Introduction to Machine Learning

The Wikipedia Guide

Contents

Chapter 1

Machine learning

For the journal, see Machine Learning (journal).

Machine learning is a subfield of computer science[1] that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.[1] Machine learning explores the construction and study of algorithms that can learn from and make predictions on data.[2] Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions,[3]:2 rather than following strictly static program instructions.

Machine learning is closely related to and often overlaps with computational statistics; a discipline that also specializes in prediction-making. It has strong ties to mathematical optimization, which deliver methods, theory and application domains to the field. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is infeasible. Example applications include spam filtering, optical character recognition (OCR),[4] search engines and computer vision. Machine learning is sometimes conflated with data mining,[5] although that focuses more on exploratory data analysis.[6] Machine learning and pattern recognition "can be viewed as two facets of the same field."[3]:vii

When employed in industrial contexts, machine learning methods may be referred to as predictive analytics or predictive modelling.

1.1 Overview

In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed".[7]

Tom M. Mitchell provided a widely quoted, more formal definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".[8] This definition is notable for its defining machine learning in fundamentally operational rather than cognitive terms, thus following Alan Turing's proposal in his paper "Computing Machinery and Intelligence" that the question "Can machines think?" be replaced with the question "Can machines do what we (as thinking entities) can do?"[9]

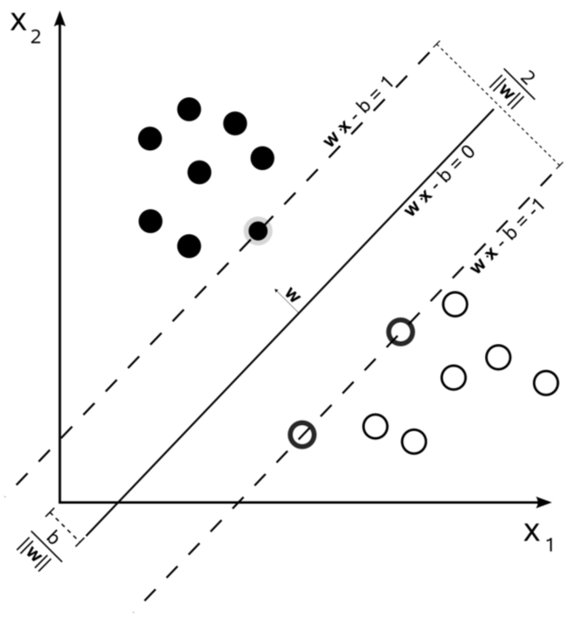
1.1.1 Types of problems and tasks

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are:[10]

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end.
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has come close to its goal or not. Another example is learning to play a game by playing against an opponent.[3]:3

Between supervised and unsupervised learning is semi-supervised learning, where the teacher gives an incomplete training signal: a training set with some (often many) of the target outputs missing. Transduction is a special case of this principle where the entire set of problem instances is known at learning time, except that part of the targets are missing.

Among other categories of machine learning problems, learning to learn learns its own inductive bias based on previous experience. Developmental learning, elaborated for robot learning, generates its own sequences (also called curriculum) of learning situations to cumulatively acquire repertoires of novel skills through autonomous



A support vector machine is a classifier that divides its input space into two regions, separated by a linear boundary. Here, it has learned to distinguish black and white circles.

self-exploration and social interaction with human teachers, and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

Another categorization of machine learning tasks arises when one considers the desired *output* of a machine-learned system:[3]:3

- In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one (or multi-label classification) or more of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".
- In regression, also a supervised problem, the outputs are continuous rather than discrete.
- In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
- Density estimation finds the distribution of inputs in some space.
- Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

1.2 History and relationships to other fields

As a scientific endeavour, machine learning grew out of the quest for artificial intelligence. Already in the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what were then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis.[10]:488

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation.[10]:488 By 1980, expert systems had come to dominate AI, and statistics was out of favor.[11] Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval.[10]:708–710; 755 Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including Hopfield, Rumelhart and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.[10]:25

Machine learning, reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics and probability theory.[11] It also benefited from the increasing availability of digitized information, and the possibility to distribute that via the internet.

Machine learning and data mining often employ the same methods and overlap significantly. They can be roughly distinguished as follows:

- Machine learning focuses on prediction, based on *known* properties learned from the training data.
- Data mining focuses on the discovery of (previously) *unknown* properties in the data. This is the analysis step of Knowledge Discovery in Databases.

The two areas overlap in many ways: data mining uses many machine learning methods, but often with a slightly different goal in mind. On the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to *reproduce known* knowledge, while in Knowledge Discovery and Data Mining (KDD) the key task is the discovery of previously *unknown* knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set examples). The difference between the two fields arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples.[12]

1.2.1 Relation to statistics

Machine learning and statistics are closely related fields. According to Michael I. Jordan, the ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics.[13] He also suggested the term data science as a placeholder to call the overall field.[13]

Leo Breiman distinguished two statistical modelling paradigms: data model and algorithmic model,[14] wherein 'algorithmic model' means more or less the machine learning algorithms like Random forest.

Some statisticians have adopted methods from machine learning, leading to a combined field that they call *statistical learning*.[15]

1.3 Theory

Main article: Computational learning theory

A core objective of a learner is to generalize from its experience.[3][16] Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

There are many similarities between machine learning theory and statistical inference, although they use different terms.

1.4 Approaches

Main article: List of machine learning algorithms

1.4.1 Decision tree learning

Main article: Decision tree learning

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

1.4.2 Association rule learning

Main article: Association rule learning

Association rule learning is a method for discovering interesting relations between variables in large databases.

1.4.3 Artificial neural networks

Main article: Artificial neural network

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is inspired by the structure and functional aspects of biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

1.4.4 Inductive logic programming

Main article: Inductive logic programming

Inductive logic programming (ILP) is an approach to rule learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as functional programs.

1.4.5 Support vector machines

Main article: Support vector machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

1.4.6 Clustering

Main article: Cluster analysis

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to some predesignated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some *similarity metric* and evaluated for example by *internal compactness* (similarity between members of the same cluster) and *separation* between different clusters. Other methods are based on *estimated density* and *graph connectivity*. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

1.4.7 Bayesian networks

Main article: Bayesian network

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning.

1.4.8 Reinforcement learning

Main article: Reinforcement learning

Reinforcement learning is concerned with how an *agent* ought to take *actions* in an *environment* so as to maximize some notion of long-term *reward*. Reinforcement learning algorithms attempt to find a *policy* that maps *states* of the world to the actions the agent ought to take in those states. Reinforcement learning differs from the supervised learning problem in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.

1.4.9 Representation learning

Main article: Representation learning

Several learning algorithms, mostly unsupervised learning algorithms, aim at discovering better representations of the inputs provided during training. Classical examples include principal components analysis and cluster analysis. Representation learning algorithms often attempt to preserve the information in their input but transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions, allowing to reconstruct the inputs coming from the unknown data generating distribution, while not being necessarily faithful for configurations that are implausible under that distribution.

Manifold learning algorithms attempt to do so under the constraint that the learned representation is low-dimensional. Sparse coding algorithms attempt to do so under the constraint that the learned representation is sparse (has many zeros). Multilinear subspace learning algorithms aim to learn low-dimensional representations directly from tensor representations for multidimensional data, without reshaping them into (high-dimensional) vectors.[17] Deep learning algorithms discover multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of (or generating) lower-level features. It has been argued that an intelligent machine is one that learns a representation that disentangles the underlying factors of variation that explain the observed data.[18]

1.4.10 Similarity and metric learning

Main article: Similarity learning

In this problem, the learning machine is given pairs of examples that are considered similar and pairs of less similar objects. It then needs to learn a similarity function (or a distance metric function) that can predict if new objects are similar. It is sometimes used in Recommendation systems.

1.4.11 Sparse dictionary learning

In this method, a datum is represented as a linear combination of basis functions, and the coefficients are assumed to be sparse. Let x be a d-dimensional datum, D be a d by n matrix, where each column of D represents a basis function. r is the coefficient to represent x using D. Mathematically, sparse dictionary learning means the following $x \approx Dr$ where r is sparse. Generally speaking, n is assumed to be larger than d to allow the freedom for a sparse representation.

Learning a dictionary along with sparse representations is strongly NP-hard and also difficult to solve approximately.[19] A popular heuristic method for sparse dictionary learning is K-SVD.

Sparse dictionary learning has been applied in several contexts. In classification, the problem is to determine which classes a previously unseen datum belongs to. Suppose a dictionary for each class has already been built. Then a new datum is associated with the class such that it's best sparsely represented by the corresponding dictionary. Sparse dictionary learning has also been applied in image de-noising. The key idea is that a clean image patch can be sparsely represented by an image dictionary, but the noise cannot.[20]

1.4.12 Genetic algorithms

Main article: Genetic algorithm

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection, and uses methods such as mutation and crossover to generate new genotype in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s.[21][22] Vice versa, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.[23]

1.5 Applications

Applications for machine learning include:

- Adaptive websites
- Affective computing
- Bioinformatics
- Brain-machine interfaces
- Cheminformatics
- Classifying DNA sequences
- Computational advertising
- Computational finance
- Computer vision, including object recognition
- Detecting credit card fraud
- Game playing[24]
- Information retrieval
- Internet fraud detection
- Machine perception
- Medical diagnosis
- Natural language processing[25]
- Optimization and metaheuristic
- Recommender systems
- Robot locomotion
- Search engines
- Sentiment analysis (or opinion mining)
- Sequence mining
- Software engineering
- Speech and handwriting recognition
- Stock market analysis
- Structural health monitoring
- Syntactic pattern recognition

In 2006, the online movie company Netflix held the first "Netflix Prize" competition to find a program to better predict user preferences and improve the accuracy on its existing Cinematch movie recommendation algorithm by at least 10%. A joint team made up of researchers from AT&T Labs-Research in collaboration with the teams Big Chaos and Pragmatic Theory built an ensemble model to win the Grand Prize in 2009 for $1 million.[26] Shortly after the prize was awarded, Netflix realized that viewers' ratings were not the best indicators of their viewing patterns ("everything is a recommendation") and they changed their recommendation engine accordingly.[27]

In 2010 The Wall Street Journal wrote about money management firm Rebellion Research's use of machine learning to predict economic movements. The article describes Rebellion Research's prediction of the financial crisis and economic recovery.[28]

In 2014 it has been reported that a machine learning algorithm has been applied in Art History to study fine art paintings, and that it may have revealed previously unrecognized influences between artists.[29]

1.6 Software

Software suites containing a variety of machine learning algorithms include the following:

1.6.1 Open-source software

- dlib
- ELKI
- Encog
- H2O
- Mahout
- mlpy
- MLPACK
- MOA (Massive Online Analysis)
- ND4J with Deeplearning4j
- OpenCV
- OpenNN
- Orange
- R
- scikit-learn
- Shogun
- Torch (machine learning)
- Spark
- Yooreeka
- Weka

1.6.2 Commercial software with open-source editions

- KNIME
- RapidMiner

1.6.3 Commercial software

- Amazon Machine Learning
- Angoss KnowledgeSTUDIO
- Databricks
- IBM SPSS Modeler
- KXEN Modeler
- LIONsolver
- Mathematica
- MATLAB
- Microsoft Azure Machine Learning
- Neural Designer
- NeuroSolutions
- Oracle Data Mining
- RCASE
- SAS Enterprise Miner
- STATISTICA Data Miner

1.7 Journals

- *Journal of Machine Learning Research*
- *Machine Learning*
- *Neural Computation*

1.8 Conferences

- Conference on Neural Information Processing Systems
- International Conference on Machine Learning

1.9 See also

- Adaptive control
- Adversarial machine learning
- Automatic reasoning
- Cache language model
- Cognitive model
- Cognitive science
- Computational intelligence
- Computational neuroscience
- Ethics of artificial intelligence
- Existential risk of artificial general intelligence
- Explanation-based learning
- Hidden Markov model
- Important publications in machine learning
- List of machine learning algorithms

1.10 References

[1] http://www.britannica.com/EBchecked/topic/1116194/machine-learning This is a tertiary source that clearly includes information from other sources but does not name them.

[2] Ron Kohavi; Foster Provost (1998). "Glossary of terms". *Machine Learning* **30**: 271–274.

[3] C. M. Bishop (2006). *Pattern Recognition and Machine Learning*. Springer. ISBN 0-387-31073-8.

[4] Wernick, Yang, Brankov, Yourganov and Strother, Machine Learning in Medical Imaging, *IEEE Signal Processing Magazine*, vol. 27, no. 4, July 2010, pp. 25-38

[5] Mannila, Heikki (1996). *Data mining: machine learning, statistics, and databases*. Int'l Conf. Scientific and Statistical Database Management. IEEE Computer Society.

[6] Friedman, Jerome H. (1998). "Data Mining and Statistics: What's the connection?". *Computing Science and Statistics* **29** (1): 3–9.

[7] Phil Simon (March 18, 2013). *Too Big to Ignore: The Business Case for Big Data*. Wiley. p. 89. ISBN 978-1-118-63817-0.

[8] • Mitchell, T. (1997). *Machine Learning*, McGraw Hill. ISBN 0-07-042807-7, p.2.

[9] Harnad, Stevan (2008), "The Annotation Game: On Turing (1950) on Computing, Machinery, and Intelligence", in Epstein, Robert; Peters, Grace, *The Turing Test Sourcebook: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, Kluwer

[10] Russell, Stuart; Norvig, Peter (2003) [1995]. *Artificial Intelligence: A Modern Approach* (2nd ed.). Prentice Hall. ISBN 978-0137903955.

[11] Langley, Pat (2011). "The changing science of machine learning". *Machine Learning* **82** (3): 275–279. doi:10.1007/s10994-011-5242-y.

[12] Le Roux, Nicolas; Bengio, Yoshua; Fitzgibbon, Andrew (2012). "Improving First and Second-Order Methods by Modeling Uncertainty". In Sra, Suvrit; Nowozin, Sebastian; Wright, Stephen J. *Optimization for Machine Learning*. MIT Press. p. 404.

[13] MI Jordan (2014-09-10). "statistics and machine learning". reddit. Retrieved 2014-10-01.

[14] http://projecteuclid.org/download/pdf_1/euclid.ss/1009213726

[15] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Springer. p. vii.

[16] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) *Foundations of Machine Learning*, MIT Press ISBN 978-0-262-01825-8.

[17] Lu, Haiping; Plataniotis, K.N.; Venetsanopoulos, A.N. (2011). "A Survey of Multilinear Subspace Learning for Tensor Data" (PDF). *Pattern Recognition* **44** (7): 1540–1551. doi:10.1016/j.patcog.2011.01.004.

[18] Yoshua Bengio (2009). *Learning Deep Architectures for AI*. Now Publishers Inc. pp. 1–3. ISBN 978-1-60198-294-0.

[19] A. M. Tillmann, "On the Computational Intractability of Exact and Approximate Dictionary Learning", IEEE Signal Processing Letters 22(1), 2015: 45–49.

[20] Aharon, M, M Elad, and A Bruckstein. 2006. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation." Signal Processing, IEEE Transactions on 54 (11): 4311-4322

[21] Goldberg, David E.; Holland, John H. (1988). "Genetic algorithms and machine learning". *Machine Learning* **3** (2): 95–99.

[22] Michie, D.; Spiegelhalter, D. J.; Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

[23] Zhang, Jun; Zhan, Zhi-hui; Lin, Ying; Chen, Ni; Gong, Yue-jiao; Zhong, Jing-hui; Chung, Henry S.H.; Li, Yun; Shi, Yu-hui (2011). "Evolutionary Computation Meets Machine Learning: A Survey" (PDF). *Computational Intelligence Magazine* (IEEE) **6** (4): 68–75.

[24] Tesauro, Gerald (March 1995). "Temporal Difference Learning and TD-Gammon". *Communications of the ACM* **38** (3).

[25] Daniel Jurafsky and James H. Martin (2009). *Speech and Language Processing*. Pearson Education. pp. 207 ff.

[26] "BelKor Home Page" research.att.com

[27]

[28]

[29] When A Machine Learning Algorithm Studied Fine Art Paintings, It Saw Things Art Historians Had Never Noticed, *The Physics at ArXiv blog*

1.11 Further reading

- Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012). *Foundations of Machine Learning*, The MIT Press. ISBN 978-0-262-01825-8.
- Ian H. Witten and Eibe Frank (2011). *Data Mining: Practical machine learning tools and techniques* Morgan Kaufmann, 664pp., ISBN 978-0-12-374856-0.
- Sergios Theodoridis, Konstantinos Koutroumbas (2009) "Pattern Recognition", 4th Edition, Academic Press, ISBN 978-1-59749-272-0.
- Mierswa, Ingo and Wurst, Michael and Klinkenberg, Ralf and Scholz, Martin and Euler, Timm: *YALE: Rapid Prototyping for Complex Data Mining Tasks*, in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006.
- Bing Liu (2007), *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data.* Springer, ISBN 3-540-37881-2
- Toby Segaran (2007), *Programming Collective Intelligence*, O'Reilly, ISBN 0-596-52932-5
- Huang T.-M., Kecman V., Kopriva I. (2006), Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning, Springer-Verlag, Berlin, Heidelberg, 260 pp. 96 illus., Hardcover, ISBN 3-540-31681-7.
- Ethem Alpaydın (2004) *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, MIT Press, ISBN 0-262-01211-1
- MacKay, D.J.C. (2003). *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press. ISBN 0-521-64298-1.
- KECMAN Vojislav (2001), Learning and Soft Computing, Support Vector Machines, Neural Networks and Fuzzy Logic Models, The MIT Press, Cambridge, MA, 608 pp., 268 illus., ISBN 0-262-11255-8.
- Trevor Hastie, Robert Tibshirani and Jerome Friedman (2001). *The Elements of Statistical Learning*, Springer. ISBN 0-387-95284-5.
- Richard O. Duda, Peter E. Hart, David G. Stork (2001) *Pattern classification* (2nd edition), Wiley, New York, ISBN 0-471-05669-3.
- Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press. ISBN 0-19-853864-2.
- Ryszard S. Michalski, George Tecuci (1994), *Machine Learning: A Multistrategy Approach*, Volume IV, Morgan Kaufmann, ISBN 1-55860-251-8.
- Sholom Weiss and Casimir Kulikowski (1991). *Computer Systems That Learn*, Morgan Kaufmann. ISBN 1-55860-065-5.
- Yves Kodratoff, Ryszard S. Michalski (1990), *Machine Learning: An Artificial Intelligence Approach, Volume III*, Morgan Kaufmann, ISBN 1-55860-119-8.
- Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (1986), *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, ISBN 0-934613-00-1.
- Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (1983), *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, ISBN 0-935382-05-4.
- Vladimir Vapnik (1998). *Statistical Learning Theory*. Wiley-Interscience, ISBN 0-471-03003-1.
- Ray Solomonoff, *An Inductive Inference Machine*, IRE Convention Record, Section on Information Theory, Part 2, pp., 56-62, 1957.
- Ray Solomonoff, "An Inductive Inference Machine" A privately circulated report from the 1956 Dartmouth Summer Research Conference on AI.

1.12 External links

- International Machine Learning Society
- Popular online course by Andrew Ng, at Coursera. It uses GNU Octave. The course is a free version of Stanford University's actual course taught by Ng, whose lectures are also available for free.
- Machine Learning Video Lectures
- mloss is an academic database of open-source machine learning software.

Chapter 2

Artificial intelligence

"AI" redirects here. For other uses, see Ai and Artificial intelligence (disambiguation).

Artificial intelligence (**AI**) is the intelligence exhibited by machines or software. It is also the name of the academic field of study which studies how to create computers and computer software that are capable of intelligent behavior. Major AI researchers and textbooks define this field as "the study and design of intelligent agents",[1] in which an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success.[2] John McCarthy, who coined the term in 1955,[3] defines it as "the science and engineering of making intelligent machines".[4]

AI research is highly technical and specialized, and is deeply divided into subfields that often fail to communicate with each other.[5] Some of the division is due to social and cultural factors: subfields have grown up around particular institutions and the work of individual researchers. AI research is also divided by several technical issues. Some subfields focus on the solution of specific problems. Others focus on one of several possible approaches or on the use of a particular tool or towards the accomplishment of particular applications.

The central problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing (communication), perception and the ability to move and manipulate objects.[6] General intelligence is still among the field's long-term goals.[7] Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are a large number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. The AI field is interdisciplinary, in which a number of sciences and professions converge, including computer science, mathematics, psychology, linguistics, philosophy and neuroscience, as well as other specialized fields such as artificial psychology.

The field was founded on the claim that a central property of humans, intelligence—the sapience of *Homo sapiens*—"can be so precisely described that a machine can be made to simulate it."[8] This raises philosophical issues about the nature of the mind and the ethics of creating artificial beings endowed with human-like intelligence, issues which have been addressed by myth, fiction and philosophy since antiquity.[9] Artificial intelligence has been the subject of tremendous optimism[10] but has also suffered stunning setbacks.[11] Today it has become an essential part of the technology industry, providing the heavy lifting for many of the most challenging problems in computer science.[12]

2.1 History

Main articles: History of artificial intelligence and Timeline of artificial intelligence

Thinking machines and artificial beings appear in Greek myths, such as Talos of Crete, the bronze robot of Hephaestus, and Pygmalion's Galatea.[13] Human likenesses believed to have intelligence were built in every major civilization: animated cult images were worshiped in Egypt and Greece[14] and humanoid automatons were built by Yan Shi, Hero of Alexandria and Al-Jazari.[15] It was also widely believed that artificial beings had been created by Jābir ibn Hayyān, Judah Loew and Paracelsus.[16] By the 19th and 20th centuries, artificial beings had become a common feature in fiction, as in Mary Shelley's *Frankenstein* or Karel Čapek's *R.U.R. (Rossum's Universal Robots)*.[17] Pamela McCorduck argues that all of these are some examples of an ancient urge, as she describes it, "to forge the gods".[9] Stories of these creatures and their fates discuss many of the same hopes, fears and ethical concerns that are presented by artificial intelligence.

Mechanical or "formal" reasoning has been developed by philosophers and mathematicians since antiquity. The study of logic led directly to the invention of the programmable digital electronic computer, based on the work of mathematician Alan Turing and others. Turing's theory of computation suggested that a machine, by shuffling symbols as simple as "0" and "1", could simulate any conceivable act of mathematical deduction.[18][19] This, along with concurrent discoveries in neurology, information theory and cybernetics, inspired a small group of researchers to begin to seriously consider the possibility of building an electronic brain.[20]

The field of AI research was founded at a conference on the campus of Dartmouth College in the summer of 1956.[21] The attendees, including John McCarthy, Marvin Minsky, Allen Newell, Arthur Samuel, and Herbert Simon, became the leaders of AI research for many decades.[22] They and their students wrote programs that were, to most people, simply astonishing:[23] computers were winning at checkers, solving word problems in algebra, proving logical theorems and speaking English.[24] By the middle of the 1960s, research in the U.S. was heavily funded by the Department of Defense[25] and laboratories had been established around the world.[26] AI's founders were profoundly optimistic about the future of the new field: Herbert Simon predicted that "machines will be capable, within twenty years, of doing any work a man can do" and Marvin Minsky agreed, writing that "within a generation ... the problem of creating 'artificial intelligence' will substantially be solved".[27]

They had failed to recognize the difficulty of some of the problems they faced.[28] In 1974, in response to the criticism of Sir James Lighthill[29] and ongoing pressure from the US Congress to fund more productive projects, both the U.S. and British governments cut off all undirected exploratory research in AI. The next few years would later be called an "AI winter",[30] a period when funding for AI projects was hard to find.

In the early 1980s, AI research was revived by the commercial success of expert systems,[31] a form of AI program that simulated the knowledge and analytical skills of one or more human experts. By 1985 the market for AI had reached over a billion dollars. At the same time, Japan's fifth generation computer project inspired the U.S and British governments to restore funding for academic research in the field.[32] However, beginning with the collapse of the Lisp Machine market in 1987, AI once again fell into disrepute, and a second, longer lasting AI winter began.[33]

In the 1990s and early 21st century, AI achieved its greatest successes, albeit somewhat behind the scenes. Artificial intelligence is used for logistics, data mining, medical diagnosis and many other areas throughout the technology industry.[12] The success was due to several factors: the increasing computational power of computers (see Moore's law), a greater emphasis on solving specific subproblems, the creation of new ties between AI and other fields working on similar problems, and a new commitment by researchers to solid mathematical methods and rigorous scientific standards.[34]

On 11 May 1997, Deep Blue became the first computer chess-playing system to beat a reigning world chess champion, Garry Kasparov.[35] In February 2011, in a *Jeopardy!* quiz show exhibition match, IBM's question answering system, Watson, defeated the two greatest Jeopardy champions, Brad Rutter and Ken Jennings, by a significant margin.[36] The Kinect, which provides a 3D body–motion interface for the Xbox 360 and the Xbox One, uses algorithms that emerged from lengthy AI research[37] as do intelligent personal assistants in smartphones.[38]

2.2 Research

2.2.1 Goals

> You awake one morning to find your brain has another lobe functioning. Invisible, this auxiliary lobe answers your questions with information beyond the realm of your own memory, suggests plausible courses of action, and asks questions that help bring out relevant facts. You quickly come to rely on the new lobe so much that you stop wondering how it works. You just use it. This is the dream of artificial intelligence.
> —*BYTE*, April 1985[39]

The general problem of simulating (or creating) intelligence has been broken down into a number of specific sub-problems. These consist of particular traits or capabilities that researchers would like an intelligent system to display. The traits described below have received the most attention.[6]

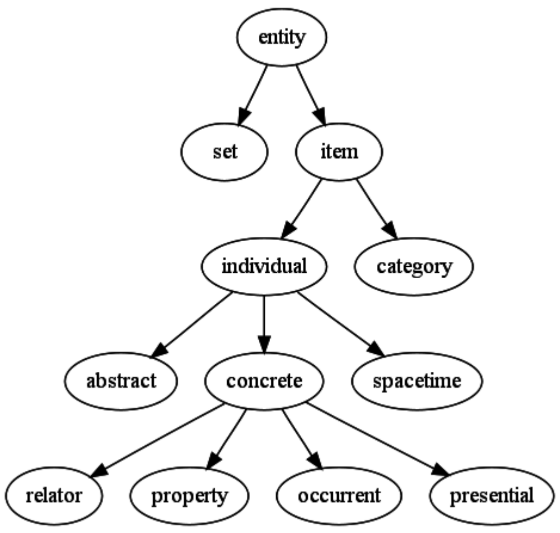
Deduction, reasoning, problem solving

Early AI researchers developed algorithms that imitated the step-by-step reasoning that humans use when they solve puzzles or make logical deductions.[40] By the late 1980s and 1990s, AI research had also developed highly successful methods for dealing with uncertain or incomplete information, employing concepts from probability and economics.[41]

For difficult problems, most of these algorithms can require enormous computational resources – most experience a "combinatorial explosion": the amount of memory or computer time required becomes astronomical when the problem goes beyond a certain size. The search for more efficient problem-solving algorithms is a high priority for AI research.[42]

Human beings solve most of their problems using fast, intuitive judgements rather than the conscious, step-by-step deduction that early AI research was able to model.[43] AI has made some progress at imitating this kind of "sub-symbolic" problem solving: embodied agent approaches emphasize the importance of sensorimotor skills to higher reasoning; neural net research attempts to simulate the structures inside the brain that give rise to this skill; statistical approaches to AI mimic the probabilistic nature of the human ability to guess.

Knowledge representation



An ontology represents knowledge as a set of concepts within a domain and the relationships between those concepts.

Main articles: Knowledge representation and Commonsense knowledge

Knowledge representation[44] and knowledge engineering[45] are central to AI research. Many of the problems machines are expected to solve will require extensive knowledge about the world. Among the things that AI needs to represent are: objects, properties, categories and relations between objects;[46] situations, events, states and time;[47] causes and effects;[48] knowledge about knowledge (what we know about what other people know);[49] and many other, less well researched domains. A representation of "what exists" is an ontology: the set of objects, relations, concepts and so on that the machine knows about. The most general are called upper ontologies, which attempt to provide a foundation for all other knowledge.[50]

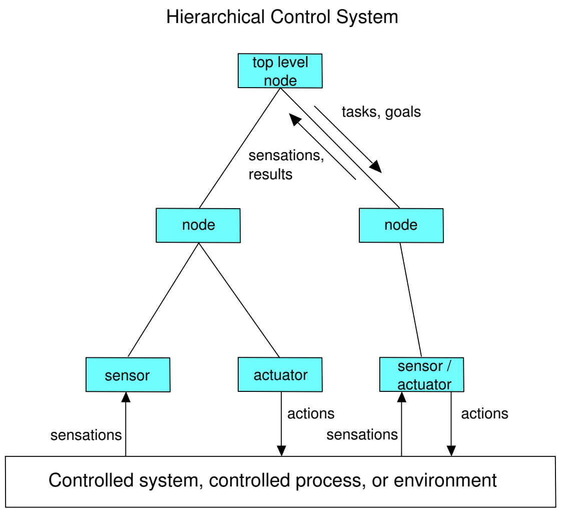
Among the most difficult problems in knowledge representation are:

Default reasoning and the qualification problem Many of the things people know take the form of "working assumptions." For example, if a bird comes up in conversation, people typically picture an animal that is fist sized, sings, and flies. None of these things are true about all birds. John McCarthy identified this problem in 1969[51] as the qualification problem: for any commonsense rule that AI researchers care to represent, there tend to be a huge number of exceptions. Almost nothing is simply true or false in the way that abstract logic requires. AI research has explored a number of solutions to this problem.[52]

The breadth of commonsense knowledge The number of atomic facts that the average person knows is astronomical. Research projects that attempt to build a complete knowledge base of commonsense knowledge (e.g., Cyc) require enormous amounts of laborious ontological engineering—they must be built, by hand, one complicated concept at a time.[53] A major goal is to have the computer understand enough concepts to be able to learn by reading from sources like the internet, and thus be able to add to its own ontology.

The subsymbolic form of some commonsense knowledge Much of what people know is not represented as "facts" or "statements" that they could express verbally. For example, a chess master will avoid a particular chess position because it "feels too exposed"[54] or an art critic can take one look at a statue and instantly realize that it is a fake.[55] These are intuitions or tendencies that are represented in the brain non-consciously and sub-symbolically.[56] Knowledge like this informs, supports and provides a context for symbolic, conscious knowledge. As with the related problem of sub-symbolic reasoning, it is hoped that situated AI, computational intelligence, or statistical AI will provide ways to represent this kind of knowledge.[56]

Planning



A hierarchical control system is a form of control system in which a set of devices and governing software is arranged in a hierarchy.

Main article: Automated planning and scheduling

Intelligent agents must be able to set goals and achieve them.[57] They need a way to visualize the future (they must have a representation of the state of the world and be able to make predictions about how their actions will change it) and be able to make choices that maximize the utility (or "value") of the available choices.[58]

In classical planning problems, the agent can assume that it is the only thing acting on the world and it can be certain what the consequences of its actions may be.[59] However, if the agent is not the only actor, it must periodically ascertain whether the world matches its predictions and it must change its plan as this becomes necessary, requiring the agent to reason under uncertainty.[60]

Multi-agent planning uses the cooperation and competition of many agents to achieve a given goal. Emergent behavior such as this is used by evolutionary algorithms and swarm intelligence.[61]

Learning

Main article: Machine learning

Machine learning is the study of computer algorithms that improve automatically through experience[62][63] and has been central to AI research since the field's inception.[64]

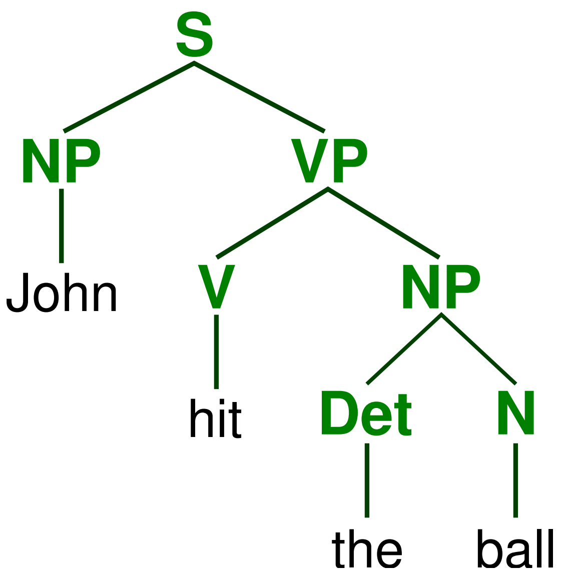
Unsupervised learning is the ability to find patterns in a stream of input. Supervised learning includes both classification and numerical regression. Classification is used to determine what category something belongs in, after seeing a number of examples of things from several categories. Regression is the attempt to produce a function that describes the relationship between inputs and outputs and predicts how the outputs should change as the inputs change. In reinforcement learning[65] the agent is rewarded for good responses and punished for bad ones. The agent uses this sequence of rewards and punishments to form a strategy for operating in its problem space. These three types of learning can be analyzed in terms of decision theory, using concepts like utility. The mathematical analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory.[66]

Within developmental robotics, developmental learning approaches were elaborated for lifelong cumulative acquisition of repertoires of novel skills by a robot, through autonomous self-exploration and social interaction with human teachers, and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation.[67][68][69][70]

Natural language processing (communication)

Main article: Natural language processing

Natural language processing[71] gives machines the ability to read and understand the languages that humans



A parse tree represents the syntactic structure of a sentence according to some formal grammar.

speak. A sufficiently powerful natural language processing system would enable natural language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval (or text mining), question answering[72] and machine translation.[73]

A common method of processing and extracting meaning from natural language is through semantic indexing. Increases in processing speeds and the drop in the cost of data storage makes indexing large volumes of abstractions of the user's input much more efficient.

Perception

Main articles: Machine perception, Computer vision and Speech recognition

Machine perception[74] is the ability to use input from sensors (such as cameras, microphones, tactile sensors, sonar and others more exotic) to deduce aspects of the world. Computer vision[75] is the ability to analyze visual input. A few selected subproblems are speech recognition,[76] facial recognition and object recognition.[77]

Motion and manipulation

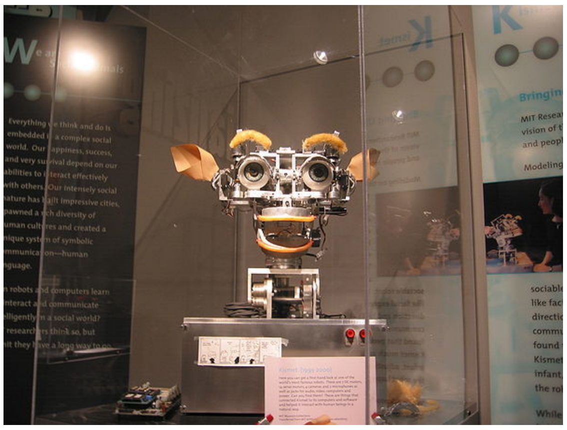
Main article: Robotics

The field of robotics[78] is closely related to AI. Intelligence is required for robots to be able to handle such tasks as object manipulation[79] and navigation, with subproblems of localization (knowing where you are, or finding out where other things are), mapping (learning what is around you, building a map of the environment), and motion planning (figuring out how to get there) or path planning (going from one point in space to another point, which may involve compliant motion – where the robot moves while maintaining physical contact with an object).[80][81]

Long-term goals

Among the long-term goals in the research pertaining to artificial intelligence are: (1) Social intelligence, (2) Creativity, and (3) General intelligence.

Social intelligence Main article: Affective computing
Affective computing is the study and development of



Kismet, a robot with rudimentary social skills

systems and devices that can recognize, interpret, process, and simulate human affects.[82][83] It is an interdisciplinary field spanning computer sciences, psychology, and cognitive science.[84] While the origins of the field may be traced as far back as to early philosophical inquiries into emotion,[85] the more modern branch of computer science originated with Rosalind Picard's 1995 paper[86] on affective computing.[87][88] A motivation for the research is the ability to simulate empathy. The machine should interpret the emotional state of humans and adapt its behaviour to them, giving an appropriate response for those emotions.

Emotion and social skills[89] play two roles for an intelligent agent. First, it must be able to predict the actions of others, by understanding their motives and emotional states. (This involves elements of game theory, decision theory, as well as the ability to model human emotions and the perceptual skills to detect emotions.) Also, in an effort to facilitate human-computer interaction, an intelligent machine might want to be able to *display* emotions—even if it does not actually experience them itself—in order to appear sensitive to the emotional dynamics of human interaction.

Creativity Main article: Computational creativity

A sub-field of AI addresses creativity both theoretically (from a philosophical and psychological perspective) and practically (via specific implementations of systems that generate outputs that can be considered creative, or systems that identify and assess creativity). Related areas of computational research are Artificial intuition and Artificial thinking.

General intelligence Main articles: Artificial general intelligence and AI-complete

Many researchers think that their work will eventually be incorporated into a machine with *general* intelligence (known as strong AI), combining all the skills above and exceeding human abilities at most or all of them.[7] A few believe that anthropomorphic features like artificial consciousness or an artificial brain may be required for such a project.[90][91]

Many of the problems above may require general intelligence to be considered solved. For example, even a straightforward, specific task like machine translation requires that the machine read and write in both languages (NLP), follow the author's argument (reason), know what is being talked about (knowledge), and faithfully reproduce the author's intention (social intelligence). A problem like machine translation is considered "AI-complete". In order to solve this particular problem, you must solve all the problems.[92]

2.2.2 Approaches

There is no established unifying theory or paradigm that guides AI research. Researchers disagree about many issues.[93] A few of the most long standing questions that have remained unanswered are these: should artificial intelligence simulate natural intelligence by studying psychology or neurology? Or is human biology as irrelevant to AI research as bird biology is to aeronautical engineering?[94] Can intelligent behavior be described using simple, elegant principles (such as logic or optimization)? Or does it necessarily require solving a large number of completely unrelated problems?[95] Can intelligence be reproduced using high-level symbols, similar to words and ideas? Or does it require "sub-symbolic" processing?[96] John Haugeland, who coined the term GOFAI (Good Old-Fashioned Artificial Intelligence), also proposed that AI should more properly be referred to as synthetic intelligence,[97] a term which has since been adopted by some non-GOFAI researchers.[98][99]

Cybernetics and brain simulation

Main articles: Cybernetics and Computational neuroscience

In the 1940s and 1950s, a number of researchers explored the connection between neurology, information theory, and cybernetics. Some of them built machines that used electronic networks to exhibit rudimentary intelligence, such as W. Grey Walter's turtles and the Johns Hopkins Beast. Many of these researchers gathered for meetings of the Teleological Society at Princeton University and the Ratio Club in England.[20] By 1960, this approach was largely abandoned, although elements of it would be revived in the 1980s.

Symbolic

Main article: Symbolic AI

When access to digital computers became possible in the middle 1950s, AI research began to explore the possibility that human intelligence could be reduced to symbol manipulation. The research was centered in three institutions: Carnegie Mellon University, Stanford and MIT, and each one developed its own style of research. John Haugeland named these approaches to AI “good old fashioned AI” or "GOFAI".[100] During the 1960s, symbolic approaches had achieved great success at simulating high-level thinking in small demonstration programs. Approaches based on cybernetics or neural networks were abandoned or pushed into the background.[101] Researchers in the 1960s and the 1970s were convinced that symbolic approaches would eventually succeed in creating a machine with artificial general intelligence and considered this the goal of their field.

Cognitive simulation Economist Herbert Simon and Allen Newell studied human problem-solving skills and attempted to formalize them, and their work laid the foundations of the field of artificial intelligence, as well as cognitive science, operations research and management science. Their research team used the results of psychological experiments to develop programs that simulated the techniques that people used to solve problems. This tradition, centered at Carnegie Mellon University would eventually culminate in the development of the Soar architecture in the middle 1980s.[102][103]

Logic-based Unlike Newell and Simon, John McCarthy felt that machines did not need to simulate human thought, but should instead try to find the essence of abstract reasoning and problem solving, regardless of whether people used the same algorithms.[94] His laboratory at Stanford (SAIL) focused on using formal logic to solve a wide variety of problems, including knowledge representation, planning and learning.[104] Logic was also the focus of the work at the University of Edinburgh and elsewhere in Europe which led to the development of the programming language Prolog and the science of logic programming.[105]

“Anti-logic” or “scruffy” Researchers at MIT (such as Marvin Minsky and Seymour Papert)[106] found that solving difficult problems in vision and natural language processing required ad-hoc solutions – they argued that there was no simple and general principle (like logic) that would capture all the aspects of intelligent behavior. Roger Schank described their “anti-logic” approaches as "scruffy" (as opposed to the "neat" paradigms at CMU and Stanford).[95] Commonsense knowledge bases (such as Doug Lenat's Cyc) are an example of “scruffy” AI, since they must be built by hand, one complicated concept at a time.[107]

Knowledge-based When computers with large memories became available around 1970, researchers from all three traditions began to build knowledge into AI applications.[108] This “knowledge revolution” led to the development and deployment of expert systems (introduced by Edward Feigenbaum), the first truly successful form of AI software.[31] The knowledge revolution was also driven by the realization that enormous amounts of knowledge would be required by many simple AI applications.

Sub-symbolic

By the 1980s progress in symbolic AI seemed to stall and many believed that symbolic systems would never be able to imitate all the processes of human cognition, especially perception, robotics, learning and pattern recognition. A number of researchers began to look into “sub-symbolic” approaches to specific AI problems.[96] r

Bottom-up, embodied, situated, behavior-based or nouvelle AI Researchers from the related field of robotics, such as Rodney Brooks, rejected symbolic AI and focused on the basic engineering problems that would allow robots to move and survive.[109] Their work revived the non-symbolic viewpoint of the early cybernetics researchers of the 1950s and reintroduced the use of control theory in AI. This coincided with the development of the embodied mind thesis in the related field of cognitive science: the idea that aspects of the body (such as movement, perception and visualization) are required for higher intelligence.

Computational intelligence and soft computing Interest in neural networks and "connectionism" was revived by David Rumelhart and others in the middle 1980s.[110] Neural networks are an example of soft computing --- they are solutions to problems which cannot be solved with complete logical certainty, and where an approximate solution is often enough. Other soft computing approaches to AI include fuzzy systems, evolutionary computation and many statistical tools. The application of soft computing to AI is studied collectively by the emerging discipline of computational intelligence.[111]

Statistical

In the 1990s, AI researchers developed sophisticated mathematical tools to solve specific subproblems. These tools are truly scientific, in the sense that their results are both measurable and verifiable, and they have been responsible for many of AI's recent successes. The shared mathematical language has also permitted a high level of collaboration with more established fields (like mathematics, economics or operations research). Stuart Russell and Peter Norvig describe this movement as nothing less than a "revolution" and "the victory of the neats."[34] Critics argue that these techniques (with few exceptions[112]) are too focused on particular problems and have failed to address the long-term goal of general intelligence.[113] There is an ongoing debate about the relevance and validity of statistical approaches in AI, exemplified in part by exchanges between Peter Norvig and Noam Chomsky.[114][115]

Integrating the approaches

Intelligent agent paradigm An intelligent agent is a system that perceives its environment and takes actions which maximize its chances of success. The simplest intelligent agents are programs that solve specific problems. More complicated agents include human beings and organizations of human beings (such as firms). The paradigm gives researchers license to study isolated problems and find solutions that are both verifiable and useful, without agreeing on one single approach. An agent that solves a specific problem can use any approach that works – some agents are symbolic and logical, some are sub-symbolic neural networks and others may use new approaches. The paradigm also gives researchers a common language to communicate with other fields—such as decision theory and economics—that also use concepts of abstract agents. The intelligent agent paradigm became widely accepted during the 1990s.[2]

Agent architectures and cognitive architectures Researchers have designed systems to build intelligent systems out of interacting intelligent agents in a multi-agent system.[116] A system with both symbolic and sub-symbolic components is a hybrid intelligent system, and the study of such systems is artificial intelligence systems integration. A hierarchical control system provides a bridge between sub-symbolic AI at its lowest, reactive levels and traditional symbolic AI at its highest levels, where relaxed time constraints permit planning and world modelling.[117] Rodney Brooks' subsumption architecture was an early proposal for such a hierarchical system.[118]

2.2.3 Tools

In the course of 50 years of research, AI has developed a large number of tools to solve the most difficult problems in computer science. A few of the most general of these methods are discussed below.

Search and optimization

Main articles: Search algorithm, Mathematical optimization and Evolutionary computation

Many problems in AI can be solved in theory by intelligently searching through many possible solutions:[119] Reasoning can be reduced to performing a search. For example, logical proof can be viewed as searching for a path that leads from premises to conclusions, where each step is the application of an inference rule.[120] Planning algorithms search through trees of goals and subgoals, attempting to find a path to a target goal, a process called means-ends analysis.[121] Robotics algorithms for moving limbs and grasping objects use local searches in configuration space.[79] Many learning algorithms use search algorithms based on optimization.

Simple exhaustive searches[122] are rarely sufficient for most real world problems: the search space (the number of places to search) quickly grows to astronomical numbers. The result is a search that is too slow or never completes. The solution, for many problems, is to use "heuristics" or "rules of thumb" that eliminate choices that are unlikely to lead to the goal (called "pruning the search tree"). Heuristics supply the program with a "best guess" for the path on which the solution lies.[123] Heuristics limit the search for solutions into a smaller sample size.[80]

A very different kind of search came to prominence in the 1990s, based on the mathematical theory of optimization. For many problems, it is possible to begin the search with some form of a guess and then refine the guess incrementally until no more refinements can be made. These algorithms can be visualized as blind hill climbing: we begin the search at a random point on the landscape, and then, by jumps or steps, we keep moving our guess uphill, until we reach the top. Other optimization algorithms are simulated annealing, beam search and random optimization.[124]

Evolutionary computation uses a form of optimization search. For example, they may begin with a population of organisms (the guesses) and then allow them to mutate and recombine, selecting only the fittest to survive each generation (refining the guesses). Forms of evolutionary computation include swarm intelligence algorithms (such as ant colony or particle swarm optimization)[125] and evolutionary algorithms (such as genetic algorithms, gene expression programming, and genetic programming).[126]

Logic

Main articles: Logic programming and Automated reasoning

Logic[127] is used for knowledge representation and problem solving, but it can be applied to other problems as well. For example, the satplan algorithm uses logic for planning[128] and inductive logic programming is a method for learning.[129]

Several different forms of logic are used in AI research. Propositional or sentential logic[130] is the logic of statements which can be true or false. First-order logic[131] also allows the use of quantifiers and predicates, and can express facts about objects, their properties, and their relations with each other. Fuzzy logic,[132] is a version of first-order logic which allows the truth of a statement to be represented as a value between 0 and 1, rather than simply True (1) or False (0). Fuzzy systems can be used for uncertain reasoning and have been widely used in modern industrial and consumer product control systems. Subjective logic[133] models uncertainty in a different and more explicit manner than fuzzy-logic: a given binomial opinion satisfies belief + disbelief + uncertainty = 1 within a Beta distribution. By this method, ignorance can be distinguished from probabilistic statements that an agent makes with high confidence.

Default logics, non-monotonic logics and circumscription[52] are forms of logic designed to help with default reasoning and the qualification problem. Several extensions of logic have been designed to handle specific domains of knowledge, such as: description logics;[46] situation calculus, event calculus and fluent calculus (for representing events and time);[47] causal calculus;[48] belief calculus; and modal logics.[49]

Probabilistic methods for uncertain reasoning

Main articles: Bayesian network, Hidden Markov model, Kalman filter, Decision theory and Utility theory

Many problems in AI (in reasoning, planning, learning, perception and robotics) require the agent to operate with incomplete or uncertain information. AI researchers have devised a number of powerful tools to solve these problems using methods from probability theory and economics.[134]

Bayesian networks[135] are a very general tool that can be used for a large number of problems: reasoning (using the Bayesian inference algorithm),[136] learning (using the expectation-maximization algorithm),[137] planning (using decision networks)[138] and perception (using dynamic Bayesian networks).[139] Probabilistic algorithms can also be used for filtering, prediction, smoothing and finding explanations for streams of data, helping perception systems to analyze processes that occur over time (e.g., hidden Markov models or Kalman filters).[139]

A key concept from the science of economics is "utility": a measure of how valuable something is to an intelligent agent. Precise mathematical tools have been developed that analyze how an agent can make choices and plan, using decision theory, decision analysis,[140] and information value theory.[58] These tools include models such as Markov decision processes,[141] dynamic decision networks,[139] game theory and mechanism design.[142]

Classifiers and statistical learning methods

Main articles: Classifier (mathematics), Statistical classification and Machine learning

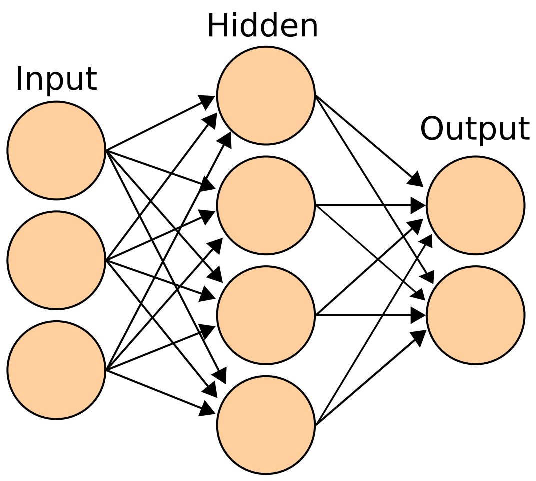
The simplest AI applications can be divided into two types: classifiers (“if shiny then diamond”) and controllers (“if shiny then pick up”). Controllers do, however, also classify conditions before inferring actions, and therefore classification forms a central part of many AI systems. Classifiers are functions that use pattern matching to determine a closest match. They can be tuned according to examples, making them very attractive for use in AI. These examples are known as observations or patterns. In supervised learning, each pattern belongs to a certain predefined class. A class can be seen as a decision that has to be made. All the observations combined with their class labels are known as a data set. When a new observation is received, that observation is classified based on previous experience.[143]

A classifier can be trained in various ways; there are many statistical and machine learning approaches. The most widely used classifiers are the neural network,[144] kernel methods such as the support vector machine,[145] k-nearest neighbor algorithm,[146] Gaussian mixture model,[147] naive Bayes classifier,[148] and decision tree.[149] The performance of these classifiers have been compared over a wide range of tasks. Classifier performance depends greatly on the characteristics of the data to be classified. There is no single classifier that works best on all given problems; this is also referred to as the "no free lunch" theorem. Determining a suitable classifier for a given problem is still more an art than science.[150]

Neural networks

Main articles: Artificial neural network and Connectionism

The study of artificial neural networks[144] began in the



A neural network is an interconnected group of nodes, akin to the vast network of neurons in the human brain.

decade before the field of AI research was founded, in the work of Walter Pitts and Warren McCullough. Other important early researchers were Frank Rosenblatt, who invented the perceptron and Paul Werbos who developed the backpropagation algorithm.[151]

The main categories of networks are acyclic or feedforward neural networks (where the signal passes in only one direction) and recurrent neural networks (which allow feedback). Among the most popular feedforward networks are perceptrons, multi-layer perceptrons and radial basis networks.[152] Among recurrent networks, the most famous is the Hopfield net, a form of attractor network, which was first described by John Hopfield in 1982.[153] Neural networks can be applied to the problem of intelligent control (for robotics) or learning, using such techniques as Hebbian learning and competitive learning.[154]

Hierarchical temporal memory is an approach that models some of the structural and algorithmic properties of the neocortex.[155] The term "deep learning" gained traction in the mid-2000s after a publication by Geoffrey Hinton and Ruslan Salakhutdinov showed how a many-layered feedforward neural network could be effectively pre-trained one layer at a time, treating each layer in turn as an unsupervised restricted Boltzmann machine, then using supervised backpropagation for fine-tuning.[156]

Control theory

Main article: Intelligent control

Control theory, the grandchild of cybernetics, has many important applications, especially in robotics.[157]

Languages

Main article: List of programming languages for artificial intelligence

AI researchers have developed several specialized languages for AI research, including Lisp[158] and Prolog.[159]

2.2.4 Evaluating progress

Main article: Progress in artificial intelligence

In 1950, Alan Turing proposed a general procedure to test the intelligence of an agent now known as the Turing test. This procedure allows almost all the major problems of artificial intelligence to be tested. However, it is a very difficult challenge and at present all agents fail.[160]

Artificial intelligence can also be evaluated on specific problems such as small problems in chemistry, handwriting recognition and game-playing. Such tests have been termed subject matter expert Turing tests. Smaller problems provide more achievable goals and there are an ever-increasing number of positive results.[161]

One classification for outcomes of an AI test is:[162]

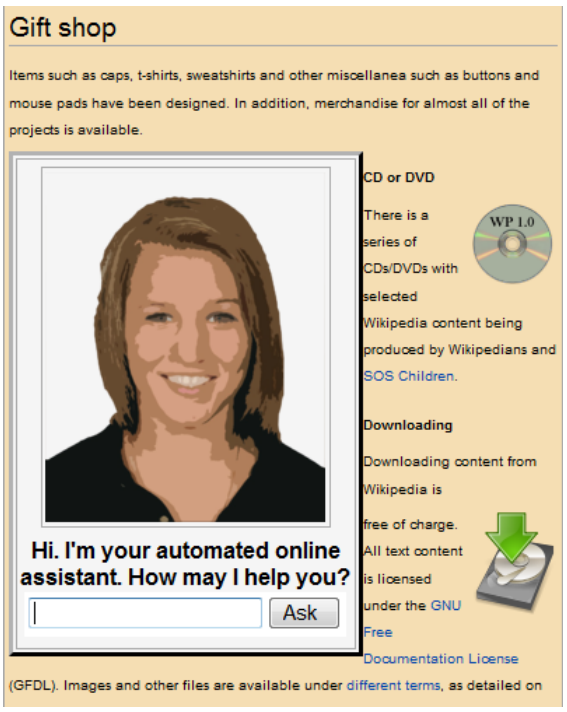
1. Optimal: it is not possible to perform better.
2. Strong super-human: performs better than all humans.
3. Super-human: performs better than most humans.
4. Sub-human: performs worse than most humans.

For example, performance at draughts (i.e. checkers) is optimal,[163] performance at chess is super-human and nearing strong super-human (see computer chess: computers versus human) and performance at many everyday tasks (such as recognizing a face or crossing a room without bumping into something) is sub-human.

A quite different approach measures machine intelligence through tests which are developed from *mathematical* definitions of intelligence. Examples of these kinds of tests start in the late nineties devising intelligence tests using notions from Kolmogorov complexity and data compression.[164] Two major advantages of mathematical definitions are their applicability to nonhuman intelligences and their absence of a requirement for human testers.

A derivative of the Turing test is the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). as the name implies, this helps to determine that a user is an actual person and not a computer posing as a human. In contrast to the standard Turing test, CAPTCHA administered by a machine and targeted to a human as opposed to being administered by a human and targeted to a machine. A computer asks a user to complete a simple test then generates a grade for that test. Computers are unable to solve the problem, so correct solutions are deemed to be the result of a person taking the test. A common type of CAPTCHA is the test that requires the typing of distorted letters, numbers or symbols that appear in an image undecipherable by a computer.[165]

2.3 Applications



Gift shop

Items such as caps, t-shirts, sweatshirts and other miscellanea such as buttons and mouse pads have been designed. In addition, merchandise for almost all of the projects is available.

CD or DVD

There is a series of CDs/DVDs with selected Wikipedia content being produced by Wikipedians and SOS Children.

Downloading

Downloading content from Wikipedia is free of charge. All text content is licensed under the GNU Free Documentation License (GFDL). Images and other files are available under different terms, as detailed on

Hi. I'm your automated online assistant. How may I help you?

An automated online assistant providing customer service on a web page – one of many very primitive applications of artificial intelligence.

Main article: Applications of artificial intelligence

Artificial intelligence techniques are pervasive and are too numerous to list. Frequently, when a technique reaches mainstream use, it is no longer considered artificial intelligence; this phenomenon is described as the AI effect.[166] An area that artificial intelligence has contributed greatly to is intrusion detection.[167]

2.3.1 Competitions and prizes

Main article: Competitions and prizes in artificial intelligence

There are a number of competitions and prizes to promote research in artificial intelligence. The main areas promoted are: general machine intelligence, conversational behavior, data-mining, robotic cars, robot soccer and games.

2.3.2 Platforms

A platform (or "computing platform") is defined as “some sort of hardware architecture or software framework (including application frameworks), that allows software to run.” As Rodney Brooks pointed out many years ago,[168] it is not just the artificial intelligence software that defines the AI features of the platform, but rather the actual platform itself that affects the AI that results, i.e., there needs to be work in AI problems on real-world platforms rather than in isolation.

A wide variety of platforms has allowed different aspects of AI to develop, ranging from expert systems, albeit PC-based but still an entire real-world system, to various robot platforms such as the widely available Roomba with open interface.[169]

2.3.3 Toys

AIBO, the first robotic pet, grew out of Sony’s Computer Science Laboratory (CSL). Famed engineer Toshitada Doi is credited as AIBO’s original progenitor: in 1994 he had started work on robots with artificial intelligence expert Masahiro Fujita, at CSL. Doi’s friend, the artist Hajime Sorayama, was enlisted to create the initial designs for the AIBO’s body. Those designs are now part of the permanent collections of Museum of Modern Art and the Smithsonian Institution, with later versions of AIBO being used in studies in Carnegie Mellon University. In 2006, AIBO was added into Carnegie Mellon University’s “Robot Hall of Fame”.

2.4 Philosophy and ethics

Main articles: Philosophy of artificial intelligence and Ethics of artificial intelligence

Alan Turing wrote in 1950 “I propose to consider the question 'can a machine think'?"[160] and began the discussion that has become the philosophy of artificial intelligence. Because “thinking” is difficult to define, there are two versions of the question that philosophers have addressed. First, can a machine be intelligent? I.e., can it solve all the problems the humans solve by using intelligence? And second, can a machine be built with a mind and the experience of subjective consciousness?[170]

The existence of an artificial intelligence that rivals or exceeds human intelligence raises difficult ethical issues, both on behalf of humans and on behalf of any possible sentient AI. The potential power of the technology inspires both hopes and fears for society.

2.4.1 The possibility/impossibility of artificial general intelligence

Main articles: philosophy of AI, Turing test, Physical symbol systems hypothesis, Dreyfus’ critique of AI, The Emperor’s New Mind and AI effect

Can a machine be intelligent? Can it “think"?

Turing’s “polite convention” We need not decide if a machine can “think"; we need only decide if a machine can act as intelligently as a human being. This approach to the philosophical problems associated with artificial intelligence forms the basis of the Turing test.[160]

The Dartmouth proposal “Every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it.” This conjecture was printed in the proposal for the Dartmouth Conference of 1956, and represents the position of most working AI researchers.[171]

Newell and Simon’s physical symbol system hypothesis “A physical symbol system has the necessary and sufficient means of general intelligent action.” Newell and Simon argue that intelligence consists of formal operations on symbols.[172] Hubert Dreyfus argued that, on the contrary, human expertise depends on unconscious instinct rather than conscious symbol manipulation and on having a “feel” for the situation rather than explicit symbolic knowledge. (See Dreyfus’ critique of AI.)[173][174]

Gödelian arguments Gödel himself,[175] John Lucas (in 1961) and Roger Penrose (in a more detailed argument from 1989 onwards) argued that humans are not reducible to Turing machines.[176] The detailed arguments are complex, but in essence they derive from Kurt Gödel's 1931 proof in his first incompleteness theorem that it is always possible to create statements that a formal system could not prove. A human being, however, can (with some thought) see the truth of these “Gödel statements”. Any Turing program designed to search for these statements can have its methods reduced to a formal system, and so will always have a “Gödel statement” derivable from its program which it can never discover. However, if humans are indeed capable of understanding mathematical truth, it doesn't seem possible that we could be limited in the same way. This is quite a general result, if accepted, since it can be shown that hardware neural nets, and computers based on random processes (e.g. annealing approaches) and quantum computers based on entangled qubits (so long as they involve no new physics) can all be reduced to Turing machines. All they do is reduce the complexity of the tasks, not permit new types of problems to be solved. Roger Penrose speculates that there may be new physics involved in our brain, perhaps at the intersection of gravity and quantum mechanics at the Planck scale. This argument, if accepted does not rule out the possibility of true artificial intelligence, but means it has to be biological in basis or based on new physical principles. The argument has been followed up by many counter arguments, and then Roger Penrose has replied to those with counter counter examples, and it is now an intricate complex debate.[177] For details see Philosophy of artificial intelligence: Lucas, Penrose and Gödel

The artificial brain argument The brain can be simulated by machines and because brains are intelligent, simulated brains must also be intelligent; thus machines can be intelligent. Hans Moravec, Ray Kurzweil and others have argued that it is technologically feasible to copy the brain directly into hardware and software, and that such a simulation will be essentially identical to the original.[91]

The AI effect Machines are *already* intelligent, but observers have failed to recognize it. When Deep Blue beat Gary Kasparov in chess, the machine was acting intelligently. However, onlookers commonly discount the behavior of an artificial intelligence program by arguing that it is not “real” intelligence after all; thus “real” intelligence is whatever intelligent behavior people can do that machines still can not. This is known as the AI Effect: “AI is whatever hasn't been done yet.”

2.4.2 Intelligent behaviour and machine ethics

As a minimum, an AI system must be able to reproduce aspects of human intelligence. This raises the issue of how ethically the machine should behave towards both humans and other AI agents. This issue was addressed by Wendell Wallach in his book titled *Moral Machines* in which he introduced the concept of artificial moral agents (AMA).[178] For Wallach, AMAs have become a part of the research landscape of artificial intelligence as guided by its two central questions which he identifies as “Does Humanity Want Computers Making Moral Decisions”[179] and “Can (Ro)bots Really Be Moral”.[180] For Wallach the question is not centered on the issue of *whether* machines can demonstrate the equivalent of moral behavior in contrast to the *constraints* which society may place on the development of AMAs.[181]

Machine ethics

Main article: Machine ethics

The field of machine ethics is concerned with giving machines ethical principles, or a procedure for discovering a way to resolve the ethical dilemmas they might encounter, enabling them to function in an ethically responsible manner through their own ethical decision making.[182] The field was delineated in the AAAI Fall 2005 Symposium on Machine Ethics: “Past research concerning the relationship between technology and ethics has largely focused on responsible and irresponsible use of technology by human beings, with a few people being interested in how human beings ought to treat machines. In all cases, only human beings have engaged in ethical reasoning. The time has come for adding an ethical dimension to at least some machines. Recognition of the ethical ramifications of behavior involving machines, as well as recent and potential developments in machine autonomy, necessitate this. In contrast to computer hacking, software property issues, privacy issues and other topics normally ascribed to computer ethics, machine ethics is concerned with the behavior of machines towards human users and other machines. Research in machine ethics is key to alleviating concerns with autonomous systems — it could be argued that the notion of autonomous machines without such a dimension is at the root of all fear concerning machine intelligence. Further, investigation of machine ethics could enable the discovery of problems with current ethical theories, advancing our thinking about Ethics.”[183] Machine ethics is sometimes referred to as machine morality, computational ethics or computational morality. A variety of perspectives of this nascent field can be found in the collected edition “Machine Ethics” [182] that stems from the AAAI Fall 2005 Symposium on Machine Ethics.[183]

Malevolent and friendly AI

Main article: Friendly AI

Political scientist Charles T. Rubin believes that AI can be neither designed nor guaranteed to be benevolent.[184] He argues that “any sufficiently advanced benevolence may be indistinguishable from malevolence.” Humans should not assume machines or robots would treat us favorably, because there is no *a priori* reason to believe that they would be sympathetic to our system of morality, which has evolved along with our particular biology (which AIs would not share). Hyper-intelligent software may not necessarily decide to support the continued existence of mankind, and would be extremely difficult to stop. This topic has also recently begun to be discussed in academic publications as a real source of risks to civilization, humans, and planet Earth.

Physicist Stephen Hawking, Microsoft founder Bill Gates and SpaceX founder Elon Musk have expressed concerns about the possibility that AI could evolve to the point that humans could not control it, with Hawking theorizing that this could "spell the end of the human race".[185]

One proposal to deal with this is to ensure that the first generally intelligent AI is 'Friendly AI', and will then be able to control subsequently developed AIs. Some question whether this kind of check could really remain in place.

Leading AI researcher Rodney Brooks writes, “I think it is a mistake to be worrying about us developing malevolent AI anytime in the next few hundred years. I think the worry stems from a fundamental error in not distinguishing the difference between the very real recent advances in a particular aspect of AI, and the enormity and complexity of building sentient volitional intelligence.”[186]

Devaluation of humanity

Main article: Computer Power and Human Reason

Joseph Weizenbaum wrote that AI applications can not, by definition, successfully simulate genuine human empathy and that the use of AI technology in fields such as customer service or psychotherapy[187] was deeply misguided. Weizenbaum was also bothered that AI researchers (and some philosophers) were willing to view the human mind as nothing more than a computer program (a position now known as computationalism). To Weizenbaum these points suggest that AI research devalues human life.[188]

Decrease in demand for human labor

Martin Ford, author of *The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future*,[189] and others argue that specialized artificial intelligence applications, robotics and other forms of automation will ultimately result in significant unemployment as machines begin to match and exceed the capability of workers to perform most routine and repetitive jobs. Ford predicts that many knowledge-based occupations—and in particular entry level jobs—will be increasingly susceptible to automation via expert systems, machine learning[190] and other AI-enhanced applications. AI-based applications may also be used to amplify the capabilities of low-wage offshore workers, making it more feasible to outsource knowledge work.[191]

2.4.3 Machine consciousness, sentience and mind

Main article: Artificial consciousness

If an AI system replicates all key aspects of human intelligence, will that system also be sentient – will it have a mind which has conscious experiences? This question is closely related to the philosophical problem as to the nature of human consciousness, generally referred to as the hard problem of consciousness.

Consciousness

Main articles: Hard problem of consciousness and Theory of mind

There are no objective criteria for knowing whether an intelligent agent is sentient – that it has conscious experiences. We assume that other people do because we do and they tell us that they do, but this is only a subjective determination. The lack of any hard criteria is known as the “hard problem” in the theory of consciousness. The problem applies not only to other people but to the higher animals and, by extension, to AI agents.

Computationalism

Main articles: Computationalism and Functionalism (philosophy of mind)

Are human intelligence, consciousness and mind products of information processing? Is the brain essentially a computer?

Computationalism is the idea that “the human mind or the human brain (or both) is an information processing system and that thinking is a form of computing”. AI, or implementing machines with human intelligence was founded on the claim that “a central property of humans, intelligence can be so precisely described that a machine can be made to simulate it”. A program can then be derived from this human human computer and implemented into an artificial one to, create efficient artificial intelligence. This program would act upon a set of outputs that result from set inputs of the internal memory of the computer, that is, the machine can only act with what it has implemented in it to start with. A long term goal for AI researchers is to provide machines with a deep understanding of the many abilities of a human being to replicate a general intelligence or STRONG AI, defined as a machine surpassing human abilities to perform the skills implanted in it, a scary thought to many, who fear losing control of such a powerful machine. Obstacles for researchers are mainly time contstraints. That is, AI scientists cannot establish much of a database for commonsense knowledge because it must be ontologically crafted into the machine which takes up a tremendous amount of time. To combat this, AI research looks to have the machine able to understand enough concepts in order to add to its own ontology, but how can it do this when machine ethics is primarily concerned with behavior of machines towards humans or other machines, limiting the extent of developing AI. In order to function like a common human AI must also display, “the ability to solve subsymbolic commonsense knowledge tasks such as how artists can tell statues are fake or how chess masters don’t move certain spots to avoid exposure,” but by developing machines who can do it all AI research is faced with the difficulty of potentially putting a lot of people out of work, while on the economy side of things businesses would boom from efficiency, thus forcing AI into a bottleneck trying to developing self improving machines.

Strong AI hypothesis

Main article: Chinese room

Searle’s strong AI hypothesis states that “The appropriately programmed computer with the right inputs and outputs would thereby have a mind in exactly the same sense human beings have minds.”[192] John Searle counters this assertion with his Chinese room argument, which asks us to look *inside* the computer and try to find where the “mind” might be.[193]

Robot rights

Main article: Robot rights

Mary Shelley's *Frankenstein* considers a key issue in the ethics of artificial intelligence: if a machine can be created that has intelligence, could it also *feel*? If it can feel, does it have the same rights as a human? The idea also appears in modern science fiction, such as the film *A.I.: Artificial Intelligence*, in which humanoid machines have the ability to feel emotions. This issue, now known as "robot rights", is currently being considered by, for example, California’s Institute for the Future, although many critics believe that the discussion is premature.[194] The subject is profoundly discussed in the 2010 documentary film *Plug & Pray*.[195]

2.4.4 Superintelligence

Main article: Superintelligence

Are there limits to how intelligent machines – or human-machine hybrids – can be? A superintelligence, hyper-intelligence, or superhuman intelligence is a hypothetical agent that would possess intelligence far surpassing that of the brightest and most gifted human mind. ”Superintelligence” may also refer to the form or degree of intelligence possessed by such an agent.

Technological singularity

Main articles: Technological singularity and Moore’s law

If research into Strong AI produced sufficiently intelligent software, it might be able to reprogram and improve itself. The improved software would be even better at improving itself, leading to recursive self-improvement.[196] The new intelligence could thus increase exponentially and dramatically surpass humans. Science fiction writer Vernor Vinge named this scenario "singularity".[197] Technological singularity is when accelerating progress in technologies will cause a runaway effect wherein artificial intelligence will exceed human intellectual capacity and control, thus radically changing or even ending civilization. Because the capabilities of such an intelligence may be impossible to comprehend, the technological singularity is an occurrence beyond which events are unpredictable or even unfathomable.[197]

Ray Kurzweil has used Moore’s law (which describes the relentless exponential improvement in digital technology) to calculate that desktop computers will have the same processing power as human brains by the year 2029, and predicts that the singularity will occur in 2045.[197]

Transhumanism

Main article: Transhumanism

Robot designer Hans Moravec, cyberneticist Kevin Warwick and inventor Ray Kurzweil have predicted that humans and machines will merge in the future into cyborgs that are more capable and powerful than either.[198] This idea, called transhumanism, which has roots in Aldous Huxley and Robert Ettinger, has been illustrated in fiction as well, for example in the manga *Ghost in the Shell* and the science-fiction series *Dune*.

In the 1980s artist Hajime Sorayama's Sexy Robots series were painted and published in Japan depicting the actual organic human form with lifelike muscular metallic skins and later “the Gynoids” book followed that was used by or influenced movie makers including George Lucas and other creatives. Sorayama never considered these organic robots to be real part of nature but always unnatural product of the human mind, a fantasy existing in the mind even when realized in actual form.

Edward Fredkin argues that “artificial intelligence is the next stage in evolution”, an idea first proposed by Samuel Butler's "Darwin among the Machines" (1863), and expanded upon by George Dyson in his book of the same name in 1998.[199]

2.5 In fiction

Main article: Artificial intelligence in fiction

The implications of artificial intelligence have been a persistent theme in science fiction. Early stories typically revolved around intelligent robots. The word “robot” itself was coined by Karel Čapek in his 1921 play *R.U.R.*, the title standing for "Rossum’s Universal Robots". Later, the SF writer Isaac Asimov developed the three laws of robotics which he subsequently explored in a long series of robot stories. These laws have since gained some traction in genuine AI research.

Other influential fictional intelligences include HAL, the computer in charge of the spaceship in *2001: A Space Odyssey*, released as both a film and a book in 1968 and written by Arthur C. Clarke.

Since then, AI has become firmly rooted in popular culture.

2.6 See also

Main article: Outline of artificial intelligence

- AI takeover
- *Artificial Intelligence* (journal)
- Artificial intelligence (video games)
- Artificial stupidity
- Nick Bostrom
- Computer Go
- Effective altruism
- Existential risk
- Existential risk of artificial general intelligence
- Future of Humanity Institute
- Human Cognome Project
- List of artificial intelligence projects

- List of artificial intelligence researchers
- List of emerging technologies
- List of important artificial intelligence publications
- List of machine learning algorithms
- List of scientific journals
- Machine ethics
- Machine learning
- Never-Ending Language Learning
- *Our Final Invention*
- Outline of artificial intelligence
- Outline of human intelligence
- Philosophy of mind
- Simulated reality
- Superintelligence

2.7 Notes

[1] Definition of AI as the study of intelligent agents:

- Poole, Mackworth & Goebel 1998, p. 1, which provides the version that is used in this article. Note that they use the term "computational intelligence" as a synonym for artificial intelligence.
- Russell & Norvig (2003) (who prefer the term "rational agent") and write "The whole-agent view is now widely accepted in the field" (Russell & Norvig 2003, p. 55).
- Nilsson 1998
- Legg & Hutter 2007.

[2] The intelligent agent paradigm:

- Russell & Norvig 2003, pp. 27, 32–58, 968–972
- Poole, Mackworth & Goebel 1998, pp. 7–21
- Luger & Stubblefield 2004, pp. 235–240
- Hutter 2005, pp. 125–126

The definition used in this article, in terms of goals, actions, perception and environment, is due to Russell & Norvig (2003). Other definitions also include knowledge and learning as additional criteria.

[3] Although there is some controversy on this point (see Crevier (1993, p. 50)), McCarthy states unequivocally "I came up with the term" in a c|net interview. (Skillings 2006) McCarthy first used the term in the proposal for the Dartmouth conference, which appeared in 1955. (McCarthy et al. 1955)

[4] McCarthy's definition of AI:

- McCarthy 2007

[5] Pamela McCorduck (2004, pp. 424) writes of "the rough shattering of AI in subfields—vision, natural language, decision theory, genetic algorithms, robotics ... and these with own sub-subfield—that would hardly have anything to say to each other."

[6] This list of intelligent traits is based on the topics covered by the major AI textbooks, including:

- Russell & Norvig 2003
- Luger & Stubblefield 2004
- Poole, Mackworth & Goebel 1998
- Nilsson 1998

[7] General intelligence (strong AI) is discussed in popular introductions to AI:

- Kurzweil 1999 and Kurzweil 2005

[8] See the Dartmouth proposal, under Philosophy, below.

[9] This is a central idea of Pamela McCorduck's *Machines Who Think*. She writes: "I like to think of artificial intelligence as the scientific apotheosis of a venerable cultural tradition." (McCorduck 2004, p. 34) "Artificial intelligence in one form or another is an idea that has pervaded Western intellectual history, a dream in urgent need of being realized." (McCorduck 2004, p. xviii) "Our history is full of attempts—nutty, eerie, comical, earnest, legendary and real—to make artificial intelligences, to reproduce what is the essential us—bypassing the ordinary means. Back and forth between myth and reality, our imaginations supplying what our workshops couldn't, we have engaged for a long time in this odd form of self-reproduction." (McCorduck 2004, p. 3) She traces the desire back to its Hellenistic roots and calls it the urge to "forge the Gods." (McCorduck 2004, pp. 340–400)

[10] The optimism referred to includes the predictions of early AI researchers (see optimism in the history of AI) as well as the ideas of modern transhumanists such as Ray Kurzweil.

[11] The "setbacks" referred to include the ALPAC report of 1966, the abandonment of perceptrons in 1970, the Lighthill Report of 1973 and the collapse of the Lisp machine market in 1987.

[12] AI applications widely used behind the scenes:

- Russell & Norvig 2003, p. 28
- Kurzweil 2005, p. 265
- NRC 1999, pp. 216–222

[13] AI in myth:

- McCorduck 2004, pp. 4–5
- Russell & Norvig 2003, p. 939

[14] Cult images as artificial intelligence:

- Crevier (1993, p. 1) (statue of Amun)
- McCorduck (2004, pp. 6–9)

These were the first machines to be believed to have true intelligence and consciousness. Hermes Trismegistus expressed the common belief that with these statues, craftsman had reproduced "the true nature of the gods", their *sensus* and *spiritus*. McCorduck makes the connection between sacred automatons and Mosaic law (developed around the same time), which expressly forbids the worship of robots (McCorduck 2004, pp. 6–9)

[15] Humanoid automata:
Yan Shi:

- Needham 1986, p. 53

Hero of Alexandria:

- McCorduck 2004, p. 6

Al-Jazari:

- "A Thirteenth Century Programmable Robot". Shef.ac.uk. Retrieved 25 April 2009.

Wolfgang von Kempelen:

- McCorduck 2004, p. 17

[16] Artificial beings:
Jābir ibn Hayyān's Takwin:

- O'Connor 1994

Judah Loew's Golem:

- McCorduck 2004, pp. 15–16
- Buchanan 2005, p. 50

Paracelsus' Homunculus:

- McCorduck 2004, pp. 13–14

[17] AI in early science fiction.

- McCorduck 2004, pp. 17–25

[18] This insight, that digital computers can simulate any process of formal reasoning, is known as the Church–Turing thesis.

[19] Formal reasoning:

- Berlinski, David (2000). *The Advent of the Algorithm*. Harcourt Books. ISBN 0-15-601391-6. OCLC 46890682.

[20] AI's immediate precursors:

- McCorduck 2004, pp. 51–107
- Crevier 1993, pp. 27–32
- Russell & Norvig 2003, pp. 15, 940
- Moravec 1988, p. 3

See also Cybernetics and early neural networks (in History of artificial intelligence). Among the researchers who laid the foundations of AI were Alan Turing, John von Neumann, Norbert Wiener, Claude Shannon, Warren McCullough, Walter Pitts and Donald Hebb.

[21] Dartmouth conference:

- McCorduck 2004, pp. 111–136
- Crevier 1993, pp. 47–49, who writes "the conference is generally recognized as the official birthdate of the new science."
- Russell & Norvig 2003, p. 17, who call the conference "the birth of artificial intelligence."
- NRC 1999, pp. 200–201

[22] Hegemony of the Dartmouth conference attendees:

- Russell & Norvig 2003, p. 17, who write "for the next 20 years the field would be dominated by these people and their students."
- McCorduck 2004, pp. 129–130

[23] Russell and Norvig write "it was astonishing whenever a computer did anything kind of smartish." Russell & Norvig 2003, p. 18

[24] "Golden years" of AI (successful symbolic reasoning programs 1956–1973):

- McCorduck 2004, pp. 243–252
- Crevier 1993, pp. 52–107
- Moravec 1988, p. 9
- Russell & Norvig 2003, pp. 18–21

The programs described are Arthur Samuel's checkers program for the IBM 701, Daniel Bobrow's STUDENT, Newell and Simon's Logic Theorist and Terry Winograd's SHRDLU.

[25] DARPA pours money into undirected pure research into AI during the 1960s:

- McCorduck 2004, pp. 131
- Crevier 1993, pp. 51, 64–65
- NRC 1999, pp. 204–205

[26] AI in England:

- Howe 1994

[27] Optimism of early AI:

- Herbert Simon quote: Simon 1965, p. 96 quoted in Crevier 1993, p. 109.
- Marvin Minsky quote: Minsky 1967, p. 2 quoted in Crevier 1993, p. 109.

[28] See The problems (in History of artificial intelligence)

[29] Lighthill 1973.

[30] First AI Winter, Mansfield Amendment, Lighthill report

- Crevier 1993, pp. 115–117
- Russell & Norvig 2003, p. 22
- NRC 1999, pp. 212–213
- Howe 1994

[31] Expert systems:

- ACM 1998, I.2.1

- Russell & Norvig 2003, pp. 22–24
- Luger & Stubblefield 2004, pp. 227–331
- Nilsson 1998, chpt. 17.4
- McCorduck 2004, pp. 327–335, 434–435
- Crevier 1993, pp. 145–62, 197–203

[32] Boom of the 1980s: rise of expert systems, Fifth Generation Project, Alvey, MCC, SCI:

- McCorduck 2004, pp. 426–441
- Crevier 1993, pp. 161–162,197–203, 211, 240
- Russell & Norvig 2003, p. 24
- NRC 1999, pp. 210–211

[33] Second AI winter:

- McCorduck 2004, pp. 430–435
- Crevier 1993, pp. 209–210
- NRC 1999, pp. 214–216

[34] Formal methods are now preferred ("Victory of the neats"):

- Russell & Norvig 2003, pp. 25–26
- McCorduck 2004, pp. 486–487

[35] McCorduck 2004, pp. 480–483

[36] Markoff 2011.

[37] Administrator. "Kinect's AI breakthrough explained". *i-programmer.info*.

[38] http://readwrite.com/2013/01/15/virtual-personal-assistants-the-future-of-your-smartphone-infographic

[39] Lemmons, Phil (April 1985). "Artificial Intelligence". *BYTE*. p. 125. Retrieved 14 February 2015.

[40] Problem solving, puzzle solving, game playing and deduction:

- Russell & Norvig 2003, chpt. 3–9,
- Poole, Mackworth & Goebel 1998, chpt. 2,3,7,9,
- Luger & Stubblefield 2004, chpt. 3,4,6,8,
- Nilsson 1998, chpt. 7–12

[41] Uncertain reasoning:

- Russell & Norvig 2003, pp. 452–644,
- Poole, Mackworth & Goebel 1998, pp. 345–395,
- Luger & Stubblefield 2004, pp. 333–381,
- Nilsson 1998, chpt. 19

[42] Intractability and efficiency and the combinatorial explosion:

- Russell & Norvig 2003, pp. 9, 21–22

[43] Psychological evidence of sub-symbolic reasoning:

- Wason & Shapiro (1966) showed that people do poorly on completely abstract problems, but if the problem is restated to allow the use of intuitive social intelligence, performance dramatically improves. (See Wason selection task)
- Kahneman, Slovic & Tversky (1982) have shown that people are terrible at elementary problems that involve uncertain reasoning. (See list of cognitive biases for several examples).
- Lakoff & Núñez (2000) have controversially argued that even our skills at mathematics depend on knowledge and skills that come from "the body", i.e. sensorimotor and perceptual skills. (See Where Mathematics Comes From)

[44] Knowledge representation:

- ACM 1998, I.2.4,
- Russell & Norvig 2003, pp. 320–363,
- Poole, Mackworth & Goebel 1998, pp. 23–46, 69–81, 169–196, 235–277, 281–298, 319–345,
- Luger & Stubblefield 2004, pp. 227–243,
- Nilsson 1998, chpt. 18

[45] Knowledge engineering:

- Russell & Norvig 2003, pp. 260–266,
- Poole, Mackworth & Goebel 1998, pp. 199–233,
- Nilsson 1998, chpt. ~17.1–17.4

[46] Representing categories and relations: Semantic networks, description logics, inheritance (including frames and scripts):

- Russell & Norvig 2003, pp. 349–354,
- Poole, Mackworth & Goebel 1998, pp. 174–177,
- Luger & Stubblefield 2004, pp. 248–258,
- Nilsson 1998, chpt. 18.3

[47] Representing events and time:Situation calculus, event calculus, fluent calculus (including solving the frame problem):

- Russell & Norvig 2003, pp. 328–341,
- Poole, Mackworth & Goebel 1998, pp. 281–298,
- Nilsson 1998, chpt. 18.2

[48] Causal calculus:

- Poole, Mackworth & Goebel 1998, pp. 335–337

[49] Representing knowledge about knowledge: Belief calculus, modal logics:

- Russell & Norvig 2003, pp. 341–344,
- Poole, Mackworth & Goebel 1998, pp. 275–277

[50] Ontology:

- Russell & Norvig 2003, pp. 320–328

[51] Qualification problem:

- McCarthy & Hayes 1969
- Russell & Norvig 2003

While McCarthy was primarily concerned with issues in the logical representation of actions, Russell & Norvig 2003 apply the term to the more general issue of default reasoning in the vast network of assumptions underlying all our commonsense knowledge.

[52] Default reasoning and default logic, non-monotonic logics, circumscription, closed world assumption, abduction (Poole *et al.* places abduction under "default reasoning". Luger *et al.* places this under "uncertain reasoning"):

- Russell & Norvig 2003, pp. 354–360,
- Poole, Mackworth & Goebel 1998, pp. 248–256, 323–335,
- Luger & Stubblefield 2004, pp. 335–363,
- Nilsson 1998, ~18.3.3

[53] Breadth of commonsense knowledge:

- Russell & Norvig 2003, p. 21,
- Crevier 1993, pp. 113–114,
- Moravec 1988, p. 13,
- Lenat & Guha 1989 (Introduction)

[54] Dreyfus & Dreyfus 1986

[55] Gladwell 2005

[56] Expert knowledge as embodied intuition:

- Dreyfus & Dreyfus 1986 (Hubert Dreyfus is a philosopher and critic of AI who was among the first to argue that most useful human knowledge was encoded sub-symbolically. See Dreyfus' critique of AI)
- Gladwell 2005 (Gladwell's *Blink* is a popular introduction to sub-symbolic reasoning and knowledge.)
- Hawkins & Blakeslee 2005 (Hawkins argues that sub-symbolic knowledge should be the primary focus of AI research.)

[57] Planning:

- ACM 1998, ~I.2.8,
- Russell & Norvig 2003, pp. 375–459,
- Poole, Mackworth & Goebel 1998, pp. 281–316,
- Luger & Stubblefield 2004, pp. 314–329,
- Nilsson 1998, chpt. 10.1–2, 22

[58] Information value theory:

- Russell & Norvig 2003, pp. 600–604

[59] Classical planning:

- Russell & Norvig 2003, pp. 375–430,
- Poole, Mackworth & Goebel 1998, pp. 281–315,
- Luger & Stubblefield 2004, pp. 314–329,
- Nilsson 1998, chpt. 10.1–2, 22

[60] Planning and acting in non-deterministic domains: conditional planning, execution monitoring, replanning and continuous planning:

- Russell & Norvig 2003, pp. 430–449

[61] Multi-agent planning and emergent behavior:

- Russell & Norvig 2003, pp. 449–455

[62] This is a form of Tom Mitchell's widely quoted definition of machine learning: "A computer program is set to learn from an experience E with respect to some task T and some performance measure P if its performance on T as measured by P improves with experience E."

[63] Learning:

- ACM 1998, I.2.6,
- Russell & Norvig 2003, pp. 649–788,
- Poole, Mackworth & Goebel 1998, pp. 397–438,
- Luger & Stubblefield 2004, pp. 385–542,
- Nilsson 1998, chpt. 3.3, 10.3, 17.5, 20

[64] Alan Turing discussed the centrality of learning as early as 1950, in his classic paper "Computing Machinery and Intelligence".(Turing 1950) In 1956, at the original Dartmouth AI summer conference, Ray Solomonoff wrote a report on unsupervised probabilistic machine learning: "An Inductive Inference Machine".(Solomonoff 1956)

[65] Reinforcement learning:

- Russell & Norvig 2003, pp. 763–788
- Luger & Stubblefield 2004, pp. 442–449

[66] Computational learning theory:

- CITATION IN PROGRESS.

[67] Weng et al. 2001.

[68] Lungarella et al. 2003.

[69] Asada et al. 2009.

[70] Oudeyer 2010.

[71] Natural language processing:

- ACM 1998, I.2.7
- Russell & Norvig 2003, pp. 790–831
- Poole, Mackworth & Goebel 1998, pp. 91–104
- Luger & Stubblefield 2004, pp. 591–632

[72] "Versatile question answering systems: seeing in synthesis", Mittal et al., IJIIDS, 5(2), 119-142, 2011

[73] Applications of natural language processing, including information retrieval (i.e. text mining) and machine translation:

- Russell & Norvig 2003, pp. 840–857,
- Luger & Stubblefield 2004, pp. 623–630

[74] Machine perception:

- Russell & Norvig 2003, pp. 537–581, 863–898
- Nilsson 1998, ~chpt. 6

[75] Computer vision:

- ACM 1998, I.2.10
- Russell & Norvig 2003, pp. 863–898
- Nilsson 1998, chpt. 6

[76] Speech recognition:

- ACM 1998, ~I.2.7
- Russell & Norvig 2003, pp. 568–578

[77] Object recognition:

- Russell & Norvig 2003, pp. 885–892

[78] Robotics:

- ACM 1998, I.2.9,
- Russell & Norvig 2003, pp. 901–942,
- Poole, Mackworth & Goebel 1998, pp. 443–460

[79] Moving and configuration space:

- Russell & Norvig 2003, pp. 916–932

[80] Tecuci 2012.

[81] Robotic mapping (localization, etc):

- Russell & Norvig 2003, pp. 908–915

[82] Thro 1993.

[83] Edelson 1991.

[84] Tao & Tan 2005.

[85] James 1884.

[86] Picard 1995.

[87] Kleine-Cosack 2006: "The introduction of emotion to computer science was done by Pickard (sic) who created the field of affective computing."

[88] Diamond 2003: "Rosalind Picard, a genial MIT professor, is the field's godmother; her 1997 book, Affective Computing, triggered an explosion of interest in the emotional side of computers and their users."

[89] Emotion and affective computing:

- Minsky 2006

[90] Gerald Edelman, Igor Aleksander and others have argued that artificial consciousness is required for strong AI. (Aleksander 1995; Edelman 2007)

[91] Artificial brain arguments: AI requires a simulation of the operation of the human brain

- Russell & Norvig 2003, p. 957
- Crevier 1993, pp. 271 and 279

A few of the people who make some form of the argument:

- Moravec 1988
- Kurzweil 2005, p. 262
- Hawkins & Blakeslee 2005

The most extreme form of this argument (the brain replacement scenario) was put forward by Clark Glymour in the mid-1970s and was touched on by Zenon Pylyshyn and John Searle in 1980.

[92] AI complete: Shapiro 1992, p. 9

[93] Nils Nilsson writes: "Simply put, there is wide disagreement in the field about what AI is all about" (Nilsson 1983, p. 10).

[94] Biological intelligence vs. intelligence in general:

- Russell & Norvig 2003, pp. 2–3, who make the analogy with aeronautical engineering.
- McCorduck 2004, pp. 100–101, who writes that there are "two major branches of artificial intelligence: one aimed at producing intelligent behavior regardless of how it was accomplioshed, and the other aimed at modeling intelligent processes found in nature, particularly human ones."
- Kolata 1982, a paper in *Science*, which describes McCarthy's indifference to biological models. Kolata quotes McCarthy as writing: "This is AI, so we don't care if it's psychologically real". McCarthy recently reiterated his position at the AI@50 conference where he said "Artificial intelligence is not, by definition, simulation of human intelligence" (Maker 2006).

[95] Neats vs. scruffies:

- McCorduck 2004, pp. 421–424, 486–489
- Crevier 1993, pp. 168
- Nilsson 1983, pp. 10–11

[96] Symbolic vs. sub-symbolic AI:

- Nilsson (1998, p. 7), who uses the term "sub-symbolic".

[97] Haugeland 1985, p. 255.

[98] Law 1994.

[99] Bach 2008.

[100] Haugeland 1985, pp. 112–117

[101] The most dramatic case of sub-symbolic AI being pushed into the background was the devastating critique of perceptrons by Marvin Minsky and Seymour Papert in 1969. See History of AI, AI winter, or Frank Rosenblatt.

[102] Cognitive simulation, Newell and Simon, AI at CMU (then called Carnegie Tech):

- McCorduck 2004, pp. 139–179, 245–250, 322–323 (EPAM)
- Crevier 1993, pp. 145–149

[103] Soar (history):

- McCorduck 2004, pp. 450–451
- Crevier 1993, pp. 258–263

[104] McCarthy and AI research at SAIL and SRI International:

- McCorduck 2004, pp. 251–259
- Crevier 1993

[105] AI research at Edinburgh and in France, birth of Prolog:

- Crevier 1993, pp. 193–196
- Howe 1994

[106] AI at MIT under Marvin Minsky in the 1960s :

- McCorduck 2004, pp. 259–305
- Crevier 1993, pp. 83–102, 163–176
- Russell & Norvig 2003, p. 19

[107] Cyc:

- McCorduck 2004, p. 489, who calls it "a determinedly scruffy enterprise"
- Crevier 1993, pp. 239–243
- Russell & Norvig 2003, p. 363–365
- Lenat & Guha 1989

[108] Knowledge revolution:

- McCorduck 2004, pp. 266–276, 298–300, 314, 421
- Russell & Norvig 2003, pp. 22–23

[109] Embodied approaches to AI:

- McCorduck 2004, pp. 454–462
- Brooks 1990
- Moravec 1988

[110] Revival of connectionism:

- Crevier 1993, pp. 214–215
- Russell & Norvig 2003, p. 25

[111] Computational intelligence

- IEEE Computational Intelligence Society

[112] Hutter 2012.

[113] Langley 2011.

[114] Katz 2012.

[115] Norvig 2012.

[116] Agent architectures, hybrid intelligent systems:

- Russell & Norvig (2003, pp. 27, 932, 970–972)
- Nilsson (1998, chpt. 25)

[117] Hierarchical control system:

- Albus 2002

[118] Subsumption architecture:

- CITATION IN PROGRESS.

[119] Search algorithms:

- Russell & Norvig 2003, pp. 59–189
- Poole, Mackworth & Goebel 1998, pp. 113–163
- Luger & Stubblefield 2004, pp. 79–164, 193–219
- Nilsson 1998, chpt. 7–12

[120] Forward chaining, backward chaining, Horn clauses, and logical deduction as search:

- Russell & Norvig 2003, pp. 217–225, 280–294
- Poole, Mackworth & Goebel 1998, pp. ~46–52
- Luger & Stubblefield 2004, pp. 62–73
- Nilsson 1998, chpt. 4.2, 7.2

[121] State space search and planning:

- Russell & Norvig 2003, pp. 382–387
- Poole, Mackworth & Goebel 1998, pp. 298–305
- Nilsson 1998, chpt. 10.1–2

[122] Uninformed searches (breadth first search, depth first search and general state space search):

- Russell & Norvig 2003, pp. 59–93
- Poole, Mackworth & Goebel 1998, pp. 113–132
- Luger & Stubblefield 2004, pp. 79–121
- Nilsson 1998, chpt. 8

[123] Heuristic or informed searches (e.g., greedy best first and A*):

- Russell & Norvig 2003, pp. 94–109,
- Poole, Mackworth & Goebel 1998, pp. pp. 132–147,
- Luger & Stubblefield 2004, pp. 133–150,
- Nilsson 1998, chpt. 9

[124] Optimization searches:

- Russell & Norvig 2003, pp. 110–116,120–129
- Poole, Mackworth & Goebel 1998, pp. 56–163
- Luger & Stubblefield 2004, pp. 127–133

[125] Artificial life and society based learning:

- Luger & Stubblefield 2004, pp. 530–541

[126] Genetic programming and genetic algorithms:

- Luger & Stubblefield 2004, pp. 509–530,
- Nilsson 1998, chpt. 4.2,
- Holland 1975,
- Koza 1992,
- Poli, Langdon & McPhee 2008.

[127] Logic:

- ACM 1998, ~I.2.3,
- Russell & Norvig 2003, pp. 194–310,

- Luger & Stubblefield 2004, pp. 35–77,
- Nilsson 1998, chpt. 13–16

[128] Satplan:

- Russell & Norvig 2003, pp. 402–407,
- Poole, Mackworth & Goebel 1998, pp. 300–301,
- Nilsson 1998, chpt. 21

[129] Explanation based learning, relevance based learning, inductive logic programming, case based reasoning:

- Russell & Norvig 2003, pp. 678–710,
- Poole, Mackworth & Goebel 1998, pp. 414–416,
- Luger & Stubblefield 2004, pp. ~422–442,
- Nilsson 1998, chpt. 10.3, 17.5

[130] Propositional logic:

- Russell & Norvig 2003, pp. 204–233,
- Luger & Stubblefield 2004, pp. 45–50
- Nilsson 1998, chpt. 13

[131] First-order logic and features such as equality:

- ACM 1998, ~I.2.4,
- Russell & Norvig 2003, pp. 240–310,
- Poole, Mackworth & Goebel 1998, pp. 268–275,
- Luger & Stubblefield 2004, pp. 50–62,
- Nilsson 1998, chpt. 15

[132] Fuzzy logic:

- Russell & Norvig 2003, pp. 526–527

[133] Subjective logic:

- CITATION IN PROGRESS.

[134] Stochastic methods for uncertain reasoning:

- ACM 1998, ~I.2.3,
- Russell & Norvig 2003, pp. 462–644,
- Poole, Mackworth & Goebel 1998, pp. 345–395,
- Luger & Stubblefield 2004, pp. 165–191, 333–381,
- Nilsson 1998, chpt. 19

[135] Bayesian networks:

- Russell & Norvig 2003, pp. 492–523,
- Poole, Mackworth & Goebel 1998, pp. 361–381,
- Luger & Stubblefield 2004, pp. ~182–190, ~363–379,
- Nilsson 1998, chpt. 19.3–4

[136] Bayesian inference algorithm:

- Russell & Norvig 2003, pp. 504–519,
- Poole, Mackworth & Goebel 1998, pp. 361–381,
- Luger & Stubblefield 2004, pp. ~363–379,
- Nilsson 1998, chpt. 19.4 & 7

[137] Bayesian learning and the expectation-maximization algorithm:

- Russell & Norvig 2003, pp. 712–724,
- Poole, Mackworth & Goebel 1998, pp. 424–433,
- Nilsson 1998, chpt. 20

[138] Bayesian decision theory and Bayesian decision networks:

- Russell & Norvig 2003, pp. 597–600

[139] Stochastic temporal models:

- Russell & Norvig 2003, pp. 537–581

Dynamic Bayesian networks:

- Russell & Norvig 2003, pp. 551–557

Hidden Markov model:

- (Russell & Norvig 2003, pp. 549–551)

Kalman filters:

- Russell & Norvig 2003, pp. 551–557

[140] decision theory and decision analysis:

- Russell & Norvig 2003, pp. 584–597,
- Poole, Mackworth & Goebel 1998, pp. 381–394

[141] Markov decision processes and dynamic decision networks:

- Russell & Norvig 2003, pp. 613–631

[142] Game theory and mechanism design:

- Russell & Norvig 2003, pp. 631–643

[143] Statistical learning methods and classifiers:

- Russell & Norvig 2003, pp. 712–754,
- Luger & Stubblefield 2004, pp. 453–541

[144] Neural networks and connectionism:

- Russell & Norvig 2003, pp. 736–748,
- Poole, Mackworth & Goebel 1998, pp. 408–414,
- Luger & Stubblefield 2004, pp. 453–505,
- Nilsson 1998, chpt. 3

[145] kernel methods such as the support vector machine:

- Russell & Norvig 2003, pp. 749–752

[146] K-nearest neighbor algorithm:

- Russell & Norvig 2003, pp. 733–736

[147] Gaussian mixture model:

- Russell & Norvig 2003, pp. 725–727

[148] Naive Bayes classifier:

- Russell & Norvig 2003, pp. 718

[149] Decision tree:

- Russell & Norvig 2003, pp. 653–664,
- Poole, Mackworth & Goebel 1998, pp. 403–408,
- Luger & Stubblefield 2004, pp. 408–417

[150] Classifier performance:

- van der Walt & Bernard 2006

[151] Backpropagation:

- Russell & Norvig 2003, pp. 744–748,
- Luger & Stubblefield 2004, pp. 467–474,
- Nilsson 1998, chpt. 3.3

[152] Feedforward neural networks, perceptrons and radial basis networks:

- Russell & Norvig 2003, pp. 739–748, 758
- Luger & Stubblefield 2004, pp. 458–467

[153] Recurrent neural networks, Hopfield nets:

- Russell & Norvig 2003, p. 758
- Luger & Stubblefield 2004, pp. 474–505

[154] Competitive learning, Hebbian coincidence learning, Hopfield networks and attractor networks:

- Luger & Stubblefield 2004, pp. 474–505

[155] Hierarchical temporal memory:

- Hawkins & Blakeslee 2005

[156] Hinton 2007.

[157] Control theory:

- ACM 1998, ~I.2.8,
- Russell & Norvig 2003, pp. 926–932

[158] Lisp:

- Luger & Stubblefield 2004, pp. 723–821
- Crevier 1993, pp. 59–62,
- Russell & Norvig 2003, p. 18

[159] Prolog:

- Poole, Mackworth & Goebel 1998, pp. 477–491,
- Luger & Stubblefield 2004, pp. 641–676, 575–581

[160] The Turing test:
Turing's original publication:

- Turing 1950

Historical influence and philosophical implications:

- Haugeland 1985, pp. 6–9
- Crevier 1993, p. 24
- McCorduck 2004, pp. 70–71
- Russell & Norvig 2003, pp. 2–3 and 948

[161] Subject matter expert Turing test:

- CITATION IN PROGRESS.

[162] Rajani 2011.

[163] Game AI:

- CITATION IN PROGRESS.

[164] Mathematical definitions of intelligence:

- Hernandez-Orallo 2000
- Dowe & Hajek 1997
- Hernandez-Orallo & Dowe 2010

[165] O'Brien & Marakas 2011.

[166] *CNN* 2006.

[167] Intrusion detection:

- Kumar & Kumar 2012

[168] Brooks 1991.

[169] "Hacking Roomba". *hackingroomba.com*.

[170] Philosophy of AI. All of these positions in this section are mentioned in standard discussions of the subject, such as:

- Russell & Norvig 2003, pp. 947–960
- Fearn 2007, pp. 38–55

[171] Dartmouth proposal:

- McCarthy et al. 1955 (the original proposal)
- Crevier 1993, p. 49 (historical significance)

[172] The physical symbol systems hypothesis:

- Newell & Simon 1976, p. 116
- McCorduck 2004, p. 153
- Russell & Norvig 2003, p. 18

[173] Dreyfus criticized the necessary condition of the physical symbol system hypothesis, which he called the "psychological assumption": "The mind can be viewed as a device operating on bits of information according to formal rules". (Dreyfus 1992, p. 156)

[174] Dreyfus' critique of artificial intelligence:

- Dreyfus 1972, Dreyfus & Dreyfus 1986
- Crevier 1993, pp. 120–132
- McCorduck 2004, pp. 211–239
- Russell & Norvig 2003, pp. 950–952,

[175] Gödel 1951: in this lecture, Kurt Gödel uses the incompleteness theorem to arrive at the following disjunction: (a) the human mind is not a consistent finite machine, or (b) there exist Diophantine equations for which it cannot decide whether solutions exist. Gödel finds (b) implausible, and thus seems to have believed the human mind was not equivalent to a finite machine, i.e., its power exceeded that of any finite machine. He recognized that this was only a conjecture, since one could never disprove (b). Yet he considered the disjunctive conclusion to be a "certain fact".

[176] The Mathematical Objection:

- Russell & Norvig 2003, p. 949
- McCorduck 2004, pp. 448–449

Making the Mathematical Objection:

- Lucas 1961
- Penrose 1989

Refuting Mathematical Objection:

- Turing 1950 under "(2) The Mathematical Objection"
- Hofstadter 1979

Background:

- Gödel 1931, Church 1936, Kleene 1935, Turing 1937

[177] Beyond the Doubting of a Shadow, A Reply to Commentaries on Shadows of the Mind, Roger Penrose 1996 The links to the original articles he responds to there are easily found in the Wayback machine: Can Physics Provide a Theory of Consciousness? Barnard J. Bars, Penrose's Gödelian Argument etc.

[178] Wendell Wallach (2010). *Moral Machines*, Oxford University Press.

[179] Wallach, pp 37–54.

[180] Wallach, pp 55–73.

[181] Wallach, Introduction chapter.

[182] Michael Anderson and Susan Leigh Anderson (2011), Machine Ethics, Cambridge University Press.

[183] "Machine Ethics". *aaai.org*.

[184] Rubin, Charles (Spring 2003). "Artificial Intelligence and Human Nature". *The New Atlantis* **1**: 88–100.

[185] Rawlinson, Kevin. "Microsoft's Bill Gates insists AI is a threat". BBC News. Retrieved 30 January 2015.

[186] Brooks, Rodney (10 November 2014). "artificial intelligence is a tool, not a threat".

[187] In the early 1970s, Kenneth Colby presented a version of Weizenbaum's ELIZA known as DOCTOR which he promoted as a serious therapeutic tool. (Crevier 1993, pp. 132–144)

[188] Joseph Weizenbaum's critique of AI:

- Weizenbaum 1976
- Crevier 1993, pp. 132–144
- McCorduck 2004, pp. 356–373
- Russell & Norvig 2003, p. 961

Weizenbaum (the AI researcher who developed the first chatterbot program, ELIZA) argued in 1976 that the misuse of artificial intelligence has the potential to devalue human life.

[189] Ford, Martin R. (2009), *The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future*, Acculant Publishing, ISBN 978-1448659814. *(e-book available free online.)*

[190] "Machine Learning: A job killer?". *econfuture - Robots, AI and Unemployment - Future Economics and Technology*.

[191] AI could decrease the demand for human labor:

- Russell & Norvig 2003, pp. 960–961
- Ford, Martin (2009). *The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future*. Acculant Publishing. ISBN 978-1-4486-5981-4.

[192] This version is from Searle (1999), and is also quoted in Dennett 1991, p. 435. Searle's original formulation was "The appropriately programmed computer really is a mind, in the sense that computers given the right programs can be literally said to understand and have other cognitive states." (Searle 1980, p. 1). Strong AI is defined similarly by Russell & Norvig (2003, p. 947): "The assertion that machines could possibly act intelligently (or, perhaps better, act as if they were intelligent) is called the 'weak AI' hypothesis by philosophers, and the assertion that machines that do so are actually thinking (as opposed to simulating thinking) is called the 'strong AI' hypothesis."

[193] Searle's Chinese room argument:

- Searle 1980. Searle's original presentation of the thought experiment.
- Searle 1999.

Discussion:

- Russell & Norvig 2003, pp. 958–960
- McCorduck 2004, pp. 443–445
- Crevier 1993, pp. 269–271

[194] Robot rights:

- Russell & Norvig 2003, p. 964
- *BBC News* 2006

Prematurity of:

- Henderson 2007

In fiction:

- McCorduck (2004, p. 190-25) discusses *Frankenstein* and identifies the key ethical issues as scientific hubris and the suffering of the monster, i.e. robot rights.

[195] maschafilm. "Content: Plug & Pray Film - Artificial Intelligence - Robots -". *plugandpray-film.de*.

[196] Omohundro, Steve (2008). *The Nature of Self-Improving Artificial Intelligence*. presented and distributed at the 2007 Singularity Summit, San Francisco, CA.

[197] Technological singularity:

- Vinge 1993
- Kurzweil 2005
- Russell & Norvig 2003, p. 963

[198] Transhumanism:

- Moravec 1988
- Kurzweil 2005
- Russell & Norvig 2003, p. 963

[199] AI as evolution:

- Edward Fredkin is quoted in McCorduck (2004, p. 401).
- Butler 1863
- Dyson 1998

2.8 References

2.8.1 AI textbooks

- Hutter, Marcus (2005). *Universal Artificial Intelligence*. Berlin: Springer. ISBN 978-3-540-22139-5.
- Luger, George; Stubblefield, William (2004). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving* (5th ed.). Benjamin/Cummings. ISBN 0-8053-4780-1.
- Neapolitan, Richard; Jiang, Xia (2012). *Contemporary Artificial Intelligence*. Chapman & Hall/CRC. ISBN 978-1-4398-4469-4.
- Nilsson, Nils (1998). *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann. ISBN 978-1-55860-467-4.
- Russell, Stuart J.; Norvig, Peter (2003), *Artificial Intelligence: A Modern Approach* (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2.
- Poole, David; Mackworth, Alan; Goebel, Randy (1998). *Computational Intelligence: A Logical Approach*. New York: Oxford University Press. ISBN 0-19-510270-3.
- Winston, Patrick Henry (1984). *Artificial Intelligence*. Reading, MA: Addison-Wesley. ISBN 0-201-08259-4.
- Rich, Elaine (1983). *Artificial Intelligence*. McGraw-Hill. ISBN 0-07-052261-8.

2.8.2 History of AI

- Crevier, Daniel (1993), *AI: The Tumultuous Search for Artificial Intelligence*, New York, NY: BasicBooks, ISBN 0-465-02997-3.
- McCorduck, Pamela (2004), *Machines Who Think* (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 1-56881-205-1.
- Nilsson, Nils (2010). *The Quest for Artificial Intelligence: A History of Ideas and Achievements*. New York: Cambridge University Press. ISBN 978-0-521-12293-1.

2.8.3 Other sources

- Asada, M.; Hosoda, K.; Kuniyoshi, Y.; Ishiguro, H.; Inui, T.; Yoshikawa, Y.; Ogino, M.; Yoshida, C. (2009). "Cognitive developmental robotics: a survey" (PDF). *IEEE Transactions on Autonomous Mental Development* **1** (1): 12–34. doi:10.1109/tamd.2009.2021702.
- "ACM Computing Classification System: Artificial intelligence". ACM. 1998. Retrieved 30 August 2007.
- Albus, J. S. (2002). "4-D/RCS: A Reference Model Architecture for Intelligent Unmanned Ground Vehicles" (PDF). In Gerhart, G.; Gunderson, R.; Shoemaker, C. *Proceedings of the SPIE AeroSense Session on Unmanned Ground Vehicle Technology* **3693**. pp. 11–20. Archived from the original (PDF) on 25 July 2004.
- Aleksander, Igor (1995). *Artificial Neuroconsciousness: An Update*. IWANN. Archived from the original on 2 March 1997. BibTex Archived 2 March 1997 at the Wayback Machine.
- Bach, Joscha (2008). "Seven Principles of Synthetic Intelligence". In Wang, Pei; Goertzel, Ben; Franklin, Stan. *Artificial General Intelligence, 2008: Proceedings of the First AGI Conference*. IOS Press. pp. 63–74. ISBN 978-1-58603-833-5.
- "Robots could demand legal rights". *BBC News*. 21 December 2006. Retrieved 3 February 2011.
- Brooks, Rodney (1990). "Elephants Don't Play Chess" (PDF). *Robotics and Autonomous Systems* **6**: 3–15. doi:10.1016/S0921-8890(05)80025-9. Archived (PDF) from the original on 9 August 2007.
- Brooks, R. A. (1991). "How to build complete creatures rather than isolated cognitive simulators". In VanLehn, K. *Architectures for Intelligence*. Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 225–239.

- Buchanan, Bruce G. (2005). "A (Very) Brief History of Artificial Intelligence" (PDF). *AI Magazine*: 53–60. Archived (PDF) from the original on 26 September 2007.
- Butler, Samuel (13 June 1863). "Darwin among the Machines". Letters to the Editor. *The Press* (Christchurch, New Zealand). Retrieved 16 October 2014 – via Victoria University of Wellington.
- "AI set to exceed human brain power". *CNN*. 26 July 2006. Archived from the original on 19 February 2008.
- Dennett, Daniel (1991). *Consciousness Explained*. The Penguin Press. ISBN 0-7139-9037-6.
- Diamond, David (December 2003). "The Love Machine; Building computers that care". Wired. Archived from the original on 18 May 2008.
- Dowe, D. L.; Hajek, A. R. (1997). "A computational extension to the Turing Test". *Proceedings of the 4th Conference of the Australasian Cognitive Science Society*.
- Dreyfus, Hubert (1972). *What Computers Can't Do*. New York: MIT Press. ISBN 0-06-011082-1.
- Dreyfus, Hubert; Dreyfus, Stuart (1986). *Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. Oxford, UK: Blackwell. ISBN 0-02-908060-6.
- Dreyfus, Hubert (1992). *What Computers* Still *Can't Do*. New York: MIT Press. ISBN 0-262-54067-3.
- Dyson, George (1998). *Darwin among the Machines*. Allan Lane Science. ISBN 0-7382-0030-1.
- Edelman, Gerald (23 November 2007). "Gerald Edelman – Neural Darwinism and Brain-based Devices". Talking Robots.
- Edelson, Edward (1991). *The Nervous System*. New York: Chelsea House. ISBN 978-0-7910-0464-7.
- Fearn, Nicholas (2007). *The Latest Answers to the Oldest Questions: A Philosophical Adventure with the World's Greatest Thinkers*. New York: Grove Press. ISBN 0-8021-1839-9.
- Gladwell, Malcolm (2005). *Blink*. New York: Little, Brown and Co. ISBN 0-316-17232-4.
- Gödel, Kurt (1951). *Some basic theorems on the foundations of mathematics and their implications*. Gibbs Lecture. In
Feferman, Solomon, ed. (1995). *Kurt Gödel: Collected Works, Vol. III: Unpublished Essays and Lectures*. Oxford University Press. pp. 304–23. ISBN 978-0-19-514722-3.
- Haugeland, John (1985). *Artificial Intelligence: The Very Idea*. Cambridge, Mass.: MIT Press. ISBN 0-262-08153-9.
- Hawkins, Jeff; Blakeslee, Sandra (2005). *On Intelligence*. New York, NY: Owl Books. ISBN 0-8050-7853-3.
- Henderson, Mark (24 April 2007). "Human rights for robots? We're getting carried away". *The Times Online* (London).
- Hernandez-Orallo, Jose (2000). "Beyond the Turing Test". *Journal of Logic, Language and Information* **9** (4): 447–466. doi:10.1023/A:1008367325700.
- Hernandez-Orallo, J.; Dowe, D. L. (2010). "Measuring Universal Intelligence: Towards an Anytime Intelligence Test". *Artificial Intelligence Journal* **174** (18): 1508–1539. doi:10.1016/j.artint.2010.09.006.
- Hinton, G. E. (2007). "Learning multiple layers of representation". *Trends in Cognitive Sciences* **11**: 428–434. doi:10.1016/j.tics.2007.09.004.
- Hofstadter, Douglas (1979). *Gödel, Escher, Bach: an Eternal Golden Braid*. New York, NY: Vintage Books. ISBN 0-394-74502-7.
- Holland, John H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press. ISBN 0-262-58111-6.
- Howe, J. (November 1994). "Artificial Intelligence at Edinburgh University: a Perspective". Retrieved 30 August 2007.
- Hutter, M. (2012). "One Decade of Universal Artificial Intelligence". *Theoretical Foundations of Artificial General Intelligence*. Atlantis Thinking Machines **4**. doi:10.2991/978-94-91216-62-6_5. ISBN 978-94-91216-61-9.
- James, William (1884). "What is Emotion". *Mind* **9**: 188–205. doi:10.1093/mind/os-IX.34.188. Cited by Tao & Tan 2005.
- Kahneman, Daniel; Slovic, D.; Tversky, Amos (1982). *Judgment under uncertainty: Heuristics and biases*. New York: Cambridge University Press. ISBN 0-521-28414-7.
- Katz, Yarden (1 November 2012). "Noam Chomsky on Where Artificial Intelligence Went Wrong". *The Atlantic*. Retrieved 26 October 2014.
- "Kismet". MIT Artificial Intelligence Laboratory, Humanoid Robotics Group. Retrieved 25 October 2014.
- Koza, John R. (1992). *Genetic Programming (On the Programming of Computers by Means of Natural Selection)*. MIT Press. ISBN 0-262-11170-5.

- Kleine-Cosack, Christian (October 2006). "Recognition and Simulation of Emotions" (PDF). Archived from the original (PDF) on 28 May 2008.
- Kolata, G. (1982). "How can computers get common sense?". *Science* **217** (4566): 1237–1238. doi:10.1126/science.217.4566.1237. PMID 17837639.
- Kumar, Gulshan; Kumar, Krishan (2012). "The Use of Artificial-Intelligence-Based Ensembles for Intrusion Detection: A Review". *Applied Computational Intelligence and Soft Computing* **2012**: 1–20. doi:10.1155/2012/850160.
- Kurzweil, Ray (1999). *The Age of Spiritual Machines*. Penguin Books. ISBN 0-670-88217-8.
- Kurzweil, Ray (2005). *The Singularity is Near*. Penguin Books. ISBN 0-670-03384-7.
- Lakoff, George; Núñez, Rafael E. (2000). *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. Basic Books. ISBN 0-465-03771-2.
- Langley, Pat (2011). "The changing science of machine learning". *Machine Learning* **82** (3): 275–279. doi:10.1007/s10994-011-5242-y.
- Law, Diane (June 1994). *Searle, Subsymbolic Functionalism and Synthetic Intelligence* (Technical report). University of Texas at Austin. p. AI94-222. CiteSeerX: 10.1.1.38.8384.
- Legg, Shane; Hutter, Marcus (15 June 2007). *A Collection of Definitions of Intelligence* (Technical report). IDSIA. arXiv:0706.3639. 07-07.
- Lenat, Douglas; Guha, R. V. (1989). *Building Large Knowledge-Based Systems*. Addison-Wesley. ISBN 0-201-51752-3.
- Lighthill, James (1973). "Artificial Intelligence: A General Survey". *Artificial Intelligence: a paper symposium*. Science Research Council.
- Lucas, John (1961). "Minds, Machines and Gödel". In Anderson, A.R. *Minds and Machines*. Archived from the original on 19 August 2007. Retrieved 30 August 2007.
- Lungarella, M.; Metta, G.; Pfeifer, R.; Sandini, G. (2003). "Developmental robotics: a survey". *Connection Science* **15**: 151–190. doi:10.1080/09540090310001655110. CiteSeerX: 10.1.1.83.7615.
- Mahmud, Ashik (June 2015), "Post/Human Beings & Techno-salvation: Exploring Artificial Intelligence in Selected Science Fictions", *Socrates Journal*, doi:10.7910/DVN/VAELLN, retrieved 2015-06-26
- Maker, Meg Houston (2006). "AI@50: AI Past, Present, Future". Dartmouth College. Archived from the original on 8 October 2008. Retrieved 16 October 2008.
- Markoff, John (16 February 2011). "Computer Wins on 'Jeopardy!': Trivial, It's Not". *The New York Times*. Retrieved 25 October 2014.
- McCarthy, John; Minsky, Marvin; Rochester, Nathan; Shannon, Claude (1955). "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence". Archived from the original on 26 August 2007. Retrieved 30 August 2007..
- McCarthy, John; Hayes, P. J. (1969). "Some philosophical problems from the standpoint of artificial intelligence". *Machine Intelligence* **4**: 463–502. Archived from the original on 10 August 2007. Retrieved 30 August 2007.
- McCarthy, John (12 November 2007). "What Is Artificial Intelligence?".
- Minsky, Marvin (1967). *Computation: Finite and Infinite Machines*. Englewood Cliffs, N.J.: Prentice-Hall. ISBN 0-13-165449-7.
- Minsky, Marvin (2006). *The Emotion Machine*. New York, NY: Simon & Schusterl. ISBN 0-7432-7663-9.
- Moravec, Hans (1988). *Mind Children*. Harvard University Press. ISBN 0-674-57616-0.
- Norvig, Peter (25 June 2012). "On Chomsky and the Two Cultures of Statistical Learning". Peter Norvig. Archived from the original on 19 October 2014.
- NRC (United States National Research Council) (1999). "Developments in Artificial Intelligence". *Funding a Revolution: Government Support for Computing Research*. National Academy Press.
- Needham, Joseph (1986). *Science and Civilization in China: Volume 2*. Caves Books Ltd.
- Newell, Allen; Simon, H. A. (1976). "Computer Science as Empirical Inquiry: Symbols and Search". *Communications of the ACM* **19** (3)..
- Nilsson, Nils (1983). "Artificial Intelligence Prepares for 2001" (PDF). *AI Magazine* **1** (1). Presidential Address to the Association for the Advancement of Artificial Intelligence.
- O'Brien, James; Marakas, George (2011). *Management Information Systems* (10th ed.). McGraw-Hill/Irwin. ISBN 978-0-07-337681-3.

- O'Connor, Kathleen Malone (1994). "The alchemical creation of life (takwin) and other concepts of Genesis in medieval Islam". University of Pennsylvania.
- Oudeyer, P-Y. (2010). "On the impact of robotics in behavioral and cognitive sciences: from insect navigation to human cognitive development" (PDF). *IEEE Transactions on Autonomous Mental Development* **2** (1): 2–16. doi:10.1109/tamd.2009.2039057.
- Penrose, Roger (1989). *The Emperor's New Mind: Concerning Computer, Minds and The Laws of Physics*. Oxford University Press. ISBN 0-19-851973-7.
- Picard, Rosalind (1995). *Affective Computing* (PDF) (Technical report). MIT. 321. Lay summary – *Abstract*.
- Poli, R.; Langdon, W. B.; McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu.com. ISBN 978-1-4092-0073-4 – via gp-field-guide.org.uk.
- Rajani, Sandeep (2011). "Artificial Intelligence – Man or Machine" (PDF). *International Journal of Information Technology and Knowledge Management* **4** (1): 173–176.
- Searle, John (1980). "Minds, Brains and Programs". *Behavioral and Brain Sciences* **3** (3): 417–457. doi:10.1017/S0140525X00005756.
- Searle, John (1999). *Mind, language and society*. New York, NY: Basic Books. ISBN 0-465-04521-9. OCLC 231867665 43689264.
- Shapiro, Stuart C. (1992). "Artificial Intelligence". In Shapiro, Stuart C. *Encyclopedia of Artificial Intelligence* (PDF) (2nd ed.). New York: John Wiley. pp. 54–57. ISBN 0-471-50306-1.
- Simon, H. A. (1965). *The Shape of Automation for Men and Management*. New York: Harper & Row.
- Skillings, Jonathan (3 July 2006). "Getting Machines to Think Like Us". *cnet*. Retrieved 3 February 2011.
- Solomonoff, Ray (1956). *An Inductive Inference Machine* (PDF). Dartmouth Summer Research Conference on Artificial Intelligence – via std.com, pdf scanned copy of the original. Later published as Solomonoff, Ray (1957). "An Inductive Inference Machine". *IRE Convention Record*. Section on Information Theory, part 2. pp. 56–62.
- Tao, Jianhua; Tan, Tieniu (2005). *Affective Computing and Intelligent Interaction*. Affective Computing: A Review. LNCS 3784. Springer. pp. 981–995. doi:10.1007/11573548.
- Tecuci, Gheorghe (March–April 2012). "Artificial Intelligence". *Wiley Interdisciplinary Reviews: Computational Statistics* (Wiley) **4** (2): 168–180. doi:10.1002/wics.200.
- Thro, Ellen (1993). *Robotics: The Marriage of Computers and Machines*. New York: Facts on File. ISBN 978-0-8160-2628-9.
- Turing, Alan (October 1950), "Computing Machinery and Intelligence", *Mind* **LIX** (236): 433–460, doi:10.1093/mind/LIX.236.433, ISSN 0026-4423, retrieved 2008-08-18.
- van der Walt, Christiaan; Bernard, Etienne (2006). "Data characteristics that determine classifier performance" (PDF). Retrieved 5 August 2009.
- Vinge, Vernor (1993). "The Coming Technological Singularity: How to Survive in the Post-Human Era".
- Wason, P. C.; Shapiro, D. (1966). "Reasoning". In Foss, B. M. *New horizons in psychology*. Harmondsworth: Penguin.
- Weizenbaum, Joseph (1976). *Computer Power and Human Reason*. San Francisco: W.H. Freeman & Company. ISBN 0-7167-0464-1.
- Weng, J.; McClelland; Pentland, A.; Sporns, O.; Stockman, I.; Sur, M.; Thelen, E. (2001). "Autonomous mental development by robots and animals" (PDF). *Science* **291**: 599–600 – via msu.edu.

2.9 Further reading

- TechCast Article Series, John Sagi, Framing Consciousness
- Boden, Margaret, Mind As Machine, Oxford University Press, 2006
- Johnston, John (2008) "The Allure of Machinic Life: Cybernetics, Artificial Life, and the New AI", MIT Press
- Myers, Courtney Boyd ed. (2009). The AI Report. Forbes June 2009
- Serenko, Alexander (2010). "The development of an AI journal ranking based on the revealed preference approach" (PDF). *Journal of Informetrics* **4** (4): 447–459. doi:10.1016/j.joi.2010.04.001.
- Serenko, Alexander; Michael Dohan (2011). "Comparing the expert survey and citation impact journal ranking methods: Example from the field of Artificial Intelligence" (PDF). *Journal of Informetrics* **5** (4): 629–649. doi:10.1016/j.joi.2011.06.002.

- Sun, R. & Bookman, L. (eds.), *Computational Architectures: Integrating Neural and Symbolic Processes*. Kluwer Academic Publishers, Needham, MA. 1994.
- Tom Simonite (29 December 2014). "2014 in Computing: Breakthroughs in Artificial Intelligence". *MIT Technology Review*.

2.10 External links

- What Is AI? – An introduction to artificial intelligence by AI founder John McCarthy.
- The Handbook of Artificial Intelligence Volume I by Avron Barr and Edward A. Feigenbaum (Stanford University)
- Artificial Intelligence entry in the *Internet Encyclopedia of Philosophy*
- Logic and Artificial Intelligence entry by Richmond Thomason in the *Stanford Encyclopedia of Philosophy*
- AI at DMOZ
- AITopics – A large directory of links and other resources maintained by the Association for the Advancement of Artificial Intelligence, the leading organization of academic AI researchers.
- Cybernetics and AI

Chapter 3

Information theory

Not to be confused with information science.

Information theory is a branch of applied mathematics, electrical engineering, and computer science involving the quantification of information. Information theory was developed by Claude E. Shannon to find fundamental limits on signal processing operations such as compressing data and on reliably storing and communicating data. Since its inception it has broadened to find applications in many other areas, including statistical inference, natural language processing, cryptography, neurobiology,[1] the evolution[2] and function[3] of molecular codes, model selection in ecology,[4] thermal physics,[5] quantum computing, linguistics, plagiarism detection,[6] pattern recognition, anomaly detection and other forms of data analysis.[7]

A key measure of information is entropy, which is usually expressed by the average number of bits needed to store or communicate one symbol in a message. Entropy quantifies the uncertainty involved in predicting the value of a random variable. For example, specifying the outcome of a fair coin flip (two equally likely outcomes) provides less information (lower entropy) than specifying the outcome from a roll of a die (six equally likely outcomes).

Applications of fundamental topics of information theory include lossless data compression (e.g. ZIP files), lossy data compression (e.g. MP3s and JPEGs), and channel coding (e.g. for Digital Subscriber Line (DSL)). The field is at the intersection of mathematics, statistics, computer science, physics, neurobiology, and electrical engineering. Its impact has been crucial to the success of the Voyager missions to deep space, the invention of the compact disc, the feasibility of mobile phones, the development of the Internet, the study of linguistics and of human perception, the understanding of black holes, and numerous other fields. Important sub-fields of information theory are source coding, channel coding, algorithmic complexity theory, algorithmic information theory, information-theoretic security, and measures of information.

3.1 Overview

Information theory studies the transmission, processing, utilization, and extraction of information. Abstractly, information can be thought of as the resolution of uncertainty. In the case of communication of information over a noisy channel, this abstract concept was made concrete in 1948 by Claude Shannon in A Mathematical Theory of Communication, in which "information" is thought of as a set of possible messages, where the goal is to send these messages over a noisy channel, and then to have the receiver reconstruct the message with low probability of error, in spite of the channel noise. Shannon's main result, the Noisy-channel coding theorem showed that, in the limit of many channel uses, the rate of information that is asymptotically achievable equal to the Channel capacity, a quantity dependent merely on the statistics of the channel over which the messages are sent.

Information theory is closely associated with a collection of pure and applied disciplines that have been investigated and reduced to engineering practice under a variety of rubrics throughout the world over the past half century or more: adaptive systems, anticipatory systems, artificial intelligence, complex systems, complexity science, cybernetics, informatics, machine learning, along with systems sciences of many descriptions. Information theory is a broad and deep mathematical theory, with equally broad and deep applications, amongst which is the vital field of coding theory.

Coding theory is concerned with finding explicit methods, called *codes*, for increasing the efficiency and reducing the error rate of data communication over noisy channels to near the Channel capacity. These codes can be roughly subdivided into data compression (source coding) and error-correction (channel coding) techniques. In the latter case, it took many years to find the methods Shannon's work proved were possible. A third class of information theory codes are cryptographic algorithms (both codes and ciphers). Concepts, methods and results from coding theory and information theory are widely used in cryptography and cryptanalysis. *See the article ban (unit) for a historical application.*

Information theory is also used in information retrieval, intelligence gathering, gambling, statistics, and even in musical composition.

3.2 Historical background

Main article: History of information theory

The landmark event that established the discipline of information theory, and brought it to immediate worldwide attention, was the publication of Claude E. Shannon's classic paper "A Mathematical Theory of Communication" in the *Bell System Technical Journal* in July and October 1948.

Prior to this paper, limited information-theoretic ideas had been developed at Bell Labs, all implicitly assuming events of equal probability. Harry Nyquist's 1924 paper, *Certain Factors Affecting Telegraph Speed*, contains a theoretical section quantifying “intelligence” and the “line speed” at which it can be transmitted by a communication system, giving the relation $W = K \log m$ (recalling Boltzmann's constant), where *W* is the speed of transmission of intelligence, *m* is the number of different voltage levels to choose from at each time step, and *K* is a constant. Ralph Hartley's 1928 paper, *Transmission of Information*, uses the word *information* as a measurable quantity, reflecting the receiver's ability to distinguish one sequence of symbols from any other, thus quantifying information as $H = \log S^n = n \log S$, where *S* was the number of possible symbols, and *n* the number of symbols in a transmission. The unit of information was therefore the decimal digit, much later renamed the hartley in his honour as a unit or scale or measure of information. Alan Turing in 1940 used similar ideas as part of the statistical analysis of the breaking of the German second world war Enigma ciphers.

Much of the mathematics behind information theory with events of different probabilities were developed for the field of thermodynamics by Ludwig Boltzmann and J. Willard Gibbs. Connections between information-theoretic entropy and thermodynamic entropy, including the important contributions by Rolf Landauer in the 1960s, are explored in *Entropy in thermodynamics and information theory*.

In Shannon's revolutionary and groundbreaking paper, the work for which had been substantially completed at Bell Labs by the end of 1944, Shannon for the first time introduced the qualitative and quantitative model of communication as a statistical process underlying information theory, opening with the assertion that

> “The fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point.”

With it came the ideas of

- the information entropy and redundancy of a source, and its relevance through the source coding theorem;
- the mutual information, and the channel capacity of a noisy channel, including the promise of perfect loss-free communication given by the noisy-channel coding theorem;
- the practical result of the Shannon–Hartley law for the channel capacity of a Gaussian channel; as well as
- the bit—a new way of seeing the most fundamental unit of information.

3.3 Quantities of information

Main article: Quantities of information

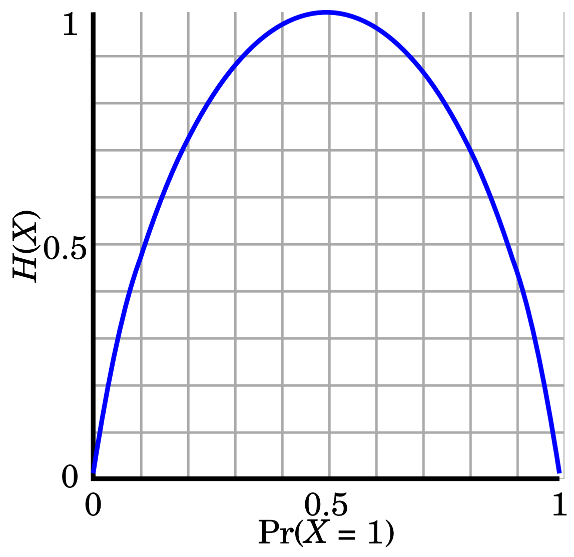
Information theory is based on probability theory and statistics. Information theory often concerns itself with measures of information of the distributions associated with random variables. Important quantities of information are entropy, a measure of information in a single random variable, and mutual information, a measure of information in common between two random variables. The former quantity is a property of the probability distribution of a random variable and gives a limit on the rate at which data generated by independent samples with the given distribution can be reliably compressed. The latter is a property of the joint distribution of two random variable, and is the maximum rate of reliable communication across a noisy channel in the limit of long block lengths, when the channel statistics are determined by the joint distribution.

The choice of logarithmic base in the following formulae determines the unit of information entropy that is used. A common unit of information is the bit, based on the binary logarithm. Other units include the nat, which is based on the natural logarithm, and the hartley, which is based on the common logarithm.

In what follows, an expression of the form $p \log p$ is considered by convention to be equal to zero whenever $p = 0$. This is justified because $\lim_{p \to 0+} p \log p = 0$ for any logarithmic base.

3.3.1 Entropy

The **entropy**, H , of a discrete random variable X intuitively is a measure of the amount of *uncertainty* associated with the value of X when only its distribution is known. So, for example, if the distribution associated with a random variable was a constant distribution, (i.e.



Entropy of a Bernoulli trial as a function of success probability, often called the ***binary entropy function****,* $H_b(p)$ *. The entropy is maximized at 1 bit per trial when the two possible outcomes are equally probable, as in an unbiased coin toss.*

equal to some known value with probability 1), then entropy is minimal, and equal to 0 . Furthermore, in the case of a distribution restricted to take on a finite number of values, entropy is maximized with a uniform distribution over the values that the distribution takes on.

Suppose one transmits 1000 bits (0s and 1s). If the value of each these bits is known to the receiver (has a specific value with certainty) ahead of transmission, it is clear that no information is transmitted. If, however, each bit is independently equally likely to be 0 or 1, 1000 shannons of information (also often called bits, in the information theoretic sense) have been transmitted. Between these two extremes, information can be quantified as follows. If $\mathbb{X}$ is the set of all messages $\{x_1, ..., x_n\}$ that X could be, and $p(x)$ is the probability of some $x \in \mathbb{X}$, then the entropy, H , of X is defined:[8]

$$H(X) = \mathbb{E}_X[I(x)] = -\sum_{x\in\mathbb{X}} p(x)\log p(x).$$

(Here, $I(x)$ is the self-information, which is the entropy contribution of an individual message, and $\mathbb{E}_X$ is the expected value.) A property of entropy is that it is maximized when all the messages in the message space are equiprobable $p(x) = 1/n$,—i.e., most unpredictable—in which case $H(X) = \log n$.

The special case of information entropy for a random variable with two outcomes is the **binary entropy function**, usually taken to the logarithmic base 2, thus having the shannon (Sh) as unit:

$$H_\mathrm{b}(p) = -p\log_2 p - (1-p)\log_2(1-p).$$

3.3.2 Joint entropy

The **joint entropy** of two discrete random variables X and Y is merely the entropy of their pairing: (X, Y) . This implies that if X and Y are independent, then their joint entropy is the sum of their individual entropies.

For example, if (X, Y) represents the position of a chess piece — X the row and Y the column, then the joint entropy of the row of the piece and the column of the piece will be the entropy of the position of the piece.

$$H(X,Y) = \mathbb{E}_{X,Y}[-\log p(x,y)] = -\sum_{x,y} p(x,y)\log p(x,y)$$

Despite similar notation, joint entropy should not be confused with **cross entropy**.

3.3.3 Conditional entropy (equivocation)

The **conditional entropy** or **conditional uncertainty** of X given random variable Y (also called the **equivocation** of X about Y) is the average conditional entropy over Y :[9]

$$H(X|Y) = \mathbb{E}_Y[H(X|y)] = -\sum_{y\in Y} p(y)\sum_{x\in X} p(x|y)\log p(x|y) = -\sum_{x,y} p$$

Because entropy can be conditioned on a random variable or on that random variable being a certain value, care should be taken not to confuse these two definitions of conditional entropy, the former of which is in more common use. A basic property of this form of conditional entropy is that:

$$H(X|Y) = H(X,Y) - H(Y).$$

3.3.4 Mutual information (transinformation)

Mutual information measures the amount of information that can be obtained about one random variable by observing another. It is important in communication where it can be used to maximize the amount of information shared between sent and received signals. The mutual information of X relative to Y is given by:

$$I(X;Y) = \mathbb{E}_{X,Y}[SI(x,y)] = \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)\,p(y)}$$

where SI (*S*pecific mutual *I*nformation) is the pointwise mutual information.

A basic property of the mutual information is that

$$I(X;Y) = H(X) - H(X|Y).$$

That is, knowing Y, we can save an average of $I(X;Y)$ bits in encoding X compared to not knowing Y.

Mutual information is symmetric:

$$I(X;Y) = I(Y;X) = H(X) + H(Y) - H(X,Y).$$

Mutual information can be expressed as the average Kullback–Leibler divergence (information gain) between the posterior probability distribution of X given the value of Y and the prior distribution on X:

$$I(X;Y) = \mathbb{E}_{p(y)}[D_{\mathrm{KL}}(p(X|Y=y)\|p(X))].$$

In other words, this is a measure of how much, on the average, the probability distribution on X will change if we are given the value of Y. This is often recalculated as the divergence from the product of the marginal distributions to the actual joint distribution:

$$I(X;Y) = D_{\mathrm{KL}}(p(X,Y)\|p(X)p(Y)).$$

Mutual information is closely related to the log-likelihood ratio test in the context of contingency tables and the multinomial distribution and to Pearson's χ^2 test: mutual information can be considered a statistic for assessing independence between a pair of variables, and has a well-specified asymptotic distribution.

3.3.5 Kullback–Leibler divergence (information gain)

The **Kullback–Leibler divergence** (or **information divergence**, **information gain**, or **relative entropy**) is a way of comparing two distributions: a "true" probability distribution *p(X)*, and an arbitrary probability distribution *q(X)*. If we compress data in a manner that assumes *q(X)* is the distribution underlying some data, when, in reality, *p(X)* is the correct distribution, the Kullback–Leibler divergence is the number of average additional bits per datum necessary for compression. It is thus defined

$$D_{\mathrm{KL}}(p(X)\|q(X)) = \sum_{x\in X} -p(x)\log q(x) - \sum_{x\in X} -p(x)\log p(x) = \sum_{x\in X} p(x)\log\frac{p(x)}{q(x)}.$$

Although it is sometimes used as a 'distance metric', KL divergence is not a true metric since it is not symmetric and does not satisfy the triangle inequality (making it a semi-quasimetric).

3.3.6 Kullback–Leibler divergence of a prior from the truth

Another interpretation of KL divergence is this: suppose a number X is about to be drawn randomly from a discrete set with probability distribution *p(x)*. If Alice knows the true distribution *p(x)*, while Bob believes (has a prior) that the distribution is *q(x)*, then Bob will be more surprised than Alice, on average, upon seeing the value of X. The KL divergence is the (objective) expected value of Bob's (subjective) surprisal minus Alice's surprisal, measured in bits if the *log* is in base 2. In this way, the extent to which Bob's prior is "wrong" can be quantified in terms of how "unnecessarily surprised" it's expected to make him.

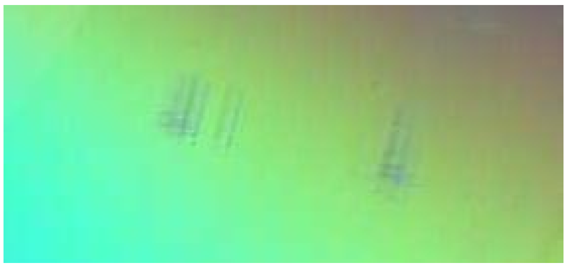
3.3.7 Other quantities

Other important information theoretic quantities include Rényi entropy (a generalization of entropy), differential entropy (a generalization of quantities of information to continuous distributions), and the conditional mutual information.

3.4 Coding theory

Main article: Coding theory

Coding theory is one of the most important and direct



A picture showing scratches on the readable surface of a CD-R. Music and data CDs are coded using error correcting codes and thus can still be read even if they have minor scratches using error detection and correction.

applications of information theory. It can be subdivided into source coding theory and channel coding theory. Using a statistical description for data, information theory quantifies the number of bits needed to describe the data, which is the information entropy of the source.

- Data compression (source coding): There are two formulations for the compression problem:

1. lossless data compression: the data must be reconstructed exactly;
2. lossy data compression: allocates bits needed to reconstruct the data, within a specified fidelity level

measured by a distortion function. This subset of Information theory is called rate–distortion theory.

- Error-correcting codes (channel coding): While data compression removes as much redundancy as possible, an error correcting code adds just the right kind of redundancy (i.e., error correction) needed to transmit the data efficiently and faithfully across a noisy channel.

This division of coding theory into compression and transmission is justified by the information transmission theorems, or source–channel separation theorems that justify the use of bits as the universal currency for information in many contexts. However, these theorems only hold in the situation where one transmitting user wishes to communicate to one receiving user. In scenarios with more than one transmitter (the multiple-access channel), more than one receiver (the broadcast channel) or intermediary "helpers" (the relay channel), or more general networks, compression followed by transmission may no longer be optimal. Network information theory refers to these multi-agent communication models.

3.4.1 Source theory

Any process that generates successive messages can be considered a **source** of information. A memoryless source is one in which each message is an independent identically distributed random variable, whereas the properties of ergodicity and stationarity impose less restrictive constraints. All such sources are stochastic. These terms are well studied in their own right outside information theory.

Rate

Information **rate** is the average entropy per symbol. For memoryless sources, this is merely the entropy of each symbol, while, in the case of a stationary stochastic process, it is

$$r = \lim_{n\to\infty} H(X_n|X_{n-1}, X_{n-2}, X_{n-3}, \ldots);$$

that is, the conditional entropy of a symbol given all the previous symbols generated. For the more general case of a process that is not necessarily stationary, the *average rate* is

$$r = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots X_n);$$

that is, the limit of the joint entropy per symbol. For stationary sources, these two expressions give the same result.[10]

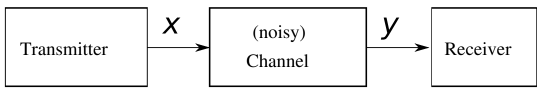
It is common in information theory to speak of the "rate" or "entropy" of a language. This is appropriate, for example, when the source of information is English prose. The rate of a source of information is related to its redundancy and how well it can be compressed, the subject of **source coding**.

3.4.2 Channel capacity

Main article: Channel capacity

Communications over a channel—such as an ethernet cable—is the primary motivation of information theory. As anyone who's ever used a telephone (mobile or landline) knows, however, such channels often fail to produce exact reconstruction of a signal; noise, periods of silence, and other forms of signal corruption often degrade quality. How much information can one hope to communicate over a noisy (or otherwise imperfect) channel?

Consider the communications process over a discrete channel. A simple model of the process is shown below:



Here X represents the space of messages transmitted, and Y the space of messages received during a unit time over our channel. Let $p(y|x)$ be the conditional probability distribution function of Y given X. We will consider $p(y|x)$ to be an inherent fixed property of our communications channel (representing the nature of the **noise** of our channel). Then the joint distribution of X and Y is completely determined by our channel and by our choice of $f(x)$, the marginal distribution of messages we choose to send over the channel. Under these constraints, we would like to maximize the rate of information, or the **signal**, we can communicate over the channel. The appropriate measure for this is the mutual information, and this maximum mutual information is called the **channel capacity** and is given by:

$$C = \max_f I(X;Y).$$

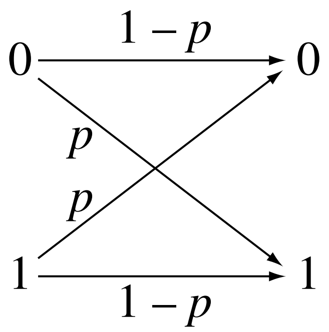
This capacity has the following property related to communicating at information rate R (where R is usually bits per symbol). For any information rate $R < C$ and coding error $\varepsilon > 0$, for large enough N, there exists a code of length N and rate $\geq$ R and a decoding algorithm, such that the maximal probability of block error is $\leq \varepsilon$; that is, it is always possible to transmit with arbitrarily small block error. In addition, for any rate $R > C$, it is impossible to transmit with arbitrarily small block error.

Channel coding is concerned with finding such nearly optimal codes that can be used to transmit data over a noisy channel with a small coding error at a rate near the channel capacity.

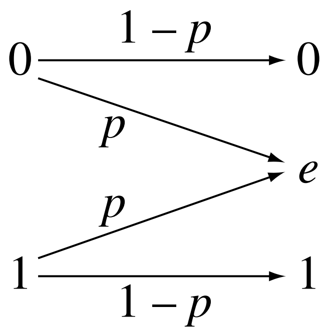
Capacity of particular channel models

- A continuous-time analog communications channel subject to Gaussian noise — see Shannon–Hartley theorem.

- A binary symmetric channel (BSC) with crossover probability p is a binary input, binary output channel that flips the input bit with probability p. The BSC has a capacity of $1 - H_b(p)$ bits per channel use, where H_b is the binary entropy function to the base 2 logarithm:



- A binary erasure channel (BEC) with erasure probability p is a binary input, ternary output channel. The possible channel outputs are *0*, *1*, and a third symbol 'e' called an erasure. The erasure represents complete loss of information about an input bit. The capacity of the BEC is *1 - p* bits per channel use.



3.5 Applications to other fields

3.5.1 Intelligence uses and secrecy applications

Information theoretic concepts apply to cryptography and cryptanalysis. Turing's information unit, the ban, was used in the Ultra project, breaking the German Enigma machine code and hastening the end of World War II in Europe. Shannon himself defined an important concept now called the unicity distance. Based on the redundancy of the plaintext, it attempts to give a minimum amount of ciphertext necessary to ensure unique decipherability.

Information theory leads us to believe it is much more difficult to keep secrets than it might first appear. A brute force attack can break systems based on asymmetric key algorithms or on most commonly used methods of symmetric key algorithms (sometimes called secret key algorithms), such as block ciphers. The security of all such methods currently comes from the assumption that no known attack can break them in a practical amount of time.

Information theoretic security refers to methods such as the one-time pad that are not vulnerable to such brute force attacks. In such cases, the positive conditional mutual information between the plaintext and ciphertext (conditioned on the key) can ensure proper transmission, while the unconditional mutual information between the plaintext and ciphertext remains zero, resulting in absolutely secure communications. In other words, an eavesdropper would not be able to improve his or her guess of the plaintext by gaining knowledge of the ciphertext but not of the key. However, as in any other cryptographic system, care must be used to correctly apply even information-theoretically secure methods; the Venona project was able to crack the one-time pads of the Soviet Union due to their improper reuse of key material.

3.5.2 Pseudorandom number generation

Pseudorandom number generators are widely available in computer language libraries and application programs. They are, almost universally, unsuited to cryptographic use as they do not evade the deterministic nature of modern computer equipment and software. A class of improved random number generators is termed cryptographically secure pseudorandom number generators, but even they require random seeds external to the software to work as intended. These can be obtained via extractors, if done carefully. The measure of sufficient randomness in extractors is min-entropy, a value related to Shannon entropy through Rényi entropy; Rényi entropy is also used in evaluating randomness in cryptographic systems. Although related, the distinctions among these measures mean that a random variable with high Shannon entropy is not necessarily satisfactory for use in an extractor and so for cryptography uses.

3.5.3 Seismic exploration

One early commercial application of information theory was in the field of seismic oil exploration. Work in this field made it possible to strip off and separate the unwanted noise from the desired seismic signal. Information theory and digital signal processing offer a major improvement of resolution and image clarity over previous analog methods.[11]

3.5.4 Semiotics

Concepts from information theory such as redundancy and code control have been used by semioticians such as Umberto Eco and Rossi-Landi to explain ideology as a form of message transmission whereby a dominant social class emits its message by using signs that exhibit a high degree of redundancy such that only one message is decoded among a selection of competing ones.[12]

3.5.5 Miscellaneous applications

Information theory also has applications in gambling and investing, black holes, bioinformatics, and music.

3.6 See also

- Algorithmic probability
- Algorithmic information theory
- Bayesian inference
- Communication theory
- Constructor theory - a generalization of information theory that includes quantum information
- Inductive probability
- Minimum message length
- Minimum description length
- List of important publications
- Philosophy of information

3.6.1 Applications

- Active networking
- Cryptanalysis
- Cryptography
- Cybernetics
- Entropy in thermodynamics and information theory
- Gambling
- Intelligence (information gathering)
- Seismic exploration

3.6.2 History

- Hartley, R.V.L.
- History of information theory
- Shannon, C.E.
- Timeline of information theory
- Yockey, H.P.

3.6.3 Theory

- Coding theory
- Detection theory
- Estimation theory
- Fisher information
- Information algebra
- Information asymmetry
- Information field theory
- Information geometry
- Information theory and measure theory
- Kolmogorov complexity
- Logic of information
- Network coding
- Philosophy of Information
- Quantum information science
- Semiotic information theory
- Source coding
- Unsolved Problems

3.6.4 Concepts

- Ban (unit)
- Channel capacity
- Channel (communications)
- Communication source
- Conditional entropy
- Covert channel
- Decoder
- Differential entropy
- Encoder

- Information entropy
- Joint entropy
- Kullback–Leibler divergence
- Mutual information
- Pointwise mutual information (PMI)
- Receiver (information theory)
- Redundancy
- Rényi entropy
- Self-information
- Unicity distance
- Variety

3.7 References

[1] F. Rieke, D. Warland, R Ruyter van Steveninck, W Bialek (1997). *Spikes: Exploring the Neural Code*. The MIT press. ISBN 978-0262681087.

[2] cf. Huelsenbeck, J. P., F. Ronquist, R. Nielsen and J. P. Bollback (2001) Bayesian inference of phylogeny and its impact on evolutionary biology, *Science* **294**:2310-2314

[3] Rando Allikmets, Wyeth W. Wasserman, Amy Hutchinson, Philip Smallwood, Jeremy Nathans, Peter K. Rogan, Thomas D. Schneider, Michael Dean (1998) Organization of the ABCR gene: analysis of promoter and splice junction sequences, *Gene* **215**:1, 111-122

[4] Burnham, K. P. and Anderson D. R. (2002) *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach, Second Edition* (Springer Science, New York) ISBN 978-0-387-95364-9.

[5] Jaynes, E. T. (1957) Information Theory and Statistical Mechanics, *Phys. Rev.* **106**:620

[6] Charles H. Bennett, Ming Li, and Bin Ma (2003) Chain Letters and Evolutionary Histories, *Scientific American* **288**:6, 76-81

[7] David R. Anderson (November 1, 2003). "Some background on why people in the empirical sciences may want to better understand the information-theoretic methods" (PDF). Retrieved 2010-06-23.

[8] Fazlollah M. Reza (1994) [1961]. *An Introduction to Information Theory*. Dover Publications, Inc., New York. ISBN 0-486-68210-2.

[9] Robert B. Ash (1990) [1965]. *Information Theory*. Dover Publications, Inc. ISBN 0-486-66521-6.

[10] Jerry D. Gibson (1998). *Digital Compression for Multimedia: Principles and Standards*. Morgan Kaufmann. ISBN 1-55860-369-7.

[11] The Corporation and Innovation, Haggerty, Patrick, Strategic Management Journal, Vol. 2, 97-118 (1981)

[12] Semiotics of Ideology, Noth, Winfried, Semiotica, Issue 148,(1981)

3.7.1 The classic work

- Shannon, C.E. (1948), "A Mathematical Theory of Communication", *Bell System Technical Journal*, 27, pp. 379–423 & 623–656, July & October, 1948. PDF.
  Notes and other formats.
- R.V.L. Hartley, "Transmission of Information", *Bell System Technical Journal*, July 1928
- Andrey Kolmogorov (1968), "Three approaches to the quantitative definition of information" in International Journal of Computer Mathematics.

3.7.2 Other journal articles

- J. L. Kelly, Jr., Saratoga.ny.us, "A New Interpretation of Information Rate" *Bell System Technical Journal*, Vol. 35, July 1956, pp. 917–26.
- R. Landauer, IEEE.org, "Information is Physical" *Proc. Workshop on Physics and Computation PhysComp'92* (IEEE Comp. Sci.Press, Los Alamitos, 1993) pp. 1–4.
- R. Landauer, IBM.com, "Irreversibility and Heat Generation in the Computing Process" *IBM J. Res. Develop.* Vol. 5, No. 3, 1961
- Timme, Nicholas; Alford, Wesley; Flecker, Benjamin; Beggs, John M. (2012). "Multivariate information measures: an experimentalist's perspective". *arXiv:111.6857v5* (Cornell University) **5**. Retrieved 7 June 2015.

3.7.3 Textbooks on information theory

- Arndt, C. *Information Measures, Information and its Description in Science and Engineering* (Springer Series: Signals and Communication Technology), 2004, ISBN 978-3-540-40855-0
- Ash, RB. *Information Theory*. New York: Interscience, 1965. ISBN 0-470-03445-9. New York: Dover 1990. ISBN 0-486-66521-6
- Gallager, R. *Information Theory and Reliable Communication.* New York: John Wiley and Sons, 1968. ISBN 0-471-29048-3
- Goldman, S. *Information Theory*. New York: Prentice Hall, 1953. New York: Dover 1968 ISBN 0-486-62209-6, 2005 ISBN 0-486-44271-3

- Cover, TM, Thomas, JA. *Elements of information theory*, 1st Edition. New York: Wiley-Interscience, 1991. ISBN 0-471-06259-6.

  2nd Edition. New York: Wiley-Interscience, 2006. ISBN 0-471-24195-4.

- Csiszar, I, Korner, J. *Information Theory: Coding Theorems for Discrete Memoryless Systems* Akademiai Kiado: 2nd edition, 1997. ISBN 963-05-7440-3
- MacKay, DJC. *Information Theory, Inference, and Learning Algorithms* Cambridge: Cambridge University Press, 2003. ISBN 0-521-64298-1
- Mansuripur, M. *Introduction to Information Theory*. New York: Prentice Hall, 1987. ISBN 0-13-484668-0
- Pierce, JR. “An introduction to information theory: symbols, signals and noise”. Dover (2nd Edition). 1961 (reprinted by Dover 1980).
- Reza, F. *An Introduction to Information Theory*. New York: McGraw-Hill 1961. New York: Dover 1994. ISBN 0-486-68210-2
- Shannon, CE. Warren Weaver. *The Mathematical Theory of Communication.* Univ of Illinois Press, 1949. ISBN 0-252-72548-4
- Stone, JV. Chapter 1 of book “Information Theory: A Tutorial Introduction”, University of Sheffield, England, 2014. ISBN 978-0956372857.
- Yeung, RW. *A First Course in Information Theory* Kluwer Academic/Plenum Publishers, 2002. ISBN 0-306-46791-7.
- Yeung, RW. *Information Theory and Network Coding* Springer 2008, 2002. ISBN 978-0-387-79233-0

3.7.4 Other books

- Leon Brillouin, *Science and Information Theory*, Mineola, N.Y.: Dover, [1956, 1962] 2004. ISBN 0-486-43918-6
- James Gleick, *The Information: A History, a Theory, a Flood*, New York: Pantheon, 2011. ISBN 978-0-375-42372-7
- A. I. Khinchin, *Mathematical Foundations of Information Theory*, New York: Dover, 1957. ISBN 0-486-60434-9
- H. S. Leff and A. F. Rex, Editors, *Maxwell’s Demon: Entropy, Information, Computing*, Princeton University Press, Princeton, New Jersey (1990). ISBN 0-691-08727-X
- Tom Siegfried, *The Bit and the Pendulum*, Wiley, 2000. ISBN 0-471-32174-5
- Charles Seife, *Decoding The Universe*, Viking, 2006. ISBN 0-670-03441-X
- Jeremy Campbell, *Grammatical Man*, Touchstone/Simon & Schuster, 1982, ISBN 0-671-44062-4
- Henri Theil, *Economics and Information Theory*, Rand McNally & Company - Chicago, 1967.
- Escolano, Suau, Bonev, *Information Theory in Computer Vision and Pattern Recognition*, Springer, 2009. ISBN 978-1-84882-296-2

3.8 External links

- Erill I. (2012), "A gentle introduction to information content in transcription factor binding sites" (University of Maryland, Baltimore County)
- Hazewinkel, Michiel, ed. (2001), “Information”, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Lambert F. L. (1999), "Shuffled Cards, Messy Desks, and Disorderly Dorm Rooms - Examples of Entropy Increase? Nonsense!", *Journal of Chemical Education*
- Schneider T. D. (2014), "Information Theory Primer"
- Srinivasa, S., "A Review on Multivariate Mutual Information"
- IEEE Information Theory Society and ITSoc review articles

Chapter 4

Computational science

Not to be confused with computer science.

Computational science (also **scientific computing** or **scientific computation**) is concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyze and solve scientific problems.[1] In practical use, it is typically the application of computer simulation and other forms of computation from numerical analysis and theoretical computer science to problems in various scientific disciplines.

The field is different from theory and laboratory experiment which are the traditional forms of science and engineering. The scientific computing approach is to gain understanding, mainly through the analysis of mathematical models implemented on computers.

Scientists and engineers develop computer programs, application software, that model systems being studied and run these programs with various sets of input parameters. In some cases, these models require massive amounts of calculations (usually floating-point) and are often executed on supercomputers or distributed computing platforms.

Numerical analysis is an important underpinning for techniques used in computational science.

4.1 Applications of computational science

Problem domains for computational science/scientific computing include:

4.1.1 Numerical simulations

Numerical simulations have different objectives depending on the nature of the task being simulated:

- Reconstruct and understand known events (e.g., earthquake, tsunamis and other natural disasters).
- Predict future or unobserved situations (e.g., weather, sub-atomic particle behaviour, and primordial explosions).

4.1.2 Model fitting and data analysis

- Appropriately tune models or solve equations to reflect observations, subject to model constraints (e.g. oil exploration geophysics, computational linguistics).
- Use graph theory to model networks, such as those connecting individuals, organizations, websites, and biological systems.

4.1.3 Computational optimization

Main article: Mathematical optimization

- Optimize known scenarios (e.g., technical and manufacturing processes, front-end engineering).
- Machine learning

4.2 Methods and algorithms

Algorithms and mathematical methods used in computational science are varied. Commonly applied methods include:

- Numerical analysis
- Application of Taylor series as convergent and asymptotic series
- Computing derivatives by Automatic differentiation (AD)
- Computing derivatives by finite differences
- Graph theoretic suites
- High order difference approximations via Taylor series and Richardson extrapolation

- Methods of integration on a uniform mesh: rectangle rule (also called *midpoint rule*), trapezoid rule, Simpson's rule
- Runge Kutta method for solving ordinary differential equations
- Monte Carlo methods
- Molecular dynamics
- Linear programming
- Branch and cut
- Branch and Bound
- Numerical linear algebra
- Computing the LU factors by Gaussian elimination
- Cholesky factorizations
- Discrete Fourier transform and applications.
- Newton's method
- Time stepping methods for dynamical systems

Programming languages and computer algebra systems commonly used for the more mathematical aspects of scientific computing applications include R (programming language), TK Solver, MATLAB, Mathematica,[2] SciLab, GNU Octave, Python (programming language) with SciPy, and PDL. The more computationally intensive aspects of scientific computing will often use some variation of C or Fortran and optimized algebra libraries such as BLAS or LAPACK.

Computational science application programs often model real-world changing conditions, such as weather, air flow around a plane, automobile body distortions in a crash, the motion of stars in a galaxy, an explosive device, etc. Such programs might create a 'logical mesh' in computer memory where each item corresponds to an area in space and contains information about that space relevant to the model. For example in weather models, each item might be a square kilometer; with land elevation, current wind direction, humidity, temperature, pressure, etc. The program would calculate the likely next state based on the current state, in simulated time steps, solving equations that describe how the system operates; and then repeat the process to calculate the next state.

The term computational scientist is used to describe someone skilled in scientific computing. This person is usually a scientist, an engineer or an applied mathematician who applies high-performance computing in different ways to advance the state-of-the-art in their respective applied disciplines in physics, chemistry or engineering. Scientific computing has increasingly also impacted on other areas including economics, biology and medicine.

Computational science is now commonly considered a third mode of science, complementing and adding to experimentation/observation and theory.[3] The essence of computational science is numerical algorithm[4] and/or computational mathematics. In fact, substantial effort in computational sciences has been devoted to the development of algorithms, the efficient implementation in programming languages, and validation of computational results. A collection of problems and solutions in computational science can be found in Steeb, Hardy, Hardy and Stoop, 2004.[5]

4.3 Reproducibility and open research computing

The complexity of computational methods is a threat to the reproducibility of research. Jon Claerbout has become prominent for pointing out that *reproducible research* requires archiving and documenting all raw data and all code used to obtain a result.[6][7][8] Nick Barnes, in the *Science Code Manifesto*, proposed five principles that should be followed when software is used in open science publication.[9] Tomi Kauppinen et al. established and defined *Linked Open Science*, an approach to interconnect scientific assets to enable transparent, reproducible and transdisciplinary research.[10]

4.4 Journals

Most scientific journals do not accept software papers because a description of a reasonably mature software usually does not meet the criterion of *novelty*. Outside computer science itself, there are only few journals dedicated to scientific software. Established journals like Elsevier's Computer Physics Communications publish papers that are not open-access (though the described software usually is). To fill this gap, a new journal entitled *Open research computation* was announced in 2010;[11] it closed in 2012 without having published a single paper, for a lack of submissions probably due to excessive quality requirements.[12] A new initiative was launched in 2012, the *Journal of Open Research Software.*[13]

4.5 Education

Scientific computation is most often studied through an applied mathematics or computer science program, or within a standard mathematics, sciences, or engineering program. At some institutions a specialization in scientific computation can be earned as a "minor" within another program (which may be at varying levels). However, there are increasingly many bachelor's and master's programs in computational science. Some schools also offer the Ph.D. in computational science, computational engineering, computational science and engineering, or scientific computation.

There are also programs in areas such as computational physics, computational chemistry, etc.

4.6 Related fields

- Bioinformatics
- Cheminformatics
- Chemometrics
- Computational archaeology
- Computational biology
- Computational chemistry
- Computational economics
- Computational electromagnetics
- Computational engineering
- Computational finance
- Computational fluid dynamics
- Computational forensics
- Computational geophysics
- Computational informatics
- Computational intelligence
- Computational law
- Computational linguistics
- Computational mathematics
- Computational mechanics
- Computational neuroscience
- Computational particle physics
- Computational physics
- Computational sociology
- Computational statistics
- Computer algebra
- Environmental simulation
- Financial modeling
- Geographic information system (GIS)
- High performance computing
- Machine learning
- Network analysis
- Neuroinformatics
- Numerical linear algebra
- Numerical weather prediction
- Pattern recognition
- Scientific visualization

4.7 See also

- Computational science and engineering
- Comparison of computer algebra systems
- List of molecular modeling software
- List of numerical analysis software
- List of statistical packages
- Timeline of scientific computing
- Simulated reality

4.8 References

[1] National Center for Computational Science. Oak Ridge National Laboratory. Retrieved 11 Nov 2012.

[2] Mathematica 6 Scientific Computing World, May 2007

[3] Graduate Education for Computational Science and Engineering.Siam.org, Society for Industrial and Applied Mathematics (SIAM) website; accessed Feb 2013.

[4] Nonweiler T. R., 1986. Computational Mathematics: An Introduction to Numerical Approximation, John Wiley and Sons

[5] Steeb W.-H., Hardy Y., Hardy A. and Stoop R., 2004. Problems and Solutions in Scientific Computing with C++ and Java Simulations, World Scientific Publishing. ISBN 981-256-112-9

[6] Sergey Fomel and Jon Claerbout, "Guest Editors' Introduction: Reproducible Research," Computing in Science and Engineering, vol. 11, no. 1, pp. 5–7, Jan./Feb. 2009, doi:10.1109/MCSE.2009.14

[7] J. B. Buckheit and D. L. Donoho, "WaveLab and Reproducible Research," Dept. of Statistics, Stanford University, Tech. Rep. 474, 1995.

[8] The Yale Law School Round Table on Data and Core Sharing: "Reproducible Research", Computing in Science and Engineering, vol. 12, no. 5, pp. 8–12, Sept/Oct 2010, doi:10.1109/MCSE.2010.113

[9] Science Code Manifesto homepage. Accessed Feb 2013.

[10] Kauppinen, T.; Espindola, G. M. D. (2011). "Linked Open Science-Communicating, Sharing and Evaluating Data, Methods and Results for Executable Papers". *Procedia Computer Science* **4**: 726. doi:10.1016/j.procs.2011.04.076.

[11] CameronNeylon.net, 13 December 2010. Open Research Computation: An ordinary journal with extraordinary aims. Retrieved 04 Nov 2012.

[12] Gaël Varoquaux's Front Page, 04 Jun 2012. A journal promoting high-quality research code: dream and reality. Retrieved 04 Nov 2012.

[13] The Journal of Open Research Software ; announced at software.ac.uk/blog/2012-03-23-announcing-journal-open-research-software-software-metajournal

4.9 Additional sources

- G. Hager and G. Wellein, Introduction to High Performance Computing for Scientists and Engineers, Chapman and Hall (2010)
- A.K. Hartmann, Practical Guide to Computer Simulations, World Scientific (2009)
- Journal Computational Methods in Science and Technology (open access), Polish Academy of Sciences
- Journal Computational Science and Discovery, Institute of Physics
- R.H. Landau, C.C. Bordeianu, and M. Jose Paez, A Survey of Computational Physics: Introductory Computational Science, Princeton University Press (2008)

4.10 External links

- John von Neumann-Institut for Computing (NIC) at Juelich (Germany)
- The National Center for Computational Science at Oak Ridge National Laboratory
- Educational Materials for Undergraduate Computational Studies
- Computational Science at the National Laboratories
- Bachelor in Computational Science, University of Medellin, Colombia, South America

Chapter 5

Exploratory data analysis

In statistics, **exploratory data analysis** (**EDA**) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA),[1] which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

5.1 Overview

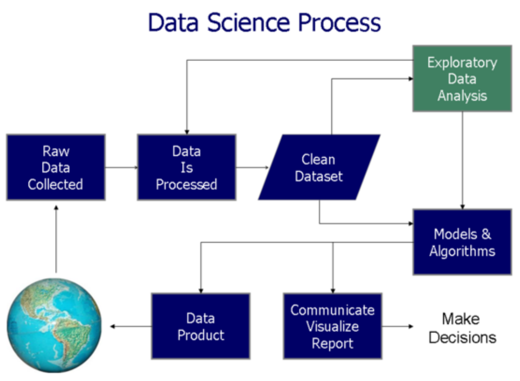
Tukey defined data analysis in 1961 as: "[P]rocedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data."[2]

Tukey's championing of EDA encouraged the development of statistical computing packages, especially *S* at Bell Labs. The *S* programming language inspired the systems 'S'-PLUS and *R*. This family of statistical-computing environments featured vastly improved dynamic visualization capabilities, which allowed statisticians to identify outliers, trends and patterns in data that merited further study.

Tukey's EDA was related to two other developments in statistical theory: Robust statistics and nonparametric statistics, both of which tried to reduce the sensitivity of statistical inferences to errors in formulating statistical models. Tukey promoted the use of five number summary of numerical data—the two extremes (maximum and minimum), the median, and the quartiles—because these median and quartiles, being functions of the empirical distribution are defined for all distributions, unlike the mean and standard deviation; moreover, the quartiles and median are more robust to skewed or heavy-tailed distributions than traditional summaries (the mean and standard deviation). The packages *S*, *S*-PLUS, and *R* included routines using resampling statistics, such as Quenouille and Tukey's jackknife and Efron 's bootstrap, which are nonparametric and robust (for many problems).

Exploratory data analysis, robust statistics, nonparametric statistics, and the development of statistical programming languages facilitated statisticians' work on scientific and engineering problems. Such problems included the fabrication of semiconductors and the understanding of communications networks, which concerned Bell Labs. These statistical developments, all championed by Tukey, were designed to complement the analytic theory of testing statistical hypotheses, particularly the Laplacian tradition's emphasis on exponential families.[3]

5.2 EDA development



Data science process flowchart

John W. Tukey wrote the book "Exploratory Data Analysis" in 1977.[4] Tukey held that too much emphasis in statistics was placed on statistical hypothesis testing (confirmatory data analysis); more emphasis needed to be placed on using data to suggest hypotheses to test. In particular, he held that confusing the two types of analyses and employing them on the same set of data can lead to systematic bias owing to the issues inherent in testing hypotheses suggested by the data.

The objectives of EDA are to:

- Suggest hypotheses about the causes of observed phenomena
- Assess assumptions on which statistical inference will be based
- Support the selection of appropriate statistical tools and techniques
- Provide a basis for further data collection through surveys or experiments[5]

Many **EDA** techniques have been adopted into data mining and are being taught to young students as a way to introduce them to statistical thinking.[6]

5.3 Techniques

There are a number of tools that are useful for EDA, but EDA is characterized more by the attitude taken than by particular techniques.[7]

Typical graphical techniques used in EDA are:

- Box plot
- Histogram
- Multi-vari chart
- Run chart
- Pareto chart
- Scatter plot
- Stem-and-leaf plot
- Parallel coordinates
- Odds ratio
- Multidimensional scaling
- Targeted projection pursuit
- Principal component analysis
- Multilinear PCA
- Projection methods such as grand tour, guided tour and manual tour
- Interactive versions of these plots

Typical quantitative techniques are:

- Median polish
- Trimean
- Ordination

5.4 History

Many EDA ideas can be traced back to earlier authors, for example:

- Francis Galton emphasized order statistics and quantiles.
- Arthur Lyon Bowley used precursors of the stemplot and five-number summary (Bowley actually used a "seven-figure summary", including the extremes, deciles and quartiles, along with the median - see his *Elementary Manual of Statistics* (3rd edn., 1920), p. 62 – he defines “the maximum and minimum, median, quartiles and two deciles” as the “seven positions”).
- Andrew Ehrenberg articulated a philosophy of data reduction (see his book of the same name).

The Open University course *Statistics in Society* (MDST 242), took the above ideas and merged them with Gottfried Noether's work, which introduced statistical inference via coin-tossing and the median test.

5.5 Example

Findings from EDA are often orthogonal to the primary analysis task. This is an example, described in more detail in.[8] The analysis task is to find the variables which best predict the tip that a dining party will give to the waiter. The variables available are tip, total bill, gender, smoking status, time of day, day of the week and size of the party. The analysis task requires that a regression model be fit with either tip or tip rate as the response variable. The fitted model is

tip rate = 0.18 - 0.01×size

which says that as the size of the dining party increase by one person tip will decrease by 1%. Making plots of the data reveals other interesting features not described by this model.

- Histogram of tips given by customers with bins equal to \$1 increments. Distribution of values is skewed right and unimodal, which says that there are few high tips, but lots of low tips.
- Histogram of tips given by customers with bins equal to 10c increments. An interesting phenomenon is visible, peaks in the counts at the full and half-dollar amounts. This corresponds to customers rounding tips. This is a behaviour that is common to other types of purchases too, like gasoline.
- Scatterplot of tips vs bill. We would expect to see a tight positive linear association, but instead see a lot more variation. In particular, there are more points

in the lower right than upper left. Points in the lower right correspond to tips that are lower than expected, and it is clear that more customers are cheap rather than generous.

- Scatterplot of tips vs bill separately by gender and smoking party. Smoking parties have a lot more variability in the tips that they give. Males tend to pay the (few) higher bills, and female non-smokers tend to be very consistent tippers (with the exception of three women).

What is learned from the graphics is different from what could be learned by the modeling. You can say that these pictures help the data tell us a story, that we have discovered some features of tipping that perhaps we didn't anticipate in advance.

5.6 Software

- GGobi is a free software for interactive data visualization data visualization
- CMU-DAP (Carnegie-Mellon University Data Analysis Package, FORTRAN source for EDA tools with English-style command syntax, 1977).
- Data Applied, a comprehensive web-based data visualization and data mining environment.
- High-D for multivariate analysis using parallel coordinates.
- JMP, an EDA package from SAS Institute.
- KNIME Konstanz Information Miner – Open-Source data exploration platform based on Eclipse.
- Orange, an open-source data mining software suite.
- SOCR provides a large number of free Internet-accessible.
- TinkerPlots (for upper elementary and middle school students).
- Weka an open source data mining package that includes visualisation and EDA tools such as targeted projection pursuit

5.7 See also

- Anscombe's quartet, on importance of exploration
- Predictive analytics
- Structured data analysis (statistics)
- Configural frequency analysis

5.8 References

[1] Chatfield, C. (1995). *Problem Solving: A Statistician's Guide* (2nd ed.). Chapman and Hall. ISBN 0412606305.

[2] John Tukey-The Future of Data Analysis-July 1961

[3] "Conversation with John W. Tukey and Elizabeth Tukey, Luisa T. Fernholz and Stephan Morgenthaler". *Statistical Science* **15** (1): 79–94. 2000. doi:10.1214/ss/1009212675.

[4] Tukey, John W. (1977). *Exploratory Data Analysis*. Pearson. ISBN 978-0201076165.

[5] Behrens-Principles and Procedures of Exploratory Data Analysis-American Psychological Association-1997

[6] Konold, C. (1999). "Statistics goes to school". *Contemporary Psychology* **44** (1): 81–82. doi:10.1037/001949.

[7] Tukey, John W. (1980). "We need both exploratory and confirmatory". *The American Statistician* **34** (1): 23–25. doi:10.1080/00031305.1980.10482706.

[8] Cook, D. and Swayne, D.F. (with A. Buja, D. Temple Lang, H. Hofmann, H. Wickham, M. Lawrence) (2007) ″Interactive and Dynamic Graphics for Data Analysis: With R and GGobi″ Springer, 978-0387717616

5.9 Bibliography

- Andrienko, N & Andrienko, G (2005) *Exploratory Analysis of Spatial and Temporal Data. A Systematic Approach*. Springer. ISBN 3-540-25994-5
- Cook, D. and Swayne, D.F. (with A. Buja, D. Temple Lang, H. Hofmann, H. Wickham, M. Lawrence). *Interactive and Dynamic Graphics for Data Analysis: With R and GGobi*. Springer. ISBN 9780387717616.
- Hoaglin, D C; Mosteller, F & Tukey, John Wilder (Eds) (1985). *Exploring Data Tables, Trends and Shapes*. ISBN 0-471-09776-4.
- Hoaglin, D C; Mosteller, F & Tukey, John Wilder (Eds) (1983). *Understanding Robust and Exploratory Data Analysis*. ISBN 0-471-09777-2.
- Inselberg, Alfred (2009). *Parallel Coordinates: Visual Multidimensional Geometry and its Applications*. London New York: Springer. ISBN 978-0-387-68628-8.
- Leinhardt, G., Leinhardt, S., *Exploratory Data Analysis: New Tools for the Analysis of Empirical Data*, Review of Research in Education, Vol. 8, 1980 (1980), pp. 85–157.
- Martinez, W. L., Martinez, A. R., and Solka, J. (2010). *Exploratory Data Analysis with MATLAB, second edition*. Chapman & Hall/CRC. ISBN 9781439812204.

- Theus, M., Urbanek, S. (2008), Interactive Graphics for Data Analysis: Principles and Examples, CRC Press, Boca Raton, FL, ISBN 978-1-58488-594-8
- Tucker, L; MacCallum, R. (1993). *Exploratory Factor Analysis*. .
- Tukey, John Wilder (1977). *Exploratory Data Analysis*. Addison-Wesley. ISBN 0-201-07616-0.
- Velleman, P. F.; Hoaglin, D. C. (1981). *Applications, Basics and Computing of Exploratory Data Analysis*. ISBN 0-87150-409-X.
- Young, F. W. Valero-Mora, P. and Friendly M. (2006) *Visual Statistics: Seeing your data with Dynamic Interactive Graphics*. Wiley ISBN 978-0-471-68160-1

5.10 External links

- Carnegie Mellon University – free online course on EDA

Chapter 6

Predictive analytics

Predictive analytics encompasses a variety of statistical techniques from modeling, machine learning, and data mining that analyze current and historical facts to make predictions about future, or otherwise unknown, events.[1][2]

In business, predictive models exploit patterns found in historical and transactional data to identify risks and opportunities. Models capture relationships among many factors to allow assessment of risk or potential associated with a particular set of conditions, guiding decision making for candidate transactions.[3]

The defining functional effect of these technical approaches is that predictive analytics provides a predictive score (probability) for each individual (customer, employee, healthcare patient, product SKU, vehicle, component, machine, or other organizational unit) in order to determine, inform, or influence organizational processes that pertain across large numbers of individuals, such as in marketing, credit risk assessment, fraud detection, manufacturing, healthcare, and government operations including law enforcement.

Predictive analytics is used in actuarial science,[4] marketing,[5] financial services,[6] insurance, telecommunications,[7] retail,[8] travel,[9] healthcare,[10] pharmaceuticals[11] and other fields.

One of the most well known applications is credit scoring,[1] which is used throughout financial services. Scoring models process a customer's credit history, loan application, customer data, etc., in order to rank-order individuals by their likelihood of making future credit payments on time.

6.1 Definition

Predictive analytics is an area of data mining that deals with extracting information from data and using it to predict trends and behavior patterns. Often the unknown event of interest is in the future, but predictive analytics can be applied to any type of unknown whether it be in the past, present or future. For example, identifying suspects after a crime has been committed, or credit card fraud as it occurs.[12] The core of predictive analytics relies on capturing relationships between explanatory variables and the predicted variables from past occurrences, and exploiting them to predict the unknown outcome. It is important to note, however, that the accuracy and usability of results will depend greatly on the level of data analysis and the quality of assumptions.

Predictive analytics is often defined as predicting at a more detailed level of granularity, i.e., generating predictive scores (probabilities) for each individual organizational element. This distinguishes it from forecasting. For example, "Predictive analytics—Technology that learns from experience (data) to predict the future behavior of individuals in order to drive better decisions."[13]

6.2 Types

Generally, the term predictive analytics is used to mean predictive modeling, "scoring" data with predictive models, and forecasting. However, people are increasingly using the term to refer to related analytical disciplines, such as descriptive modeling and decision modeling or optimization. These disciplines also involve rigorous data analysis, and are widely used in business for segmentation and decision making, but have different purposes and the statistical techniques underlying them vary.

6.2.1 Predictive models

Predictive models are models of the relation between the specific performance of a unit in a sample and one or more known attributes or features of the unit. The objective of the model is to assess the likelihood that a similar unit in a different sample will exhibit the specific performance. This category encompasses models in many areas, such as marketing, where they seek out subtle data patterns to answer questions about customer performance, or fraud detection models. Predictive models often perform calculations during live transactions, for example, to evaluate the risk or opportunity of a given customer or transaction, in order to guide a decision. With advancements in computing speed, individual agent modeling systems have become capable of simulating human behaviour or reactions to given stimuli or scenarios.

The available sample units with known attributes and known performances is referred to as the “training sample.” The units in other samples, with known attributes but unknown performances, are referred to as “out of [training] sample” units. The out of sample bear no chronological relation to the training sample units. For example, the training sample may consists of literary attributes of writings by Victorian authors, with known attribution, and the out-of sample unit may be newly found writing with unknown authorship; a predictive model may aid in attributing a work to a known author. Another example is given by analysis of blood splatter in simulated crime scenes in which the out of sample unit is the actual blood splatter pattern from a crime scene. The out of sample unit may be from the same time as the training units, from a previous time, or from a future time.

6.2.2 Descriptive models

Descriptive models quantify relationships in data in a way that is often used to classify customers or prospects into groups. Unlike predictive models that focus on predicting a single customer behavior (such as credit risk), descriptive models identify many different relationships between customers or products. Descriptive models do not rank-order customers by their likelihood of taking a particular action the way predictive models do. Instead, descriptive models can be used, for example, to categorize customers by their product preferences and life stage. Descriptive modeling tools can be utilized to develop further models that can simulate large number of individualized agents and make predictions.

6.2.3 Decision models

Decision models describe the relationship between all the elements of a decision — the known data (including results of predictive models), the decision, and the forecast results of the decision — in order to predict the results of decisions involving many variables. These models can be used in optimization, maximizing certain outcomes while minimizing others. Decision models are generally used to develop decision logic or a set of business rules that will produce the desired action for every customer or circumstance.

6.3 Applications

Although predictive analytics can be put to use in many applications, we outline a few examples where predictive analytics has shown positive impact in recent years.

6.3.1 Analytical customer relationship management (CRM)

Analytical Customer Relationship Management is a frequent commercial application of Predictive Analysis. Methods of predictive analysis are applied to customer data to pursue CRM objectives, which involve constructing a holistic view of the customer no matter where their information resides in the company or the department involved. CRM uses predictive analysis in applications for marketing campaigns, sales, and customer services to name a few. These tools are required in order for a company to posture and focus their efforts effectively across the breadth of their customer base. They must analyze and understand the products in demand or have the potential for high demand, predict customers’ buying habits in order to promote relevant products at multiple touch points, and proactively identify and mitigate issues that have the potential to lose customers or reduce their ability to gain new ones. Analytical Customer Relationship Management can be applied throughout the customers lifecycle (acquisition, relationship growth, retention, and win-back). Several of the application areas described below (direct marketing, cross-sell, customer retention) are part of Customer Relationship Managements.

6.3.2 Clinical decision support systems

Experts use predictive analysis in health care primarily to determine which patients are at risk of developing certain conditions, like diabetes, asthma, heart disease, and other lifetime illnesses. Additionally, sophisticated clinical decision support systems incorporate predictive analytics to support medical decision making at the point of care. A working definition has been proposed by Robert Hayward of the Centre for Health Evidence: “Clinical Decision Support Systems link health observations with health knowledge to influence health choices by clinicians for improved health care.”

6.3.3 Collection analytics

Many portfolios have a set of delinquent customers who do not make their payments on time. The financial institution has to undertake collection activities on these customers to recover the amounts due. A lot of collection resources are wasted on customers who are difficult or impossible to recover. Predictive analytics can help optimize the allocation of collection resources by identifying the most effective collection agencies, contact strategies, legal actions and other strategies to each customer, thus significantly increasing recovery at the same time reducing collection costs.

6.3.4 Cross-sell

Often corporate organizations collect and maintain abundant data (e.g. customer records, sale transactions) as exploiting hidden relationships in the data can provide a competitive advantage. For an organization that offers multiple products, predictive analytics can help analyze customers' spending, usage and other behavior, leading to efficient cross sales, or selling additional products to current customers.[2] This directly leads to higher profitability per customer and stronger customer relationships.

6.3.5 Customer retention

With the number of competing services available, businesses need to focus efforts on maintaining continuous consumer satisfaction, rewarding consumer loyalty and minimizing customer attrition. In addition, small increases in customer retention have been shown to increase profits disproportionately. One study concluded that a 5% increase in customer retention rates will increase profits by 25% to 95%.[14] Businesses tend to respond to customer attrition on a reactive basis, acting only after the customer has initiated the process to terminate service. At this stage, the chance of changing the customer's decision is almost impossible. Proper application of predictive analytics can lead to a more proactive retention strategy. By a frequent examination of a customer's past service usage, service performance, spending and other behavior patterns, predictive models can determine the likelihood of a customer terminating service sometime soon.[7] An intervention with lucrative offers can increase the chance of retaining the customer. Silent attrition, the behavior of a customer to slowly but steadily reduce usage, is another problem that many companies face. Predictive analytics can also predict this behavior, so that the company can take proper actions to increase customer activity.

6.3.6 Direct marketing

When marketing consumer products and services, there is the challenge of keeping up with competing products and consumer behavior. Apart from identifying prospects, predictive analytics can also help to identify the most effective combination of product versions, marketing material, communication channels and timing that should be used to target a given consumer. The goal of predictive analytics is typically to lower the cost per order or cost per action.

6.3.7 Fraud detection

Fraud is a big problem for many businesses and can be of various types: inaccurate credit applications, fraudulent transactions (both offline and online), identity thefts and false insurance claims. These problems plague firms of all sizes in many industries. Some examples of likely victims are credit card issuers, insurance companies,[15] retail merchants, manufacturers, business-to-business suppliers and even services providers. A predictive model can help weed out the "bads" and reduce a business's exposure to fraud.

Predictive modeling can also be used to identify high-risk fraud candidates in business or the public sector. Mark Nigrini developed a risk-scoring method to identify audit targets. He describes the use of this approach to detect fraud in the franchisee sales reports of an international fast-food chain. Each location is scored using 10 predictors. The 10 scores are then weighted to give one final overall risk score for each location. The same scoring approach was also used to identify high-risk check kiting accounts, potentially fraudulent travel agents, and questionable vendors. A reasonably complex model was used to identify fraudulent monthly reports submitted by divisional controllers.[16]

The Internal Revenue Service (IRS) of the United States also uses predictive analytics to mine tax returns and identify tax fraud.[15]

Recent advancements in technology have also introduced predictive behavior analysis for web fraud detection. This type of solution utilizes heuristics in order to study normal web user behavior and detect anomalies indicating fraud attempts.

6.3.8 Portfolio, product or economy-level prediction

Often the focus of analysis is not the consumer but the product, portfolio, firm, industry or even the economy. For example, a retailer might be interested in predicting store-level demand for inventory management purposes. Or the Federal Reserve Board might be interested in predicting the unemployment rate for the next year. These types of problems can be addressed by predictive analytics using time series techniques (see below). They can also be addressed via machine learning approaches which transform the original time series into a feature vector space, where the learning algorithm finds patterns that have predictive power.[17][18]

6.3.9 Risk management

When employing risk management techniques, the results are always to predict and benefit from a future scenario. The Capital asset pricing model (CAP-M) "predicts" the best portfolio to maximize return, Probabilistic Risk Assessment (PRA)--when combined with mini-Delphi Techniques and statistical approaches yields accurate forecasts and RiskAoA is a stand-alone predictive tool.[19] These are three examples of approaches that can extend from project to market, and from near to long term. Underwriting (see below) and other business approaches identify risk management as a predictive method.

6.3.10 Underwriting

Many businesses have to account for risk exposure due to their different services and determine the cost needed to cover the risk. For example, auto insurance providers need to accurately determine the amount of premium to charge to cover each automobile and driver. A financial company needs to assess a borrower's potential and ability to pay before granting a loan. For a health insurance provider, predictive analytics can analyze a few years of past medical claims data, as well as lab, pharmacy and other records where available, to predict how expensive an enrollee is likely to be in the future. Predictive analytics can help underwrite these quantities by predicting the chances of illness, default, bankruptcy, etc. Predictive analytics can streamline the process of customer acquisition by predicting the future risk behavior of a customer using application level data.[4] Predictive analytics in the form of credit scores have reduced the amount of time it takes for loan approvals, especially in the mortgage market where lending decisions are now made in a matter of hours rather than days or even weeks. Proper predictive analytics can lead to proper pricing decisions, which can help mitigate future risk of default.

6.4 Technology and big data influences

Big data is a collection of data sets that are so large and complex that they become awkward to work with using traditional database management tools. The volume, variety and velocity of big data have introduced challenges across the board for capture, storage, search, sharing, analysis, and visualization. Examples of big data sources include web logs, RFID, sensor data, social networks, Internet search indexing, call detail records, military surveillance, and complex data in astronomic, biogeochemical, genomics, and atmospheric sciences. Big Data is the core of most predictive analytic services offered by IT organizations.[20] Thanks to technological advances in computer hardware — faster CPUs, cheaper memory, and MPP architectures — and new technologies such as Hadoop, MapReduce, and in-database and text analytics for processing big data, it is now feasible to collect, analyze, and mine massive amounts of structured and unstructured data for new insights.[15] Today, exploring big data and using predictive analytics is within reach of more organizations than ever before and new methods that are capable for handling such datasets are proposed [21] [22]

6.5 Analytical Techniques

The approaches and techniques used to conduct predictive analytics can broadly be grouped into regression techniques and machine learning techniques.

6.5.1 Regression techniques

Regression models are the mainstay of predictive analytics. The focus lies on establishing a mathematical equation as a model to represent the interactions between the different variables in consideration. Depending on the situation, there are a wide variety of models that can be applied while performing predictive analytics. Some of them are briefly discussed below.

Linear regression model

The linear regression model analyzes the relationship between the response or dependent variable and a set of independent or predictor variables. This relationship is expressed as an equation that predicts the response variable as a linear function of the parameters. These parameters are adjusted so that a measure of fit is optimized. Much of the effort in model fitting is focused on minimizing the size of the residual, as well as ensuring that it is randomly distributed with respect to the model predictions.

The goal of regression is to select the parameters of the model so as to minimize the sum of the squared residuals. This is referred to as **ordinary least squares** (OLS) estimation and results in best linear unbiased estimates (BLUE) of the parameters if and only if the Gauss-Markov assumptions are satisfied.

Once the model has been estimated we would be interested to know if the predictor variables belong in the model – i.e. is the estimate of each variable's contribution reliable? To do this we can check the statistical significance of the model's coefficients which can be measured using the t-statistic. This amounts to testing whether the coefficient is significantly different from zero. How well the model predicts the dependent variable based on the value of the independent variables can be assessed by using the R^2 statistic. It measures predictive power of the model i.e. the proportion of the total variation in the dependent variable that is "explained" (accounted for) by variation in the independent variables.

Discrete choice models

Multivariate regression (above) is generally used when the response variable is continuous and has an unbounded range. Often the response variable may not be continuous but rather discrete. While mathematically it is feasible to apply multivariate regression to discrete ordered dependent variables, some of the assumptions behind the theory of multivariate linear regression no longer hold, and there are other techniques such as discrete choice models which are better suited for this type of analysis. If the dependent variable is discrete, some of those superior methods are logistic regression, multinomial logit and probit models. Logistic regression and probit models are used when the dependent variable is binary.

Logistic regression

For more details on this topic, see logistic regression.

In a classification setting, assigning outcome probabilities to observations can be achieved through the use of a logistic model, which is basically a method which transforms information about the binary dependent variable into an unbounded continuous variable and estimates a regular multivariate model (See Allison's Logistic Regression for more information on the theory of Logistic Regression).

The Wald and likelihood-ratio test are used to test the statistical significance of each coefficient *b* in the model (analogous to the t tests used in OLS regression; see above). A test assessing the goodness-of-fit of a classification model is the "percentage correctly predicted".

Multinomial logistic regression

An extension of the binary logit model to cases where the dependent variable has more than 2 categories is the multinomial logit model. In such cases collapsing the data into two categories might not make good sense or may lead to loss in the richness of the data. The multinomial logit model is the appropriate technique in these cases, especially when the dependent variable categories are not ordered (for examples colors like red, blue, green). Some authors have extended multinomial regression to include feature selection/importance methods such as Random multinomial logit.

Probit regression

Probit models offer an alternative to logistic regression for modeling categorical dependent variables. Even though the outcomes tend to be similar, the underlying distributions are different. Probit models are popular in social sciences like economics.

A good way to understand the key difference between probit and logit models is to assume that there is a latent variable z.

We do not observe z but instead observe y which takes the value 0 or 1. In the logit model we assume that y follows a logistic distribution. In the probit model we assume that y follows a standard normal distribution. Note that in social sciences (e.g. economics), probit is often used to model situations where the observed variable y is continuous but takes values between 0 and 1.

Logit versus probit

The Probit model has been around longer than the logit model. They behave similarly, except that the logistic distribution tends to be slightly flatter tailed. One of the reasons the logit model was formulated was that the probit model was computationally difficult due to the requirement of numerically calculating integrals. Modern computing however has made this computation fairly simple. The coefficients obtained from the logit and probit model are fairly close. However, the odds ratio is easier to interpret in the logit model.

Practical reasons for choosing the probit model over the logistic model would be:

- There is a strong belief that the underlying distribution is normal
- The actual event is not a binary outcome (*e.g.*, bankruptcy status) but a proportion (*e.g.*, proportion of population at different debt levels).

Time series models

Time series models are used for predicting or forecasting the future behavior of variables. These models account for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for. As a result standard regression techniques cannot be applied to time series data and methodology has been developed to decompose the trend, seasonal and cyclical component of the series. Modeling the dynamic path of a variable can improve forecasts since the predictable component of the series can be projected into the future.

Time series models estimate difference equations containing stochastic components. Two commonly used forms of these models are autoregressive models (AR) and moving average (MA) models. The Box-Jenkins methodology (1976) developed by George Box and G.M. Jenkins combines the AR and MA models to produce the ARMA (autoregressive moving average) model which is the cornerstone of stationary time series analysis. ARIMA (autoregressive integrated moving average models) on the other hand are used to describe non-stationary time series. Box and Jenkins suggest differencing a non stationary time series to obtain a stationary series to which an ARMA model can be applied. Non stationary time series have a pronounced trend and do not have a constant long-run mean or variance.

Box and Jenkins proposed a three stage methodology which includes: model identification, estimation and validation. The identification stage involves identifying if the series is stationary or not and the presence of seasonality by examining plots of the series, autocorrelation and partial autocorrelation functions. In the estimation stage, models are estimated using non-linear time series or maximum likelihood estimation procedures. Finally the validation stage involves diagnostic checking such as plotting the residuals to detect outliers and evidence of model fit.

In recent years time series models have become more sophisticated and attempt to model conditional heteroskedasticity with models such as ARCH (autoregressive conditional heteroskedasticity) and GARCH (generalized autoregressive conditional heteroskedasticity) models frequently used for financial time series. In addition time series models are also used to understand inter-relationships among economic variables represented by systems of equations using VAR (vector autoregression) and structural VAR models.

Survival or duration analysis

Survival analysis is another name for time to event analysis. These techniques were primarily developed in the medical and biological sciences, but they are also widely used in the social sciences like economics, as well as in engineering (reliability and failure time analysis).

Censoring and non-normality, which are characteristic of survival data, generate difficulty when trying to analyze the data using conventional statistical models such as multiple linear regression. The normal distribution, being a symmetric distribution, takes positive as well as negative values, but duration by its very nature cannot be negative and therefore normality cannot be assumed when dealing with duration/survival data. Hence the normality assumption of regression models is violated.

The assumption is that if the data were not censored it would be representative of the population of interest. In survival analysis, censored observations arise whenever the dependent variable of interest represents the time to a terminal event, and the duration of the study is limited in time.

An important concept in survival analysis is the hazard rate, defined as the probability that the event will occur at time t conditional on surviving until time t. Another concept related to the hazard rate is the survival function which can be defined as the probability of surviving to time t.

Most models try to model the hazard rate by choosing the underlying distribution depending on the shape of the hazard function. A distribution whose hazard function slopes upward is said to have positive duration dependence, a decreasing hazard shows negative duration dependence whereas constant hazard is a process with no memory usually characterized by the exponential distribution. Some of the distributional choices in survival models are: F, gamma, Weibull, log normal, inverse normal, exponential etc. All these distributions are for a non-negative random variable.

Duration models can be parametric, non-parametric or semi-parametric. Some of the models commonly used are Kaplan-Meier and Cox proportional hazard model (non parametric).

Classification and regression trees

Main article: decision tree learning

Globally-optimal classification tree analysis (GO-CTA) (also called hierarchical optimal discriminant analysis) is a generalization of optimal discriminant analysis that may be used to identify the statistical model that has maximum accuracy for predicting the value of a categorical dependent variable for a dataset consisting of categorical and continuous variables. The output of HODA is a non-orthogonal tree that combines categorical variables and cut points for continuous variables that yields maximum predictive accuracy, an assessment of the exact Type I error rate, and an evaluation of potential cross-generalizability of the statistical model. Hierarchical optimal discriminant analysis may be thought of as a generalization of Fisher's linear discriminant analysis. Optimal discriminant analysis is an alternative to ANOVA (analysis of variance) and regression analysis, which attempt to express one dependent variable as a linear combination of other features or measurements. However, ANOVA and regression analysis give a dependent variable that is a numerical variable, while hierarchical optimal discriminant analysis gives a dependent variable that is a class variable.

Classification and regression trees (CART) are a non-parametric decision tree learning technique that produces either classification or regression trees, depending on whether the dependent variable is categorical or numeric, respectively.

Decision trees are formed by a collection of rules based on variables in the modeling data set:

- Rules based on variables' values are selected to get the best split to differentiate observations based on the dependent variable
- Once a rule is selected and splits a node into two, the same process is applied to each "child" node (i.e. it is a recursive procedure)
- Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met. (Alternatively, the data are split as much as possible and then the tree is later pruned.)

Each branch of the tree ends in a terminal node. Each observation falls into one and exactly one terminal node, and each terminal node is uniquely defined by a set of rules.

A very popular method for predictive analytics is Leo Breiman's Random forests or derived versions of this technique like Random multinomial logit.

Multivariate adaptive regression splines

Multivariate adaptive regression splines (MARS) is a non-parametric technique that builds flexible models by fitting piecewise linear regressions.

An important concept associated with regression splines is that of a knot. Knot is where one local regression model gives way to another and thus is the point of intersection between two splines.

In multivariate and adaptive regression splines, basis functions are the tool used for generalizing the search for knots. Basis functions are a set of functions used to represent the information contained in one or more variables. Multivariate and Adaptive Regression Splines model almost always creates the basis functions in pairs.

Multivariate and adaptive regression spline approach deliberately overfits the model and then prunes to get to the optimal model. The algorithm is computationally very intensive and in practice we are required to specify an upper limit on the number of basis functions.

6.5.2 Machine learning techniques

Machine learning, a branch of artificial intelligence, was originally employed to develop techniques to enable computers to learn. Today, since it includes a number of advanced statistical methods for regression and classification, it finds application in a wide variety of fields including medical diagnostics, credit card fraud detection, face and speech recognition and analysis of the stock market. In certain applications it is sufficient to directly predict the dependent variable without focusing on the underlying relationships between variables. In other cases, the underlying relationships can be very complex and the mathematical form of the dependencies unknown. For such cases, machine learning techniques emulate human cognition and learn from training examples to predict future events.

A brief discussion of some of these methods used commonly for predictive analytics is provided below. A detailed study of machine learning can be found in Mitchell (1997).

Neural networks

Neural networks are nonlinear sophisticated modeling techniques that are able to model complex functions. They can be applied to problems of prediction, classification or control in a wide spectrum of fields such as finance, cognitive psychology/neuroscience, medicine, engineering, and physics.

Neural networks are used when the exact nature of the relationship between inputs and output is not known. A key feature of neural networks is that they learn the relationship between inputs and output through training. There are three types of training in neural networks used by different networks, supervised and unsupervised training, reinforcement learning, with supervised being the most common one.

Some examples of neural network training techniques are backpropagation, quick propagation, conjugate gradient descent, projection operator, Delta-Bar-Delta etc. Some unsupervised network architectures are multilayer perceptrons, Kohonen networks, Hopfield networks, etc.

Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) consists of an input and an output layer with one or more hidden layers of nonlinearly-activating nodes or sigmoid nodes. This is determined by the weight vector and it is necessary to adjust the weights of the network. The backpropagation employs gradient fall to minimize the squared error between the network output values and desired values for those outputs. The weights adjusted by an iterative process of repetitive present of attributes. Small changes in the weight to get the desired values are done by the process called training the net and is done by the training set (learning rule).

Radial basis functions

A radial basis function (RBF) is a function which has built into it a distance criterion with respect to a center. Such functions can be used very efficiently for interpolation and for smoothing of data. Radial basis functions have been applied in the area of neural networks where they are used as a replacement for the sigmoidal transfer function. Such networks have 3 layers, the input layer, the hidden layer with the RBF non-linearity and a linear output layer. The most popular choice for the non-linearity is the Gaussian. RBF networks have the advantage of not being locked into local minima as do the feed-forward networks such as the multilayer perceptron.

Support vector machines

Support Vector Machines (SVM) are used to detect and exploit complex patterns in data by clustering, classifying and ranking the data. They are learning machines that are used to perform binary classifications and regression estimations. They commonly use kernel based methods to apply linear classification techniques to non-linear classification problems. There are a number of types of SVM such as linear, polynomial, sigmoid etc.

Naïve Bayes

Naïve Bayes based on Bayes conditional probability rule is used for performing classification tasks. Naïve Bayes assumes the predictors are statistically independent which makes it an effective classification tool that is easy to interpret. It is best employed when faced with the problem of 'curse of dimensionality' i.e. when the number of predictors is very high.

k-nearest neighbours

The nearest neighbour algorithm (KNN) belongs to the class of pattern recognition statistical methods. The method does not impose a priori any assumptions about the distribution from which the modeling sample is drawn. It involves a training set with both positive and negative values. A new sample is classified by calculating the distance to the nearest neighbouring training case. The sign of that point will determine the classification of the sample. In the k-nearest neighbour classifier, the k nearest points are considered and the sign of the majority is used to classify the sample. The performance of the kNN algorithm is influenced by three main factors: (1) the distance measure used to locate the nearest neighbours; (2) the decision rule used to derive a classification from the k-nearest neighbours; and (3) the number of neighbours used to classify the new sample. It can be proved that, unlike other methods, this method is universally asymptotically convergent, i.e.: as the size of the training set increases, if the observations are independent and identically distributed (i.i.d.), regardless of the distribution from which the sample is drawn, the predicted class will converge to the class assignment that minimizes misclassification error. See Devroy et al.

Geospatial predictive modeling

Conceptually, geospatial predictive modeling is rooted in the principle that the occurrences of events being modeled are limited in distribution. Occurrences of events are neither uniform nor random in distribution – there are spatial environment factors (infrastructure, sociocultural, topographic, etc.) that constrain and influence where the locations of events occur. Geospatial predictive modeling attempts to describe those constraints and influences by spatially correlating occurrences of historical geospatial locations with environmental factors that represent those constraints and influences. Geospatial predictive modeling is a process for analyzing events through a geographic filter in order to make statements of likelihood for event occurrence or emergence.

6.6 Tools

Historically, using predictive analytics tools—as well as understanding the results they delivered—required advanced skills. However, modern predictive analytics tools are no longer restricted to IT specialists. As more organizations adopt predictive analytics into decision-making processes and integrate it into their operations, they are creating a shift in the market toward business users as the primary consumers of the information. Business users want tools they can use on their own. Vendors are responding by creating new software that removes the mathematical complexity, provides user-friendly graphic interfaces and/or builds in short cuts that can, for example, recognize the kind of data available and suggest an appropriate predictive model.[23] Predictive analytics tools have become sophisticated enough to adequately present and dissect data problems, so that any data-savvy information worker can utilize them to analyze data and retrieve meaningful, useful results.[2] For example, modern tools present findings using simple charts, graphs, and scores that indicate the likelihood of possible outcomes.[24]

There are numerous tools available in the marketplace that help with the execution of predictive analytics. These range from those that need very little user sophistication to those that are designed for the expert practitioner. The difference between these tools is often in the level of customization and heavy data lifting allowed.

Notable open source predictive analytic tools include:

- scikit-learn
- KNIME
- OpenNN
- Orange
- R
- Weka
- GNU Octave
- Apache Mahout

Notable commercial predictive analytic tools include:

- Alpine Data Labs
- BIRT Analytics
- Angoss KnowledgeSTUDIO
- IBM SPSS Statistics and IBM SPSS Modeler
- KXEN Modeler
- Mathematica
- MATLAB

- Minitab
- Neural Designer
- Oracle Data Mining (ODM)
- Pervasive
- Predixion Software
- RapidMiner
- RCASE
- Revolution Analytics
- SAP
- SAS and SAS Enterprise Miner
- STATA
- STATISTICA
- TIBCO

The most popular commercial predictive analytics software packages according to the Rexer Analytics Survey for 2013 are IBM SPSS Modeler, SAS Enterprise Miner, and Dell Statistica <http://www.rexeranalytics.com/Data-Miner-Survey-2013-Intro.html>

6.6.1 PMML

In an attempt to provide a standard language for expressing predictive models, the Predictive Model Markup Language (PMML) has been proposed. Such an XML-based language provides a way for the different tools to define predictive models and to share these between PMML compliant applications. PMML 4.0 was released in June, 2009.

6.7 Criticism

There are plenty of skeptics when it comes to computers and algorithms abilities to predict the future, including Gary King, a professor from Harvard University and the director of the Institute for Quantitative Social Science. [25] People are influenced by their environment in innumerable ways. Trying to understand what people will do next assumes that all the influential variables can be known and measured accurately. "People's environments change even more quickly than they themselves do. Everything from the weather to their relationship with their mother can change the way people think and act. All of those variables are unpredictable. How they will impact a person is even less predictable. If put in the exact same situation tomorrow, they may make a completely different decision. This means that a statistical prediction is only valid in sterile laboratory conditions, which suddenly isn't as useful as it seemed before." [26]

6.8 See also

- Criminal Reduction Utilising Statistical History
- Data mining
- Learning analytics
- Odds algorithm
- Pattern recognition
- Prescriptive analytics
- Predictive modeling
- RiskAoA a predictive tool for discriminating future decisions.

6.9 References

[1] Nyce, Charles (2007), *Predictive Analytics White Paper* (PDF), American Institute for Chartered Property Casualty Underwriters/Insurance Institute of America, p. 1

[2] Eckerson, Wayne (May 10, 2007), *Extending the Value of Your Data Warehousing Investment*, The Data Warehouse Institute

[3] Coker, Frank (2014). *Pulse: Understanding the Vital Signs of Your Business* (1st ed.). Bellevue, WA: Ambient Light Publishing. pp. 30, 39, 42,more. ISBN 978-0-9893086-0-1.

[4] Conz, Nathan (September 2, 2008), "Insurers Shift to Customer-focused Predictive Analytics Technologies", *Insurance & Technology*

[5] Fletcher, Heather (March 2, 2011), "The 7 Best Uses for Predictive Analytics in Multichannel Marketing", *Target Marketing*

[6] Korn, Sue (April 21, 2011), "The Opportunity for Predictive Analytics in Finance", *HPC Wire*

[7] Barkin, Eric (May 2011), "CRM + Predictive Analytics: Why It All Adds Up", *Destination CRM*

[8] Das, Krantik; Vidyashankar, G.S. (July 1, 2006), "Competitive Advantage in Retail Through Analytics: Developing Insights, Creating Value", *Information Management*

[9] McDonald, Michèle (September 2, 2010), "New Technology Taps 'Predictive Analytics' to Target Travel Recommendations", *Travel Market Report*

[10] Stevenson, Erin (December 16, 2011), "Tech Beat: Can you pronounce health care predictive analytics?", *Times-Standard*

[11] McKay, Lauren (August 2009), "The New Prescription for Pharma", *Destination CRM*

[12] Finlay, Steven (2014). *Predictive Analytics, Data Mining and Big Data. Myths, Misconceptions and Methods* (1st ed.). Basingstoke: Palgrave Macmillan. p. 237. ISBN 1137379278.

[13] Siegel, Eric (2013). *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die* (1st ed.). Wiley. ISBN 978-1-1183-5685-2.

[14] Reichheld, Frederick; Schefter, Phil. "The Economics of E-Loyalty". *http://hbswk.hbs.edu/". Havard Business School. Retrieved 10 November 2014.*

[15] Schiff, Mike (March 6, 2012), *BI Experts: Why Predictive Analytics Will Continue to Grow*, The Data Warehouse Institute

[16] Nigrini, Mark (June 2011). "Forensic Analytics: Methods and Techniques for Forensic Accounting Investigations". Hoboken, NJ: John Wiley & Sons Inc. ISBN 978-0-470-89046-2.

[17] Dhar, Vasant (April 2011). "Prediction in Financial Markets: The Case for Small Disjuncts". *ACM Transactions on Intelligent Systems and Technologies* **2** (3).

[18] Dhar, Vasant; Chou, Dashin and Provost Foster (October 2000). "Discovering Interesting Patterns in Investment Decision Making with GLOWER – A Genetic Learning Algorithm Overlaid With Entropy Reduction". *Data Mining and Knowledge Discovery* **4** (4).

[19] https://acc.dau.mil/CommunityBrowser.aspx?id=126070

[20] http://www.hcltech.com/sites/default/files/key_to_monetizing_big_data_via_predictive_analytics.pdf

[21] Ben-Gal I. Dana A., Shkolnik N. and Singer (2014). "Efficient Construction of Decision Trees by the Dual Information Distance Method" (PDF). Quality Technology & Quantitative Management (QTQM), 11(1), 133-147.

[22] Ben-Gal I., Shavitt Y., Weinsberg E., Weinsberg U. (2014). "Peer-to-peer information retrieval using shared-content clustering" (PDF). Knowl Inf Syst DOI 10.1007/s10115-013-0619-9.

[23] Halper, Fern (November 1, 2011), "The Top 5 Trends in Predictive Analytics", *Information Management*

[24] MacLennan, Jamie (May 1, 2012), *5 Myths about Predictive Analytics*, The Data Warehouse Institute

[25] Temple-Raston, Dina (Oct 8, 2012), *Predicting The Future: Fantasy Or A Good Algorithm?*, NPR

[26] Alverson, Cameron (Sep 2012), *Polling and Statistical Models Can't Predict the Future*, Cameron Alverson

6.10 Further reading

- Agresti, Alan (2002). *Categorical Data Analysis*. Hoboken: John Wiley and Sons. ISBN 0-471-36093-7.
- Coggeshall, Stephen, Davies, John, Jones, Roger., and Schutzer, Daniel, "Intelligent Security Systems," in Freedman, Roy S., Flein, Robert A., and Lederman, Jess, Editors (1995). *Artificial Intelligence in the Capital Markets*. Chicago: Irwin. ISBN 1-55738-811-3.
- L. Devroye, L. Györfi, G. Lugosi (1996). *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag.
- Enders, Walter (2004). *Applied Time Series Econometrics*. Hoboken: John Wiley and Sons. ISBN 0-521-83919-X.
- Greene, William (2012). *Econometric Analysis, 7th Ed*. London: Prentice Hall. ISBN 978-0-13-139538-1.
- Guidère, Mathieu; Howard N, Sh. Argamon (2009). *Rich Language Analysis for Counterterrrorism*. Berlin, London, New York: Springer-Verlag. ISBN 978-3-642-01140-5.
- Mitchell, Tom (1997). *Machine Learning*. New York: McGraw-Hill. ISBN 0-07-042807-7.
- Siegel, Eric (2013). *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die*. John Wiley. ISBN 978-1-1183-5685-2.
- Tukey, John (1977). *Exploratory Data Analysis*. New York: Addison-Wesley. ISBN 0-201-07616-0.
- Finlay, Steven (2014). *Predictive Analytics, Data Mining and Big Data. Myths, Misconceptions and Methods*. Basingstoke: Palgrave Macmillan. ISBN 978-1-137-37927-6.
- Coker, Frank (2014). *Pulse: Understanding the Vital Signs of Your Business*. Bellevue, WA: Ambient Light Publishing. ISBN 978-0-9893086-0-1.

Chapter 7

Business intelligence

Business intelligence (**BI**) is the set of techniques and tools for the transformation of raw data into meaningful and useful information for business analysis purposes. BI technologies are capable of handling large amounts of unstructured data to help identify, develop and otherwise create new strategic business opportunities. The goal of BI is to allow for the easy interpretation of these large volumes of data. Identifying new opportunities and implementing an effective strategy based on insights can provide businesses with a competitive market advantage and long-term stability.[1]

BI technologies provide historical, current and predictive views of business operations. Common functions of business intelligence technologies are reporting, online analytical processing, analytics, data mining, process mining, complex event processing, business performance management, benchmarking, text mining, predictive analytics and prescriptive analytics.

BI can be used to support a wide range of business decisions ranging from operational to strategic. Basic operating decisions include product positioning or pricing. Strategic business decisions include priorities, goals and directions at the broadest level. In all cases, BI is most effective when it combines data derived from the market in which a company operates (external data) with data from company sources internal to the business such as financial and operations data (internal data). When combined, external and internal data can provide a more complete picture which, in effect, creates an "intelligence" that cannot be derived by any singular set of data.[2]

7.1 Components

Business intelligence is made up of an increasing number of components including:

- Multidimensional aggregation and allocation
- Denormalization, tagging and standardization
- Realtime reporting with analytical alert
- A method of interfacing with unstructured data sources
- Group consolidation, budgeting and rolling forecasts
- Statistical inference and probabilistic simulation
- Key performance indicators optimization
- Version control and process management
- Open item management

7.2 History

The term "Business Intelligence" was originally coined by Richard Millar Devens' in the 'Cyclopædia of Commercial and Business Anecdotes' from 1865. Devens used the term to describe how the banker, Sir Henry Furnese, gained profit by receiving and acting upon information about his environment, prior to his competitors. "*Throughout Holland, Flanders, France, and Germany, he maintained a complete and perfect train of business intelligence. The news of the many battles fought was thus received first by him, and the fall of Namur added to his profits, owing to his early receipt of the news.*" (Devens, (1865), p. 210). The ability to collect and react accordingly based on the information retrieved, an ability that Furnese excelled in, is today still at the very heart of BI.[3]

In a 1958 article, IBM researcher Hans Peter Luhn used the term business intelligence. He employed the Webster's dictionary definition of intelligence: "the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal."[4]

Business intelligence as it is understood today is said to have evolved from the decision support systems (DSS) that began in the 1960s and developed throughout the mid-1980s. DSS originated in the computer-aided models created to assist with decision making and planning. From DSS, data warehouses, Executive Information Systems, OLAP and business intelligence came into focus beginning in the late 80s.

In 1988, an Italian-Dutch-French-English consortium organized an international meeting on the Multiway Data Analysis in Rome.[5] The ultimate goal is to reduce the multiple dimensions down to one or two (by detecting the patterns within the data) that can then be presented to human decision-makers.

In 1989, Howard Dresner (later a Gartner Group analyst) proposed “business intelligence” as an umbrella term to describe “concepts and methods to improve business decision making by using fact-based support systems.”[6] It was not until the late 1990s that this usage was widespread.[7]

7.3 Data warehousing

Often BI applications use data gathered from a data warehouse (DW) or from a data mart, and the concepts of BI and DW sometimes combine as "**BI/DW**"[8] or as "**BIDW**". A data warehouse contains a copy of analytical data that facilitates decision support. However, not all data warehouses serve for business intelligence, nor do all business intelligence applications require a data warehouse.

To distinguish between the concepts of business intelligence and data warehouses, Forrester Research defines business intelligence in one of two ways:

1. Using a broad definition: “Business Intelligence is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making.”[9] Under this definition, business intelligence also includes technologies such as data integration, data quality, data warehousing, master-data management, text- and content-analytics, and many others that the market sometimes lumps into the "Information Management" segment. Therefore, Forrester refers to *data preparation* and *data usage* as two separate but closely linked segments of the business-intelligence architectural stack.

2. Forrester defines the narrower business-intelligence market as, "...referring to just the top layers of the BI architectural stack such as reporting, analytics and dashboards.”[10]

7.4 Comparison with competitive intelligence

Though the term business intelligence is sometimes a synonym for competitive intelligence (because they both support decision making), BI uses technologies, processes, and applications to analyze mostly internal, structured data and business processes while competitive intelligence gathers, analyzes and disseminates information with a topical focus on company competitors. If understood broadly, business intelligence can include the subset of competitive intelligence.[11]

7.5 Comparison with business analytics

Business intelligence and business analytics are sometimes used interchangeably, but there are alternate definitions.[12] One definition contrasts the two, stating that the term business intelligence refers to collecting business data to find information primarily through asking questions, reporting, and online analytical processes. Business analytics, on the other hand, uses statistical and quantitative tools for explanatory and predictive modeling.[13]

In an alternate definition, Thomas Davenport, professor of information technology and management at Babson College argues that business intelligence should be divided into querying, reporting, Online analytical processing (OLAP), an “alerts” tool, and business analytics. In this definition, business analytics is the subset of BI focusing on statistics, prediction, and optimization, rather than the reporting functionality.[14]

7.6 Applications in an enterprise

Business intelligence can be applied to the following business purposes, in order to drive business value.

1. Measurement – program that creates a hierarchy of performance metrics (see also Metrics Reference Model) and benchmarking that informs business leaders about progress towards business goals (business process management).

2. Analytics – program that builds quantitative processes for a business to arrive at optimal decisions and to perform business knowledge discovery. Frequently involves: data mining, process mining, statistical analysis, predictive analytics, predictive modeling, business process modeling, data lineage, complex event processing and prescriptive analytics.

3. Reporting/enterprise reporting – program that builds infrastructure for strategic reporting to serve the strategic management of a business, not operational reporting. Frequently involves data visualization, executive information system and OLAP.

4. Collaboration/collaboration platform – program that gets different areas (both inside and outside the business) to work together through data sharing and electronic data interchange.

5. Knowledge management – program to make the company data-driven through strategies and practices to identify, create, represent, distribute, and enable adoption of insights and experiences that are true business knowledge. Knowledge management leads to learning management and regulatory compliance.

In addition to the above, business intelligence can provide a pro-active approach, such as alert functionality that immediately notifies the end-user if certain conditions are met. For example, if some business metric exceeds a pre-defined threshold, the metric will be highlighted in standard reports, and the business analyst may be alerted via e-mail or another monitoring service. This end-to-end process requires data governance, which should be handled by the expert.

7.7 Prioritization of projects

It can be difficult to provide a positive business case for business intelligence initiatives, and often the projects must be prioritized through strategic initiatives. BI projects can attain higher prioritization within the organization if managers consider the following:

- As described by Kimball[15] the BI manager must determine the tangible benefits such as eliminated cost of producing legacy reports.
- Data access for the entire organization must be enforced.[16] In this way even a small benefit, such as a few minutes saved, makes a difference when multiplied by the number of employees in the entire organization.
- As described by Ross, Weil & Roberson for Enterprise Architecture,[17] managers should also consider letting the BI project be driven by other business initiatives with excellent business cases. To support this approach, the organization must have enterprise architects who can identify suitable business projects.
- Using a structured and quantitative methodology to create defensible prioritization in line with the actual needs of the organization, such as a weighted decision matrix.[18]

7.8 Success factors of implementation

According to Kimball et al., there are three critical areas that organizations should assess before getting ready to do a BI project:[19]

1. The level of commitment and sponsorship of the project from senior management
2. The level of business need for creating a BI implementation
3. The amount and quality of business data available.

7.8.1 Business sponsorship

The commitment and sponsorship of senior management is according to Kimball *et al.*, the most important criteria for assessment.[20] This is because having strong management backing helps overcome shortcomings elsewhere in the project. However, as Kimball *et al.* state: “even the most elegantly designed DW/BI system cannot overcome a lack of business [management] sponsorship”.[21]

It is important that personnel who participate in the project have a vision and an idea of the benefits and drawbacks of implementing a BI system. The best business sponsor should have organizational clout and should be well connected within the organization. It is ideal that the business sponsor is demanding but also able to be realistic and supportive if the implementation runs into delays or drawbacks. The management sponsor also needs to be able to assume accountability and to take responsibility for failures and setbacks on the project. Support from multiple members of the management ensures the project does not fail if one person leaves the steering group. However, having many managers work together on the project can also mean that there are several different interests that attempt to pull the project in different directions, such as if different departments want to put more emphasis on their usage. This issue can be countered by an early and specific analysis of the business areas that benefit the most from the implementation. All stakeholders in the project should participate in this analysis in order for them to feel invested in the project and to find common ground.

Another management problem that may be encountered before the start of an implementation is an overly aggressive business sponsor. Problems of scope creep occur when the sponsor requests data sets that were not specified in the original planning phase.

7.8.2 Business needs

Because of the close relationship with senior management, another critical thing that must be assessed before the project begins is whether or not there is a business need and whether there is a clear business benefit by doing the implementation.[22] The needs and benefits of the implementation are sometimes driven by competition and the need to gain an advantage in the market. Another reason for a business-driven approach to implementation of BI is the acquisition of other organizations that enlarge the original organization it can sometimes be beneficial to implement DW or BI in order to create more oversight.

Companies that implement BI are often large, multinational organizations with diverse subsidiaries.[23] A well-designed BI solution provides a consolidated view of key business data not available anywhere else in the organization, giving management visibility and control over measures that otherwise would not exist.

7.8.3 Amount and quality of available data

Without proper data, or with too little quality data, any BI implementation fails; it does not matter how good the management sponsorship or business-driven motivation is. Before implementation it is a good idea to do data profiling. This analysis identifies the "content, consistency and structure [..]"[22] of the data. This should be done as early as possible in the process and if the analysis shows that data is lacking, put the project on hold temporarily while the IT department figures out how to properly collect data.

When planning for business data and business intelligence requirements, it is always advisable to consider specific scenarios that apply to a particular organization, and then select the business intelligence features best suited for the scenario.

Often, scenarios revolve around distinct business processes, each built on one or more data sources. These sources are used by features that present that data as information to knowledge workers, who subsequently act on that information. The business needs of the organization for each business process adopted correspond to the essential steps of business intelligence. These essential steps of business intelligence include but are not limited to:

1. Go through business data sources in order to collect needed data
2. Convert business data to information and present appropriately
3. Query and analyze data
4. Act on the collected data

The **quality aspect** in business intelligence should cover all the process from the source data to the final reporting. At each step, the **quality gates** are different:

1. Source Data:
   - Data Standardization: make data comparable (same unit, same pattern...)
   - Master Data Management: unique referential
2. Operational Data Store (ODS):
   - Data Cleansing: detect & correct inaccurate data
   - Data Profiling: check inappropriate value, null/empty
3. Data warehouse:
   - Completeness: check that all expected data are loaded
   - Referential integrity: unique and existing referential over all sources
   - Consistency between sources: check consolidated data vs sources
4. Reporting:
   - Uniqueness of indicators: only one share dictionary of indicators
   - Formula accuracy: local reporting formula should be avoided or checked

7.9 User aspect

Some considerations must be made in order to successfully integrate the usage of business intelligence systems in a company. Ultimately the BI system must be accepted and utilized by the users in order for it to add value to the organization.[24][25] If the usability of the system is poor, the users may become frustrated and spend a considerable amount of time figuring out how to use the system or may not be able to really use the system. If the system does not add value to the users´ mission, they simply don't use it.[25]

To increase user acceptance of a BI system, it can be advisable to consult business users at an early stage of the DW/BI lifecycle, for example at the requirements gathering phase.[24] This can provide an insight into the business process and what the users need from the BI system. There are several methods for gathering this information, such as questionnaires and interview sessions.

When gathering the requirements from the business users, the local IT department should also be consulted in order to determine to which degree it is possible to fulfill the business's needs based on the available data.[24]

Taking a user-centered approach throughout the design and development stage may further increase the chance of rapid user adoption of the BI system.[25]

Besides focusing on the user experience offered by the BI applications, it may also possibly motivate the users to utilize the system by adding an element of competition. Kimball[24] suggests implementing a function on the Business Intelligence portal website where reports on system usage can be found. By doing so, managers can see how well their departments are doing and compare themselves to others and this may spur them to encourage their staff to utilize the BI system even more.

In a 2007 article, H. J. Watson gives an example of how the competitive element can act as an incentive.[26] Watson describes how a large call centre implemented performance dashboards for all call agents, with monthly incentive bonuses tied to performance metrics. Also, agents could compare their performance to other team members. The implementation of this type of performance measurement and competition significantly improved agent performance.

BI chances of success can be improved by involving senior management to help make BI a part of the organizational culture, and by providing the users with necessary tools, training, and support.[26] Training encourages more people to use the BI application.[24]

Providing user support is necessary to maintain the BI system and resolve user problems.[25] User support can be incorporated in many ways, for example by creating a website. The website should contain great content and tools for finding the necessary information. Furthermore, helpdesk support can be used. The help desk can be manned by power users or the DW/BI project team.[24]

7.10 BI Portals

A **Business Intelligence portal** (BI portal) is the primary access interface for Data Warehouse (DW) and Business Intelligence (BI) applications. The BI portal is the user's first impression of the DW/BI system. It is typically a browser application, from which the user has access to all the individual services of the DW/BI system, reports and other analytical functionality. The BI portal must be implemented in such a way that it is easy for the users of the DW/BI application to call on the functionality of the application.[27]

The BI portal's main functionality is to provide a navigation system of the DW/BI application. This means that the portal has to be implemented in a way that the user has access to all the functions of the DW/BI application.

The most common way to design the portal is to custom fit it to the business processes of the organization for which the DW/BI application is designed, in that way the portal can best fit the needs and requirements of its users.[28]

The BI portal needs to be easy to use and understand, and if possible have a look and feel similar to other applications or web content of the organization the DW/BI application is designed for (consistency).

The following is a list of desirable features for web portals in general and BI portals in particular:

Usable User should easily find what they need in the BI tool.

Content Rich The portal is not just a report printing tool, it should contain more functionality such as advice, help, support information and documentation.

Clean The portal should be designed so it is easily understandable and not over complex as to confuse the users

Current The portal should be updated regularly.

Interactive The portal should be implemented in a way that makes it easy for the user to use its functionality and encourage them to use the portal. Scalability and customization give the user the means to fit the portal to each user.

Value Oriented It is important that the user has the feeling that the DW/BI application is a valuable resource that is worth working on.

7.11 Marketplace

There are a number of business intelligence vendors, often categorized into the remaining independent "pure-play" vendors and consolidated "megavendors" that have entered the market through a recent trend[29] of acquisitions in the BI industry.[30] The business intelligence market is gradually growing. In 2012 business intelligence services brought in \$13.1 billion in revenue.[31]

Some companies adopting BI software decide to pick and choose from different product offerings (best-of-breed) rather than purchase one comprehensive integrated solution (full-service).[32]

7.11.1 Industry-specific

Specific considerations for business intelligence systems have to be taken in some sectors such as governmental banking regulations. The information collected by banking institutions and analyzed with BI software must be protected from some groups or individuals, while being fully available to other groups or individuals. Therefore, BI solutions must be sensitive to those needs and be flexible enough to adapt to new regulations and changes to existing law.

7.12 Semi-structured or unstructured data

Businesses create a huge amount of valuable information in the form of e-mails, memos, notes from call-centers, news, user groups, chats, reports, web-pages, presentations, image-files, video-files, and marketing material and news. According to Merrill Lynch, more than 85% of all business information exists in these forms. These information types are called either *semi-structured* or *unstructured* data. However, organizations often only use these documents once.[33]

The management of semi-structured data is recognized as a major unsolved problem in the information technology industry.[34] According to projections from Gartner (2003), white collar workers spend anywhere from 30 to 40 percent of their time searching, finding and assessing unstructured data. BI uses both structured and unstructured data, but the former is easy to search, and the latter contains a large quantity of the information needed for analysis and decision making.[34][35] Because of the difficulty of properly searching, finding and assessing unstructured or semi-structured data, organizations may not draw upon these vast reservoirs of information, which could influence a particular decision, task or project. This can ultimately lead to poorly informed decision making.[33]

Therefore, when designing a business intelligence/DW-solution, the specific problems associated with semi-structured and unstructured data must be accommodated for as well as those for the structured data.[35]

7.12.1 Unstructured data vs. semi-structured data

Unstructured and semi-structured data have different meanings depending on their context. In the context of relational database systems, unstructured data cannot be stored in predictably ordered columns and rows. One type of unstructured data is typically stored in a BLOB (binary large object), a catch-all data type available in most relational database management systems. Unstructured data may also refer to irregularly or randomly repeated column patterns that vary from row to row within each file or document.

Many of these data types, however, like e-mails, word processing text files, PPTs, image-files, and video-files conform to a standard that offers the possibility of metadata. Metadata can include information such as author and time of creation, and this can be stored in a relational database. Therefore, it may be more accurate to talk about this as semi-structured documents or data,[34] but no specific consensus seems to have been reached.

Unstructured data can also simply be the knowledge that business users have about future business trends. Business forecasting naturally aligns with the BI system because business users think of their business in aggregate terms. Capturing the business knowledge that may only exist in the minds of business users provides some of the most important data points for a complete BI solution.

7.12.2 Problems with semi-structured or unstructured data

There are several challenges to developing BI with semi-structured data. According to Inmon & Nesavich,[36] some of those are:

1. Physically accessing unstructured textual data – unstructured data is stored in a huge variety of formats.
2. Terminology – Among researchers and analysts, there is a need to develop a standardized terminology.
3. Volume of data – As stated earlier, up to 85% of all data exists as semi-structured data. Couple that with the need for word-to-word and semantic analysis.
4. Searchability of unstructured textual data – A simple search on some data, e.g. apple, results in links where there is a reference to that precise search term. (Inmon & Nesavich, 2008)[36] gives an example: “a search is made on the term felony. In a simple search, the term felony is used, and everywhere there is a reference to felony, a hit to an unstructured document is made. But a simple search is crude. It does not find references to crime, arson, murder, embezzlement, vehicular homicide, and such, even though these crimes are types of felonies.”

7.12.3 The use of metadata

To solve problems with searchability and assessment of data, it is necessary to know something about the content. This can be done by adding context through the use of metadata.[33] Many systems already capture some metadata (e.g. filename, author, size, etc.), but more useful would be metadata about the actual content – e.g. summaries, topics, people or companies mentioned. Two technologies designed for generating metadata about content are automatic categorization and information extraction.

7.13 Future

A 2009 paper predicted[37] these developments in the business intelligence market:

- Because of lack of information, processes, and tools, through 2012, more than 35 percent of the top 5,000 global companies regularly fail to make insightful decisions about significant changes in their business and markets.
- By 2012, business units will control at least 40 percent of the total budget for business intelligence.
- By 2012, one-third of analytic applications applied to business processes will be delivered through coarse-grained application mashups.

A 2009 *Information Management* special report predicted the top BI trends: "green computing, social networking services, data visualization, mobile BI, predictive analytics, composite applications, cloud computing and multitouch.".[38] Research undertaken in 2014 indicated that employees are more likely to have access to, and more likely to engage with, cloud-based BI tools than traditional tools.[39]

Other business intelligence trends include the following:

- Third party SOA-BI products increasingly address ETL issues of volume and throughput.
- Companies embrace in-memory processing, 64-bit processing, and pre-packaged analytic BI applications.
- Operational applications have callable BI components, with improvements in response time, scaling, and concurrency.
- Near or real time BI analytics is a baseline expectation.
- Open source BI software replaces vendor offerings.

Other lines of research include the combined study of business intelligence and uncertain data.[40][41] In this context, the data used is not assumed to be precise, accurate and complete. Instead, data is considered uncertain and therefore this uncertainty is propagated to the results produced by BI.

According to a study by the Aberdeen Group, there has been increasing interest in Software-as-a-Service (SaaS) business intelligence over the past years, with twice as many organizations using this deployment approach as one year ago – 15% in 2009 compared to 7% in 2008.[42]

An article by InfoWorld's Chris Kanaracus points out similar growth data from research firm IDC, which predicts the SaaS BI market will grow 22 percent each year through 2013 thanks to increased product sophistication, strained IT budgets, and other factors.[43]

An analysis of top 100 Business Intelligence and Analytics scores and ranks the firms based on several open variables [44]

7.14 See also

- Accounting intelligence
- Analytic applications
- Artificial intelligence marketing
- Business Intelligence 2.0
- Business process discovery
- Business process management
- Business activity monitoring
- Business service management
- Customer dynamics
- Data Presentation Architecture
- Data visualization
- Decision engineering
- Enterprise planning systems
- Document intelligence
- Integrated business planning
- Location intelligence
- Media intelligence
- Meteorological intelligence
- Mobile business intelligence
- Multiway Data Analysis
- Operational intelligence
- Business Information Systems
- Business intelligence tools
- Process mining
- Real-time business intelligence
- Runtime intelligence
- Sales intelligence
- Spend management
- Test and learn

7.15 References

[1] (Rud, Olivia (2009). *Business Intelligence Success Factors: Tools for Aligning Your Business in the Global Economy*. Hoboken, N.J: Wiley & Sons. ISBN 978-0-470-39240-9.)

[2] Coker, Frank (2014). *Pulse: Understanding the Vital Signs of Your Business*. Ambient Light Publishing. pp. 41–42. ISBN 978-0-9893086-0-1.

[3] Miller Devens, Richard. *Cyclopaedia of Commercial and Business Anecdotes; Comprising Interesting Reminiscences and Facts, Remarkable Traits and Humors of Merchants, Traders, Bankers Etc. in All Ages and Countries*. D. Appleton and company. p. 210. Retrieved 15 February 2014.

[4] H P Luhn (1958). "A Business Intelligence System" (PDF). *IBM Journal* **2** (4): 314. doi:10.1147/rd.24.0314.

[5] Pieter M. Kroonenberg, Applied Multiway Data Analysis, Wiley 2008, pp. xv.

[6] D. J. Power (10 March 2007). "A Brief History of Decision Support Systems, version 4.0". DSSResources.COM. Retrieved 10 July 2008.

[7] Power, D. J. "A Brief History of Decision Support Systems". Retrieved 1 November 2010.

[8] Golden, Bernard (2013). *Amazon Web Services For Dummies*. For dummies. John Wiley & Sons. p. 234. ISBN 9781118652268. Retrieved 2014-07-06. [...] traditional business intelligence or data warehousing tools (the terms are used so interchangeably that they're often referred to as BI/DW) are extremely expensive [...]

[9] Evelson, Boris (21 November 2008). "Topic Overview: Business Intelligence".

[10] Evelson, Boris (29 April 2010). "Want to know what Forrester's lead data analysts are thinking about BI and the data domain?".

[11] Kobielus, James (30 April 2010). "What's Not BI? Oh, Don't Get Me Started....Oops Too Late...Here Goes....". "Business" intelligence is a non-domain-specific catchall for all the types of analytic data that can be delivered to users in reports, dashboards, and the like. When you specify the subject domain for this intelligence, then you can refer to "competitive intelligence," "market intelligence," "social intelligence," "financial intelligence," "HR intelligence," "supply chain intelligence," and the like.

[12] "Business Analytics vs Business Intelligence?". timoelliott.com. 2011-03-09. Retrieved 2014-06-15.

[13] "Difference between Business Analytics and Business Intelligence". businessanalytics.com. 2013-03-15. Retrieved 2014-06-15.

[14] Henschen, Doug (4 January 2010). *Analytics at Work: Q&A with Tom Davenport*. (Interview).

[15] Kimball et al., 2008: 29

[16] "Are You Ready for the New Business Intelligence?". Dell.com. Retrieved 19 June 2012.

[17] Jeanne W. Ross, Peter Weill, David C. Robertson (2006) *Enterprise Architecture As Strategy*, p. 117 ISBN 1-59139-839-8.

[18] Krapohl, Donald. "A Structured Methodology for Group Decision Making". AugmentedIntel. Retrieved 22 April 2013.

[19] Kimball et al. 2008: p. 298

[20] Kimball et al., 2008: 16

[21] Kimball et al., 2008: 18

[22] Kimball et al., 2008: 17

[23] "How Companies Are Implementing Business Intelligence Competency Centers" (PDF). Computer World. Retrieved 1 April 2014.

[24] Kimball

[25] Swain Scheps *Business Intelligence for Dummies*, 2008, ISBN 978-0-470-12723-0

[26] Watson, Hugh J.; Wixom, Barbara H. (2007). "The Current State of Business Intelligence". *Computer* **40** (9): 96. doi:10.1109/MC.2007.331.

[27] *The Data Warehouse Lifecycle Toolkit (2nd ed.). Ralph Kimball (2008).*

[28] *Microsoft Data Warehouse Toolkit. Wiley Publishing. (2006)*

[29] Andrew Brust (2013-02-14). "Gartner releases 2013 BI Magic Quadrant". ZDNet. Retrieved 21 August 2013.

[30] Pendse, Nigel (7 March 2008). "Consolidations in the BI industry". *The OLAP Report*.

[31] "Why Business Intelligence Is Key For Competitive Advantage". *Boston University*. Retrieved 23 October 2014.

[32] Imhoff, Claudia (4 April 2006). "Three Trends in Business Intelligence Technology".

[33] Rao, R. (2003). "From unstructured data to actionable intelligence" (PDF). *IT Professional* **5** (6): 29. doi:10.1109/MITP.2003.1254966.

[34] Blumberg, R. & S. Atre (2003). "The Problem with Unstructured Data" (PDF). *DM Review*: 42–46.

[35] Negash, S (2004). "Business Intelligence" (PDF). *Communications of the Association of Information Systems* **13**: 177–195.

[36] Inmon, B. & A. Nesavich, "Unstructured Textual Data in the Organization" from "Managing Unstructured data in the organization", Prentice Hall 2008, pp. 1–13

[37] Gartner Reveals Five Business Intelligence Predictions for 2009 and Beyond. gartner.com. 15 January 2009

[38] Campbell, Don (23 June 2009). "10 Red Hot BI Trends". *Information Management*.

[39] Lock, Michael (27 March 2014). "Cloud Analytics in 2014: Infusing the Workforce with Insight".

[40] Rodriguez, Carlos; Daniel, Florian; Casati, Fabio; Cappiello, Cinzia (2010). "Toward Uncertain Business Intelligence: The Case of Key Indicators". *IEEE Internet Computing* **14** (4): 32. doi:10.1109/MIC.2010.59.

[41] Rodriguez, C., Daniel, F., Casati, F. & Cappiello, C. (2009), *Computing Uncertain Key Indicators from Uncertain Data* (PDF), pp. 106–120

[42] Lock, Michael. "http://baroi.aberdeen.com/pdfs/5874-RA-BIDashboards-MDL-06-NSP.pdf" (PDF). *Aberdeen*. Aberdeen Group. Retrieved 23 October 2014.

[43] SaaS BI growth will soar in 2010 | Cloud Computing. InfoWorld (2010-02-01). Retrieved 17 January 2012.

[44] Top 100 Business Intelligence startups

7.16 Bibliography

- Ralph Kimball *et al.* “The Data warehouse Lifecycle Toolkit” (2nd ed.) Wiley ISBN 0-470-47957-4
- Peter Rausch, Alaa Sheta, Aladdin Ayesh : *Business Intelligence and Performance Management: Theory, Systems, and Industrial Applications*, Springer Verlag U.K., 2013, ISBN 978-1-4471-4865-4.

7.17 External links

- Chaudhuri, Surajit; Dayal, Umeshwar; Narasayya, Vivek (August 2011). “An Overview Of Business Intelligence Technology”. *Communications of the ACM* **54** (8): 88–98. doi:10.1145/1978542.1978562. Retrieved 26 October 2011.

Chapter 8

Analytics

For the ice hockey term, see Analytics (ice hockey).

Analytics is the discovery and communication of meaningful patterns in data. Especially valuable in areas rich with recorded information, analytics relies on the simultaneous application of statistics, computer programming and operations research to quantify performance. Analytics often favors data visualization to communicate insight.

Firms may commonly apply analytics to business data, to describe, predict, and improve business performance. Specifically, areas within analytics include predictive analytics, enterprise decision management, retail analytics, store assortment and stock-keeping unit optimization, marketing optimization and marketing mix modeling, web analytics, sales force sizing and optimization, price and promotion modeling, predictive science, credit risk analysis, and fraud analytics. Since analytics can require extensive computation (see big data), the algorithms and software used for analytics harness the most current methods in computer science, statistics, and mathematics.[1]

8.1 Analytics vs. analysis

Analytics is a multidimensional discipline. There is extensive use of mathematics and statistics, the use of descriptive techniques and predictive models to gain valuable knowledge from data—data analysis. The insights from data are used to recommend action or to guide decision making rooted in business context. Thus, analytics is not so much concerned with individual analyses or analysis steps, but with the entire methodology. There is a pronounced tendency to use the term *analytics* in business settings e.g. text analytics vs. the more generic text mining to emphasize this broader perspective. . There is an increasing use of the term *advanced analytics*, typically used to describe the technical aspects of analytics, especially in the emerging fields such as the use of machine learning techniques like neural networks to do predictive modeling.

8.2 Examples

8.2.1 Marketing optimization

Marketing has evolved from a creative process into a highly data-driven process. Marketing organizations use analytics to determine the outcomes of campaigns or efforts and to guide decisions for investment and consumer targeting. Demographic studies, customer segmentation, conjoint analysis and other techniques allow marketers to use large amounts of consumer purchase, survey and panel data to understand and communicate marketing strategy.

Web analytics allows marketers to collect session-level information about interactions on a website using an operation called sessionization. Google Analytics is an example of a popular free analytics tools that marketers use for this purpose. Those interactions provide the web analytics information systems with the information to track the referrer, search keywords, IP address, and activities of the visitor. With this information, a marketer can improve the marketing campaigns, site creative content, and information architecture.

Analysis techniques frequently used in marketing include marketing mix modeling, pricing and promotion analyses, sales force optimization, customer analytics e.g.: segmentation. Web analytics and optimization of web sites and online campaigns now frequently work hand in hand with the more traditional marketing analysis techniques. A focus on digital media has slightly changed the vocabulary so that marketing mix modeling is commonly referred to as attribution modeling in the digital or Marketing mix modeling context.

These tools and techniques support both strategic marketing decisions (such as how much overall to spend on marketing and how to allocate budgets across a portfolio of brands and the marketing mix) and more tactical campaign support in terms of targeting the best potential customer with the optimal message in the most cost effective medium at the ideal time.

8.2.2 Portfolio analysis

A common application of business analytics is portfolio analysis. In this, a bank or lending agency has a collection of accounts of varying value and risk. The accounts may differ by the social status (wealthy, middle-class, poor, etc.) of the holder, the geographical location, its net value, and many other factors. The lender must balance the return on the loan with the risk of default for each loan. The question is then how to evaluate the portfolio as a whole.

The least risk loan may be to the very wealthy, but there are a very limited number of wealthy people. On the other hand there are many poor that can be lent to, but at greater risk. Some balance must be struck that maximizes return and minimizes risk. The analytics solution may combine time series analysis with many other issues in order to make decisions on when to lend money to these different borrower segments, or decisions on the interest rate charged to members of a portfolio segment to cover any losses among members in that segment.

8.2.3 Risk analytics

Predictive models in the banking industry are developed to bring certainty across the risk scores for individual customers. Credit scores are built to predict individual's delinquency behaviour and widely used to evaluate the credit worthiness of each applicant. Furthermore, risk analyses are carried out in the scientific world and the insurance industry.

8.2.4 Digital analytics

Digital analytics is a set of business and technical activities that define, create, collect, verify or transform digital data into reporting, research, analyses, recommendations, optimizations, predictions, and automations.[2]

8.2.5 Security analytics

Security analytics refers to information technology (IT) solutions that gather and analyze security events to bring situational awareness and enable IT staff to understand and analyze events that pose the greatest risk.[3] Solutions in this area include Security information and event management solutions and user behavior analytics solutions.

8.2.6 Software analytics

Main article: Software analytics

Software analytics is the process of collecting information about the way a piece of software is used and produced.

8.3 Challenges

In the industry of commercial analytics software, an emphasis has emerged on solving the challenges of analyzing massive, complex data sets, often when such data is in a constant state of change. Such data sets are commonly referred to as big data. Whereas once the problems posed by big data were only found in the scientific community, today big data is a problem for many businesses that operate transactional systems online and, as a result, amass large volumes of data quickly.[4]

The analysis of unstructured data types is another challenge getting attention in the industry. Unstructured data differs from structured data in that its format varies widely and cannot be stored in traditional relational databases without significant effort at data transformation.[5] Sources of unstructured data, such as email, the contents of word processor documents, PDFs, geospatial data, etc., are rapidly becoming a relevant source of business intelligence for businesses, governments and universities.[6] For example, in Britain the discovery that one company was illegally selling fraudulent doctor's notes in order to assist people in defrauding employers and insurance companies,[7] is an opportunity for insurance firms to increase the vigilance of their unstructured data analysis. The McKinsey Global Institute estimates that big data analysis could save the American health care system $300 billion per year and the European public sector €250 billion.[8]

These challenges are the current inspiration for much of the innovation in modern analytics information systems, giving birth to relatively new machine analysis concepts such as complex event processing, full text search and analysis, and even new ideas in presentation.[9] One such innovation is the introduction of grid-like architecture in machine analysis, allowing increases in the speed of massively parallel processing by distributing the workload to many computers all with equal access to the complete data set.[10]

Analytics is increasingly used in education, particularly at the district and government office levels. However, the complexity of student performance measures presents challenges when educators try to understand and use analytics to discern patterns in student performance, predict graduation likelihood, improve chances of student success, etc. For example, in a study involving districts known for strong data use, 48% of teachers had difficulty posing questions prompted by data, 36% did not comprehend given data, and 52% incorrectly interpreted data.[11] To combat this, some analytics tools for educators adhere to an over-the-counter data format (embedding labels, supplemental documentation, and a help system, and making key package/display and content decisions) to improve educators' understanding and use of the analytics being displayed.[12]

One more emerging challenge is dynamic regulatory needs. For example, in the banking industry, Basel III and future capital adequacy needs are likely to make even smaller banks adopt internal risk models. In such cases, cloud computing and open source R (programming language) can help smaller banks to adopt risk analytics and support branch level monitoring by applying predictive analytics.

8.4 Risks

The main risk for the people is discrimination like Price discrimination or Statistical discrimination.

There is also the risk that a developer could profit from the ideas or work done by users, like this example: Users could write new ideas in a note taking app, which could then be sent as a custom event, and the developers could profit from those ideas. This can happen because the ownership of content is usually unclear in the law.[13]

If a user's identity is not protected, there are more risks; for example, the risk that private information about users is made public on the internet.

In the extreme, there is the risk that governments could gather too much private information, now that the governments are giving themselves more powers to access citizens' information.

Further information: Telecommunications data retention

8.5 See also

8.6 References

[1] Kohavi, Rothleder and Simoudis (2002). "Emerging Trends in Business Analytics". *Communications of the ACM* **45** (8): 45–48. doi:10.1145/545151.545177.

[2] Phillips, Judah "Building a Digital Analytics Organization" Financial Times Press, 2013, pp 7–8.

[3] "Security analytics shores up hope for breach detection". Enterprise Innovation. Retrieved April 27, 2015.

[4] Naone, Erica. "The New Big Data". Technology Review, MIT. Retrieved August 22, 2011.

[5] Inmon, Bill; Nesavich, Anthony (2007). *Tapping Into Unstructured Data*. Prentice-Hall. ISBN 978-0-13-236029-6.

[6] Wise, Lyndsay. "Data Analysis and Unstructured Data". Dashboard Insight. Retrieved February 14, 2011.

[7] "Fake doctors' sick notes for Sale for £25, NHS fraud squad warns". London: The Telegraph. Retrieved August 2008.

[8] "Big Data: The next frontier for innovation, competition and productivity as reported in Building with Big Data". *The Economist*. May 26, 2011. Archived from the original on 3 June 2011. Retrieved May 26, 2011.

[9] Ortega, Dan. "Mobililty: Fueling a Brainier Business Intelligence". IT Business Edge. Retrieved June 21, 2011.

[10] Khambadkone, Krish. "Are You Ready for Big Data?". InfoGain. Retrieved February 10, 2011.

[11] U.S. Department of Education Office of Planning, Evaluation and Policy Development (2009). *Implementing data-informed decision making in schools: Teacher access, supports and use.* United States Department of Education (ERIC Document Reproduction Service No. ED504191)

[12] Rankin, J. (2013, March 28). How data Systems & reports can either fight or propagate the data analysis error epidemic, and how educator leaders can help. *Presentation conducted from Technology Information Center for Administrative Leadership (TICAL) School Leadership Summit.*

[13] http://www.techrepublic.com/blog/10-things/10-reasons-why-i-avoid-social-networking-services/

•

8.7 External links

- INFORMS' bi-monthly, digital magazine on the analytics profession
- Glossary of popular analytical terms

Chapter 9

Data mining

Not to be confused with analytics, information extraction, or data analysis.

Data mining (the analysis step of the “Knowledge Discovery in Databases” process, or KDD),[1] an interdisciplinary subfield of computer science,[2][3][4] is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems.[2] The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.[2] Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating.[2]

The term is a misnomer, because the goal is the extraction of patterns and knowledge from large amount of data, not the extraction of data itself.[5] It also is a buzzword[6] and is frequently applied to any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) as well as any application of computer decision support system, including artificial intelligence, machine learning, and business intelligence. The popular book “Data mining: Practical machine learning tools and techniques with Java”[7] (which covers mostly machine learning material) was originally to be named just “Practical machine learning”, and the term “data mining” was only added for marketing reasons.[8] Often the more general terms "(large scale) data analysis", or "analytics" – or when referring to actual methods, artificial intelligence and machine learning – are more appropriate.

The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation, nor result interpretation and reporting are part of the data mining step, but do belong to the overall KDD process as additional steps.

The related terms *data dredging*, *data fishing*, and *data snooping* refer to the use of data mining methods to sample parts of a larger population data set that are (or may be) too small for reliable statistical inferences to be made about the validity of any patterns discovered. These methods can, however, be used in creating new hypotheses to test against the larger data populations.

9.1 Etymology

In the 1960s, statisticians used terms like “Data Fishing” or “Data Dredging” to refer to what they considered the bad practice of analyzing data without an a-priori hypothesis. The term “Data Mining” appeared around 1990 in the database community. For a short time in 1980s, a phrase “database mining"™, was used, but since it was trademarked by HNC, a San Diego-based company, to pitch their Database Mining Workstation;[9] researchers consequently turned to “data mining”. Other terms used include Data Archaeology, Information Harvesting, Information Discovery, Knowledge Extraction, etc. Gregory Piatetsky-Shapiro coined the term “Knowledge Discovery in Databases” for the first workshop on the same topic (KDD-1989) and this term became more popular in AI and Machine Learning Community. However, the term data mining became more popular in the business and press communities.[10] Currently, Data Mining and Knowledge Discovery are used interchangeably. Since about 2007, “Predictive Analytics” and since 2011, “Data Science” terms were also used to describe this field.

9.2 Background

The manual extraction of patterns from data has occurred for centuries. Early methods of identifying patterns in data include Bayes' theorem (1700s) and regression analysis (1800s). The proliferation, ubiquity and increasing power of computer technology has dramatically increased data collection, storage, and manipulation ability. As data sets have grown in size and complexity, direct "hands-on" data analysis has increasingly been augmented with indirect, automated data processing, aided by other discoveries in computer science, such as neural networks, cluster analysis, genetic algorithms (1950s), decision trees and decision rules (1960s), and support vector machines (1990s). Data mining is the process of applying these methods with the intention of uncovering hidden patterns[11] in large data sets. It bridges the gap from applied statistics and artificial intelligence (which usually provide the mathematical background) to database management by exploiting the way data is stored and indexed in databases to execute the actual learning and discovery algorithms more efficiently, allowing such methods to be applied to ever larger data sets.

9.2.1 Research and evolution

The premier professional body in the field is the Association for Computing Machinery's (ACM) Special Interest Group (SIG) on Knowledge Discovery and Data Mining (SIGKDD).[12][13] Since 1989 this ACM SIG has hosted an annual international conference and published its proceedings,[14] and since 1999 it has published a biannual academic journal titled "SIGKDD Explorations".[15]

Computer science conferences on data mining include:

- CIKM Conference – ACM Conference on Information and Knowledge Management
- DMIN Conference – International Conference on Data Mining
- DMKD Conference – Research Issues on Data Mining and Knowledge Discovery
- ECDM Conference – European Conference on Data Mining
- ECML-PKDD Conference – European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases
- EDM Conference – International Conference on Educational Data Mining
- ICDM Conference – IEEE International Conference on Data Mining
- KDD Conference – ACM SIGKDD Conference on Knowledge Discovery and Data Mining
- MLDM Conference – Machine Learning and Data Mining in Pattern Recognition
- PAKDD Conference – The annual Pacific-Asia Conference on Knowledge Discovery and Data Mining
- PAW Conference – Predictive Analytics World
- SDM Conference – SIAM International Conference on Data Mining (SIAM)
- SSTD Symposium – Symposium on Spatial and Temporal Databases
- WSDM Conference – ACM Conference on Web Search and Data Mining

Data mining topics are also present on many data management/database conferences such as the ICDE Conference, SIGMOD Conference and International Conference on Very Large Data Bases

9.3 Process

The **Knowledge Discovery in Databases (KDD) process** is commonly defined with the stages:

(1) Selection

(2) Pre-processing

(3) Transformation

(4) *Data Mining*

(5) Interpretation/Evaluation.[1]

It exists, however, in many variations on this theme, such as the Cross Industry Standard Process for Data Mining (CRISP-DM) which defines six phases:

(1) Business Understanding

(2) Data Understanding

(3) Data Preparation

(4) Modeling

(5) Evaluation

(6) Deployment

or a simplified process such as (1) pre-processing, (2) data mining, and (3) results validation.

Polls conducted in 2002, 2004, and 2007 show that the CRISP-DM methodology is the leading methodology used by data miners.[16][17][18] The only other data mining standard named in these polls was SEMMA. However, 3-4 times as many people reported using CRISP-DM. Several teams of researchers have published reviews of data mining process models,[19][20] and Azevedo and Santos conducted a comparison of CRISP-DM and SEMMA in 2008.[21]

9.3.1 Pre-processing

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns actually present in the data, the target data set must be large enough to contain these patterns while remaining concise enough to be mined within an acceptable time limit. A common source for data is a data mart or data warehouse. Pre-processing is essential to analyze the multivariate data sets before data mining. The target set is then cleaned. Data cleaning removes the observations containing noise and those with missing data.

9.3.2 Data mining

Data mining involves six common classes of tasks:[1]

- Anomaly detection (Outlier/change/deviation detection) – The identification of unusual data records, that might be interesting or data errors that require further investigation.
- Association rule learning (Dependency modelling) – Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.
- Clustering – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.
- Classification – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".
- Regression – attempts to find a function which models the data with the least error.
- Summarization – providing a more compact representation of the data set, including visualization and report generation.

9.3.3 Results validation

Data mining can unintentionally be misused, and can then produce results which appear to be significant; but which do not actually predict future behavior and cannot be reproduced on a new sample of data and bear little use. Often this results from investigating too many hypotheses and not performing proper statistical hypothesis testing. A simple version of this problem in machine learning is known as overfitting, but the same problem can arise at different phases of the process and thus a train/test split - when applicable at all - may not be sufficient to prevent this from happening.

The final step of knowledge discovery from data is to verify that the patterns produced by the data mining algorithms occur in the wider data set. Not all patterns found by the data mining algorithms are necessarily valid. It is common for the data mining algorithms to find patterns in the training set which are not present in the general data set. This is called overfitting. To overcome this, the evaluation uses a test set of data on which the data mining algorithm was not trained. The learned patterns are applied to this test set, and the resulting output is compared to the desired output. For example, a data mining algorithm trying to distinguish "spam" from "legitimate" emails would be trained on a training set of sample e-mails. Once trained, the learned patterns would be applied to the test set of e-mails on which it had *not* been trained. The accuracy of the patterns can then be measured from how many e-mails they correctly classify. A number of statistical methods may be used to evaluate the algorithm, such as ROC curves.

If the learned patterns do not meet the desired standards, subsequently it is necessary to re-evaluate and change the pre-processing and data mining steps. If the learned patterns do meet the desired standards, then the final step is to interpret the learned patterns and turn them into knowledge.

9.4 Standards

There have been some efforts to define standards for the data mining process, for example the 1999 European Cross Industry Standard Process for Data Mining (CRISP-DM 1.0) and the 2004 Java Data Mining standard (JDM 1.0). Development on successors to these processes (CRISP-DM 2.0 and JDM 2.0) was active in 2006, but has stalled since. JDM 2.0 was withdrawn without reaching a final draft.

For exchanging the extracted models – in particular for use in predictive analytics – the key standard is the Predictive Model Markup Language (PMML), which is an XML-based language developed by the Data Mining Group (DMG) and supported as exchange format by many data mining applications. As the name suggests, it only covers prediction models, a particular data mining task of high importance to business applications. However, extensions to cover (for example) subspace clustering have been proposed independently of the DMG.[22]

9.5 Notable uses

See also: Category:Applied data mining.

9.5.1 Games

Since the early 1960s, with the availability of oracles for certain combinatorial games, also called tablebases (e.g. for 3x3-chess) with any beginning configuration, small-board dots-and-boxes, small-board-hex, and certain endgames in chess, dots-and-boxes, and hex; a new area for data mining has been opened. This is the extraction of human-usable strategies from these oracles. Current pattern recognition approaches do not seem to fully acquire the high level of abstraction required to be applied successfully. Instead, extensive experimentation with the tablebases – combined with an intensive study of tablebase-answers to well designed problems, and with knowledge of prior art (i.e., pre-tablebase knowledge) – is used to yield insightful patterns. Berlekamp (in dots-and-boxes, etc.) and John Nunn (in chess endgames) are notable examples of researchers doing this work, though they were not – and are not – involved in tablebase generation.

9.5.2 Business

In business, data mining is the analysis of historical business activities, stored as static data in data warehouse databases. The goal is to reveal hidden patterns and trends. Data mining software uses advanced pattern recognition algorithms to sift through large amounts of data to assist in discovering previously unknown strategic business information. Examples of what businesses use data mining for include performing market analysis to identify new product bundles, finding the root cause of manufacturing problems, to prevent customer attrition and acquire new customers, cross-selling to existing customers, and profiling customers with more accuracy.[23]

- In today's world raw data is being collected by companies at an exploding rate. For example, Walmart processes over 20 million point-of-sale transactions every day. This information is stored in a centralized database, but would be useless without some type of data mining software to analyze it. If Walmart analyzed their point-of-sale data with data mining techniques they would be able to determine sales trends, develop marketing campaigns, and more accurately predict customer loyalty.[24][25]

- Every time a credit card or a store loyalty card is being used, or a warranty card is being filled, data is being collected about the users behavior. Many people find the amount of information stored about us from companies, such as Google, Facebook, and Amazon, disturbing and are concerned about privacy. Although there is the potential for our personal data to be used in harmful, or unwanted, ways it is also being used to make our lives better. For example, Ford and Audi hope to one day collect information about customer driving patterns so they can recommend safer routes and warn drivers about dangerous road conditions.[26]

- Data mining in customer relationship management applications can contribute significantly to the bottom line. Rather than randomly contacting a prospect or customer through a call center or sending mail, a company can concentrate its efforts on prospects that are predicted to have a high likelihood of responding to an offer. More sophisticated methods may be used to optimize resources across campaigns so that one may predict to which channel and to which offer an individual is most likely to respond (across all potential offers). Additionally, sophisticated applications could be used to automate mailing. Once the results from data mining (potential prospect/customer and channel/offer) are determined, this "sophisticated application" can either automatically send an e-mail or a regular mail. Finally, in cases where many people will take an action without an offer, "uplift modeling" can be used to determine which people have the greatest increase in response if given an offer. Uplift modeling thereby enables marketers to focus mailings and offers on persuadable people, and not to send offers to people who will buy the product without an offer. Data clustering can also be used to automatically discover the segments or groups within a customer data set.

- Businesses employing data mining may see a return on investment, but also they recognize that the number of predictive models can quickly become very large. For example, rather than using one model to predict how many customers will churn, a business may choose to build a separate model for each region and customer type. In situations where a large number of models need to be maintained, some businesses turn to more automated data mining methodologies.

- Data mining can be helpful to human resources (HR) departments in identifying the characteristics of their most successful employees. Information obtained – such as universities attended by highly successful employees – can help HR focus recruiting efforts accordingly. Additionally, Strategic Enterprise Management applications help a company translate corporate-level goals, such as profit and margin share targets, into operational decisions, such as production plans and workforce levels.[27]

- Market basket analysis, relates to data-mining use in retail sales. If a clothing store records the purchases of customers, a data mining system could identify those customers who favor silk shirts over cotton ones. Although some explanations of relationships may be difficult, taking advantage of it is easier. The example deals with association rules within transaction-based data. Not all data are transaction based and logical, or inexact rules may also be present within a database.

- Market basket analysis has been used to identify the purchase patterns of the Alpha Consumer. Analyzing the data collected on this type of user has allowed companies to predict future buying trends and forecast supply demands.

- Data mining is a highly effective tool in the catalog marketing industry. Catalogers have a rich database of history of their customer transactions for millions of customers dating back a number of years. Data mining tools can identify patterns among customers and help identify the most likely customers to respond to upcoming mailing campaigns.

- Data mining for business applications can be integrated into a complex modeling and decision making process.[28] Reactive business intelligence (RBI) advocates a "holistic" approach that integrates data mining, modeling, and interactive visualization into an end-to-end discovery and continuous innovation process powered by human and automated learning.[29]

- In the area of decision making, the RBI approach has been used to mine knowledge that is progressively acquired from the decision maker, and then self-tune the decision method accordingly.[30] The relation between the quality of a data mining system and the amount of investment that the decision maker is willing to make was formalized by providing an economic perspective on the value of "extracted knowledge" in terms of its payoff to the organization[28] This decision-theoretic classification framework[28] was applied to a real-world semiconductor wafer manufacturing line, where decision rules for effectively monitoring and controlling the semiconductor wafer fabrication line were developed.[31]

- An example of data mining related to an integrated-circuit (IC) production line is described in the paper "Mining IC Test Data to Optimize VLSI Testing."[32] In this paper, the application of data mining and decision analysis to the problem of die-level functional testing is described. Experiments mentioned demonstrate the ability to apply a system of mining historical die-test data to create a probabilistic model of patterns of die failure. These patterns are then utilized to decide, in real time, which die to test next and when to stop testing. This system has been shown, based on experiments with historical test data, to have the potential to improve profits on mature IC products. Other examples[33][34] of the application of data mining methodologies in semiconductor manufacturing environments suggest that data mining methodologies may be particularly useful when data is scarce, and the various physical and chemical parameters that affect the process exhibit highly complex interactions. Another implication is that on-line monitoring of the semiconductor manufacturing process using data mining may be highly effective.

9.5.3 Science and engineering

In recent years, data mining has been used widely in the areas of science and engineering, such as bioinformatics, genetics, medicine, education and electrical power engineering.

- In the study of human genetics, sequence mining helps address the important goal of understanding the mapping relationship between the inter-individual variations in human DNA sequence and the variability in disease susceptibility. In simple terms, it aims to find out how the changes in an individual's DNA sequence affects the risks of developing common diseases such as cancer, which is of great importance to improving methods of diagnosing, preventing, and treating these diseases. One data mining method that is used to perform this task is known as multifactor dimensionality reduction.[35]

- In the area of electrical power engineering, data mining methods have been widely used for condition monitoring of high voltage electrical equipment. The purpose of condition monitoring is to obtain valuable information on, for example, the status of the insulation (or other important safety-related parameters). Data clustering techniques – such as the self-organizing map (SOM), have been applied to vibration monitoring and analysis of transformer on-load tap-changers (OLTCS). Using vibration monitoring, it can be observed that each tap change operation generates a signal that contains information about the condition of the tap changer contacts and the drive mechanisms. Obviously, different tap positions will generate different signals. However, there was considerable variability amongst normal condition signals for exactly the same tap position. SOM has been applied to detect abnormal conditions and to hypothesize about the nature of the abnormalities.[36]

- Data mining methods have been applied to dissolved gas analysis (DGA) in power transformers. DGA, as a diagnostics for power transformers, has been available for many years. Methods such as SOM has been applied to analyze generated data and to determine trends which are not obvious to the standard DGA ratio methods (such as Duval Triangle).[36]

- In educational research, where data mining has been used to study the factors leading students to choose to engage in behaviors which reduce their learning,[37] and to understand factors influencing university student retention.[38] A similar example of social application of data mining is its use in expertise finding systems, whereby descriptors of human expertise are extracted, normalized, and classified so as to facilitate the finding of experts, particularly in scientific and technical fields. In this way, data mining can facilitate institutional memory.

- Data mining methods of biomedical data facilitated by domain ontologies,[39] mining clinical trial data,[40] and traffic analysis using SOM.[41]

- In adverse drug reaction surveillance, the Uppsala Monitoring Centre has, since 1998, used data mining methods to routinely screen for reporting patterns indicative of emerging drug safety issues in the WHO global database of 4.6 million suspected adverse drug reaction incidents.[42] Recently, similar methodology has been developed to mine large collections of electronic health records for temporal patterns associating drug prescriptions to medical diagnoses.[43]

- Data mining has been applied to software artifacts within the realm of software engineering: Mining Software Repositories.

9.5.4 Human rights

Data mining of government records – particularly records of the justice system (i.e., courts, prisons) – enables the discovery of systemic human rights violations in connection to generation and publication of invalid or fraudulent legal records by various government agencies.[44][45]

9.5.5 Medical data mining

Some machine learning algorithms can be applied in medical field as second-opinion diagnostic tools and as tools for the knowledge extraction phase in the process of knowledge discovery in databases. One of these classifiers (called *Prototype exemplar learning classifier* (PEL-C)[46] is able to discover syndromes as well as atypical clinical cases.

In 2011, the case of Sorrell v. IMS Health, Inc., decided by the Supreme Court of the United States, ruled that pharmacies may share information with outside companies. This practice was authorized under the 1st Amendment of the Constitution, protecting the “freedom of speech.”[47] However, the passage of the Health Information Technology for Economic and Clinical Health Act (HITECH Act) helped to initiate the adoption of the electronic health record (EHR) and supporting technology in the United States.[48] The HITECH Act was signed into law on February 17, 2009 as part of the American Recovery and Reinvestment Act (ARRA) and helped to open the door to medical data mining.[49] Prior to the signing of this law, estimates of only 20% of United States-based physicians were utilizing electronic patient records.[48] Søren Brunak notes that “the patient record becomes as information-rich as possible” and thereby “maximizes the data mining opportunities.”[48] Hence, electronic patient records further expands the possibilities regarding medical data mining thereby opening the door to a vast source of medical data analysis.

9.5.6 Spatial data mining

Spatial data mining is the application of data mining methods to spatial data. The end objective of spatial data mining is to find patterns in data with respect to geography. So far, data mining and Geographic Information Systems (GIS) have existed as two separate technologies, each with its own methods, traditions, and approaches to visualization and data analysis. Particularly, most contemporary GIS have only very basic spatial analysis functionality. The immense explosion in geographically referenced data occasioned by developments in IT, digital mapping, remote sensing, and the global diffusion of GIS emphasizes the importance of developing data-driven inductive approaches to geographical analysis and modeling.

Data mining offers great potential benefits for GIS-based applied decision-making. Recently, the task of integrating these two technologies has become of critical importance, especially as various public and private sector organizations possessing huge databases with thematic and geographically referenced data begin to realize the huge potential of the information contained therein. Among those organizations are:

- offices requiring analysis or dissemination of geo-referenced statistical data

- public health services searching for explanations of disease clustering

- environmental agencies assessing the impact of changing land-use patterns on climate change

- geo-marketing companies doing customer segmentation based on spatial location.

Challenges in Spatial mining: Geospatial data repositories tend to be very large. Moreover, existing GIS datasets are often splintered into feature and attribute components that are conventionally archived in hybrid data management systems. Algorithmic requirements differ substantially for relational (attribute) data management and for topological (feature) data management.[50] Related to this is the range and diversity of geographic data formats, which present unique challenges. The digital geographic data revolution is creating new types of data formats beyond the traditional "vector" and "raster" formats. Geographic data repositories increasingly include ill-structured data, such as imagery and geo-referenced multi-media.[51]

There are several critical research challenges in geographic knowledge discovery and data mining. Miller and Han[52] offer the following list of emerging research topics in the field:

- **Developing and supporting geographic data warehouses (GDW's)**: Spatial properties are often reduced to simple aspatial attributes in mainstream data warehouses. Creating an integrated GDW requires solving issues of spatial and temporal data interoperability – including differences in semantics, referencing systems, geometry, accuracy, and position.

- **Better spatio-temporal representations in geographic knowledge discovery**: Current geographic knowledge discovery (GKD) methods generally use very simple representations of geographic objects and spatial relationships. Geographic data mining methods should recognize more complex geographic objects (i.e., lines and polygons) and relationships (i.e., non-Euclidean distances, direction, connectivity, and interaction through attributed geographic space such as terrain). Furthermore, the time dimension needs to be more fully integrated into these geographic representations and relationships.

- **Geographic knowledge discovery using diverse data types**: GKD methods should be developed that can handle diverse data types beyond the traditional raster and vector models, including imagery and geo-referenced multimedia, as well as dynamic data types (video streams, animation).

9.5.7 Temporal data mining

Data may contain attributes generated and recorded at different times. In this case finding meaningful relationships in the data may require considering the temporal order of the attributes. A temporal relationship may indicate a causal relationship, or simply an association.

9.5.8 Sensor data mining

Wireless sensor networks can be used for facilitating the collection of data for spatial data mining for a variety of applications such as air pollution monitoring.[53] A characteristic of such networks is that nearby sensor nodes monitoring an environmental feature typically register similar values. This kind of data redundancy due to the spatial correlation between sensor observations inspires the techniques for in-network data aggregation and mining. By measuring the spatial correlation between data sampled by different sensors, a wide class of specialized algorithms can be developed to develop more efficient spatial data mining algorithms.[54]

9.5.9 Visual data mining

In the process of turning from analogical into digital, large data sets have been generated, collected, and stored discovering statistical patterns, trends and information which is hidden in data, in order to build predictive patterns. Studies suggest visual data mining is faster and much more intuitive than is traditional data mining.[55][56][57] See also Computer vision.

9.5.10 Music data mining

Data mining techniques, and in particular co-occurrence analysis, has been used to discover relevant similarities among music corpora (radio lists, CD databases) for purposes including classifying music into genres in a more objective manner.[58]

9.5.11 Surveillance

Data mining has been used by the U.S. government. Programs include the Total Information Awareness (TIA) program, Secure Flight (formerly known as Computer-Assisted Passenger Prescreening System (CAPPS II)), Analysis, Dissemination, Visualization, Insight, Semantic Enhancement (ADVISE),[59] and the Multi-state Anti-Terrorism Information Exchange (MATRIX).[60] These programs have been discontinued due to controversy over whether they violate the 4th Amendment to the United States Constitution, although many programs that were formed under them continue to be funded by different organizations or under different names.[61]

In the context of combating terrorism, two particularly plausible methods of data mining are "pattern mining" and "subject-based data mining".

9.5.12 Pattern mining

"Pattern mining" is a data mining method that involves finding existing patterns in data. In this context *patterns* often means association rules. The original motivation for searching association rules came from the desire to analyze supermarket transaction data, that is, to examine customer behavior in terms of the purchased products. For example, an association rule "beer ⇒ potato chips (80%)" states that four out of five customers that bought beer also bought potato chips.

In the context of pattern mining as a tool to identify terrorist activity, the National Research Council provides the following definition: "Pattern-based data mining looks for patterns (including anomalous data patterns) that might be associated with terrorist activity — these patterns might be regarded as small signals in a large ocean of noise."[62][63][64] Pattern Mining includes new areas such a Music Information Retrieval (MIR) where patterns seen both in the temporal and non temporal domains are imported to classical knowledge discovery search methods.

9.5.13 Subject-based data mining

"Subject-based data mining" is a data mining method involving the search for associations between individuals in data. In the context of combating terrorism, the National Research Council provides the following definition: "Subject-based data mining uses an initiating individual or other datum that is considered, based on other information, to be of high interest, and the goal is to determine what other persons or financial transactions or movements, etc., are related to that initiating datum."[63]

9.5.14 Knowledge grid

Knowledge discovery "On the Grid" generally refers to conducting knowledge discovery in an open environment using grid computing concepts, allowing users to integrate data from various online data sources, as well make use of remote resources, for executing their data mining tasks. The earliest example was the Discovery Net,[65][66] developed at Imperial College London, which won the "Most Innovative Data-Intensive Application Award" at the ACM SC02 (Supercomputing 2002) conference and exhibition, based on a demonstration of a fully interactive distributed knowledge discovery application for a bioinformatics application. Other examples include work conducted by researchers at the University of Calabria, who developed a Knowledge Grid architecture for distributed knowledge discovery, based on grid computing.[67][68]

9.6 Privacy concerns and ethics

While the term "data mining" itself has no ethical implications, it is often associated with the mining of information in relation to peoples' behavior (ethical and otherwise).[69]

The ways in which data mining can be used can in some cases and contexts raise questions regarding privacy, legality, and ethics.[70] In particular, data mining government or commercial data sets for national security or law enforcement purposes, such as in the Total Information Awareness Program or in ADVISE, has raised privacy concerns.[71][72]

Data mining requires data preparation which can uncover information or patterns which may compromise confidentiality and privacy obligations. A common way for this to occur is through data aggregation. Data aggregation involves combining data together (possibly from various sources) in a way that facilitates analysis (but that also might make identification of private, individual-level data deducible or otherwise apparent).[73] This is not data mining *per se*, but a result of the preparation of data before – and for the purposes of – the analysis. The threat to an individual's privacy comes into play when the data, once compiled, cause the data miner, or anyone who has access to the newly compiled data set, to be able to identify specific individuals, especially when the data were originally anonymous.[74][75][76]

It is recommended that an individual is made aware of the following **before** data are collected:[73]

- the purpose of the data collection and any (known) data mining projects;
- how the data will be used;
- who will be able to mine the data and use the data and their derivatives;
- the status of security surrounding access to the data;
- how collected data can be updated.

Data may also be modified so as to *become* anonymous, so that individuals may not readily be identified.[73] However, even "de-identified"/"anonymized" data sets can potentially contain enough information to allow identification of individuals, as occurred when journalists were able to find several individuals based on a set of search histories that were inadvertently released by AOL.[77]

9.6.1 Situation in Europe

Europe has rather strong privacy laws, and efforts are underway to further strengthen the rights of the consumers. However, the U.S.-E.U. Safe Harbor Principles currently effectively expose European users to privacy exploitation by U.S. companies. As a consequence of Edward Snowden's Global surveillance disclosure, there has been increased discussion to revoke this agreement, as in particular the data will be fully exposed to the National Security Agency, and attempts to reach an agreement have failed.

9.6.2 Situation in the United States

In the United States, privacy concerns have been addressed by the US Congress via the passage of regulatory controls such as the Health Insurance Portability and Accountability Act (HIPAA). The HIPAA requires individuals to give their “informed consent” regarding information they provide and its intended present and future uses. According to an article in *Biotech Business Week', "'[i]n practice, HIPAA may not offer any greater protection than the longstanding regulations in the research arena,' says the AAHC. More importantly, the rule’s goal of protection through informed consent is undermined by the complexity of consent forms that are required of patients and participants, which approach a level of incomprehensibility to average individuals.'"[78] This underscores the necessity for data anonymity in data aggregation and mining practices.*

U.S. information privacy legislation such as HIPAA and the Family Educational Rights and Privacy Act (FERPA) applies only to the specific areas that each such law addresses. Use of data mining by the majority of businesses in the U.S. is not controlled by any legislation.

9.7 Copyright Law

9.7.1 Situation in Europe

Due to a lack of flexibilities in European copyright and database law, the mining of in-copyright works such as web mining without the permission of the copyright owner is not legal. Where a database is pure data in Europe there is likely to be no copyright, but database rights may exist so data mining becomes subject to regulations by the Database Directive. On the recommendation of the Hargreaves review this led to the UK government to amend its copyright law in 2014[79] to allow content mining as a limitation and exception. Only the second country in the world to do so after Japan, which introduced an exception in 2009 for data mining. However due to the restriction of the Copyright Directive, the UK exception only allows content mining for non-commercial purposes. UK copyright law also does not allow this provision to be overridden by contractual terms and conditions. The European Commission facilitated stakeholder discussion on text and data mining in 2013, under the title of Licences for Europe.[80] The focus on the solution to this legal issue being licences and not limitations and exceptions led to representatives of universities, researchers, libraries, civil society groups and open access publishers to leave the stakeholder dialogue in May 2013.[81]

9.7.2 Situation in the United States

By contrast to Europe, the flexible nature of US copyright law, and in particular fair use means that content mining in America, as well as other fair use countries such as Israel, Taiwan and South Korea is viewed as being legal. As content mining is transformative, that is it does not supplant the original work, it is viewed as being lawful under fair use. For example as part of the Google Book settlement the presiding judge on the case ruled that Google’s digitisation project of in-copyright books was lawful, in part because of the transformative uses that the digitisation project displayed - one being text and data mining.[82]

9.8 Software

See also: Category:Data mining and machine learning software.

9.8.1 Free open-source data mining software and applications

- Carrot2: Text and search results clustering framework.
- Chemicalize.org: A chemical structure miner and web search engine.
- ELKI: A university research project with advanced cluster analysis and outlier detection methods written in the Java language.
- GATE: a natural language processing and language engineering tool.
- KNIME: The Konstanz Information Miner, a user friendly and comprehensive data analytics framework.
- ML-Flex: A software package that enables users to integrate with third-party machine-learning packages written in any programming language, execute classification analyses in parallel across multiple computing nodes, and produce HTML reports of classification results.
- MLPACK library: a collection of ready-to-use machine learning algorithms written in the C++ language.
- Massive Online Analysis (MOA): a real-time big data stream mining with concept drift tool in the Java programming language.
- NLTK (Natural Language Toolkit): A suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python language.
- OpenNN: Open neural networks library.

- Orange: A component-based data mining and machine learning software suite written in the Python language.
- R: A programming language and software environment for statistical computing, data mining, and graphics. It is part of the GNU Project.
- SCaViS: Java cross-platform data analysis framework developed at Argonne National Laboratory.
- SenticNet API: A semantic and affective resource for opinion mining and sentiment analysis.
- Tanagra: A visualisation-oriented data mining software, also for teaching.
- Torch: An open source deep learning library for the Lua programming language and scientific computing framework with wide support for machine learning algorithms.
- UIMA: The UIMA (Unstructured Information Management Architecture) is a component framework for analyzing unstructured content such as text, audio and video – originally developed by IBM.
- Weka: A suite of machine learning software applications written in the Java programming language.

9.8.2 Commercial data-mining software and applications

- Angoss KnowledgeSTUDIO: data mining tool provided by Angoss.
- Clarabridge: enterprise class text analytics solution.
- HP Vertica Analytics Platform: data mining software provided by HP.
- IBM SPSS Modeler: data mining software provided by IBM.
- KXEN Modeler: data mining tool provided by KXEN.
- Grapheme: data mining and visualization software provided by iChrome.
- LIONsolver: an integrated software application for data mining, business intelligence, and modeling that implements the Learning and Intelligent OptimizatioN (LION) approach.
- Microsoft Analysis Services: data mining software provided by Microsoft.
- NetOwl: suite of multilingual text and entity analytics products that enable data mining.
- Oracle Data Mining: data mining software by Oracle.
- RapidMiner: An environment for machine learning and data mining experiments.
- SAS Enterprise Miner: data mining software provided by the SAS Institute.
- STATISTICA Data Miner: data mining software provided by StatSoft.
- Qlucore Omics Explorer: data mining software provided by Qlucore.

9.8.3 Marketplace surveys

Several researchers and organizations have conducted reviews of data mining tools and surveys of data miners. These identify some of the strengths and weaknesses of the software packages. They also provide an overview of the behaviors, preferences and views of data miners. Some of these reports include:

- 2011 Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery[83]
- Rexer Analytics Data Miner Surveys (2007–2013)[84]
- Forrester Research 2010 Predictive Analytics and Data Mining Solutions report[85]
- Gartner 2008 “Magic Quadrant” report[86]
- Robert A. Nisbet’s 2006 Three Part Series of articles “Data Mining Tools: Which One is Best For CRM?"[87]
- Haughton et al.'s 2003 Review of Data Mining Software Packages in *The American Statistician*[88]
- Goebel & Gruenwald 1999 “A Survey of Data Mining a Knowledge Discovery Software Tools” in SIGKDD Explorations[89]

9.9 See also

Methods

- Anomaly/outlier/change detection
- Association rule learning
- Classification
- Cluster analysis
- Decision tree
- Factor analysis
- Genetic algorithms

- Intention mining
- Multilinear subspace learning
- Neural networks
- Regression analysis
- Sequence mining
- Structured data analysis
- Support vector machines
- Text mining
- Online analytical processing (OLAP)

Application domains

- Analytics
- Bioinformatics
- Business intelligence
- Data analysis
- Data warehouse
- Decision support system
- Drug discovery
- Exploratory data analysis
- Predictive analytics
- Web mining

Application examples

See also: Category:Applied data mining.

- Customer analytics
- Data mining in agriculture
- Data mining in meteorology
- Educational data mining
- National Security Agency
- Police-enforced ANPR in the UK
- Quantitative structure–activity relationship
- Surveillance / Mass surveillance (e.g., Stellar Wind)

Related topics

Data mining is about *analyzing* data; for information about extracting information out of data, see:

- Data integration
- Data transformation
- Electronic discovery
- Information extraction
- Information integration
- Named-entity recognition
- Profiling (information science)
- Web scraping

9.10 References

[1] Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic (1996). "From Data Mining to Knowledge Discovery in Databases" (PDF). Retrieved 17 December 2008.

[2] "Data Mining Curriculum". ACM SIGKDD. 2006-04-30. Retrieved 2014-01-27.

[3] Clifton, Christopher (2010). "Encyclopædia Britannica: Definition of Data Mining". Retrieved 2010-12-09.

[4] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009). "The Elements of Statistical Learning: Data Mining, Inference, and Prediction". Retrieved 2012-08-07.

[5] Han, Jiawei; Kamber, Micheline (2001). *Data mining: concepts and techniques*. Morgan Kaufmann. p. 5. ISBN 9781558604896. Thus, data mining should habe been more appropriately named "knowledge mining from data," which is unfortunately somewhat long

[6] See e.g. OKAIRP 2005 Fall Conference, Arizona State University About.com: Datamining

[7] Witten, Ian H.; Frank, Eibe; Hall, Mark A. (30 January 2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3 ed.). Elsevier. ISBN 978-0-12-374856-0.

[8] Bouckaert, Remco R.; Frank, Eibe; Hall, Mark A.; Holmes, Geoffrey; Pfahringer, Bernhard; Reutemann, Peter; Witten, Ian H. (2010). "WEKA Experiences with a Java open-source project". *Journal of Machine Learning Research* **11**: 2533–2541. the original title, "Practical machine learning", was changed ... The term "data mining" was [added] primarily for marketing reasons.

[9] Mena, Jesús (2011). *Machine Learning Forensics for Law Enforcement, Security, and Intelligence*. Boca Raton, FL: CRC Press (Taylor & Francis Group). ISBN 978-1-4398-6069-4.

[10] Piatetsky-Shapiro, Gregory; Parker, Gary (2011). "Lesson: Data Mining, and Knowledge Discovery: An Introduction". *Introduction to Data Mining*. KD Nuggets. Retrieved 30 August 2012.

[11] Kantardzic, Mehmed (2003). *Data Mining: Concepts, Models, Methods, and Algorithms.* John Wiley & Sons. ISBN 0-471-22852-4. OCLC 50055336.

[12] "Microsoft Academic Search: Top conferences in data mining". Microsoft Academic Search.

[13] "Google Scholar: Top publications - Data Mining & Analysis". Google Scholar.

[14] Proceedings, International Conferences on Knowledge Discovery and Data Mining, ACM, New York.

[15] SIGKDD Explorations, ACM, New York.

[16] Gregory Piatetsky-Shapiro (2002) *KDnuggets Methodology Poll*

[17] Gregory Piatetsky-Shapiro (2004) *KDnuggets Methodology Poll*

[18] Gregory Piatetsky-Shapiro (2007) *KDnuggets Methodology Poll*

[19] Óscar Marbán, Gonzalo Mariscal and Javier Segovia (2009); *A Data Mining & Knowledge Discovery Process Model*. In Data Mining and Knowledge Discovery in Real Life Applications, Book edited by: Julio Ponce and Adem Karahoca, ISBN 978-3-902613-53-0, pp. 438–453, February 2009, I-Tech, Vienna, Austria.

[20] Lukasz Kurgan and Petr Musilek (2006); *A survey of Knowledge Discovery and Data Mining process models*. The Knowledge Engineering Review. Volume 21 Issue 1, March 2006, pp 1–24, Cambridge University Press, New York, NY, USA doi:10.1017/S0269888906000737

[21] Azevedo, A. and Santos, M. F. KDD, SEMMA and CRISP-DM: a parallel overview. In Proceedings of the IADIS European Conference on Data Mining 2008, pp 182–185.

[22] Günnemann, Stephan; Kremer, Hardy; Seidl, Thomas (2011). "An extension of the PMML standard to subspace clustering models". *Proceedings of the 2011 workshop on Predictive markup language modeling - PMML '11*. p. 48. doi:10.1145/2023598.2023605. ISBN 9781450308373.

[23] O'Brien, J. A., & Marakas, G. M. (2011). Management Information Systems. New York, NY: McGraw-Hill/Irwin.

[24] Alexander, D. (n.d.). Data Mining. Retrieved from The University of Texas at Austin: College of Liberal Arts: http://www.laits.utexas.edu/~{}anorman/BUS.FOR/course.mat/Alex/

[25] "Daniele Medri: Big Data & Business: An on-going revolution". Statistics Views. 21 Oct 2013.

[26] Goss, S. (2013, April 10). Data-mining and our personal privacy. Retrieved from The Telegraph: http://www.macon.com/2013/04/10/2429775/data-mining-and-our-personal-privacy.html

[27] Monk, Ellen; Wagner, Bret (2006). *Concepts in Enterprise Resource Planning, Second Edition*. Boston, MA: Thomson Course Technology. ISBN 0-619-21663-8. OCLC 224465825.

[28] Elovici, Yuval; Braha, Dan (2003). "A Decision-Theoretic Approach to Data Mining" (PDF). *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans* **33** (1).

[29] Battiti, Roberto; and Brunato, Mauro; *Reactive Business Intelligence. From Data to Models to Insight*, Reactive Search Srl, Italy, February 2011. ISBN 978-88-905795-0-9.

[30] Battiti, Roberto; Passerini, Andrea (2010). "Brain-Computer Evolutionary Multi-Objective Optimization (BC-EMO): a genetic algorithm adapting to the decision maker" (PDF). *IEEE Transactions on Evolutionary Computation* **14** (15): 671–687. doi:10.1109/TEVC.2010.2058118.

[31] Braha, Dan; Elovici, Yuval; Last, Mark (2007). "Theory of actionable data mining with application to semiconductor manufacturing control" (PDF). *International Journal of Production Research* **45** (13).

[32] Fountain, Tony; Dietterich, Thomas; and Sudyka, Bill (2000); *Mining IC Test Data to Optimize VLSI Testing*, in Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM Press, pp. 18–25

[33] Braha, Dan; Shmilovici, Armin (2002). "Data Mining for Improving a Cleaning Process in the Semiconductor Industry" (PDF). *IEEE Transactions on Semiconductor Manufacturing* **15** (1).

[34] Braha, Dan; Shmilovici, Armin (2003). "On the Use of Decision Tree Induction for Discovery of Interactions in a Photolithographic Process" (PDF). *IEEE Transactions on Semiconductor Manufacturing* **16** (4).

[35] Zhu, Xingquan; Davidson, Ian (2007). *Knowledge Discovery and Data Mining: Challenges and Realities.* New York, NY: Hershey. p. 18. ISBN 978-1-59904-252-7.

[36] McGrail, Anthony J.; Gulski, Edward; Allan, David; Birtwhistle, David; Blackburn, Trevor R.; Groot, Edwin R. S. "Data Mining Techniques to Assess the Condition of High Voltage Electrical Plant". *CIGRÉ WG 15.11 of Study Committee 15*.

[37] Baker, Ryan S. J. d. "Is Gaming the System State-or-Trait? Educational Data Mining Through the Multi-Contextual Application of a Validated Behavioral Model". *Workshop on Data Mining for User Modeling 2007*.

[38] Superby Aguirre, Juan Francisco; Vandamme, Jean-Philippe; Meskens, Nadine. "Determination of factors influencing the achievement of the first-year university students using data mining methods". *Workshop on Educational Data Mining 2006*.

[39] Zhu, Xingquan; Davidson, Ian (2007). *Knowledge Discovery and Data Mining: Challenges and Realities.* New York, NY: Hershey. pp. 163–189. ISBN 978-1-59904-252-7.

[40] Zhu, Xingquan; Davidson, Ian (2007). *Knowledge Discovery and Data Mining: Challenges and Realities.* New York, NY: Hershey. pp. 31–48. ISBN 978-1-59904-252-7.

[41] Chen, Yudong; Zhang, Yi; Hu, Jianming; Li, Xiang (2006). “Traffic Data Analysis Using Kernel PCA and Self-Organizing Map”. *IEEE Intelligent Vehicles Symposium*.

[42] Bate, Andrew; Lindquist, Marie; Edwards, I. Ralph; Olsson, Sten; Orre, Roland; Lansner, Anders; de Freitas, Rogelio Melhado (Jun 1998). “A Bayesian neural network method for adverse drug reaction signal generation” (PDF). *European Journal of Clinical Pharmacology* **54** (4): 315–21. doi:10.1007/s002280050466. PMID 9696956.

[43] Norén, G. Niklas; Bate, Andrew; Hopstadius, Johan; Star, Kristina; and Edwards, I. Ralph (2008); Temporal Pattern Discovery for Trends and Transient Effects: Its Application to Patient Records. *Proceedings of the Fourteenth International Conference on Knowledge Discovery and Data Mining (SIGKDD 2008), Las Vegas, NV*, pp. 963–971.

[44] Zernik, Joseph; Data Mining as a Civic Duty – Online Public Prisoners’ Registration Systems, *International Journal on Social Media: Monitoring, Measurement, Mining*, 1: 84–96 (2010)

[45] Zernik, Joseph; Data Mining of Online Judicial Records of the Networked US Federal Courts, *International Journal on Social Media: Monitoring, Measurement, Mining*, 1:69–83 (2010)

[46] Gagliardi, F (2011). “Instance-based classifiers applied to medical databases: Diagnosis and knowledge extraction”. *Artificial Intelligence in Medicine* **52** (3): 123–139. doi:10.1016/j.artmed.2011.04.002.

[47] David G. Savage (2011-06-24). “Pharmaceutical industry: Supreme Court sides with pharmaceutical industry in two decisions”. *Los Angeles Times*. Retrieved 2012-11-07.

[48] Analyzing Medical Data. (2012). *Communications of the ACM* 55(6), 13-15. doi:10.1145/2184319.2184324

[49] http://searchhealthit.techtarget.com/definition/HITECH-Act

[50] Healey, Richard G. (1991); *Database Management Systems*, in Maguire, David J.; Goodchild, Michael F.; and Rhind, David W., (eds.), *Geographic Information Systems: Principles and Applications*, London, GB: Longman

[51] Camara, Antonio S.; and Raper, Jonathan (eds.) (1999); *Spatial Multimedia and Virtual Reality*, London, GB: Taylor and Francis

[52] Miller, Harvey J.; and Han, Jiawei (eds.) (2001); *Geographic Data Mining and Knowledge Discovery*, London, GB: Taylor & Francis

[53] Ma, Y.; Richards, M.; Ghanem, M.; Guo, Y.; Hassard, J. (2008). “Air Pollution Monitoring and Mining Based on Sensor Grid in London”. *Sensors* **8** (6): 3601. doi:10.3390/s8063601.

[54] Ma, Y.; Guo, Y.; Tian, X.; Ghanem, M. (2011). “Distributed Clustering-Based Aggregation Algorithm for Spatial Correlated Sensor Networks”. *IEEE Sensors Journal* **11** (3): 641. doi:10.1109/JSEN.2010.2056916.

[55] Zhao, Kaidi; and Liu, Bing; Tirpark, Thomas M.; and Weimin, Xiao; *A Visual Data Mining Framework for Convenient Identification of Useful Knowledge*

[56] Keim, Daniel A.; *Information Visualization and Visual Data Mining*

[57] Burch, Michael; Diehl, Stephan; Weißgerber, Peter; *Visual Data Mining in Software Archives*

[58] Pachet, François; Westermann, Gert; and Laigre, Damien; *Musical Data Mining for Electronic Music Distribution*, Proceedings of the 1st WedelMusic Conference,Firenze, Italy, 2001, pp. 101–106.

[59] Government Accountability Office, *Data Mining: Early Attention to Privacy in Developing a Key DHS Program Could Reduce Risks*, GAO-07-293 (February 2007), Washington, DC

[60] Secure Flight Program report, MSNBC

[61] “Total/Terrorism Information Awareness (TIA): Is It Truly Dead?". *Electronic Frontier Foundation (official website)*. 2003. Retrieved 2009-03-15.

[62] Agrawal, Rakesh; Mannila, Heikki; Srikant, Ramakrishnan; Toivonen, Hannu; and Verkamo, A. Inkeri; *Fast discovery of association rules*, in *Advances in knowledge discovery and data mining*, MIT Press, 1996, pp. 307–328

[63] National Research Council, *Protecting Individual Privacy in the Struggle Against Terrorists: A Framework for Program Assessment*, Washington, DC: National Academies Press, 2008

[64] Haag, Stephen; Cummings, Maeve; Phillips, Amy (2006). *Management Information Systems for the information age*. Toronto: McGraw-Hill Ryerson. p. 28. ISBN 0-07-095569-7. OCLC 63194770.

[65] Ghanem, Moustafa; Guo, Yike; Rowe, Anthony; Wendel, Patrick (2002). “Grid-based knowledge discovery services for high throughput informatics”. *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*. p. 416. doi:10.1109/HPDC.2002.1029946. ISBN 0-7695-1686-6.

[66] Ghanem, Moustafa; Curcin, Vasa; Wendel, Patrick; Guo, Yike (2009). “Building and Using Analytical Workflows in Discovery Net”. *Data Mining Techniques in Grid Computing Environments*. p. 119. doi:10.1002/9780470699904.ch8. ISBN 9780470699904.

[67] Cannataro, Mario; Talia, Domenico (January 2003). “The Knowledge Grid: An Architecture for Distributed Knowledge Discovery” (PDF). *Communications of the ACM* **46** (1): 89–93. doi:10.1145/602421.602425. Retrieved 17 October 2011.

[68] Talia, Domenico; Trunfio, Paolo (July 2010). “How distributed data mining tasks can thrive as knowledge services” (PDF). *Communications of the ACM* **53** (7): 132–137. doi:10.1145/1785414.1785451. Retrieved 17 October 2011.

[69] Seltzer, William. "The Promise and Pitfalls of Data Mining: Ethical Issues" (PDF).

[70] Pitts, Chip (15 March 2007). "The End of Illegal Domestic Spying? Don't Count on It". *Washington Spectator*.

[71] Taipale, Kim A. (15 December 2003). "Data Mining and Domestic Security: Connecting the Dots to Make Sense of Data". *Columbia Science and Technology Law Review* **5** (2). OCLC 45263753. SSRN 546782.

[72] Resig, John; and Teredesai, Ankur (2004). "A Framework for Mining Instant Messaging Services". *Proceedings of the 2004 SIAM DM Conference*.

[73] *Think Before You Dig: Privacy Implications of Data Mining & Aggregation*, NASCIO Research Brief, September 2004

[74] Ohm, Paul. "Don't Build a Database of Ruin". Harvard Business Review.

[75] Darwin Bond-Graham, Iron Cagebook - The Logical End of Facebook's Patents, Counterpunch.org, 2013.12.03

[76] Darwin Bond-Graham, Inside the Tech industry's Startup Conference, Counterpunch.org, 2013.09.11

[77] *AOL search data identified individuals*, SecurityFocus, August 2006

[78] Biotech Business Week Editors (June 30, 2008); *BIOMEDICINE; HIPAA Privacy Rule Impedes Biomedical Research*, Biotech Business Week, retrieved 17 November 2009 from LexisNexis Academic

[79] UK Researchers Given Data Mining Right Under New UK Copyright Laws. *Out-Law.com.* Retrieved 14 November 2014

[80] "Licences for Europe - Structured Stakeholder Dialogue 2013". *European Commission*. Retrieved 14 November 2014.

[81] "Text and Data Mining:Its importance and the need for change in Europe". *Association of European Research Libraries*. Retrieved 14 November 2014.

[82] "Judge grants summary judgment in favor of Google Books — a fair use victory". *Lexology.com*. Antonelli Law Ltd. Retrieved 14 November 2014.

[83] Mikut, Ralf; Reischl, Markus (September–October 2011). "Data Mining Tools". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **1** (5): 431–445. doi:10.1002/widm.24. Retrieved October 21, 2011.

[84] Karl Rexer, Heather Allen, & Paul Gearan (2011); *Understanding Data Miners*, Analytics Magazine, May/June 2011 (INFORMS: Institute for Operations Research and the Management Sciences).

[85] Kobielus, James; *The Forrester Wave: Predictive Analytics and Data Mining Solutions, Q1 2010*, Forrester Research, 1 July 2008

[86] Herschel, Gareth; *Magic Quadrant for Customer Data-Mining Applications*, Gartner Inc., 1 July 2008

[87] Nisbet, Robert A. (2006); *Data Mining Tools: Which One is Best for CRM? Part 1*, Information Management Special Reports, January 2006

[88] Haughton, Dominique; Deichmann, Joel; Eshghi, Abdolreza; Sayek, Selin; Teebagy, Nicholas; and Topi, Heikki (2003); *A Review of Software Packages for Data Mining*, The American Statistician, Vol. 57, No. 4, pp. 290–309

[89] Goebel, Michael; Gruenwald, Le (1999); *A Survey of Data Mining and Knowledge Discovery Software Tools*, SIGKDD Explorations, Vol. 1, Issue 1, pp. 20–33

9.11 Further reading

- Cabena, Peter; Hadjnian, Pablo; Stadler, Rolf; Verhees, Jaap; and Zanasi, Alessandro (1997); *Discovering Data Mining: From Concept to Implementation*, Prentice Hall, ISBN 0-13-743980-6
- M.S. Chen, J. Han, P.S. Yu (1996) "Data mining: an overview from a database perspective". *Knowledge and data Engineering, IEEE Transactions* on 8 (6), 866-883
- Feldman, Ronen; and Sanger, James; *The Text Mining Handbook*, Cambridge University Press, ISBN 978-0-521-83657-9
- Guo, Yike; and Grossman, Robert (editors) (1999); *High Performance Data Mining: Scaling Algorithms, Applications and Systems*, Kluwer Academic Publishers
- Han, Jiawei, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.
- Hastie, Trevor, Tibshirani, Robert and Friedman, Jerome (2001); *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, ISBN 0-387-95284-5
- Liu, Bing (2007); *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*, Springer, ISBN 3-540-37881-2
- Murphy, Chris (16 May 2011). "Is Data Mining Free Speech?". *InformationWeek* (UMB): 12.
- Nisbet, Robert; Elder, John; Miner, Gary (2009); *Handbook of Statistical Analysis & Data Mining Applications*, Academic Press/Elsevier, ISBN 978-0-12-374765-5
- Poncelet, Pascal; Masseglia, Florent; and Teisseire, Maguelonne (editors) (October 2007); "Data Mining Patterns: New Methods and Applications", *Information Science Reference*, ISBN 978-1-59904-162-9

- Tan, Pang-Ning; Steinbach, Michael; and Kumar, Vipin (2005); *Introduction to Data Mining*, ISBN 0-321-32136-7
- Theodoridis, Sergios; and Koutroumbas, Konstantinos (2009); *Pattern Recognition*, 4th Edition, Academic Press, ISBN 978-1-59749-272-0
- Weiss, Sholom M.; and Indurkhya, Nitin (1998); *Predictive Data Mining*, Morgan Kaufmann
- Witten, Ian H.; Frank, Eibe; Hall, Mark A. (30 January 2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3 ed.). Elsevier. ISBN 978-0-12-374856-0. (See also Free Weka software)
- Ye, Nong (2003); *The Handbook of Data Mining*, Mahwah, NJ: Lawrence Erlbaum

9.12 External links

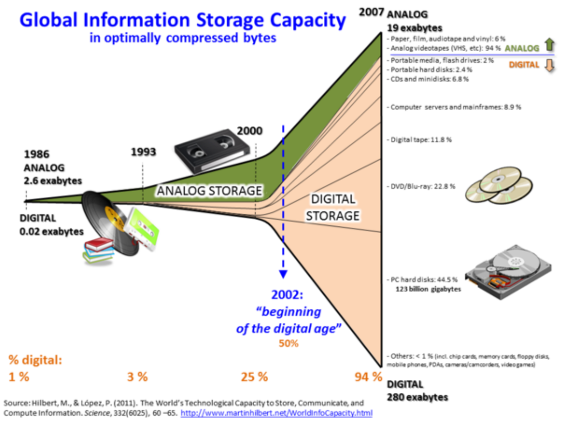
Chapter 10

Big data

This article is about large collections of data. For the graph database, see Graph database. For the band, see Big Data (band).

Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, and information privacy. The term often refers simply to the use of predictive analytics or other certain advanced methods to extract value from data, and seldom to a particular size of data set. Accuracy in big data may lead to more confident decision making. And better decisions can mean greater operational efficiency, cost reductions and reduced risk.

Visualization of daily Wikipedia edits created by IBM. At multiple terabytes in size, the text and images of Wikipedia are an example of big data.



Growth of and Digitization of Global Information Storage Capacity Source

Analysis of data sets can find new correlations, to "spot business trends, prevent diseases, combat crime and so on."[1] Scientists, business executives, practitioners of media and advertising and governments alike regularly meet difficulties with large data sets in areas including Internet search, finance and business informatics. Scientists encounter limitations in e-Science work, including meteorology, genomics,[2] connectomics, complex physics simulations,[3] and biological and environmental research.[4]

Data sets grow in size in part because they are increasingly being gathered by cheap and numerous information-sensing mobile devices, aerial (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers, and wireless sensor networks.[5][6][7] The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s;[8] as of 2012, every day 2.5 exabytes (2.5×10^{18}) of data were created;[9] The challenge for large enterprises is determining who should own big data initiatives that straddle the entire organization.[10]

Work with big data is necessarily uncommon; most analysis is of "PC size" data, on a desktop PC or notebook[11] that can handle the available data set.

Relational database management systems and desktop statistics and visualization packages often have difficulty handling big data. The work instead requires "massively parallel software running on tens, hundreds, or even thousands of servers".[12] What is considered "big data" varies depending on the capabilities of the users and their tools, and expanding capabilities make Big Data a moving target. Thus, what is considered to be "Big" in one year will become ordinary in later years. "For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration."[13]

10.1 Definition

Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time.[14] Big data "size" is a constantly moving target, as of 2012 ranging from a few dozen terabytes to many petabytes of data. Big data is a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex, and of a massive scale.[15]

In a 2001 research report[16] and related lectures, META Group (now Gartner) analyst Doug Laney defined data growth challenges and opportunities as being three-dimensional, i.e. increasing volume (amount of data), velocity (speed of data in and out), and variety (range of data types and sources). Gartner, and now much of the industry, continue to use this "3Vs" model for describing big data.[17] In 2012, Gartner updated its definition as follows: "Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization."[18] Additionally, a new V "Veracity" is added by some organizations to describe it.[19]

If Gartner's definition (the 3Vs) is still widely used, the growing maturity of the concept fosters a more sound difference between big data and Business Intelligence, regarding data and their use:[20]

- Business Intelligence uses descriptive statistics with data with high information density to measure things, detect trends etc.;
- Big data uses inductive statistics and concepts from nonlinear system identification [21] to infer laws (regressions, nonlinear relationships, and causal effects) from large sets of data with low information density[22] to reveal relationships, dependencies and perform predictions of outcomes and behaviors.[21][23]

A more recent, consensual definition states that "Big Data represents the Information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value".[24]

10.2 Characteristics

Big data can be described by the following characteristics:

Volume – The quantity of data that is generated is very important in this context. It is the size of the data which determines the value and potential of the data under consideration and whether it can actually be considered Big Data or not. The name 'Big Data' itself contains a term which is related to size and hence the characteristic.

Variety - The next aspect of Big Data is its variety. This means that the category to which Big Data belongs to is also an essential fact that needs to be known by the data analysts. This helps the people, who are closely analyzing the data and are associated with it, to effectively use the data to their advantage and thus upholding the importance of the Big Data.

Velocity - The term 'velocity' in the context refers to the speed of generation of data or how fast the data is generated and processed to meet the demands and the challenges which lie ahead in the path of growth and development.

Variability - This is a factor which can be a problem for those who analyse the data. This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

Veracity - The quality of the data being captured can vary greatly. Accuracy of analysis depends on the veracity of the source data.

Complexity - Data management can become a very complex process, especially when large volumes of data come from multiple sources. These data need to be linked, connected and correlated in order to be able to grasp the information that is supposed to be conveyed by these data. This situation, is therefore, termed as the 'complexity' of Big Data.

Factory work and Cyber-physical systems may have a 6C system:

1. Connection (sensor and networks),
2. Cloud (computing and data on demand),
3. Cyber (model and memory),
4. content/context (meaning and correlation),
5. community (sharing and collaboration), and
6. customization (personalization and value).

In this scenario and in order to provide useful insight to the factory management and gain correct content, data has to be processed with advanced tools (analytics and algorithms) to generate meaningful information. Considering the presence of visible and invisible issues in an industrial factory, the information generation algorithm has to be capable of detecting and addressing invisible issues such as machine degradation, component wear, etc. in the factory floor.[25][26]

10.3 Architecture

In 2000, Seisint Inc. developed C++ based distributed file sharing framework for data storage and querying. Structured, semi-structured and/or unstructured data is stored and distributed across multiple servers. Querying of data is done by modified C++ called ECL which uses apply scheme on read method to create structure of stored data during time of query. In 2004 LexisNexis acquired Seisint Inc.[27] and 2008 acquired ChoicePoint, Inc.[28] and their high speed parallel processing platform. The two platforms were merged into HPCC Systems and in 2011 was open sourced under Apache v2.0 License. Currently HPCC and Quantcast File System[29] are the only publicly available platforms capable of analyzing multiple exabytes of data.

In 2004, Google published a paper on a process called MapReduce that used such an architecture. The MapReduce framework provides a parallel processing model and associated implementation to process huge amounts of data. With MapReduce, queries are split and distributed across parallel nodes and processed in parallel (the Map step). The results are then gathered and delivered (the Reduce step). The framework was very successful,[30] so others wanted to replicate the algorithm. Therefore, an implementation of the MapReduce framework was adopted by an Apache open source project named Hadoop.[31]

MIKE2.0 is an open approach to information management that acknowledges the need for revisions due to big data implications in an article titled “Big Data Solution Offering”.[32] The methodology addresses handling big data in terms of useful permutations of data sources, complexity in interrelationships, and difficulty in deleting (or modifying) individual records.[33]

Recent studies show that the use of a multiple layer architecture is an option for dealing with big data. The Distributed Parallel architecture distributes data across multiple processing units and parallel processing units provide data much faster, by improving processing speeds. This type of architecture inserts data into a parallel DBMS, which implements the use of MapReduce and Hadoop frameworks. This type of framework looks to make the processing power transparent to the end user by using a front end application server.[34]

Big Data Analytics for Manufacturing Applications can be based on a 5C architecture (connection, conversion, cyber, cognition, and configuration).[35]

Big Data Lake - With the changing face of business and IT sector, capturing and storage of data has emerged into a sophisticated system. The big data lake allows an organization to shift its focus from centralized control to a shared model to respond to the changing dynamics of information management. This enables quick segregation of data into the data lake thereby reducing the overhead time.[36]

10.4 Technologies

Big data requires exceptional technologies to efficiently process large quantities of data within tolerable elapsed times. A 2011 McKinsey report[37] suggests suitable technologies include A/B testing, crowdsourcing, data fusion and integration, genetic algorithms, machine learning, natural language processing, signal processing, simulation, time series analysis and visualisation. Multidimensional big data can also be represented as tensors, which can be more efficiently handled by tensor-based computation,[38] such as multilinear subspace learning.[39] Additional technologies being applied to big data include massively parallel-processing (MPP) databases, search-based applications, data mining, distributed file systems, distributed databases, cloud based infrastructure (applications, storage and computing resources) and the Internet.

Some but not all MPP relational databases have the ability to store and manage petabytes of data. Implicit is the ability to load, monitor, back up, and optimize the use of the large data tables in the RDBMS.[40]

DARPA’s Topological Data Analysis program seeks the fundamental structure of massive data sets and in 2008 the technology went public with the launch of a company called Ayasdi.[41]

The practitioners of big data analytics processes are generally hostile to slower shared storage,[42] preferring direct-attached storage (DAS) in its various forms from solid state drive (SSD) to high capacity SATA disk buried inside parallel processing nodes. The perception of shared storage architectures—Storage area network (SAN) and Network-attached storage (NAS) —is that they are relatively slow, complex, and expensive. These qualities are not consistent with big data analytics systems that thrive on system performance, commodity infrastructure, and low cost.

Real or near-real time information delivery is one of the defining characteristics of big data analytics. Latency is therefore avoided whenever and wherever possible. Data in memory is good—data on spinning disk at the other end of a FC SAN connection is not. The cost of a SAN at the scale needed for analytics applications is very much higher than other storage techniques.

There are advantages as well as disadvantages to shared storage in big data analytics, but big data analytics practitioners as of 2011 did not favour it.[43]

10.5 Applications

Big data has increased the demand of information management specialists in that Software AG, Oracle Corporation, IBM, Microsoft, SAP, EMC, HP and Dell have spent more than $15 billion on software firms specializing

Bus wrapped with SAP Big data parked outside IDF13.

in data management and analytics. In 2010, this industry was worth more than $100 billion and was growing at almost 10 percent a year: about twice as fast as the software business as a whole.[1]

Developed economies make increasing use of data-intensive technologies. There are 4.6 billion mobile-phone subscriptions worldwide and between 1 billion and 2 billion people accessing the internet.[1] Between 1990 and 2005, more than 1 billion people worldwide entered the middle class which means more and more people who gain money will become more literate which in turn leads to information growth. The world's effective capacity to exchange information through telecommunication networks was 281 petabytes in 1986, 471 petabytes in 1993, 2.2 exabytes in 2000, 65 exabytes in 2007[8] and it is predicted that the amount of traffic flowing over the internet will reach 667 exabytes annually by 2014.[1] It is estimated that one third of the globally stored information is in the form of alphanumeric text and still image data,[44] which is the format most useful for most big data applications. This also shows the potential of yet unused data (i.e. in the form of video and audio content).

While many vendors offer off-the-shelf solutions for Big Data, experts recommend the development of in-house solutions custom-tailored to solve the company's problem at hand if the company has sufficient technical capabilities.[45]

10.5.1 Government

The use and adoption of Big Data within governmental processes is beneficial and allows efficiencies in terms of cost, productivity, and innovation. That said, this process does not come without its flaws. Data analysis often requires multiple parts of government (central and local) to work in collaboration and create new and innovative processes to deliver the desired outcome. Below are the thought leading examples within the Governmental Big Data space.

United States of America

- In 2012, the Obama administration announced the Big Data Research and Development Initiative, to explore how big data could be used to address important problems faced by the government.[46] The initiative is composed of 84 different big data programs spread across six departments.[47]
- Big data analysis played a large role in Barack Obama's successful 2012 re-election campaign.[48]
- The United States Federal Government owns six of the ten most powerful supercomputers in the world.[49]
- The Utah Data Center is a data center currently being constructed by the United States National Security Agency. When finished, the facility will be able to handle a large amount of information collected by the NSA over the Internet. The exact amount of storage space is unknown, but more recent sources claim it will be on the order of a few exabytes.[50][51][52]

India

- Big data analysis was, in parts, responsible for the BJP and its allies to win a highly successful Indian General Election 2014.[53]
- The Indian Government utilises numerous techniques to ascertain how the Indian electorate is responding to government action, as well as ideas for policy augmentation

United Kingdom

Examples of uses of big data in public services:

- Data on prescription drugs: by connecting origin, location and the time of each prescription, a research unit was able to exemplify the considerable delay between the release of any given drug, and a UK-wide adaptation of the National Institute for Health and Care Excellence guidelines. This suggests that new/most up-to-date drugs take some time to filter through to the general patient.
- Joining up data: a local authority blended data about services, such as road gritting rotas, with services for people at risk, such as 'meals on wheels'. The connection of data allowed the local authority to avoid any weather related delay.

10.5.2 International development

Research on the effective usage of information and communication technologies for development (also known as

ICT4D) suggests that big data technology can make important contributions but also present unique challenges to International development.[54][55] Advancements in big data analysis offer cost-effective opportunities to improve decision-making in critical development areas such as health care, employment, economic productivity, crime, security, and natural disaster and resource management.[56][57][58] However, longstanding challenges for developing regions such as inadequate technological infrastructure and economic and human resource scarcity exacerbate existing concerns with big data such as privacy, imperfect methodology, and interoperability issues.[56]

10.5.3 Manufacturing

Based on TCS 2013 Global Trend Study, improvements in supply planning and product quality provide the greatest benefit of big data for manufacturing.[59] Big data provides an infrastructure for transparency in manufacturing industry, which is the ability to unravel uncertainties such as inconsistent component performance and availability. Predictive manufacturing as an applicable approach toward near-zero downtime and transparency requires vast amount of data and advanced prediction tools for a systematic process of data into useful information.[60] A conceptual framework of predictive manufacturing begins with data acquisition where different type of sensory data is available to acquire such as acoustics, vibration, pressure, current, voltage and controller data. Vast amount of sensory data in addition to historical data construct the big data in manufacturing. The generated big data acts as the input into predictive tools and preventive strategies such as Prognostics and Health Management (PHM).[61]

Cyber-Physical Models

Current PHM implementations mostly utilize data during the actual usage while analytical algorithms can perform more accurately when more information throughout the machine's lifecycle, such as system configuration, physical knowledge and working principles, are included. There is a need to systematically integrate, manage and analyze machinery or process data during different stages of machine life cycle to handle data/information more efficiently and further achieve better transparency of machine health condition for manufacturing industry.

With such motivation a cyber-physical (coupled) model scheme has been developed. Please see http://www.imscenter.net/cyber-physical-platform The coupled model is a digital twin of the real machine that operates in the cloud platform and simulates the health condition with an integrated knowledge from both data driven analytical algorithms as well as other available physical knowledge. It can also be described as a 5S systematic approach consisting of Sensing, Storage, Synchronization, Synthesis and Service. The coupled model first constructs a digital image from the early design stage. System information and physical knowledge are logged during product design, based on which a simulation model is built as a reference for future analysis. Initial parameters may be statistically generalized and they can be tuned using data from testing or the manufacturing process using parameter estimation. After which, the simulation model can be considered as a mirrored image of the real machine, which is able to continuously record and track machine condition during the later utilization stage. Finally, with ubiquitous connectivity offered by cloud computing technology, the coupled model also provides better accessibility of machine condition for factory managers in cases where physical access to actual equipment or machine data is limited.[26][62]

10.5.4 Media

Internet of Things (IoT)

Main article: Internet of Things

To understand how the media utilises Big Data, it is first necessary to provide some context into the mechanism used for media process. It has been suggested by Nick Couldry and Joseph Turow that practitioners in Media and Advertising approach big data as many actionable points of information about millions of individuals. The industry appears to be moving away from the traditional approach of using specific media environments such as newspapers, magazines, or television shows and instead tap into consumers with technologies that reach targeted people at optimal times in optimal locations. The ultimate aim is to serve, or convey, a message or content that is (statistically speaking) in line with the consumers mindset. For example, publishing environments are increasingly tailoring messages (advertisements) and content (articles) to appeal to consumers that have been exclusively gleaned through various data-mining activities.[63]

- Targeting of consumers (for advertising by marketers)
- Data-capture

Big Data and the IoT work in conjunction. From a media perspective, data is the key derivative of device inter connectivity and allows accurate targeting. The Internet of Things, with the help of big data, therefore transforms the media industry, companies and even governments, opening up a new era of economic growth and competitiveness. The intersection of people, data and intelligent algorithms have far-reaching impacts on media efficiency. The wealth of data generated allows an elaborate layer on the present targeting mechanisms of the industry.

Technology

- eBay.com uses two data warehouses at 7.5 petabytes and 40PB as well as a 40PB Hadoop cluster for search, consumer recommendations, and merchandising. Inside eBay's 90PB data warehouse
- Amazon.com handles millions of back-end operations every day, as well as queries from more than half a million third-party sellers. The core technology that keeps Amazon running is Linux-based and as of 2005 they had the world's three largest Linux databases, with capacities of 7.8 TB, 18.5 TB, and 24.7 TB.[64]
- Facebook handles 50 billion photos from its user base.[65]
- As of August 2012, Google was handling roughly 100 billion searches per month.[66]
- Oracle NoSQL Database has been tested to past the 1M ops/sec mark with 8 shards and proceeded to hit 1.2M ops/sec with 10 shards.[67]

10.5.5 Private sector

Retail

- Walmart handles more than 1 million customer transactions every hour, which are imported into databases estimated to contain more than 2.5 petabytes (2560 terabytes) of data – the equivalent of 167 times the information contained in all the books in the US Library of Congress.[1]

Retail Banking

- FICO Card Detection System protects accounts world-wide.[68]
- The volume of business data worldwide, across all companies, doubles every 1.2 years, according to estimates.[69][70]

Real Estate

- Windermere Real Estate uses anonymous GPS signals from nearly 100 million drivers to help new home buyers determine their typical drive times to and from work throughout various times of the day.[71]

10.5.6 Science

The Large Hadron Collider experiments represent about 150 million sensors delivering data 40 million times per second. There are nearly 600 million collisions per second. After filtering and refraining from recording more than 99.99995% [72] of these streams, there are 100 collisions of interest per second.[73][74][75]

- As a result, only working with less than 0.001% of the sensor stream data, the data flow from all four LHC experiments represents 25 petabytes annual rate before replication (as of 2012). This becomes nearly 200 petabytes after replication.
- If all sensor data were to be recorded in LHC, the data flow would be extremely hard to work with. The data flow would exceed 150 million petabytes annual rate, or nearly 500 exabytes per day, before replication. To put the number in perspective, this is equivalent to 500 quintillion (5×10^{20}) bytes per day, almost 200 times more than all the other sources combined in the world.

The Square Kilometre Array is a telescope which consists of millions of antennas and is expected to be operational by 2024. Collectively, these antennas are expected to gather 14 exabytes and store one petabyte per day.[76][77] It is considered to be one of the most ambitious scientific projects ever undertaken.

Science and Research

- When the Sloan Digital Sky Survey (SDSS) began collecting astronomical data in 2000, it amassed more in its first few weeks than all data collected in the history of astronomy. Continuing at a rate of about 200 GB per night, SDSS has amassed more than 140 terabytes of information. When the Large Synoptic Survey Telescope, successor to SDSS, comes online in 2016 it is anticipated to acquire that amount of data every five days.[1]
- Decoding the human genome originally took 10 years to process, now it can be achieved in less than a day: the DNA sequencers have divided the sequencing cost by 10,000 in the last ten years, which is 100 times cheaper than the reduction in cost predicted by Moore's Law.[78]
- The NASA Center for Climate Simulation (NCCS) stores 32 petabytes of climate observations and simulations on the Discover supercomputing cluster.[79]

10.6 Research activities

Encrypted search and cluster formation in big data was demonstrated in March 2014 at the American Society of Engineering Education. Gautam Siwach engaged at *Tackling the challenges of Big Data* by MIT Computer Science and Artificial Intelligence Laboratory and Dr.

Amir Esmailpour at UNH Research Group investigated the key features of big data as formation of clusters and their interconnections. They focused on the security of big data and the actual orientation of the term towards the presence of different type of data in an encrypted form at cloud interface by providing the raw definitions and real time examples within the technology. Moreover, they proposed an approach for identifying the encoding technique to advance towards an expedited search over encrypted text leading to the security enhancements in big data.[80]

In March 2012, The White House announced a national “Big Data Initiative” that consisted of six Federal departments and agencies committing more than $200 million to big data research projects.[81]

The initiative included a National Science Foundation “Expeditions in Computing” grant of $10 million over 5 years to the AMPLab[82] at the University of California, Berkeley.[83] The AMPLab also received funds from DARPA, and over a dozen industrial sponsors and uses big data to attack a wide range of problems from predicting traffic congestion[84] to fighting cancer.[85]

The White House Big Data Initiative also included a commitment by the Department of Energy to provide $25 million in funding over 5 years to establish the Scalable Data Management, Analysis and Visualization (SDAV) Institute,[86] led by the Energy Department’s Lawrence Berkeley National Laboratory. The SDAV Institute aims to bring together the expertise of six national laboratories and seven universities to develop new tools to help scientists manage and visualize data on the Department’s supercomputers.

The U.S. state of Massachusetts announced the Massachusetts Big Data Initiative in May 2012, which provides funding from the state government and private companies to a variety of research institutions.[87] The Massachusetts Institute of Technology hosts the Intel Science and Technology Center for Big Data in the MIT Computer Science and Artificial Intelligence Laboratory, combining government, corporate, and institutional funding and research efforts.[88]

The European Commission is funding the 2-year-long Big Data Public Private Forum through their Seventh Framework Program to engage companies, academics and other stakeholders in discussing big data issues. The project aims to define a strategy in terms of research and innovation to guide supporting actions from the European Commission in the successful implementation of the big data economy. Outcomes of this project will be used as input for Horizon 2020, their next framework program.[89]

The British government announced in March 2014 the founding of the Alan Turing Institute, named after the computer pioneer and code-breaker, which will focus on new ways of collecting and analysing large sets of data.[90]

At the University of Waterloo Stratford Campus Canadian Open Data Experience (CODE) Inspiration Day, it was demonstrated how using data visualization techniques can increase the understanding and appeal of big data sets in order to communicate a story to the world.[91]

In order to make manufacturing more competitive in the United States (and globe), there is a need to integrate more American ingenuity and innovation into manufacturing ; Therefore, National Science Foundation has granted the Industry University cooperative research center for Intelligent Maintenance Systems (IMS) at university of Cincinnati to focus on developing advanced predictive tools and techniques to be applicable in a big data environment.[61][92] In May 2013, IMS Center held an industry advisory board meeting focusing on big data where presenters from various industrial companies discussed their concerns, issues and future goals in Big Data environment.

Computational social sciences — Anyone can use Application Programming Interfaces (APIs) provided by Big Data holders, such as Google and Twitter, to do research in the social and behavioral sciences.[93] Often these APIs are provided for free.[93] Tobias Preis *et al.* used Google Trends data to demonstrate that Internet users from countries with a higher per capita gross domestic product (GDP) are more likely to search for information about the future than information about the past. The findings suggest there may be a link between online behaviour and real-world economic indicators.[94][95][96] The authors of the study examined Google queries logs made by ratio of the volume of searches for the coming year (‘2011’) to the volume of searches for the previous year (‘2009’), which they call the ‘future orientation index’.[97] They compared the future orientation index to the per capita GDP of each country and found a strong tendency for countries in which Google users enquire more about the future to exhibit a higher GDP. The results hint that there may potentially be a relationship between the economic success of a country and the information-seeking behavior of its citizens captured in big data.

Tobias Preis and his colleagues Helen Susannah Moat and H. Eugene Stanley introduced a method to identify online precursors for stock market moves, using trading strategies based on search volume data provided by Google Trends.[98] Their analysis of Google search volume for 98 terms of varying financial relevance, published in *Scientific Reports*,[99] suggests that increases in search volume for financially relevant search terms tend to precede large losses in financial markets.[100][101][102][103][104][105][106][107]

10.7 Critique

Critiques of the big data paradigm come in two flavors, those that question the implications of the approach itself, and those that question the way it is currently done.

ARRIVALS
BIG DATA
TG'11
"Your recent Amazon purchases, Tweet score and location history makes you 23.5% welcome here."

Cartoon critical of big data application, by T. Gregorius

10.7.1 Critiques of the big data paradigm

"A crucial problem is that we do not know much about the underlying empirical micro-processes that lead to the emergence of the[se] typical network characteristics of Big Data".[14] In their critique, Snijders, Matzat, and Reips point out that often very strong assumptions are made about mathematical properties that may not at all reflect what is really going on at the level of micro-processes. Mark Graham has leveled broad critiques at Chris Anderson's assertion that big data will spell the end of theory: focusing in particular on the notion that big data will always need to be contextualized in their social, economic and political contexts.[108] Even as companies invest eight- and nine-figure sums to derive insight from information streaming in from suppliers and customers, less than 40% of employees have sufficiently mature processes and skills to do so. To overcome this insight deficit, "big data", no matter how comprehensive or well analyzed, needs to be complemented by "big judgment", according to an article in the Harvard Business Review.[109]

Much in the same line, it has been pointed out that the decisions based on the analysis of big data are inevitably "informed by the world as it was in the past, or, at best, as it currently is".[56] Fed by a large number of data on past experiences, algorithms can predict future development if the future is similar to the past. If the systems dynamics of the future change, the past can say little about the future. For this, it would be necessary to have a thorough understanding of the systems dynamic, which implies theory.[110] As a response to this critique it has been suggested to combine big data approaches with computer simulations, such as agent-based models[56] and Complex Systems.[111] Agent-based models are increasingly getting better in predicting the outcome of social complexities of even unknown future scenarios through computer simulations that are based on a collection of mutually interdependent algorithms.[112][113] In addition, use of multivariate methods that probe for the latent structure of the data, such as factor analysis and cluster analysis, have proven useful as analytic approaches that go well beyond the bi-variate approaches (cross-tabs) typically employed with smaller data sets.

In health and biology, conventional scientific approaches are based on experimentation. For these approaches, the limiting factor is the relevant data that can confirm or refute the initial hypothesis.[114] A new postulate is accepted now in biosciences: the information provided by the data in huge volumes (omics) without prior hypothesis is complementary and sometimes necessary to conventional approaches based on experimentation. In the massive approaches it is the formulation of a relevant hypothesis to explain the data that is the limiting factor. The search logic is reversed and the limits of induction ("Glory of Science and Philosophy scandal", C. D. Broad, 1926) are to be considered.

Privacy advocates are concerned about the threat to privacy represented by increasing storage and integration of personally identifiable information; expert panels have released various policy recommendations to conform practice to expectations of privacy.[115][116][117]

10.7.2 Critiques of big data execution

Big data has been called a "fad" in scientific research and its use was even made fun of as an absurd practice in a satirical example on "pig data".[93] Researcher danah boyd has raised concerns about the use of big data in science neglecting principles such as choosing a representative sample by being too concerned about actually handling the huge amounts of data.[118] This approach may lead to results bias in one way or another. Integration across heterogeneous data resources—some that might be considered "big data" and others not—presents formidable logistical as well as analytical challenges, but many researchers argue that such integrations are likely to represent the most promising new frontiers in science.[119] In the provocative article "Critical Questions for Big Data",[120] the authors title big data a part of mythology: "large data sets offer a higher form of intelligence and knowledge [...], with the aura of truth, objectivity, and accuracy". Users of big data are often "lost in the sheer volume of numbers", and "working with Big Data is still subjective, and what it quantifies does not necessarily have a closer claim on objective truth".[120] Recent developments in BI domain, such as pro-active reporting especially target improvements in usability of Big Data, through automated filtering of non-useful data and correlations.[121]

Big data analysis is often shallow compared to analysis of smaller data sets.[122] In many big data projects, there is no large data analysis happening, but the challenge is the extract, transform, load part of data preprocessing.[122]

Big data is a buzzword and a “vague term”,[123] but at the same time an “obsession”[123] with entrepreneurs, consultants, scientists and the media. Big data showcases such as Google Flu Trends failed to deliver good predictions in recent years, overstating the flu outbreaks by a factor of two. Similarly, Academy awards and election predictions solely based on Twitter were more often off than on target. Big data often poses the same challenges as small data; and adding more data does not solve problems of bias, but may emphasize other problems. In particular data sources such as Twitter are not representative of the overall population, and results drawn from such sources may then lead to wrong conclusions. Google Translate - which is based on big data statistical analysis of text - does a remarkably good job at translating web pages. However, results from specialized domains may be dramatically skewed. On the other hand, big data may also introduce new problems, such as the multiple comparisons problem: simultaneously testing a large set of hypotheses is likely to produce many false results that mistakenly appear to be significant. Ioannidis argued that “most published research findings are false” [124] due to essentially the same effect: when many scientific teams and researchers each perform many experiments (i.e. process a big amount of scientific data; although not with big data technology), the likelihood of a “significant” result being actually false grows fast - even more so, when only positive results are published.

10.8 See also

- Apache Accumulo
- Apache Hadoop
- Big Data to Knowledge
- Data Defined Storage
- Data mining
- Cask (company)
- Cloudera
- HPCC Systems
- Intelligent Maintenance Systems
- Internet of Things
- MapReduce
- Hortonworks
- Oracle NoSQL Database
- Nonlinear system identification
- Operations research
- Programming with Big Data in R (a series of R packages)
- Sqrrl
- Supercomputer
- Talend
- Transreality gaming
- Tuple space
- Unstructured data

10.9 References

[1] “Data, data everywhere”. *The Economist*. 25 February 2010. Retrieved 9 December 2012.

[2] “Community cleverness required”. *Nature* **455** (7209): 1. 4 September 2008. doi:10.1038/455001a.

[3] “Sandia sees data management challenges spiral”. *HPC Projects*. 4 August 2009.

[4] Reichman, O.J.; Jones, M.B.; Schildhauer, M.P. (2011). “Challenges and Opportunities of Open Data in Ecology”. *Science* **331** (6018): 703–5. doi:10.1126/science.1197962. PMID 21311007.

[5] “Data Crush by Christopher Surdak”. Retrieved 14 February 2014.

[6] Hellerstein, Joe (9 November 2008). “Parallel Programming in the Age of Big Data”. *Gigaom Blog*.

[7] Segaran, Toby; Hammerbacher, Jeff (2009). *Beautiful Data: The Stories Behind Elegant Data Solutions*. O'Reilly Media. p. 257. ISBN 978-0-596-15711-1.

[8] Hilbert & López 2011

[9] “IBM What is big data? — Bringing big data to the enterprise”. www.ibm.com. Retrieved 2013-08-26.

[10] Oracle and FSN, “Mastering Big Data: CFO Strategies to Transform Insight into Opportunity”, December 2012

[11] “Computing Platforms for Analytics, Data Mining, Data Science”. *kdnuggets.com*. Retrieved 15 April 2015.

[12] Jacobs, A. (6 July 2009). “The Pathologies of Big Data”. *ACMQueue*.

[13] Magoulas, Roger; Lorica, Ben (February 2009). “Introduction to Big Data”. *Release 2.0* (Sebastopol CA: O’Reilly Media) (11).

[14] Snijders, C.; Matzat, U.; Reips, U.-D. (2012). "'Big Data': Big gaps of knowledge in the field of Internet”. *International Journal of Internet Science* **7**: 1–5.

[15] Ibrahim; Targio Hashem, Abaker; Yaqoob, Ibrar; Badrul Anuar, Nor; Mokhtar, Salimah; Gani, Abdullah; Ullah Khan, Samee (2015). "big data" on cloud computing: Review and open research issues". *Information Systems* **47**: 98–115. doi:10.1016/j.is.2014.07.006.

[16] Laney, Douglas. "3D Data Management: Controlling Data Volume, Velocity and Variety" (PDF). Gartner. Retrieved 6 February 2001.

[17] Beyer, Mark. "Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data". Gartner. Archived from the original on 10 July 2011. Retrieved 13 July 2011.

[18] Laney, Douglas. "The Importance of 'Big Data': A Definition". Gartner. Retrieved 21 June 2012.

[19] "What is Big Data?". Villanova University.

[20] http://www.bigdataparis.com/presentation/mercredi/PDelort.pdf?PHPSESSID=tv7k70pcr3egpi2r6fi3qbjtj6#page=4

[21] Billings S.A. "Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains". Wiley, 2013

[22] Delort P., Big data Paris 2013 http://www.andsi.fr/tag/dsi-big-data/

[23] Delort P., Big Data car Low-Density Data ? La faible densité en information comme facteur discriminant http://lecercle.lesechos.fr/entrepreneur/tendances-innovation/221169222/big-data-low-density-data-faible-densite-information-com

[24] De Mauro, Andrea; Greco, Marco; Grimaldi, Michele (2015). "What is big data? A consensual definition and a review of key research topics". *AIP Conference Proceedings* **1644**: 97–104. doi:10.1063/1.4907823.

[25] Lee, Jay; Bagheri, Behrad; Kao, Hung-An (2014). "Recent Advances and Trends of Cyber-Physical Systems and Big Data Analytics in Industrial Informatics". *IEEE Int. Conference on Industrial Informatics (INDIN) 2014*.

[26] Lee, Jay; Lapira, Edzel; Bagheri, Behrad; Kao, Hung-an. "Recent advances and trends in predictive manufacturing systems in big data environment". *Manufacturing Letters* **1** (1): 38–41. doi:10.1016/j.mfglet.2013.09.005.

[27] "LexisNexis To Buy Seisint For $775 Million". Washington Post. Retrieved 15 July 2004.

[28] "LexisNexis Parent Set to Buy ChoicePoint". Washington Post. Retrieved 22 February 2008.

[29] "Quantcast Opens Exabyte-Ready File System". www.datanami.com. Retrieved 1 October 2012.

[30] Bertolucci, Jeff "Hadoop: From Experiment To Leading Big Data Platform", "Information Week", 2013. Retrieved on 14 November 2013.

[31] Webster, John. "MapReduce: Simplified Data Processing on Large Clusters", "Search Storage", 2004. Retrieved on 25 March 2013.

[32] "Big Data Solution Offering". MIKE2.0. Retrieved 8 Dec 2013.

[33] "Big Data Definition". MIKE2.0. Retrieved 9 March 2013.

[34] Boja, C; Pocovnicu, A; Bătăgan, L. (2012). "Distributed Parallel Architecture for Big Data". *Informatica Economica* **16** (2): 116–127.

[35] Intelligent Maintenance System

[36] http://www.hcltech.com/sites/default/files/solving_key_businesschallenges_with_big_data_lake_0.pdf

[37] Manyika, James; Chui, Michael; Bughin, Jaques; Brown, Brad; Dobbs, Richard; Roxburgh, Charles; Byers, Angela Hung (May 2011). *Big Data: The next frontier for innovation, competition, and productivity*. McKinsey Global Institute.

[38] "Future Directions in Tensor-Based Computation and Modeling" (PDF). May 2009.

[39] Lu, Haiping; Plataniotis, K.N.; Venetsanopoulos, A.N. (2011). "A Survey of Multilinear Subspace Learning for Tensor Data" (PDF). *Pattern Recognition* **44** (7): 1540–1551. doi:10.1016/j.patcog.2011.01.004.

[40] Monash, Curt (30 April 2009). "eBay's two enormous data warehouses".
Monash, Curt (6 October 2010). "eBay followup — Greenplum out, Teradata > 10 petabytes, Hadoop has some value, and more".

[41] "Resources on how Topological Data Analysis is used to analyze big data". Ayasdi.

[42] CNET News (1 April 2011). "Storage area networks need not apply".

[43] "How New Analytic Systems will Impact Storage". September 2011.

[44] "What Is the Content of the World's Technologically Mediated Information and Communication Capacity: How Much Text, Image, Audio, and Video?", Martin Hilbert (2014), The Information Society; free access to the article through this link: martinhilbert.net/WhatsTheContent_Hilbert.pdf

[45] Rajpurohit, Anmol (2014-07-11). "Interview: Amy Gershkoff, Director of Customer Analytics & Insights, eBay on How to Design Custom In-House BI Tools". *KDnuggets*. Retrieved 2014-07-14. Dr. Amy Gershkoff: "Generally, I find that off-the-shelf business intelligence tools do not meet the needs of clients who want to derive custom insights from their data. Therefore, for medium-to-large organizations with access to strong technical talent, I usually recommend building custom, in-house solutions."

[46] Kalil, Tom. "Big Data is a Big Deal". White House. Retrieved 26 September 2012.

[47] Executive Office of the President (March 2012). "Big Data Across the Federal Government" (PDF). White House. Retrieved 26 September 2012.

[48] Lampitt, Andrew. "The real story of how big data analytics helped Obama win". *Infoworld*. Retrieved 31 May 2014.

[49] Hoover, J. Nicholas. "Government's 10 Most Powerful Supercomputers". *Information Week*. UBM. Retrieved 26 September 2012.

[50] Bamford, James (15 March 2012). "The NSA Is Building the Country's Biggest Spy Center (Watch What You Say)". *Wired Magazine*. Retrieved 2013-03-18.

[51] "Groundbreaking Ceremony Held for $1.2 Billion Utah Data Center". National Security Agency Central Security Service. Retrieved 2013-03-18.

[52] Hill, Kashmir. "TBlueprints Of NSA's Ridiculously Expensive Data Center In Utah Suggest It Holds Less Info Than Thought". *Forbes*. Retrieved 2013-10-31.

[53] "News: Live Mint". *Are Indian companies making enough sense of Big Data?*. Live Mint - http://www.livemint.com/. 2014-06-23. Retrieved 2014-11-22.

[54] UN GLobal Pulse (2012). Big Data for Development: Opportunities and Challenges (White p. by Letouzé, E.). New York: United Nations. Retrieved from http://www.unglobalpulse.org/projects/BigDataforDevelopment

[55] WEF (World Economic Forum), & Vital Wave Consulting. (2012). Big Data, Big Impact: New Possibilities for International Development. World Economic Forum. Retrieved 24 August 2012, from http://www.weforum.org/reports/big-data-big-impact-new-possibilities-international-development

[56] "Big Data for Development: From Information- to Knowledge Societies", Martin Hilbert (2013), SSRN Scholarly Paper No. ID 2205145). Rochester, NY: Social Science Research Network; http://papers.ssrn.com/abstract=2205145

[57] "Elena Kvochko, Four Ways To talk About Big Data (Information Communication Technologies for Development Series)". worldbank.org. Retrieved 2012-05-30.

[58] "Daniele Medri: Big Data & Business: An on-going revolution". Statistics Views. 21 Oct 2013.

[59] "Manufacturing: Big Data Benefits and Challenges". *TCS Big Data Study*. Mumbai, India: Tata Consultancy Services Limited. Retrieved 2014-06-03.

[60] Lee, Jay; Wu, F.; Zhao, W.; Ghaffari, M.; Liao, L (Jan 2013). "Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications". *Mechanical Systems and Signal Processing* **42** (1).

[61] "Center for Intelligent Maintenance Systems (IMS Center)".

[62] Predictive manufacturing system

[63] Couldry, Nick; Turow, Joseph (2014). "Advertising, Big Data, and the Clearance of the Public Realm: Marketers' New Approaches to the Content Subsidy". *International Journal of Communication* **8**: 1710–1726.

[64] Layton, Julia. "Amazon Technology". Money.howstuffworks.com. Retrieved 2013-03-05.

[65] "Scaling Facebook to 500 Million Users and Beyond". Facebook.com. Retrieved 2013-07-21.

[66] "Google Still Doing At Least 1 Trillion Searches Per Year". *Search Engine Land*. 16 January 2015. Retrieved 15 April 2015.

[67] Lamb, Charles. "Oracle NoSQL Database Exceeds 1 Million Mixed YCSB Ops/Sec".

[68] "FICO® Falcon® Fraud Manager". Fico.com. Retrieved 2013-07-21.

[69] "eBay Study: How to Build Trust and Improve the Shopping Experience". Knowwpcarey.com. 2012-05-08. Retrieved 2013-03-05.

[70] Leading Priorities for Big Data for Business and IT. eMarketer. October 2013. Retrieved January 2014.

[71] Wingfield, Nick (2013-03-12). "Predicting Commutes More Accurately for Would-Be Home Buyers - NYTimes.com". Bits.blogs.nytimes.com. Retrieved 2013-07-21.

[72] Alexandru, Dan. "Prof" (PDF). *cds.cern.ch*. CERN. Retrieved 24 March 2015.

[73] "LHC Brochure, English version. A presentation of the largest and the most powerful particle accelerator in the world, the Large Hadron Collider (LHC), which started up in 2008. Its role, characteristics, technologies, etc. are explained for the general public.". *CERN-Brochure-2010-006-Eng. LHC Brochure, English version*. CERN. Retrieved 20 January 2013.

[74] "LHC Guide, English version. A collection of facts and figures about the Large Hadron Collider (LHC) in the form of questions and answers.". *CERN-Brochure-2008-001-Eng. LHC Guide, English version*. CERN. Retrieved 20 January 2013.

[75] Brumfiel, Geoff (19 January 2011). "High-energy physics: Down the petabyte highway". *Nature* **469**. pp. 282–83. doi:10.1038/469282a.

[76] http://www.zurich.ibm.com/pdf/astron/CeBIT%202013%20Background%20DOME.pdf

[77] "Future telescope array drives development of exabyte processing". *Ars Technica*. Retrieved 15 April 2015.

[78] Delort P., OECD ICCP Technology Foresight Forum, 2012. http://www.oecd.org/sti/ieconomy/Session_3_Delort.pdf#page=6

[79] Webster, Phil. "Supercomputing the Climate: NASA's Big Data Mission". *CSC World*. Computer Sciences Corporation. Retrieved 2013-01-18.

[80] Siwach, Gautam; Esmailpour, Amir (March 2014). *Encrypted Search & Cluster Formation in Big Data* (PDF). ASEE 2014 Zone I Conference. University of Bridgeport, Bridgeport, Connecticut, USA.

[81] "Obama Administration Unveils "Big Data" Initiative: Announces $200 Million In New R&D Investments" (PDF). The White House.

[82] "AMPLab at the University of California, Berkeley". Amplab.cs.berkeley.edu. Retrieved 2013-03-05.

[83] "NSF Leads Federal Efforts In Big Data". National Science Foundation (NSF). 29 March 2012.

[84] Timothy Hunter; Teodor Moldovan; Matei Zaharia; Justin Ma; Michael Franklin; Pieter Abbeel; Alexandre Bayen (October 2011). *Scaling the Mobile Millennium System in the Cloud*.

[85] David Patterson (5 December 2011). "Computer Scientists May Have What It Takes to Help Cure Cancer". The New York Times.

[86] "Secretary Chu Announces New Institute to Help Scientists Improve Massive Data Set Research on DOE Supercomputers". "energy.gov".

[87] "Governor Patrick announces new initiative to strengthen Massachusetts' position as a World leader in Big Data". Commonwealth of Massachusetts.

[88] "Big Data @ CSAIL". Bigdata.csail.mit.edu. 2013-02-22. Retrieved 2013-03-05.

[89] "Big Data Public Private Forum". Cordis.europa.eu. 2012-09-01. Retrieved 2013-03-05.

[90] "Alan Turing Institute to be set up to research big data". BBC News. 19 March 2014. Retrieved 2014-03-19.

[91] "Inspiration day at University of Waterloo, Stratford Campus". http://www.betakit.com/. Retrieved 2014-02-28.

[92] Lee, Jay; Lapira, Edzel; Bagheri, Behrad; Kao, Hung-An (2013). "Recent Advances and Trends in Predictive Manufacturing Systems in Big Data Environment". *Manufacturing Letters* **1** (1): 38–41. doi:10.1016/j.mfglet.2013.09.005.

[93] Reips, Ulf-Dietrich; Matzat, Uwe (2014). "Mining "Big Data" using Big Data Services". *International Journal of Internet Science* **1** (1): 1–8.

[94] Preis, Tobias; Moat,, Helen Susannah; Stanley, H. Eugene; Bishop, Steven R. (2012). "Quantifying the Advantage of Looking Forward". *Scientific Reports* **2**: 350. doi:10.1038/srep00350. PMC 3320057. PMID 22482034.

[95] Marks, Paul (5 April 2012). "Online searches for future linked to economic success". *New Scientist*. Retrieved 9 April 2012.

[96] Johnston, Casey (6 April 2012). "Google Trends reveals clues about the mentality of richer nations". *Ars Technica*. Retrieved 9 April 2012.

[97] Tobias Preis (2012-05-24). "Supplementary Information: The Future Orientation Index is available for download" (PDF). Retrieved 2012-05-24.

[98] Philip Ball (26 April 2013). "Counting Google searches predicts market movements". *Nature*. Retrieved 9 August 2013.

[99] Tobias Preis, Helen Susannah Moat and H. Eugene Stanley (2013). "Quantifying Trading Behavior in Financial Markets Using Google Trends". *Scientific Reports* **3**: 1684. doi:10.1038/srep01684.

[100] Nick Bilton (26 April 2013). "Google Search Terms Can Predict Stock Market, Study Finds". *New York Times*. Retrieved 9 August 2013.

[101] Christopher Matthews (26 April 2013). "Trouble With Your Investment Portfolio? Google It!". *TIME Magazine*. Retrieved 9 August 2013.

[102] Philip Ball (26 April 2013). "Counting Google searches predicts market movements". *Nature*. Retrieved 9 August 2013.

[103] Bernhard Warner (25 April 2013). "'Big Data' Researchers Turn to Google to Beat the Markets". *Bloomberg Businessweek*. Retrieved 9 August 2013.

[104] Hamish McRae (28 April 2013). "Hamish McRae: Need a valuable handle on investor sentiment? Google it". *The Independent* (London). Retrieved 9 August 2013.

[105] Richard Waters (25 April 2013). "Google search proves to be new word in stock market prediction". *Financial Times*. Retrieved 9 August 2013.

[106] David Leinweber (26 April 2013). "Big Data Gets Bigger: Now Google Trends Can Predict The Market". *Forbes*. Retrieved 9 August 2013.

[107] Jason Palmer (25 April 2013). "Google searches predict market moves". *BBC*. Retrieved 9 August 2013.

[108] Graham M. (9 March 2012). "Big data and the end of theory?". *The Guardian* (London).

[109] "Good Data Won't Guarantee Good Decisions. Harvard Business Review". *Shah, Shvetank; Horne, Andrew; Capellá, Jaime;*. HBR.org. Retrieved 8 September 2012.

[110] Anderson, C. (2008, 23 June). The End of Theory: The Data Deluge Makes the Scientific Method Obsolete. Wired Magazine, (Science: Discoveries). http://www.wired.com/science/discoveries/magazine/16-07/pb_theory

[111] Braha, D.; Stacey, B.; Bar-Yam, Y. (2011). "Corporate Competition: A Self-organized Network". Social Networks 33: 219–230.

[112] Rauch, J. (2002). Seeing Around Corners. The Atlantic, (April), 35–48. http://www.theatlantic.com/magazine/archive/2002/04/seeing-around-corners/302471/

[113] Epstein, J. M., & Axtell, R. L. (1996). Growing Artificial Societies: Social Science from the Bottom Up. A Bradford Book.

[114] Delort P., Big data in Biosciences, Big Data Paris, 2012 http://www.bigdataparis.com/documents/Pierre-Delort-INSERM.pdf#page=5

[115] Ohm, Paul. "Don't Build a Database of Ruin". Harvard Business Review.

[116] Darwin Bond-Graham, *Iron Cagebook - The Logical End of Facebook's Patents,* Counterpunch.org, 2013.12.03

[117] Darwin Bond-Graham, *Inside the Tech industry's Startup Conference,* Counterpunch.org, 2013.09.11

[118] danah boyd (2010-04-29). "Privacy and Publicity in the Context of Big Data". *WWW 2010 conference*. Retrieved 2011-04-18.

[119] Jones, MB; Schildhauer, MP; Reichman, OJ; Bowers, S (2006). "The New Bioinformatics: Integrating Ecological Data from the Gene to the Biosphere" (PDF). *Annual Review of Ecology, Evolution, and Systematics* **37** (1): 519–544. doi:10.1146/annurev.ecolsys.37.091305.110031.

[120] Boyd, D.; Crawford, K. (2012). "Critical Questions for Big Data". *Information, Communication & Society* **15** (5): 662. doi:10.1080/1369118X.2012.678878.

[121] Failure to Launch: From Big Data to Big Decisions, Forte Wares.

[122] Gregory Piatetsky (2014-08-12). "Interview: Michael Berthold, KNIME Founder, on Research, Creativity, Big Data, and Privacy, Part 2". KDnuggets. Retrieved 2014-08-13.

[123] Harford, Tim (2014-03-28). "Big data: are we making a big mistake?". *Financial Times*. Financial Times. Retrieved 2014-04-07.

[124] Ioannidis, J. P. A. (2005). "Why Most Published Research Findings Are False". *PLoS Medicine* **2** (8): e124. doi:10.1371/journal.pmed.0020124. PMC 1182327. PMID 16060722.

10.10 Further reading

- Sharma, Sugam; Tim, Udoyara S; Wong, Johnny; Gadia, Shashi; Sharma, Subhash (2014). "A BRIEF REVIEW ON LEADING BIG DATA MODELS". *Data Science Journal* **13**.
- Big Data Computing and Clouds: Challenges, Solutions, and Future Directions. Marcos D. Assuncao, Rodrigo N. Calheiros, Silvia Bianchi, Marco A. S. Netto, Rajkumar Buyya. Technical Report CLOUDS-TR-2013-1, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, 17 Dec. 2013.
- Encrypted search & cluster formation in Big Data. Gautam Siwach, Dr. A. Esmailpour. American Society for Engineering Education, Conference at the University of Bridgeport, Bridgeport, Connecticut 3–5 April 2014.
- "Big Data for Good" (PDF). ODBMS.org. 5 June 2012. Retrieved 2013-11-12.
- Hilbert, Martin; López, Priscila (2011). "The World's Technological Capacity to Store, Communicate, and Compute Information". *Science* **332** (6025): 60–65. doi:10.1126/science.1200970. PMID 21310967.
- "The Rise of Industrial Big Data". GE Intelligent Platforms. Retrieved 2013-11-12.
- History of Big Data Timeline. A visual history of Big Data with links to supporting articles.

10.11 External links

- Media related to Big data at Wikimedia Commons
- The dictionary definition of big data at Wiktionary
- MIT Big Data Initiative / Tackling the Challenges of Big Data

Chapter 11

Euclidean distance

In mathematics, the **Euclidean distance** or **Euclidean metric** is the “ordinary” (i.e straight line) distance between two points in Euclidean space. With this distance, Euclidean space becomes a metric space. The associated norm is called the **Euclidean norm.** Older literature refers to the metric as **Pythagorean metric**. A generalized term for the Euclidean norm is the **L^2 norm** or L^2 distance.

11.1 Definition

The **Euclidean distance** between points **p** and **q** is the length of the line segment connecting them ($\overline{\mathbf{pq}}$).

In Cartesian coordinates, if **p** = (p_1, p_2,..., pn) and **q** = (q_1, q_2,..., qn) are two points in Euclidean n-space, then the distance (d) from **p** to **q**, or from **q** to **p** is given by the Pythagorean formula:

The position of a point in a Euclidean n-space is a Euclidean vector. So, **p** and **q** are Euclidean vectors, starting from the origin of the space, and their tips indicate two points. The **Euclidean norm**, or **Euclidean length**, or **magnitude** of a vector measures the length of the vector:

$$\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + \cdots + p_n^2} = \sqrt{\mathbf{p}\cdot\mathbf{p}},$$

where the last equation involves the dot product.

A vector can be described as a directed line segment from the origin of the Euclidean space (vector tail), to a point in that space (vector tip). If we consider that its length is actually the distance from its tail to its tip, it becomes clear that the Euclidean norm of a vector is just a special case of Euclidean distance: the Euclidean distance between its tail and its tip.

The distance between points **p** and **q** may have a direction (e.g. from **p** to **q**), so it may be represented by another vector, given by

$$\mathbf{q} - \mathbf{p} = (q_1 - p_1, q_2 - p_2, \cdots, q_n - p_n)$$

In a three-dimensional space (n=3), this is an arrow from **p** to **q**, which can be also regarded as the position of **q** relative to **p**. It may be also called a displacement vector if **p** and **q** represent two positions of the same point at two successive instants of time.

The Euclidean distance between **p** and **q** is just the Euclidean length of this distance (or displacement) vector:

which is equivalent to equation 1, and also to:

$$\|\mathbf{q} - \mathbf{p}\| = \sqrt{\|\mathbf{p}\|^2 + \|\mathbf{q}\|^2 - 2\mathbf{p}\cdot\mathbf{q}}.$$

11.1.1 One dimension

In one dimension, the distance between two points on the real line is the absolute value of their numerical difference. Thus if x and y are two points on the real line, then the distance between them is given by:

$$\sqrt{(x-y)^2} = |x - y|.$$

In one dimension, there is a single homogeneous, translation-invariant metric (in other words, a distance that is induced by a norm), up to a scale factor of length, which is the Euclidean distance. In higher dimensions there are other possible norms.

11.1.2 Two dimensions

In the Euclidean plane, if **p** = (p_1, p_2) and **q** = (q_1, q_2) then the distance is given by

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

This is equivalent to the Pythagorean theorem.

Alternatively, it follows from (2) that if the polar coordinates of the point **p** are (r_1, θ$_1$) and those of **q** are (r_2, θ$_2$), then the distance between the points is

$$\sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\theta_1 - \theta_2)}.$$

11.1.3 Three dimensions

In three-dimensional Euclidean space, the distance is

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.$$

11.1.4 *n* dimensions

In general, for an n-dimensional space, the distance is

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

11.1.5 Squared Euclidean distance

The standard Euclidean distance can be squared in order to place progressively greater weight on objects that are farther apart. In this case, the equation becomes

$$d^2(p,q) = (p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2.$$

Squared Euclidean Distance is not a metric as it does not satisfy the triangle inequality, however, it is frequently used in optimization problems in which distances only have to be compared.

It is also referred to as quadrance within the field of rational trigonometry.

11.2 See also

- Chebyshev distance measures distance assuming only the most significant dimension is relevant.
- Euclidean distance matrix
- Hamming distance identifies the difference bit by bit of two strings
- Mahalanobis distance normalizes based on a covariance matrix to make the distance metric scale-invariant.
- Manhattan distance measures distance following only axis-aligned directions.
- Metric
- Minkowski distance is a generalization that unifies Euclidean distance, Manhattan distance, and Chebyshev distance.
- Pythagorean addition

11.3 References

- Deza, Elena; Deza, Michel Marie (2009). *Encyclopedia of Distances*. Springer. p. 94.
- "Cluster analysis". March 2, 2011.

Chapter 12

Hamming distance

In information theory, the **Hamming distance** between two strings of equal length is the number of positions at which the corresponding symbols are different. In another way, it measures the minimum number of *substitutions* required to change one string into the other, or the minimum number of *errors* that could have transformed one string into the other.

A major application is in coding theory, more specifically to block codes, in which the equal-length strings are vectors over a finite field.

12.1 Examples

The Hamming distance between:

- "**karolin**" and "**kathrin**" is 3.
- "**karolin**" and "**kerstin**" is 3.
- **1011101** and **1001001** is 2.
- **2173896** and **2233796** is 3.

On a two-dimensional grid such as a chessboard, the Hamming distance is the minimum number of moves it would take a rook to move from one cell to the other.

12.2 Properties

For a fixed length n, the Hamming distance is a metric on the vector space of the words of length n (also known as a Hamming space), as it fulfills the conditions of non-negativity, identity of indiscernibles and symmetry, and it can be shown by complete induction that it satisfies the triangle inequality as well.[1] The Hamming distance between two words a and b can also be seen as the Hamming weight of a–b for an appropriate choice of the – operator.

For binary strings a and b the Hamming distance is equal to the number of ones (population count) in a XOR b. The metric space of length-n binary strings, with the Hamming distance, is known as the *Hamming cube*; it is equivalent as a metric space to the set of distances between vertices in a hypercube graph. One can also view a binary string of length n as a vector in R^n by treating each symbol in the string as a real coordinate; with this embedding, the strings form the vertices of an n-dimensional hypercube, and the Hamming distance of the strings is equivalent to the Manhattan distance between the vertices.

12.3 Error detection and error correction

The Hamming distance is used to define some essential notions in coding theory, such as error detecting and error correcting codes. In particular, a code C is said to be *k-errors detecting* if any two codewords c_1 and c_2 from C that have a Hamming distance less than k coincide; Otherwise put it, a code is k-errors detecting if and only if the minimum Hamming distance between any two of its codewords is at least k+1.[1]

A code C is said to be *k-errors correcting* if for every word w in the underlying Hamming space H there exists at most one codeword c (from C) such that the Hamming distance between w and c is less than k. In other words, a code is k-errors correcting if and only if the minimum Hamming distance between any two of its codewords is at least 2k+1. This is more easily understood geometrically as any closed balls of radius k centered on distinct codewords being disjoint.[1] These balls are also called **Hamming spheres** in this context.[2]

Thus a code with minimum Hamming distance d between its codewords can detect at most $d-1$ errors and can correct $\lfloor(d-1)/2\rfloor$ errors.[1] The latter number is also called the *packing radius* or the *error-correcting capability* of the code.[2]

12.4 History and applications

The Hamming distance is named after Richard Hamming, who introduced it in his fundamental paper on Hamming codes *Error detecting and error correcting* *codes* in 1950.[3] Hamming weight analysis of bits is used in several disciplines including information theory, coding theory, and cryptography.

It is used in telecommunication to count the number of flipped bits in a fixed-length binary word as an estimate of error, and therefore is sometimes called the **signal distance**. For q-ary strings over an alphabet of size $q \geq 2$ the Hamming distance is applied in case of the q-ary symmetric channel, while the Lee distance is used for phase-shift keying or more generally channels susceptible to synchronization errors because the Lee distance accounts for errors of ± 1.[4] If $q = 2$ or $q = 3$ both distances coincide because Z/2Z and Z/3Z are also fields, but Z/4Z is not a field but only a ring.

The Hamming distance is also used in systematics as a measure of genetic distance.[5]

However, for comparing strings of different lengths, or strings where not just substitutions but also insertions or deletions have to be expected, a more sophisticated metric like the Levenshtein distance is more appropriate.

12.5 Algorithm example

The Python function hamming_distance() computes the Hamming distance between two strings (or other iterable objects) of equal length, by creating a sequence of Boolean values indicating mismatches and matches between corresponding positions in the two inputs, and then summing the sequence with False and True values being interpreted as zero and one.

def hamming_distance(s1, s2): """Return the Hamming distance between equal-length sequences""" if len(s1) != len(s2): raise ValueError(“Undefined for sequences of unequal length”) return sum(ch1 != ch2 for ch1, ch2 in zip(s1, s2))

The following C function will compute the Hamming distance of two integers (considered as binary values, that is, as sequences of bits). The running time of this procedure is proportional to the Hamming distance rather than to the number of bits in the inputs. It computes the bitwise exclusive or of the two inputs, and then finds the Hamming weight of the result (the number of nonzero bits) using an algorithm of Wegner (1960) that repeatedly finds and clears the lowest-order nonzero bit.

int hamming_distance(unsigned x, unsigned y) { int dist = 0; unsigned val = x ^ y; // Count the number of bits set while (val != 0) { // A bit is set, so increment the count and clear the bit dist++; val &= val - 1; } // Return the number of differing bits return dist; }

12.6 See also

- Closest string
- Damerau–Levenshtein distance
- Euclidean distance
- Mahalanobis distance
- Jaccard index
- String metric
- Sørensen similarity index
- Word ladder

12.7 Notes

[1] Derek J.S. Robinson (2003). *An Introduction to Abstract Algebra*. Walter de Gruyter. pp. 255–257. ISBN 978-3-11-019816-4.

[2] Cohen, G.; Honkala, I.; Litsyn, S.; Lobstein, A. (1997), *Covering Codes*, North-Holland Mathematical Library **54**, Elsevier, pp. 16–17, ISBN 9780080530079

[3] Hamming (1950).

[4] Ron Roth (2006). *Introduction to Coding Theory*. Cambridge University Press. p. 298. ISBN 978-0-521-84504-5.

[5] Pilcher, Wong & Pillai (2008).

12.8 References

- This article incorporates public domain material from the General Services Administration document “Federal Standard 1037C”.
- Hamming, Richard W. (1950), “Error detecting and error correcting codes” (PDF), *Bell System Technical Journal* **29** (2): 147–160, doi:10.1002/j.1538-7305.1950.tb00463.x, MR 0035935.
- Pilcher, C. D.; Wong, J. K.; Pillai, S. K. (March 2008), “Inferring HIV transmission dynamics from phylogenetic sequence relationships”, *PLoS Med.* **5** (3): e69, doi:10.1371/journal.pmed.0050069, PMC 2267810, PMID 18351799.
- Wegner, Peter (1960), “A technique for counting ones in a binary computer”, *Communications of the ACM* **3** (5): 322, doi:10.1145/367236.367286.

Chapter 13

Norm (mathematics)

This article is about linear algebra and analysis. For field theory, see Field norm. For ideals, see Ideal norm. For group theory, see Norm (group). For norms in descriptive set theory, see prewellordering.

In linear algebra, functional analysis and related areas of mathematics, a **norm** is a function that assigns a strictly positive *length* or *size* to each vector in a vector space—save for the zero vector, which is assigned a length of zero. A **seminorm**, on the other hand, is allowed to assign zero length to some non-zero vectors (in addition to the zero vector).

A norm must also satisfy certain properties pertaining to scalability and additivity which are given in the formal definition below.

A simple example is the 2-dimensional Euclidean space $\mathbf{R}^2$ equipped with the Euclidean norm. Elements in this vector space (e.g., (3, 7)) are usually drawn as arrows in a 2-dimensional cartesian coordinate system starting at the origin (0, 0). The Euclidean norm assigns to each vector the length of its arrow. Because of this, the Euclidean norm is often known as the magnitude.

A vector space on which a norm is defined is called a normed vector space. Similarly, a vector space with a seminorm is called a seminormed vector space. It is often possible to supply a norm for a given vector space in more than one way.

13.1 Definition

Given a vector space V over a subfield F of the complex numbers, a **norm** on V is a function $p\colon V \to \mathbf{R}$ with the following properties:[1]

For all $a \in F$ and all $\mathbf{u}, \mathbf{v} \in V$,

1. $p(a\mathbf{v}) = |a|\, p(\mathbf{v})$, (*absolute homogeneity* or *absolute scalability*).
2. $p(\mathbf{u} + \mathbf{v}) \le p(\mathbf{u}) + p(\mathbf{v})$ (*triangle inequality* or *subadditivity*).
3. If $p(\mathbf{v}) = 0$ then $\mathbf{v}$ is the zero vector (*separates points*).

By the first axiom, absolute homogeneity, we have $p(\mathbf{0}) = 0$ and $p(-\mathbf{v}) = p(\mathbf{v})$, so that by the triangle inequality

$p(\mathbf{v}) \ge 0$ (*positivity*).

A **seminorm** on V is a function $p : V \to \mathbf{R}$ with the properties 1. and 2. above.

Every vector space V with seminorm p induces a normed space V/W, called the quotient space, where W is the subspace of V consisting of all vectors $\mathbf{v}$ in V with $p(\mathbf{v}) = 0$. The induced norm on V/W is clearly well-defined and is given by:

$$p(W + \mathbf{v}) = p(\mathbf{v}).$$

Two norms (or seminorms) p and q on a vector space V are **equivalent** if there exist two real constants c and C, with $c > 0$ such that

for every vector $\mathbf{v}$ in V, one has that: $c\, q(\mathbf{v}) \le p(\mathbf{v}) \le C\, q(\mathbf{v})$.

A topological vector space is called **normable** (**seminormable**) if the topology of the space can be induced by a norm (seminorm).

13.2 Notation

If a norm $p : V \to \mathbf{R}$ is given on a vector space V then the norm of a vector $\mathbf{v} \in V$ is usually denoted by enclosing it within double vertical lines: $\|\mathbf{v}\| = p(\mathbf{v})$. Such notation is also sometimes used if p is only a seminorm.

For the length of a vector in Euclidean space (which is an example of a norm, as explained below), the notation $|\mathbf{v}|$ with single vertical lines is also widespread.

In Unicode, the codepoint of the "double vertical line" character ‖ is U+2016. The double vertical line should not be confused with the "parallel to" symbol, Unicode U+2225 (∥). This is usually not a problem because the former is used in parenthesis-like fashion, whereas the latter is used as an infix operator. The double vertical line used here should also not be confused with the symbol used to denote lateral clicks, Unicode U+01C1 (ǁ). The single vertical line | is called “vertical line” in Unicode and its codepoint is U+007C.

13.3 Examples

- All norms are seminorms.
- The *trivial seminorm* has $p(\mathbf{x}) = 0$ for all $\mathbf{x}$ in V.
- Every linear form f on a vector space defines a seminorm by $\mathbf{x} \to |f(\mathbf{x})|$.

13.3.1 Absolute-value norm

The absolute value

$$\|x\| = |x|$$

is a norm on the one-dimensional vector spaces formed by the real or complex numbers.

13.3.2 Euclidean norm

Main article: Euclidean distance

On an n-dimensional Euclidean space $\mathbf{R}^n$, the intuitive notion of length of the vector $\boldsymbol{x} = (x_1, x_2, ..., xn)$ is captured by the formula

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$$

This gives the ordinary distance from the origin to the point $\boldsymbol{x}$, a consequence of the Pythagorean theorem. The Euclidean norm is by far the most commonly used norm on $\mathbf{R}^n$, but there are other norms on this vector space as will be shown below. However all these norms are equivalent in the sense that they all define the same topology.

On an n-dimensional complex space $\mathbf{C}^n$ the most common norm is

$$\|\boldsymbol{z}\| := \sqrt{|z_1|^2 + \cdots + |z_n|^2} = \sqrt{z_1 \bar{z}_1 + \cdots + z_n \bar{z}_n}.$$

In both cases we can also express the norm as the square root of the inner product of the vector and itself:

$$\|\boldsymbol{x}\| := \sqrt{\boldsymbol{x}^* \, \boldsymbol{x}},$$

where $\boldsymbol{x}$ is represented as a column vector ($[x_1; x_2; ...; xn]$), and $\boldsymbol{x}^*$ denotes its conjugate transpose.

This formula is valid for any inner product space, including Euclidean and complex spaces. For Euclidean spaces, the inner product is equivalent to the dot product. Hence, in this specific case the formula can be also written with the following notation:

$$\|\boldsymbol{x}\| := \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}.$$

The Euclidean norm is also called the **Euclidean length**, **L^2 distance**, **ℓ^2 distance**, **L^2 norm**, or **ℓ^2 norm**; see L^p space.

The set of vectors in $\mathbf{R}^{n+1}$ whose Euclidean norm is a given positive constant forms an n-sphere.

Euclidean norm of a complex number

The Euclidean norm of a complex number is the absolute value (also called the **modulus**) of it, if the complex plane is identified with the Euclidean plane $\mathbf{R}^2$. This identification of the complex number $\mathbf{x} + \boldsymbol{i}\mathbf{y}$ as a vector in the Euclidean plane, makes the quantity $\sqrt{x^2 + y^2}$ (as first suggested by Euler) the Euclidean norm associated with the complex number.

13.3.3 Taxicab norm or Manhattan norm

Main article: Taxicab geometry

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i| \,.$$

The name relates to the distance a taxi has to drive in a rectangular street grid to get from the origin to the point x.

The set of vectors whose 1-norm is a given constant forms the surface of a cross polytope of dimension equivalent to that of the norm minus 1. The Taxicab norm is also called the **L_1 norm**. The distance derived from this norm is called the Manhattan distance or **L_1 distance**.

The 1-norm is simply the sum of the absolute values of the columns.

In contrast,

$$\sum_{i=1}^{n} x_i$$

is not a norm because it may yield negative results.

13.3.4 *p*-norm

Main article: Lᵖ space

Let $p \geq 1$ be a real number.

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}.$$

Note that for $p = 1$ we get the taxicab norm, for $p = 2$ we get the Euclidean norm, and as p approaches ∞ the p-norm approaches the infinity norm or maximum norm. Note that the p-norm is related to the Hölder mean.

This definition is still of some interest for $0 < p < 1$, but the resulting function does not define a norm,[2] because it violates the triangle inequality. What is true for this case of $0 < p < 1$, even in the measurable analog, is that the corresponding L^p class is a vector space, and it is also true that the function

$$\int_X |f(x) - g(x)|^p \, \mathrm{d}\mu$$

(without pth root) defines a distance that makes $L^p(X)$ into a complete metric topological vector space. These spaces are of great interest in functional analysis, probability theory, and harmonic analysis. However, outside trivial cases, this topological vector space is not locally convex and has no continuous nonzero linear forms. Thus the topological dual space contains only the zero functional.

The derivative of the p-norm is given by

$$\frac{\partial}{\partial x_k}\|\mathbf{x}\|_p = \frac{x_k\,|x_k|^{p-2}}{\|\mathbf{x}\|_p^{p-1}}.$$

For the special case of $p = 2$, this becomes

$$\frac{\partial}{\partial x_k}\|\mathbf{x}\|_2 = \frac{x_k}{\|\mathbf{x}\|_2},$$

or

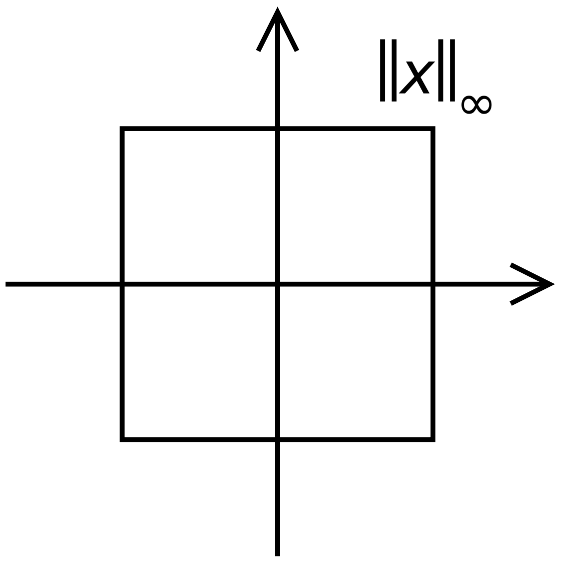
$$\frac{\partial}{\partial \mathbf{x}}\|\mathbf{x}\|_2 = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}.$$

13.3.5 Maximum norm (special case of: infinity norm, uniform norm, or supremum norm)

Main article: Maximum norm

$$\|\mathbf{x}\|_\infty := \max\left(|x_1|, \ldots, |x_n|\right).$$

The set of vectors whose infinity norm is a given constant, c, forms the surface of a hypercube with edge length $2c$.



$\|x\|_\infty = 1$

13.3.6 Zero norm

In probability and functional analysis, the zero norm induces a complete metric topology for the space of measureable functions and for the F-space of sequences with F–norm $(x_n) \mapsto \sum_n 2^{-n} x_n/(1+x_n)$, which is discussed by Stefan Rolewicz in *Metric Linear Spaces*.[3]

Hamming distance of a vector from zero

See also: Hamming distance and discrete metric

In metric geometry, the discrete metric takes the value one for distinct points and zero otherwise. When applied coordinate-wise to the elements of a vector space, the discrete distance defines the *Hamming distance*, which is important in coding and information theory. In the field of real or complex numbers, the distance of the discrete metric from zero is not homogeneous in the non-zero point; indeed, the distance from zero remains one as its non-zero argument approaches zero. However, the discrete distance of a number from zero does satisfy the other properties of a norm, namely the triangle inequality and positive definiteness. When applied component-wise to vectors, the discrete distance from zero behaves like a non-homogeneous "norm", which counts the number of non-zero components in its vector argument; again, this non-homogeneous "norm" is discontinuous.

In signal processing and statistics, David Donoho referred to the *zero* *"norm"* with quotation marks. Following Donoho's notation, the zero "norm" of **x** is simply the number of non-zero coordinates of **x**, or the Hamming distance of the vector from zero. When this "norm" is localized to a bounded set, it is the limit of p-norms as p approaches 0. Of course, the zero "norm" is **not** a B-norm, because it is not positive homogeneous. It is not even an F-norm, because it is discontinuous, jointly and severally, with respect to the scalar argument in scalar–vector multiplication and with respect to its vector argument. Abusing terminology, some engineers omit Donoho's quotation marks and inappropriately call the number-of-nonzeros function the L0 norm (sic.), also misusing the notation for the Lebesgue space of measurable functions.

13.3.7 Other norms

Other norms on $\mathbf{R}^n$ can be constructed by combining the above; for example

$$\|x\| := 2\left|x_1\right| + \sqrt{3\left|x_2\right|^2 + \max(\left|x_3\right|, 2\left|x_4\right|)^2}$$

is a norm on $\mathbf{R}^4$.

For any norm and any injective linear transformation A we can define a new norm of x, equal to

$$\|Ax\|.$$

In 2D, with A a rotation by 45° and a suitable scaling, this changes the taxicab norm into the maximum norm. In 2D, each A applied to the taxicab norm, up to inversion and interchanging of axes, gives a different unit ball: a parallelogram of a particular shape, size and orientation. In 3D this is similar but different for the 1-norm (octahedrons) and the maximum norm (prisms with parallelogram base).

All the above formulas also yield norms on $\mathbf{C}^n$ without modification.

13.3.8 Infinite-dimensional case

The generalization of the above norms to an infinite number of components leads to the L^p spaces, with norms

$$\|x\|_p = \left(\sum_{i\in\mathbb{N}} \left|x_i\right|^p\right)^{1/p} \text{ resp. } \|f\|_{p,X} = \left(\int_X \left|f(x)\right|^p \,\mathrm{d}x\right)^{1/p}$$

(for complex-valued sequences x resp. functions f defined on $X \subset \mathbb{R}$), which can be further generalized (see Haar measure).

Any inner product induces in a natural way the norm $\|x\| := \sqrt{\langle x, x\rangle}$.

Other examples of infinite dimensional normed vector spaces can be found in the Banach space article.

13.4 Properties

The concept of unit circle (the set of all vectors of norm 1) is different in different norms: for the 1-norm the unit circle in $\mathbf{R}^2$ is a square, for the 2-norm (Euclidean norm) it is the well-known unit circle, while for the infinity norm it is a different square. For any p-norm it is a superellipse (with congruent axes). See the accompanying illustration. Note that due to the definition of the norm, the unit circle is always convex and centrally symmetric (therefore, for example, the unit ball may be a rectangle but cannot be a triangle).

In terms of the vector space, the seminorm defines a topology on the space, and this is a Hausdorff topology precisely when the seminorm can distinguish between distinct vectors, which is again equivalent to the seminorm being a norm. The topology thus defined (by either a norm or a seminorm) can be understood either in terms of sequences or open sets. A sequence of vectors $\{v_n\}$ is said to converge in norm to v if $\|v_n - v\| \to 0$ as $n \to \infty$. Equivalently, the topology consists of all sets that can be represented as a union of open balls.

Two norms ‖•‖α and ‖•‖β on a vector space V are called *equivalent* if there exist positive real numbers C and D such that for all x in V

$$C\|x\|_\alpha \leq \|x\|_\beta \leq D\|x\|_\alpha.$$

For instance, on $\mathbf{C}^n$, if $p > r > 0$, then

$$\|x\|_p \leq \|x\|_r \leq n^{(1/r-1/p)}\|x\|_p.$$

In particular,

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$$
$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$$
$$\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty.$$

If the vector space is a finite-dimensional real or complex one, all norms are equivalent. On the other hand, in the case of infinite-dimensional vector spaces, not all norms are equivalent.

Equivalent norms define the same notions of continuity and convergence and for many purposes do not need to be distinguished. To be more precise the uniform structure defined by equivalent norms on the vector space is uniformly isomorphic.

Every (semi)-norm is a sublinear function, which implies that every norm is a convex function. As a result, finding a global optimum of a norm-based objective function is often tractable.

Given a finite family of seminorms pi on a vector space the sum

$$p(x) := \sum_{i=0}^{n} p_i(x)$$

is again a seminorm.

For any norm p on a vector space V, we have that for all **u** and **v** ∈ V:

$$p(\mathbf{u} \pm \mathbf{v}) \geq |p(\mathbf{u}) - p(\mathbf{v})|.$$

Proof: Applying the triangular inequality to both $p(u-0)$ and $p(v-0)$:

$$p(u-0) \leq p(u-v)+p(v-0) \Rightarrow p(u-v) \geq p(u)-p(v)$$

$$p(u-0) \leq p(u+v)+p(0-v) \Rightarrow p(u+v) \geq p(u)-p(v)$$

$$p(v-0) \leq p(u-v)+p(u-0) \Rightarrow p(u-v) \geq p(v)-p(u)$$

$$p(v-0) \leq p(u+v)+p(0-u) \Rightarrow p(u+v) \geq p(v)-p(u)$$

Thus, $p(\mathbf{u} \pm \mathbf{v}) \geq |p(\mathbf{u}) - p(\mathbf{v})|$.

If X and Y are normed spaces and $u : X \to Y$ is a continuous linear map, then the norm of u and the norm of the transpose of u are equal.[4]

For the l^p norms, we have Hölder's inequality[5]

$$\left|x^\mathsf{T} y\right| \leq \|x\|_p \|y\|_q \qquad \frac{1}{p} + \frac{1}{q} = 1.$$

A special case of this is the Cauchy–Schwarz inequality:[5]

$$\left|x^\mathsf{T} y\right| \leq \|x\|_2 \|y\|_2 .$$

13.5 Classification of seminorms: absolutely convex absorbing sets

All seminorms on a vector space V can be classified in terms of absolutely convex absorbing sets in V. To each such set, A, corresponds a seminorm pA called the **gauge** of A, defined as

$$pA(x) := \inf\{\alpha : \alpha > 0, x \in \alpha A\}$$

with the property that

$$\{x : pA(x) < 1\} \subseteq A \subseteq \{x : pA(x) \leq 1\}.$$

Conversely:

Any locally convex topological vector space has a local basis consisting of absolutely convex sets. A common method to construct such a basis is to use a family (p) of seminorms p that separates points: the collection of all finite intersections of sets $\{p < 1/n\}$ turns the space into a locally convex topological vector space so that every p is continuous.

Such a method is used to design weak and weak* topologies.

norm case:

> Suppose now that (p) contains a single p: since (p) is separating, p is a norm, and $A = \{p < 1\}$ is its open unit ball. Then A is an absolutely convex bounded neighbourhood of 0, and $p = pA$ is continuous.

> The converse is due to Kolmogorov: any locally convex and locally bounded topological vector space is normable. Precisely:
>
> If V is an absolutely convex bounded neighbourhood of 0, the gauge gV (so that $V = \{gV < 1\}$) is a norm.

13.6 Generalizations

There are several generalizations of norms and semi-norms. If p is absolute homogeneity but in place of subadditivity we require that

then p satisfies the triangle inequality but is called a **quasi-seminorm** and the smallest value of b for which this holds is called the **multiplier of *p***; if in addition p separates points then it is called a **quasi-norm**.

On the other hand, if p satisfies the triangle inequality but in place of absolute homogeneity we require that

then p is called a ***k*-seminorm**.

We have the following relationship between quasi-seminorms and k-seminorms:

> Suppose that q is a quasi-seminorm on a vector space X with multiplier b. If $0 < k < \log_2^2 b$ then there exists k-seminorm p on X equivalent to q.

13.7 See also

- Normed vector space
- Asymmetric norm
- Matrix norm

- Gowers norm
- Mahalanobis distance
- Manhattan distance
- Relation of norms and metrics

13.8 Notes

[1] Prugovečki 1981, page 20

[2] Except in $\mathbf{R}^1$, where it coincides with the Euclidean norm, and $\mathbf{R}^0$, where it is trivial.

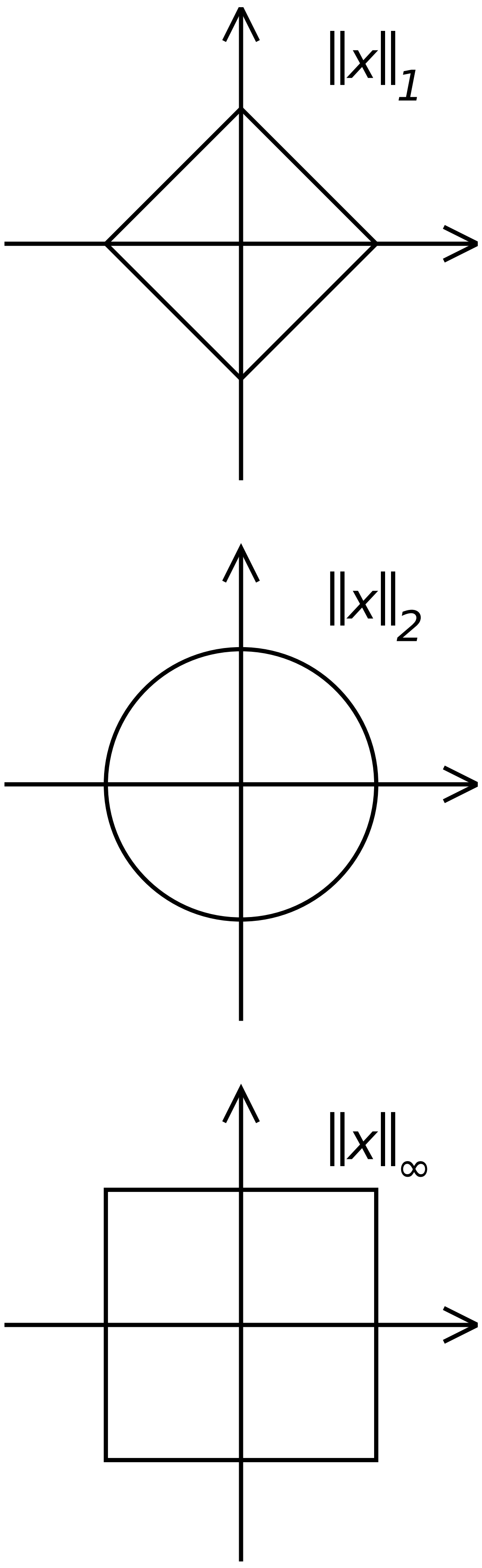
[3] Rolewicz, Stefan (1987), *Functional analysis and control theory: Linear systems*, Mathematics and its Applications (East European Series) **29** (Translated from the Polish by Ewa Bednarczuk ed.), Dordrecht; Warsaw: D. Reidel Publishing Co.; PWN—Polish Scientific Publishers, pp. xvi,524, ISBN 90-277-2186-6, MR 920371, OCLC 13064804

[4] Treves pp. 242–243

[5] Golub, Gene; Van Loan, Charles F. (1996). *Matrix Computations* (Third ed.). Baltimore: The Johns Hopkins University Press. p. 53. ISBN 0-8018-5413-X.

13.9 References

- Bourbaki, Nicolas (1987). “Chapters 1–5”. *Topological vector spaces*. Springer. ISBN 3-540-13627-4.
- Prugovečki, Eduard (1981). *Quantum mechanics in Hilbert space* (2nd ed.). Academic Press. p. 20. ISBN 0-12-566060-X.
- Trèves, François (1995). *Topological Vector Spaces, Distributions and Kernels*. Academic Press, Inc. pp. 136–149, 195–201, 240–252, 335–390, 420–433. ISBN 0-486-45352-9.
- Khaleelulla, S. M. (1982). *Counterexamples in Topological Vector Spaces*. Lecture Notes in Mathematics **936**. Springer-Verlag. pp. 3–5. ISBN 978-3-540-11565-6. Zbl 0482.46002.



Illustrations of unit circles in different norms.

Chapter 14

Regularization (mathematics)

For other uses in related fields, see Regularization (disambiguation).

Regularization, in mathematics and statistics and particularly in the fields of machine learning and inverse problems, refers to a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting. This information is usually of the form of a penalty for complexity, such as restrictions for smoothness or bounds on the vector space norm.

A theoretical justification for regularization is that it attempts to impose Occam's razor on the solution. From a Bayesian point of view, many regularization techniques correspond to imposing certain prior distributions on model parameters.

The same idea arose in many fields of science. For example, the least-squares method can be viewed as a very simple form of regularization. A simple form of regularization applied to integral equations, generally termed Tikhonov regularization after Andrey Nikolayevich Tikhonov, is essentially a trade-off between fitting the data and reducing a norm of the solution. More recently, non-linear regularization methods, including total variation regularization have become popular.

14.1 Regularization in statistics and machine learning

In statistics and machine learning, regularization methods are used for model selection, in particular to prevent overfitting by penalizing models with extreme parameter values. The most common variants in machine learning are L_1 and L_2 regularization, which can be added to learning algorithms that minimize a loss function $E(X, Y)$ by instead minimizing $E(X, Y) + \alpha\|w\|$, where w is the model's weight vector, $\|\cdot\|$ is either the L_1 norm or the squared L_2 norm, and α is a free parameter that needs to be tuned empirically (typically by cross-validation; see hyperparameter optimization). This method applies to many models. When applied in linear regression, the resulting models are termed lasso or ridge regression, but regularization is also employed in (binary and multiclass) logistic regression, neural nets, support vector machines, conditional random fields and some matrix decomposition methods. L_2 regularization may also be called "weight decay", in particular in the setting of neural nets.

L_1 regularization is often preferred because it produces sparse models and thus performs feature selection within the learning algorithm, but since the L_1 norm is not differentiable, it may require changes to learning algorithms, in particular gradient-based learners.[1][2]

Bayesian learning methods make use of a prior probability that (usually) gives lower probability to more complex models. Well-known model selection techniques include the Akaike information criterion (AIC), minimum description length (MDL), and the Bayesian information criterion (BIC). Alternative methods of controlling overfitting not involving regularization include cross-validation.

Regularization can be used to fine-tune model complexity using an augmented error function with cross-validation. The data sets used in complex models can produce a levelling-off of validation as complexity of the models increases. Training data sets errors decrease while the validation data set error remains constant. Regularization introduces a second factor which weights the penalty against more complex models with an increasing variance in the data errors. This gives an increasing penalty as model complexity increases.[3]

Examples of applications of different methods of regularization to the linear model are:

A linear combination of the LASSO and ridge regression methods is elastic net regularization.

14.2 See also

- Bayesian interpretation of regularization
- Regularization by spectral filtering

14.3 Notes

[1] Andrew, Galen; Gao, Jianfeng (2007). “Scalable training of L_1-regularized log-linear models”. *Proceedings of the 24th International Conference on Machine Learning*. doi:10.1145/1273496.1273501. ISBN 9781595937933.

[2] Tsuruoka, Y.; Tsujii, J.; Ananiadou, S. (2009). *Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty* (PDF). Proceedings of the AFNLP/ACL.

[3] Alpaydin, Ethem (2004). *Introduction to machine learning*. Cambridge, Mass. [u.a.]: MIT Press. pp. 79, 80. ISBN 978-0-262-01211-9. Retrieved 9 March 2011.

[4] Bishop, Christopher M. (2007). *Pattern recognition and machine learning* (Corr. printing. ed.). New York: Springer. ISBN 978-0387310732.

[5] Duda, Richard O. (2004). *Pattern classification + computer manual : hardcover set* (2. ed. ed.). New York [u.a.]: Wiley. ISBN 978-0471703501.

[6] Tibshirani, Robert (1996). “Regression Shrinkage and Selection via the Lasso” (POSTSCRIPT). *Journal of the Royal Statistical Society, Series B* **58** (1): 267–288. MR 1379242. Retrieved 2009-03-19.

[7] Li Wang, Michael D. Gordon & Ji Zhu (2006). “Regularized Least Absolute Deviations Regression and an Efficient Algorithm for Parameter Tuning”. *Sixth International Conference on Data Mining*. pp. 690–700. doi:10.1109/ICDM.2006.134.

[8] Candes, Emmanuel; Tao, Terence (2007). “The Dantzig selector: Statistical estimation when *p* is much larger than *n*". *Annals of Statistics* **35** (6): 2313–2351. arXiv:math/0506081. doi:10.1214/009053606000001523. MR 2382644.

[9] Małgorzata Bogdan, Ewout van den Berg, Weijie Su & Emmanuel J. Candes (2013). “Statistical estimation and testing via the ordered L1 norm” (PDF). *arXiv preprint arXiv:1310.1969*. arXiv:1310.1969v2.

14.4 References

- A. Neumaier, Solving ill-conditioned and singular linear systems: A tutorial on regularization, SIAM Review 40 (1998), 636-666. Available in pdf from author’s website.

Chapter 15

Loss function

In mathematical optimization, statistics, decision theory and machine learning, a **loss function** or **cost function** is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event. An optimization problem seeks to minimize a loss function. An **objective function** is either a loss function or its negative (sometimes called a reward function, a profit function, a utility function, etc.), in which case it is to be maximized.

In statistics, typically a loss function is used for parameter estimation, and the event in question is some function of the difference between estimated and true values for an instance of data. The concept, as old as Laplace, was reintroduced in statistics by Abraham Wald in the middle of the 20th century.[1] In the context of economics, for example, this is usually economic cost or regret. In classification, it is the penalty for an incorrect classification of an example. In actuarial science, it is used in an insurance context to model benefits paid over premiums, particularly since the works of Harald Cramér in the 1920s.[2] In optimal control the loss is the penalty for failing to achieve a desired value. In financial risk management the function is precisely mapped to a monetary loss.

15.1 Use in statistics

Parameter estimation for supervised learning tasks such as regression or classification can be formulated as the minimization of a loss function over a training set. The goal of estimation is to find a function that models its input well: if it were applied to the training set, it should predict the values (or class labels) associated with the samples in that set. The loss function quantifies the amount by which the prediction deviates from the actual values.

15.1.1 Definition

Formally, we begin by considering some family of distributions for a random variable X, that is indexed by some θ.

More intuitively, we can think of X as our "data", perhaps $X = (X_1, \ldots, X_n)$, where $X_i \sim F_\theta$ i.i.d. The X is the set of things the decision rule will be making decisions on. There exists some number of possible ways F_θ to model our data X, which our decision function can use to make decisions. For a finite number of models, we can thus think of θ as the *index* to this family of probability models. For an infinite family of models, it is a set of parameters to the family of distributions.

On a more practical note, it is important to understand that, while it is tempting to think of loss functions as necessarily parametric (since they seem to take θ as a "parameter"), the fact that θ is infinite-dimensional is completely incompatible with this notion; for example, if the family of probability functions is uncountably infinite, θ indexes an uncountably infinite space.

From here, given a set A of possible actions, a **decision rule** is a function $\delta : \mathcal{X} \rightarrow A$.

A **loss function** is a real lower-bounded function L on $\Theta \times A$ for some $\theta \in \Theta$. The value $L(\theta, \delta(X))$ is the *cost* of action $\delta(X)$ under parameter θ.[3]

15.2 Expected loss

The value of the loss function itself is a random quantity because it depends on the outcome of a random variable X. Both frequentist and Bayesian statistical theory involve making a decision based on the expected value of the loss function: however this quantity is defined differently under the two paradigms.

15.2.1 Frequentist expected loss

We first define the expected loss in the frequentist context. It is obtained by taking the expected value with respect to the probability distribution, $P\theta$, of the observed data, X. This is also referred to as the **risk function**[4] [5][6][7] of the decision rule δ and the parameter θ. Here the decision rule depends on the outcome of X. The risk function is given by[8]

$$R(\theta,\delta) = \mathbb{E}_\theta L\big(\theta,\delta(X)\big) = \int_X L\big(\theta,\delta(x)\big)\,\mathrm{d}\,P_\theta(x).$$

15.2.2 Bayesian expected loss

In a Bayesian approach, the expectation is calculated using the posterior distribution π^* of the parameter θ:

$$\rho(\pi^*,a) = \int_\Theta L(\theta,a)\,\mathrm{d}\,\pi^*(\theta)$$

One then should choose the action a^* which minimises the expected loss. Although this will result in choosing the same action as would be chosen using the frequentist risk, the emphasis of the Bayesian approach is that one is only interested in choosing the optimal action under the actual observed data, whereas choosing the actual Bayes optimal decision rule, which is a function of all possible observations, is a much more difficult problem.

15.2.3 Economic choice under uncertainty

In economics, decision-making under uncertainty is often modelled using the von Neumann-Morgenstern utility function of the uncertain variable of interest, such as end-of-period wealth. Since the value of this variable is uncertain, so is the value of the utility function; it is the expected value of utility that is maximized.

15.2.4 Examples

- For a scalar parameter θ, a decision function whose output $\hat\theta$ is an estimate of θ, and a quadratic loss function

  $$L(\theta,\hat\theta) = (\theta-\hat\theta)^2,$$

  the risk function becomes the mean squared error of the estimate,

  $$R(\theta,\hat\theta) = E_\theta(\theta-\hat\theta)^2.$$

- In density estimation, the unknown parameter is probability density itself. The loss function is typically chosen to be a norm in an appropriate function space. For example, for L^2 norm,

  $$L(f,\hat f) = \|f-\hat f\|_2^2\,,$$

  the risk function becomes the mean integrated squared error

  $$R(f,\hat f) = E\|f-\hat f\|^2.$$

15.3 Decision rules

A decision rule makes a choice using an optimality criterion. Some commonly used criteria are:

- **Minimax**: Choose the decision rule with the lowest worst loss — that is, minimize the worst-case (maximum possible) loss:

  $$\operatorname*{arg\,min}_\delta \ \max_{\theta\in\Theta} \ R(\theta,\delta).$$

- **Invariance**: Choose the optimal decision rule which satisfies an invariance requirement.
- Choose the decision rule with the lowest average loss (i.e. minimize the expected value of the loss function):

$$\operatorname*{arg\,min}_\delta \mathbb{E}_{\theta\in\Theta}[R(\theta,\delta)] = \operatorname*{arg\,min}_\delta \int_{\theta\in\Theta} R(\theta,\delta)\,p(\theta)\,d\theta.$$

15.4 Selecting a loss function

Sound statistical practice requires selecting an estimator consistent with the actual acceptable variation experienced in the context of a particular applied problem. Thus, in the applied use of loss functions, selecting which statistical method to use to model an applied problem depends on knowing the losses that will be experienced from being wrong under the problem's particular circumstances.[9]

A common example involves estimating "location." Under typical statistical assumptions, the mean or average is the statistic for estimating location that minimizes the expected loss experienced under the squared-error loss function, while the median is the estimator that minimizes expected loss experienced under the absolute-difference loss function. Still different estimators would be optimal under other, less common circumstances.

In economics, when an agent is risk neutral, the objective function is simply expressed in monetary terms, such as profit, income, or end-of-period wealth.

But for risk-averse (or risk-loving) agents, loss is measured as the negative of a utility function, which represents satisfaction and is usually interpreted in ordinal terms rather than in cardinal (absolute) terms.

Other measures of cost are possible, for example mortality or morbidity in the field of public health or safety engineering.

For most optimization algorithms, it is desirable to have a loss function that is globally continuous and differentiable.

Two very commonly used loss functions are the squared loss, $L(a) = a^2$, and the absolute loss, $L(a) = |a|$. However the absolute loss has the disadvantage that it is not differentiable at $a = 0$. The squared loss has the disadvantage that it has the tendency to be dominated by outliers—when summing over a set of a 's (as in $\sum_{i=1}^{n} L(a_i)$), the final sum tends to be the result of a few particularly large a-values, rather than an expression of the average a-value.

The choice of a loss function is not arbitrary. It is very restrictive and sometimes the loss function may be characterized by its desirable properties.[10] Among the choice principles are, for example, the requirement of completeness of the class of symmetric statistics in the case of i.i.d. observations, the principle of complete information, and some others.

15.5 Loss functions in Bayesian statistics

One of the consequences of Bayesian inference is that in addition to experimental data, the loss function does not in itself wholly determine a decision. What is important is the relationship between the loss function and the posterior probability. So it is possible to have two different loss functions which lead to the same decision when the prior probability distributions associated with each compensate for the details of each loss function.

Combining the three elements of the prior probability, the data, and the loss function then allows decisions to be based on maximizing the subjective expected utility, a concept introduced by Leonard J. Savage.

15.6 Regret

Main article: Regret (decision theory)

Savage also argued that using non-Bayesian methods such as minimax, the loss function should be based on the idea of *regret*, i.e., the loss associated with a decision should be the difference between the consequences of the best decision that could have been taken had the underlying circumstances been known and the decision that was in fact taken before they were known.

15.7 Quadratic loss function

The use of a quadratic loss function is common, for example when using least squares techniques. It is often more mathematically tractable than other loss functions because of the properties of variances, as well as being symmetric: an error above the target causes the same loss as the same magnitude of error below the target. If the target is *t*, then a quadratic loss function is

$$\lambda(x) = C(t - x)^2$$

for some constant *C*; the value of the constant makes no difference to a decision, and can be ignored by setting it equal to 1.

Many common statistics, including t-tests, regression models, design of experiments, and much else, use least squares methods applied using linear regression theory, which is based on the quadratic loss function.

The quadratic loss function is also used in linear-quadratic optimal control problems. In these problems, even in the absence of uncertainty, it may not be possible to achieve the desired values of all target variables. Often loss is expressed as a quadratic form in the deviations of the variables of interest from their desired values; this approach is tractable because it results in linear first-order conditions. In the context of stochastic control, the expected value of the quadratic form is used.

15.8 0-1 loss function

In statistics and decision theory, a frequently used loss function is the *0-1 loss function*

$$L(\hat{y}, y) = I(\hat{y} \neq y),$$

where I is the indicator notation.

15.9 See also

- Discounted maximum loss
- Hinge loss
- Scoring rule

15.10 References

[1] Wald, A. (1950). *Statistical Decision Functions*. Wiley.

[2] Cramér, H. (1930). *On the mathematical theory of risk*. *Centraltryckeriet*.

[3] Nikulin, M.S. (2001), "Loss function", in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

[4] Nikulin, M.S. (2001), "Risk of a statistical procedure", in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

[5] Berger, James O. (1985). *Statistical decision theory and Bayesian Analysis* (2nd ed.). New York: Springer-Verlag. ISBN 0-387-96098-8. MR 0804611.

[6] DeGroot, Morris (2004) [1970]. *Optimal Statistical Decisions*. Wiley Classics Library. ISBN 0-471-68029-X. MR 2288194.

[7] Robert, Christian P. (2007). *The Bayesian Choice* (2nd ed.). New York: Springer. doi:10.1007/0-387-71599-1. ISBN 0-387-95231-4. MR 1835885.

[8] Here,

- θ is a fixed but possibly unknown state of nature;
- X is a vector of observations stochastically drawn from a population;
- $\mathbb{E}_\theta$ is the expectation over all population values of X;
- $dP\theta$ is a probability measure over the event space of X, parametrized by θ; and
- the integral is evaluated over the entire support of X.

[9] Pfanzagl, J. (1994). *Parametric Statistical Theory*. Berlin: Walter de Gruyter. ISBN 3-11-013863-8.

[10] Detailed information on mathematical principles of the loss function choice is given in Chapter 2 of the book Robust and Non-Robust Models in Statistics by Lev B. Klebanov, Svetlozat T. Rachev and Frank J. Fabozzi, Nova Scientific Publishers, Inc. New York, 2009 (and references there).

15.11 Further reading

- Aretz, Kevin; Bartram, Söhnke M.; Pope, Peter F. (April–June 2011). "Asymmetric Loss Functions and the Rationality of Expected Stock Returns". *International Journal of Forecasting* **27** (2): 413–437. doi:10.1016/j.ijforecast.2009.10.008.
- Berger, James O. (1985). *Statistical decision theory and Bayesian Analysis* (2nd ed.). New York: Springer-Verlag. ISBN 0-387-96098-8. MR 0804611.

Chapter 16

Least squares

The method of **least squares** is a standard approach in regression analysis to the approximate solution of overdetermined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation.

The most important application is in data fitting. The best fit in the least-squares sense minimizes the sum of squared residuals, a residual being the difference between an observed value and the fitted value provided by a model. When the problem has substantial uncertainties in the independent variable (the x variable), then simple regression and least squares methods have problems; in such cases, the methodology required for fitting errors-in-variables models may be considered instead of that for least squares.

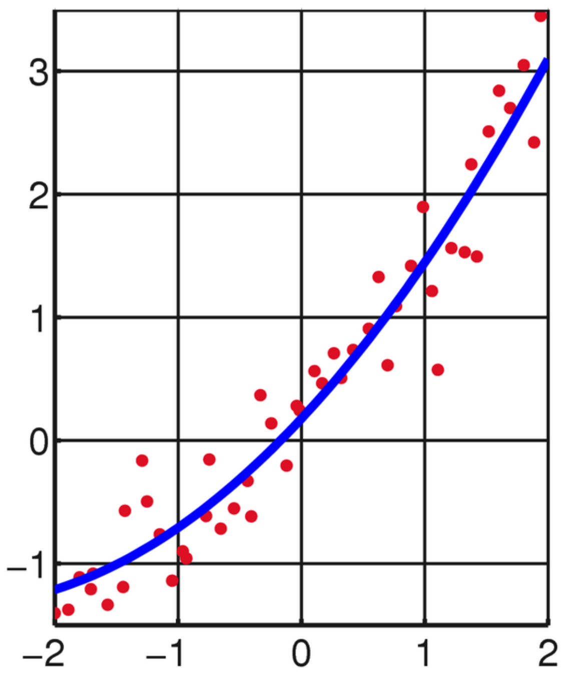
Least squares problems fall into two categories: linear or ordinary least squares and non-linear least squares, depending on whether or not the residuals are linear in all unknowns. The linear least-squares problem occurs in statistical regression analysis; it has a closed-form solution. The non-linear problem is usually solved by iterative refinement; at each iteration the system is approximated by a linear one, and thus the core calculation is similar in both cases.

Polynomial least squares describes the variance in a prediction of the dependent variable as a function of the independent variable and the deviations from the fitted curve.

When the observations come from an exponential family and mild conditions are satisfied, least-squares estimates and maximum-likelihood estimates are identical.[1] The method of least squares can also be derived as a method of moments estimator.

The following discussion is mostly presented in terms of linear functions but the use of least-squares is valid and practical for more general families of functions. Also, by iteratively applying local quadratic approximation to the likelihood (through the Fisher information), the least-squares method may be used to fit a generalized linear model.

For the topic of approximating a function by a sum of others using an objective function based on squared distances, see least squares (function approximation).



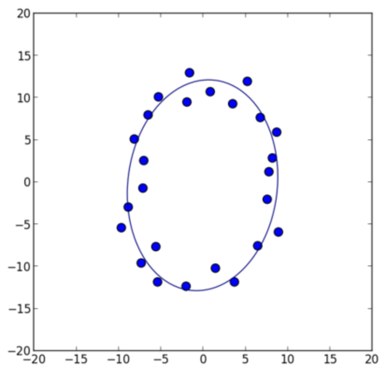
The result of fitting a set of data points with a quadratic function.

The least-squares method is usually credited to Carl Friedrich Gauss (1795),[2] but it was first published by Adrien-Marie Legendre.[3]

16.1 History

16.1.1 Context

The method of least squares grew out of the fields of astronomy and geodesy as scientists and mathematicians sought to provide solutions to the challenges of navigating the Earth's oceans during the Age of Exploration. The accurate description of the behavior of celestial bodies was the key to enabling ships to sail in open seas, where sailors could no longer rely on land sightings for navigation.



Conic fitting a set of points using least-squares approximation.

The method was the culmination of several advances that took place during the course of the eighteenth century:[4]

- The combination of different observations as being the best estimate of the true value; errors decrease with aggregation rather than increase, perhaps first expressed by Roger Cotes in 1722.

- The combination of different observations taken under the *same* conditions contrary to simply trying one's best to observe and record a single observation accurately. The approach was known as the method of averages. This approach was notably used by Tobias Mayer while studying the librations of the moon in 1750, and by Pierre-Simon Laplace in his work in explaining the differences in motion of Jupiter and Saturn in 1788.

- The combination of different observations taken under *different* conditions. The method came to be known as the method of least absolute deviation. It was notably performed by Roger Joseph Boscovich in his work on the shape of the earth in 1757 and by Pierre-Simon Laplace for the same problem in 1799.

- The development of a criterion that can be evaluated to determine when the solution with the minimum error has been achieved. Laplace tried to specify a mathematical form of the probability density for the errors and define a method of estimation that minimizes the error of estimation. For this purpose, Laplace used a symmetric two-sided exponential distribution we now call Laplace distribution to model the error distribution, and used the sum of absolute deviation as error of estimation. He felt these to be the simplest assumptions he could make, and he had hoped to obtain the arithmetic mean as the best estimate. Instead, his estimator was the posterior median.

Carl Friedrich Gauss

16.1.2 The method

The first clear and concise exposition of the method of least squares was published by Legendre in 1805.[5] The technique is described as an algebraic procedure for fitting linear equations to data and Legendre demonstrates the new method by analyzing the same data as Laplace for the shape of the earth. The value of Legendre's method of least squares was immediately recognized by leading astronomers and geodesists of the time.

In 1809 Carl Friedrich Gauss published his method of calculating the orbits of celestial bodies. In that work he claimed to have been in possession of the method of least squares since 1795. This naturally led to a priority dispute with Legendre. However, to Gauss's credit, he went beyond Legendre and succeeded in connecting the method of least squares with the principles of probability and to the normal distribution. He had managed to complete Laplace's program of specifying a mathematical form of the probability density for the observations, depending on a finite number of unknown parameters, and define a method of estimation that minimizes the error of estimation. Gauss showed that arithmetic mean is indeed the best estimate of the location parameter by changing both the probability density and the method of estimation. He then turned the problem around by asking what form the density should have and what method of estimation should be used to get the arithmetic mean as estimate of the location parameter. In this attempt, he invented the normal distribution.

An early demonstration of the strength of Gauss' Method came when it was used to predict the future location of the newly discovered asteroid Ceres. On 1 January 1801, the Italian astronomer Giuseppe Piazzi discovered

Ceres and was able to track its path for 40 days before it was lost in the glare of the sun. Based on this data, astronomers desired to determine the location of Ceres after it emerged from behind the sun without solving the complicated Kepler's nonlinear equations of planetary motion. The only predictions that successfully allowed Hungarian astronomer Franz Xaver von Zach to relocate Ceres were those performed by the 24-year-old Gauss using least-squares analysis.

In 1810, after reading Gauss's work, Laplace, after proving the central limit theorem, used it to give a large sample justification for the method of least square and the normal distribution. In 1822, Gauss was able to state that the least-squares approach to regression analysis is optimal in the sense that in a linear model where the errors have a mean of zero, are uncorrelated, and have equal variances, the best linear unbiased estimator of the coefficients is the least-squares estimator. This result is known as the Gauss–Markov theorem.

The idea of least-squares analysis was also independently formulated by the American Robert Adrain in 1808. In the next two centuries workers in the theory of errors and in statistics found many different ways of implementing least squares.[6]

16.2 Problem statement

The objective consists of adjusting the parameters of a model function to best fit a data set. A simple data set consists of *n* points (data pairs) (x_i, y_i), *i* = 1, ..., *n*, where $x_i$is an independent variable and $y_i$is a dependent variable whose value is found by observation. The model function has the form $f(x, \beta)$, where *m* adjustable parameters are held in the vector $\boldsymbol{\beta}$. The goal is to find the parameter values for the model which "best" fits the data. The least squares method finds its optimum when the sum, *S*, of squared residuals

$$S = \sum_{i=1}^{n} {r_i}^2$$

is a minimum. A residual is defined as the difference between the actual value of the dependent variable and the value predicted by the model.

$$r_i = y_i - f(x_i, \boldsymbol{\beta}).$$

An example of a model is that of the straight line in two dimensions. Denoting the intercept as β_0 and the slope as β_1 , the model function is given by $f(x, \boldsymbol{\beta}) = \beta_0 + \beta_1 x$. See linear least squares for a fully worked out example of this model.

A data point may consist of more than one independent variable. For example, when fitting a plane to a set of height measurements, the plane is a function of two independent variables, *x* and *z*, say. In the most general case there may be one or more independent variables and one or more dependent variables at each data point.

16.3 Limitations

This regression formulation considers only residuals in the dependent variable. There are two rather different contexts in which different implications apply:

- Regression for prediction. Here a model is fitted to provide a prediction rule for application in a similar situation to which the data used for fitting apply. Here the dependent variables corresponding to such future application would be subject to the same types of observation error as those in the data used for fitting. It is therefore logically consistent to use the least-squares prediction rule for such data.
- Regression for fitting a "true relationship". In standard regression analysis, that leads to fitting by least squares, there is an implicit assumption that errors in the independent variable are zero or strictly controlled so as to be negligible. When errors in the independent variable are non-negligible, models of measurement error can be used; such methods can lead to parameter estimates, hypothesis testing and confidence intervals that take into account the presence of observation errors in the independent variables.[7] An alternative approach is to fit a model by total least squares; this can be viewed as taking a pragmatic approach to balancing the effects of the different sources of error in formulating an objective function for use in model-fitting.

16.4 Solving the least squares problem

The minimum of the sum of squares is found by setting the gradient to zero. Since the model contains *m* parameters, there are *m* gradient equations:

$$\frac{\partial S}{\partial \beta_j} = 2\sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0,\ j = 1, \ldots, m,$$

and since $r_i = y_i - f(x_i, \boldsymbol{\beta})$, the gradient equations become

$$-2\sum_i r_i \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \beta_j} = 0,\ j = 1, \ldots, m.$$

The gradient equations apply to all least squares problems. Each particular problem requires particular expressions for the model and its partial derivatives.

16.4.1 Linear least squares

Main article: Linear least squares

A regression model is a linear one when the model comprises a linear combination of the parameters, i.e.,

$$f(x,\beta)=\sum_{j=1}^{m}\beta_j\phi_j(x),$$

where the function ϕ_j is a function of x .

Letting

$$X_{ij}=\frac{\partial f(x_i,\boldsymbol{\beta})}{\partial\beta_j}=\phi_j(x_i),$$

we can then see that in that case the least square estimate (or estimator, in the context of a random sample), $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}}=(X^TX)^{-1}X^T\boldsymbol{y}.$$

For a derivation of this estimate see Linear least squares (mathematics).

16.4.2 Non-linear least squares

Main article: Non-linear least squares

There is no closed-form solution to a non-linear least squares problem. Instead, numerical algorithms are used to find the value of the parameters β that minimizes the objective. Most algorithms involve choosing initial values for the parameters. Then, the parameters are refined iteratively, that is, the values are obtained by successive approximation:

$${\beta_j}^{k+1}={\beta_j}^k+\Delta\beta_j,$$

where *k* is an iteration number, and the vector of increments $\Delta\beta_j$ is called the shift vector. In some commonly used algorithms, at each iteration the model may be linearized by approximation to a first-order Taylor series expansion about $\boldsymbol{\beta}^k$:

$$\begin{aligned}f(x_i,\boldsymbol{\beta})&=f^k(x_i,\boldsymbol{\beta})+\sum_j\frac{\partial f(x_i,\boldsymbol{\beta})}{\partial\beta_j}\left(\beta_j-{\beta_j}^k\right)\\&=f^k(x_i,\boldsymbol{\beta})+\sum_j J_{ij}\Delta\beta_j.\end{aligned}$$

The Jacobian **J** is a function of constants, the independent variable *and* the parameters, so it changes from one iteration to the next. The residuals are given by

$$r_i=y_i-f^k(x_i,\boldsymbol{\beta})-\sum_{k=1}^{m}J_{ik}\Delta\beta_k=\Delta y_i-\sum_{j=1}^{m}J_{ij}\Delta\beta_j.$$

To minimize the sum of squares of r_i , the gradient equation is set to zero and solved for $\Delta\beta_j$:

$$-2\sum_{i=1}^{n}J_{ij}\left(\Delta y_i-\sum_{k=1}^{m}J_{ik}\Delta\beta_k\right)=0,$$

which, on rearrangement, become *m* simultaneous linear equations, the **normal equations**:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}J_{ij}J_{ik}\Delta\beta_k=\sum_{i=1}^{n}J_{ij}\Delta y_i\qquad(j=1,\ldots,m).$$

The normal equations are written in matrix notation as

$$\left(\mathbf{J}^\mathbf{T}\mathbf{J}\right)\square\boldsymbol{\beta}=\mathbf{J}^\mathbf{T}\square\mathbf{y}.$$

These are the defining equations of the Gauss–Newton algorithm.

16.4.3 Differences between linear and non-linear least squares

- The model function, *f*, in LLSQ (linear least squares) is a linear combination of parameters of the form $f=X_{i1}\beta_1+X_{i2}\beta_2+\cdots$ The model may represent a straight line, a parabola or any other linear combination of functions. In NLLSQ (non-linear least squares) the parameters appear as functions, such as $\beta^2,e^{\beta x}$ and so forth. If the derivatives $\partial f/\partial\beta_j$ are either constant or depend only on the values of the independent variable, the model is linear in the parameters. Otherwise the model is non-linear.
- Algorithms for finding the solution to a NLLSQ problem require initial values for the parameters, LLSQ does not.
- Like LLSQ, solution algorithms for NLLSQ often require that the Jacobian be calculated. Analytical expressions for the partial derivatives can be complicated. If analytical expressions are impossible to obtain either the partial derivatives must be calculated by numerical approximation or an estimate must be made of the Jacobian.
- In NLLSQ non-convergence (failure of the algorithm to find a minimum) is a common phenomenon whereas the LLSQ is globally concave so non-convergence is not an issue.

- NLLSQ is usually an iterative process. The iterative process has to be terminated when a convergence criterion is satisfied. LLSQ solutions can be computed using direct methods, although problems with large numbers of parameters are typically solved with iterative methods, such as the Gauss–Seidel method.
- In LLSQ the solution is unique, but in NLLSQ there may be multiple minima in the sum of squares.
- Under the condition that the errors are uncorrelated with the predictor variables, LLSQ yields unbiased estimates, but even under that condition NLLSQ estimates are generally biased.

These differences must be considered whenever the solution to a non-linear least squares problem is being sought.

16.5 Least squares, regression analysis and statistics

The method of least squares is often used to generate estimators and other statistics in regression analysis.

Consider a simple example drawn from physics. A spring should obey Hooke's law which states that the extension of a spring y is proportional to the force, *F*, applied to it.

$$y = f(F, k) = kF$$

constitutes the model, where *F* is the independent variable. To estimate the force constant, *k*, a series of *n* measurements with different forces will produce a set of data, $(F_i, y_i),\ i = 1, \dots, n$, where *yi* is a measured spring extension. Each experimental observation will contain some error. If we denote this error ε , we may specify an empirical model for our observations,

$$y_i = kF_i + \varepsilon_i.$$

There are many methods we might use to estimate the unknown parameter *k*. Noting that the *n* equations in the *m* variables in our data comprise an overdetermined system with one unknown and *n* equations, we may choose to estimate *k* using least squares. The sum of squares to be minimized is

$$S = \sum_{i=1}^{n} (y_i - kF_i)^2 .$$

The least squares estimate of the force constant, *k*, is given by

$$\hat{k} = \frac{\sum_i F_i y_i}{\sum_i {F_i}^2}.$$

Here it is assumed that application of the force ***causes*** the spring to expand and, having derived the force constant by least squares fitting, the extension can be predicted from Hooke's law.

In regression analysis the researcher specifies an empirical model. For example, a very common model is the straight line model which is used to test if there is a linear relationship between dependent and independent variable. If a linear relationship is found to exist, the variables are said to be correlated. However, correlation does not prove causation, as both variables may be correlated with other, hidden, variables, or the dependent variable may "reverse" cause the independent variables, or the variables may be otherwise spuriously correlated. For example, suppose there is a correlation between deaths by drowning and the volume of ice cream sales at a particular beach. Yet, both the number of people going swimming and the volume of ice cream sales increase as the weather gets hotter, and presumably the number of deaths by drowning is correlated with the number of people going swimming. Perhaps an increase in swimmers causes both the other variables to increase.

In order to make statistical tests on the results it is necessary to make assumptions about the nature of the experimental errors. A common (but not necessary) assumption is that the errors belong to a normal distribution. The central limit theorem supports the idea that this is a good approximation in many cases.

- The Gauss–Markov theorem. In a linear model in which the errors have expectation zero conditional on the independent variables, are uncorrelated and have equal variances, the best linear unbiased estimator of any linear combination of the observations, is its least-squares estimator. "Best" means that the least squares estimators of the parameters have minimum variance. The assumption of equal variance is valid when the errors all belong to the same distribution.
- In a linear model, if the errors belong to a normal distribution the least squares estimators are also the maximum likelihood estimators.

However, if the errors are not normally distributed, a central limit theorem often nonetheless implies that the parameter estimates will be approximately normally distributed so long as the sample is reasonably large. For this reason, given the important property that the error mean is independent of the independent variables, the distribution of the error term is not an important issue in regression analysis. Specifically, it is not typically important whether the error term follows a normal distribution.

In a least squares calculation with unit weights, or in linear regression, the variance on the *j*th parameter, denoted $\mathrm{var}(\hat{\beta}_j)$, is usually estimated with

$$\mathrm{var}(\hat{\beta}_j) = \sigma^2 \left(\left[X^T X \right]^{-1} \right)_{jj} \approx \frac{S}{n-m} \left(\left[X^T X \right]^{-1} \right)_{jj},$$

where the true residual variance σ^2 is replaced by an estimate based on the minimised value of the sum of squares objective function *S*. The denominator, $n - m$, is the statistical degrees of freedom; see effective degrees of freedom for generalizations.

Confidence limits can be found if the probability distribution of the parameters is known, or an asymptotic approximation is made, or assumed. Likewise statistical tests on the residuals can be made if the probability distribution of the residuals is known or assumed. The probability distribution of any linear combination of the dependent variables can be derived if the probability distribution of experimental errors is known or assumed. Inference is particularly straightforward if the errors are assumed to follow a normal distribution, which implies that the parameter estimates and residuals will also be normally distributed conditional on the values of the independent variables.

16.6 Weighted least squares

See also: Weighted mean and Linear least squares (mathematics) § Weighted linear least squares

A special case of generalized least squares called **weighted least squares** occurs when all the off-diagonal entries of *Ω* (the correlation matrix of the residuals) are null; the variances of the observations (along the covariance matrix diagonal) may still be unequal (heteroskedasticity).

The expressions given above are based on the implicit assumption that the errors are uncorrelated with each other and with the independent variables and have equal variance. The Gauss–Markov theorem shows that, when this is so, $\hat{\boldsymbol{\beta}}$ is a best linear unbiased estimator (BLUE). If, however, the measurements are uncorrelated but have different uncertainties, a modified approach might be adopted. Aitken showed that when a weighted sum of squared residuals is minimized, $\hat{\boldsymbol{\beta}}$ is the BLUE if each weight is equal to the reciprocal of the variance of the measurement

$$S = \sum_{i=1}^{n} W_{ii} {r_i}^2, \qquad W_{ii} = \frac{1}{{\sigma_i}^2}$$

The gradient equations for this sum of squares are

$$-2 \sum_i W_{ii} \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \beta_j} r_i = 0, \qquad j = 1, \ldots, n$$

which, in a linear least squares system give the modified normal equations,

$$\sum_{i=1}^{n} \sum_{k=1}^{m} X_{ij} W_{ii} X_{ik} \hat{\beta}_k = \sum_{i=1}^{n} X_{ij} W_{ii} y_i, \qquad j = 1, \ldots, m.$$

When the observational errors are uncorrelated and the weight matrix, **W**, is diagonal, these may be written as

$$\left(\mathbf{X^T W X}\right) \hat{\boldsymbol{\beta}} = \mathbf{X^T W y}.$$

If the errors are correlated, the resulting estimator is the BLUE if the weight matrix is equal to the inverse of the variance-covariance matrix of the observations.

When the errors are uncorrelated, it is convenient to simplify the calculations to factor the weight matrix as $\mathbf{w_{ii}} = \sqrt{\mathbf{W_{ii}}}$. The normal equations can then be written as

$$\left(\mathbf{X'^T X'}\right) \hat{\boldsymbol{\beta}} = \mathbf{X'^T y'}$$

where

$$\mathbf{X'} = \mathbf{wX}, \mathbf{y'} = \mathbf{wy}.$$

For non-linear least squares systems a similar argument shows that the normal equations should be modified as follows.

$$\left(\mathbf{J^T W J}\right) \boldsymbol{\Delta} \beta = \mathbf{J^T W \Delta y}.$$

Note that for empirical tests, the appropriate **W** is not known for sure and must be estimated. For this feasible generalized least squares (FGLS) techniques may be used.

16.7 Relationship to principal components

The first principal component about the mean of a set of points can be represented by that line which most closely approaches the data points (as measured by squared distance of closest approach, i.e. perpendicular to the line). In contrast, linear least squares tries to minimize the distance in the y direction only. Thus, although the two use a similar error metric, linear least squares is a method that treats one dimension of the data preferentially, while PCA treats all dimensions equally.

16.8 Regularized versions

16.8.1 Tikhonov regularization

Main article: Tikhonov regularization

In some contexts a regularized version of the least squares solution may be preferable. Tikhonov regularization (or ridge regression) adds a constraint that $\|\beta\|^2$, the L^2-norm of the parameter vector, is not greater than a given value. Equivalently, it may solve an unconstrained minimization of the least-squares penalty with $\alpha\|\beta\|^2$ added, where α is a constant (this is the Lagrangian form of the constrained problem). In a Bayesian context, this is equivalent to placing a zero-mean normally distributed prior on the parameter vector.

16.8.2 Lasso method

An alternative regularized version of least squares is *lasso* (least absolute shrinkage and selection operator), which uses the constraint that $\|\beta\|_1$, the L^1-norm of the parameter vector, is no greater than a given value.[8][9][10] (As above, this is equivalent to an unconstrained minimization of the least-squares penalty with $\alpha\|\beta\|_1$ added.) In a Bayesian context, this is equivalent to placing a zero-mean Laplace prior distribution on the parameter vector.[11] The optimization problem may be solved using quadratic programming or more general convex optimization methods, as well as by specific algorithms such as the least angle regression algorithm.

One of the prime differences between Lasso and ridge regression is that in ridge regression, as the penalty is increased, all parameters are reduced while still remaining non-zero, while in Lasso, increasing the penalty will cause more and more of the parameters to be driven to zero. This is an advantage of Lasso over ridge regression, as driving parameters to zero deselects the features from the regression. Thus, Lasso automatically selects more relevant features and discards the others, whereas Ridge regression never fully discards any features. Some feature selection techniques are developed based on the LASSO including Bolasso which bootstraps samples,[12] and FeaLect which analyzes the regression coefficients corresponding to different values of α to score all the features.[13]

The L^1-regularized formulation is useful in some contexts due to its tendency to prefer solutions with fewer nonzero parameter values, effectively reducing the number of variables upon which the given solution is dependent.[8] For this reason, the Lasso and its variants are fundamental to the field of compressed sensing. An extension of this approach is elastic net regularization.

16.9 See also

- Adjustment of observations
- Bayesian MMSE estimator
- Best linear unbiased estimator (BLUE)
- Best linear unbiased prediction (BLUP)
- Gauss–Markov theorem
- L_2 norm
- Least absolute deviation
- Measurement uncertainty
- Proximal gradient methods for learning
- Quadratic loss function
- Root mean square
- Squared deviations

16.10 References

[1] Charnes, A.; Frome, E. L.; Yu, P. L. (1976). "The Equivalence of Generalized Least Squares and Maximum Likelihood Estimates in the Exponential Family". *Journal of the American Statistical Association* **71** (353): 169. doi:10.1080/01621459.1976.10481508.

[2] Bretscher, Otto (1995). *Linear Algebra With Applications* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

[3] Stigler, Stephen M. (1981). "Gauss and the Invention of Least Squares". *Ann. Statist.* **9** (3): 465–474. doi:10.1214/aos/1176345451.

[4] Stigler, Stephen M. (1986). *The History of Statistics: The Measurement of Uncertainty Before 1900*. Cambridge, MA: Belknap Press of Harvard University Press. ISBN 0-674-40340-1.

[5] Legendre, Adrien-Marie (1805), *Nouvelles méthodes pour la détermination des orbites des comètes* [*New Methods for the Determination of the Orbits of Comets*] (in French), Paris: F. Didot

[6] Aldrich, J. (1998). "Doing Least Squares: Perspectives from Gauss and Yule". *International Statistical Review* **66** (1): 61–81. doi:10.1111/j.1751-5823.1998.tb00406.x.

[7] For a good introduction to error-in-variables, please see Fuller W., "Measurement Error Models", John Wiley & Sons, 1987

[8] Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso". *Journal of the Royal Statistical Society, Series B* **58** (1): 267–288. JSTOR 2346178.

[9] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome H. (2009). "The Elements of Statistical Learning" (second ed.). Springer-Verlag. ISBN 978-0-387-84858-7.

[10] Bühlmann, Peter; van de Geer, Sara (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer. ISBN 9783642201929.

[11] Park, Trevor; Casella, George (2008). "The Bayesian Lasso". *Journal of the American Statistical Association* **103** (482): 681–686. doi:10.1198/016214508000000337.

[12] Bach, Francis R (2008). "Bolasso: model consistent lasso estimation through the bootstrap". *Proceedings of the 25th international conference on Machine learning*: 33–40. doi:10.1145/1390156.1390161.

[13] Zare, Habil (2013). "Scoring relevancy of features based on combinatorial analysis of Lasso with application to lymphoma diagnosis". *BMC genomics* **14**: S14. doi:10.1186/1471-2164-14-S1-S14.

16.11 Further reading

- Björck, Å. (1996). *Numerical Methods for Least Squares Problems*. SIAM. ISBN 978-0-89871-360-2.
- Rao, C. R.; Toutenburg, H.; Fieger, A.; Heumann, C.; Nittner, T.; Scheid, S. (1999). *Linear Models: Least Squares and Alternatives*. Springer Series in Statistics.
- Kariya, T.; Kurata, H. (2004). *Generalized Least Squares*. Wiley.
- Wolberg, J. (2005). *Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments*. Springer. ISBN 3-540-25674-1.
- Strutz, T. (2010). *Data Fitting and Uncertainty (A practical introduction to weighted least squares and beyond)*. Vieweg+Teubner. ISBN 978-3-8348-1022-9.

Chapter 17

Newton's method

This article is about Newton's method for finding roots. For Newton's method for finding minima, see Newton's method in optimization.

In numerical analysis, **Newton's method** (also known as the **Newton–Raphson method**), named after Isaac Newton and Joseph Raphson, is a method for finding successively better approximations to the roots (or zeroes) of a real-valued function.

$$x : f(x) = 0\,.$$

The Newton–Raphson method in one variable is implemented as follows:

Given a function f defined over the reals x, and its derivative f', we begin with a first guess x_0 for a root of the function f. Provided the function satisfies all the assumptions made in the derivation of the formula, a better approximation x_1 is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}\,.$$

Geometrically, $(x_1, 0)$ is the intersection with the x-axis of the tangent to the graph of f at $(x_0, f(x_0))$.

The process is repeated as

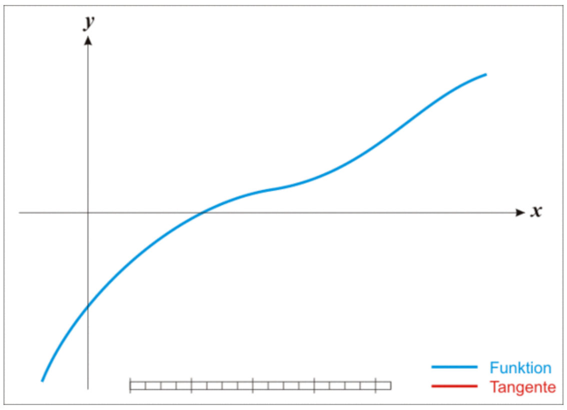
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

until a sufficiently accurate value is reached.

This algorithm is first in the class of Householder's methods, succeeded by Halley's method. The method can also be extended to complex functions and to systems of equations.

17.1 Description

The idea of the method is as follows: one starts with an initial guess which is reasonably close to the true root, then the function is approximated by its tangent line (which can be computed using the tools of calculus), and one computes the x-intercept of this tangent line (which is easily done with elementary algebra). This x-intercept will typically be a better approximation to the function's root than the original guess, and the method can be iterated.



The function f *is shown in blue and the tangent line is in red. We see that* x_n*+1 is a better approximation than* x_n *for the root* x *of the function* f.

Suppose $f : [a, b] \to \mathbf{R}$ is a differentiable function defined on the interval $[a, b]$ with values in the real numbers $\mathbf{R}$. The formula for converging on the root can be easily derived. Suppose we have some current approximation xn. Then we can derive the formula for a better approximation, xn_{+1} by referring to the diagram on the right. The equation of the tangent line to the curve $y = f(x)$ at the point x=xn is

$$y = f'(x_n)\,(x - x_n) + f(x_n),$$

where, f' denotes the derivative of the function f.

The x-intercept of this line (the value of x such that y=0) is then used as the next approximation to the root, xn_{+1}. In other words, setting y to zero and x to xn_{+1} gives

$$0 = f'(x_n)\,(x_{n+1} - x_n) + f(x_n).$$

Solving for xn_{+1} gives

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

We start the process off with some arbitrary initial value x_0. (The closer to the zero, the better. But, in the absence of any intuition about where the zero might lie, a "guess and check" method might narrow the possibilities to a reasonably small interval by appealing to the intermediate value theorem.) The method will usually converge, provided this initial guess is close enough to the unknown zero, and that $f'(x_0) \neq 0$. Furthermore, for a zero of multiplicity 1, the convergence is at least quadratic (see rate of convergence) in a neighbourhood of the zero, which intuitively means that the number of correct digits roughly at least doubles in every step. More details can be found in the analysis section below.

The Householder's methods are similar but have higher order for even faster convergence. However, the extra computations required for each step can slow down the overall performance relative to Newton's method, particularly if f or its derivatives are computationally expensive to evaluate.

17.2 History

The name "Newton's method" is derived from Isaac Newton's description of a special case of the method in *De analysi per aequationes numero terminorum infinitas* (written in 1669, published in 1711 by William Jones) and in *De metodis fluxionum et serierum infinitarum* (written in 1671, translated and published as *Method of Fluxions* in 1736 by John Colson). However, his method differs substantially from the modern method given above: Newton applies the method only to polynomials. He does not compute the successive approximations x_n , but computes a sequence of polynomials, and only at the end arrives at an approximation for the root x. Finally, Newton views the method as purely algebraic and makes no mention of the connection with calculus. Newton may have derived his method from a similar but less precise method by Vieta. The essence of Vieta's method can be found in the work of the Persian mathematician Sharaf al-Din al-Tusi, while his successor Jamshīd al-Kāshī used a form of Newton's method to solve $x^P - N = 0$ to find roots of N (Ypma 1995). A special case of Newton's method for calculating square roots was known much earlier and is often called the Babylonian method.

Newton's method was used by 17th-century Japanese mathematician Seki Kōwa to solve single-variable equations, though the connection with calculus was missing.

Newton's method was first published in 1685 in *A Treatise of Algebra both Historical and Practical* by John Wallis. In 1690, Joseph Raphson published a simplified description in *Analysis aequationum universalis*. Raphson again viewed Newton's method purely as an algebraic method and restricted its use to polynomials, but he describes the method in terms of the successive approximations *xn* instead of the more complicated sequence of polynomials used by Newton. Finally, in 1740, Thomas Simpson described Newton's method as an iterative method for solving general nonlinear equations using calculus, essentially giving the description above. In the same publication, Simpson also gives the generalization to systems of two equations and notes that Newton's method can be used for solving optimization problems by setting the gradient to zero.

Arthur Cayley in 1879 in *The Newton-Fourier imaginary problem* was the first to notice the difficulties in generalizing Newton's method to complex roots of polynomials with degree greater than 2 and complex initial values. This opened the way to the study of the theory of iterations of rational functions.

17.3 Practical considerations

Newton's method is an extremely powerful technique—in general the convergence is quadratic: as the method converges on the root, the difference between the root and the approximation is squared (the number of accurate digits roughly doubles) at each step. However, there are some difficulties with the method.

17.3.1 Difficulty in calculating derivative of a function

Newton's method requires that the derivative be calculated directly. An analytical expression for the derivative may not be easily obtainable and could be expensive to evaluate. In these situations, it may be appropriate to approximate the derivative by using the slope of a line through two nearby points on the function. Using this approximation would result in something like the secant method whose convergence is slower than that of Newton's method.

17.3.2 Failure of the method to converge to the root

It is important to review the proof of quadratic convergence of Newton's Method before implementing it. Specifically, one should review the assumptions made in the proof. For situations where the method fails to converge, it is because the assumptions made in this proof are not met.

Overshoot

If the first derivative is not well behaved in the neighborhood of a particular root, the method may overshoot, and diverge from that root. An example of a function with one root, for which the derivative is not well behaved in the neighborhood of the root, is

$$f(x) = |x|^a, \quad 0 < a < \tfrac{1}{2}$$

for which the root will be overshot and the sequence of x will diverge. For a = 1/2, the root will still be overshot, but the sequence will oscillate between two values. For 1/2 < a < 1, the root will still be overshot but the sequence will converge, and for a ≥ 1 the root will not be overshot at all.

In some cases, Newton's method can be stabilized by using successive over-relaxation, or the speed of convergence can be increased by using the same method.

Stationary point

If a stationary point of the function is encountered, the derivative is zero and the method will terminate due to division by zero.

Poor initial estimate

A large error in the initial estimate can contribute to non-convergence of the algorithm.

Mitigation of non-convergence

In a robust implementation of Newton's method, it is common to place limits on the number of iterations, bound the solution to an interval known to contain the root, and combine the method with a more robust root finding method.

17.3.3 Slow convergence for roots of multiplicity > 1

If the root being sought has multiplicity greater than one, the convergence rate is merely linear (errors reduced by a constant factor at each step) unless special steps are taken. When there are two or more roots that are close together then it may take many iterations before the iterates get close enough to one of them for the quadratic convergence to be apparent. However, if the multiplicity m of the root is known, one can use the following modified algorithm that preserves the quadratic convergence rate:

$$x_{n+1} = x_n - m\tfrac{f(x_n)}{f'(x_n)}.$$ [1]

This is equivalent to using successive over-relaxation. On the other hand, if the multiplicity m of the root is not known, it is possible to estimate m after carrying out one or two iterations, and then use that value to increase the rate of convergence.

17.4 Analysis

Suppose that the function f has a zero at α, i.e., f(α) = 0, and f is differentiable in a neighborhood of α.

If f is continuously differentiable and its derivative is nonzero at α, then there exists a neighborhood of α such that for all starting values x_0 in that neighborhood, the sequence {xn} will converge to α.[2]

If the function is continuously differentiable and its derivative is not 0 at α and it has a second derivative at α then the convergence is quadratic or faster. If the second derivative is not 0 at α then the convergence is merely quadratic. If the third derivative exists and is bounded in a neighborhood of α, then:

$$\Delta x_{i+1} = \frac{f''(\alpha)}{2f'(\alpha)}(\Delta x_i)^2 + O[\Delta x_i]^3\,,$$

where $\Delta x_i \triangleq x_i - \alpha\,.$

If the derivative is 0 at α, then the convergence is usually only linear. Specifically, if f is twice continuously differentiable, $f\,'(\alpha) = 0$ and $f\,''(\alpha) \neq 0$, then there exists a neighborhood of α such that for all starting values x_0 in that neighborhood, the sequence of iterates converges linearly, with rate $\log_{10} 2$ (Süli & Mayers, Exercise 1.6). Alternatively if $f\,'(\alpha) = 0$ and $f\,'(x) \neq 0$ for $x \neq \alpha$, x in a neighborhood U of α, α being a zero of multiplicity r, and if $f \in C^r(U)$ then there exists a neighborhood of α such that for all starting values x_0 in that neighborhood, the sequence of iterates converges linearly.

However, even linear convergence is not guaranteed in pathological situations.

In practice these results are local, and the neighborhood of convergence is not known in advance. But there are also some results on global convergence: for instance, given a right neighborhood U+ of α, if f is twice differentiable in U+ and if $f' \neq 0$, $f \cdot f'' > 0$in U+, then, for each x_0 in U_+ the sequence xk is monotonically decreasing to α.

17.4.1 Proof of quadratic convergence for Newton's iterative method

According to Taylor's theorem, any function $f(x)$ which has a continuous second derivative can be represented by an expansion about a point that is close to a root of f(x). Suppose this root is α . Then the expansion of f(α) about xn is:

where the Lagrange form of the Taylor series expansion remainder is

$$R_1 = \frac{1}{2!}f''(\xi_n)(\alpha - x_n)^2\,,$$

where ξ*n* is in between *xn* and α .

Since α is the root, (1) becomes:

Dividing equation (2) by $f'(x_n)$ and rearranging gives

Remembering that *xn*+1 is defined by

one finds that

$$\underbrace{\alpha - x_{n+1}}_{\epsilon_{n+1}} = \frac{-f''(\xi_n)}{2f'(x_n)}\underbrace{(\alpha - x_n)}_{\epsilon_n}^2\,.$$

That is,

Taking absolute value of both sides gives

Equation (6) shows that the rate of convergence is quadratic if the following conditions are satisfied:

1. $f'(x) \neq 0; \forall x \in I$ where ,I interval the is $[\alpha - r, \alpha + r]$ some for $r \geq |(\alpha - x_0)|$;
2. $f''(x)$ finite is , $\forall x \in I$;
3. x_0 *sufficiently* close to the root α

The term *sufficiently* close in this context means the following:

(a) Taylor approximation is accurate enough such that we can ignore higher order terms,

(b) $\frac{1}{2}\left|\frac{f''(x_n)}{f'(x_n)}\right| < C\left|\frac{f''(\alpha)}{f'(\alpha)}\right|$, some for $C < \infty$,

(c) $C\left|\frac{f''(\alpha)}{f'(\alpha)}\right|\epsilon_n < 1$, for $n \in$ Z$^+ \cup \{0\}$ and C (b) condition satisfying .

Finally, (6) can be expressed in the following way:

$$|\epsilon_{n+1}| \leq M{\epsilon_n}^2$$

where M is the supremum of the variable coefficient of ${\epsilon_n}^2$ on the interval I defined in the condition 1, that is:

$$M = \sup_{x \in I} \frac{1}{2}\left|\frac{f''(x)}{f'(x)}\right|.$$

The initial point x_0 has to be chosen such that conditions 1 through 3 are satisfied, where the third condition requires that $M\,|\epsilon_0| < 1$.

17.4.2 Basins of attraction

The basins of attraction—the regions of the real number line such that within each region iteration from any point leads to one particular root—can be infinite in number and arbitrarily small. For example,[3] for the function $f(x) = x^3 - 2x^2 - 11x + 12$, the following initial conditions are in successive basins of attraction:

2.35287527 converges to 4;

2.35284172 converges to –3;

2.35283735 converges to 4;

2.352836327 converges to –3;

2.352836323 converges to 1.

17.5 Failure analysis

Newton's method is only guaranteed to converge if certain conditions are satisfied. If the assumptions made in the proof of quadratic convergence are met, the method will converge. For the following subsections, failure of the method to converge indicates that the assumptions made in the proof were not met.

17.5.1 Bad starting points

In some cases the conditions on the function that are necessary for convergence are satisfied, but the point chosen as the initial point is not in the interval where the method converges. This can happen, for example, if the function whose root is sought approaches zero asymptotically as x goes to ∞ or $-\infty$. In such cases a different method, such as bisection, should be used to obtain a better estimate for the zero to use as an initial point.

Iteration point is stationary

Consider the function:

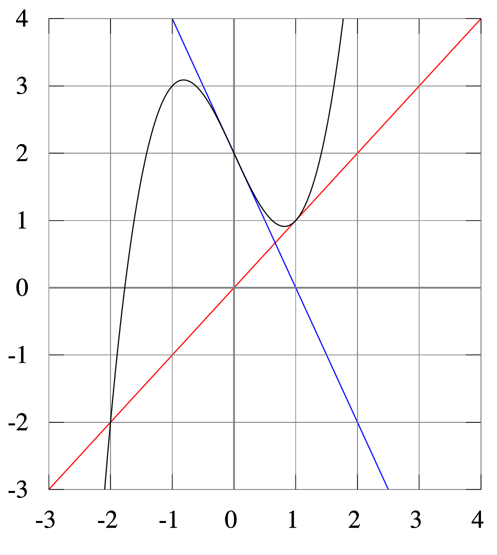
$f(x) = 1 - x^2.$

It has a maximum at $x = 0$ and solutions of $f(x) = 0$ at $x = \pm 1$. If we start iterating from the stationary point $x_0 = 0$ (where the derivative is zero), x_1 will be undefined, since the tangent at (0,1) is parallel to the x-axis:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0 - \frac{1}{0}.$$

The same issue occurs if, instead of the starting point, any iteration point is stationary. Even if the derivative is small but not zero, the next iteration will be a far worse approximation.

Starting point enters a cycle



The tangent lines of x^3 *- 2*x *+ 2 at 0 and 1 intersect the* x*-axis at 1 and 0 respectively, illustrating why Newton's method oscillates between these values for some starting points.*

For some functions, some starting points may enter an infinite cycle, preventing convergence. Let

$f(x) = x^3 - 2x + 2$

and take 0 as the starting point. The first iteration produces 1 and the second iteration returns to 0 so the sequence will alternate between the two without converging to a root. In fact, this 2-cycle is stable: there are neighborhoods around 0 and around 1 from which all points iterate asymptotically to the 2-cycle (and hence not to the root of the function). In general, the behavior of the sequence can be very complex (see Newton fractal).

17.5.2 Derivative issues

If the function is not continuously differentiable in a neighborhood of the root then it is possible that Newton's method will always diverge and fail, unless the solution is guessed on the first try.

Derivative does not exist at root

A simple example of a function where Newton's method diverges is the cube root, which is continuous and infinitely differentiable, except for $x = 0$, where its derivative is undefined (this, however, does not affect the algorithm, since it will never require the derivative if the solution is already found):

$f(x) = \sqrt[3]{x}.$

For any iteration point *xn*, the next iteration point will be:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{{x_n}^{\frac{1}{3}}}{\frac{1}{3}\,{x_n}^{\frac{1}{3}-1}} = x_n - 3\,x_n = -2\,x_n.$$

The algorithm overshoots the solution and lands on the other side of the y-axis, farther away than it initially was; applying Newton's method actually doubles the distances from the solution at each iteration.

In fact, the iterations diverge to infinity for every $f(x) = |x|^\alpha$, where $0 < \alpha < \frac{1}{2}$. In the limiting case of $\alpha = \frac{1}{2}$ (square root), the iterations will alternate indefinitely between points x_0 and $-x_0$, so they do not converge in this case either.

Discontinuous derivative

If the derivative is not continuous at the root, then convergence may fail to occur in any neighborhood of the root. Consider the function

$$f(x) = \begin{cases} 0 & \text{if} x = 0, \\ x + x^2 \sin\left(\frac{2}{x}\right) & \text{if} x \neq 0. \end{cases}$$

Its derivative is:

$$f'(x) = \begin{cases} 1 & \text{if} x = 0, \\ 1 + 2\,x\,\sin\left(\frac{2}{x}\right) - 2\,\cos\left(\frac{2}{x}\right) & \text{if} x \neq 0. \end{cases}$$

Within any neighborhood of the root, this derivative keeps changing sign as x approaches 0 from the right (or from the left) while $f(x) \geq x - x^2 > 0$ for $0 < x < 1$.

So $f(x)/f'(x)$ is unbounded near the root, and Newton's method will diverge almost everywhere in any neighborhood of it, even though:

- the function is differentiable (and thus continuous) everywhere;
- the derivative at the root is nonzero;
- *f* is infinitely differentiable except at the root; and
- the derivative is bounded in a neighborhood of the root (unlike *f*(*x*)/*f*'(*x*)).

17.5.3 Non-quadratic convergence

In some cases the iterates converge but do not converge as quickly as promised. In these cases simpler methods converge just as quickly as Newton's method.

Zero derivative

If the first derivative is zero at the root, then convergence will not be quadratic. Indeed, let

$$f(x) = x^2$$

then $f'(x) = 2x$ and consequently $x - f(x)/f'(x) = x/2$. So convergence is not quadratic, even though the function is infinitely differentiable everywhere.

Similar problems occur even when the root is only "nearly" double. For example, let

$$f(x) = x^2(x - 1000) + 1.$$

Then the first few iterates starting at $x_0 = 1$ are 1, 0.500250376, 0.251062828, 0.127507934, 0.067671976, 0.041224176, 0.032741218, 0.031642362; it takes six iterations to reach a point where the convergence appears to be quadratic.

No second derivative

If there is no second derivative at the root, then convergence may fail to be quadratic. Indeed, let

$$f(x) = x + x^{\frac{4}{3}}.$$

Then

$$f'(x) = 1 + \frac{4}{3}x^{\frac{1}{3}}.$$

And

$$f''(x) = \frac{4}{9}x^{-\frac{2}{3}}$$

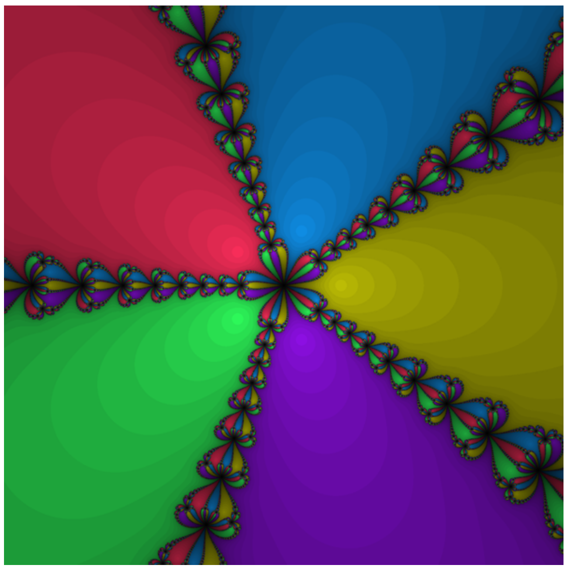
except when $x = 0$ where it is undefined. Given x_n,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = \frac{\frac{1}{3}x_n{}^{\frac{4}{3}}}{(1 + \frac{4}{3}x_n{}^{\frac{1}{3}})}$$

which has approximately 4/3 times as many bits of precision as x_n has. This is less than the 2 times as many which would be required for quadratic convergence. So the convergence of Newton's method (in this case) is not quadratic, even though: the function is continuously differentiable everywhere; the derivative is not zero at the root; and *f* is infinitely differentiable except at the desired root.

17.6 Generalizations

17.6.1 Complex functions



Basins of attraction for x^5 - 1 = 0; darker means more iterations to converge.

Main article: Newton fractal

When dealing with complex functions, Newton's method can be directly applied to find their zeroes. Each zero has a basin of attraction in the complex plane, the set of all starting values that cause the method to converge to that particular zero. These sets can be mapped as in the image shown. For many complex functions, the boundaries of the basins of attraction are fractals.

In some cases there are regions in the complex plane which are not in any of these basins of attraction, meaning the iterates do not converge. For example,[4] if one uses a real initial condition to seek a root of $x^2 + 1$, all subsequent iterates will be real numbers and so the iterations cannot converge to either root, since both roots are non-real. In this case almost all real initial conditions lead to chaotic behavior, while some initial conditions iterate either to infinity or to repeating cycles of any finite length.

17.6.2 Nonlinear systems of equations

k variables, k functions

One may also use Newton's method to solve systems of k (non-linear) equations, which amounts to finding the zeroes of continuously differentiable functions $F : \mathbf{R}^k \to \mathbf{R}^k$. In the formulation given above, one then has to left multiply with the inverse of the k-by-k Jacobian matrix $JF(xn)$ instead of dividing by $f\,'(xn)$.

Rather than actually computing the inverse of this matrix, one can save time by solving the system of linear equations

$$J_F(x_n)(x_{n+1} - x_n) = -F(x_n)$$

for the unknown $xn_{+1} - xn$.

k variables, m equations, with m > k

The k-dimensional Newton's method can be used to solve systems of >k (non-linear) equations as well if the algorithm uses the generalized inverse of the non-square Jacobian matrix $J^+ = ((J^TJ)^{-1})J^T$ instead of the inverse of J. If the nonlinear system has no solution, the method attempts to find a solution in the non-linear least squares sense. See Gauss–Newton algorithm for more information.

17.6.3 Nonlinear equations in a Banach space

Another generalization is Newton's method to find a root of a functional F defined in a Banach space. In this case the formulation is

$$X_{n+1} = X_n - [F'(X_n)]^{-1}F(X_n),$$

where $F'(X_n)$ is the Fréchet derivative computed at X_n . One needs the Fréchet derivative to be boundedly invertible at each X_n in order for the method to be applicable. A condition for existence of and convergence to a root is given by the Newton–Kantorovich theorem.

17.6.4 Nonlinear equations over *p*-adic numbers

In p-adic analysis, the standard method to show a polynomial equation in one variable has a p-adic root is Hensel's lemma, which uses the recursion from Newton's method on the p-adic numbers. Because of the more stable behavior of addition and multiplication in the p-adic numbers compared to the real numbers (specifically, the unit ball in the p-adics is a ring), convergence in Hensel's lemma can be guaranteed under much simpler hypotheses than in the classical Newton's method on the real line.

17.6.5 Newton-Fourier method

The Newton-Fourier method is Joseph Fourier's extension of Newton's method to provide bounds on the absolute error of the root approximation, while still providing quadratic convergence.

Assume that $f(x)$ is twice continuously differentiable on $[a, b]$ and that f contains a root in this interval. Assume that $f'(x)f''(x) \neq 0$ on this interval (this is the case for instance if $f(a) < 0$, $f(b) > 0$, and $f'(x) > 0$, and $f''(x) > 0$ on this interval). This guarantees that there is a unique root on this interval, call it α . If it is concave down instead of concave up then replace $f(x)$ by $-f(x)$ since they have the same roots.

Let $x_0 = b$ be the right endpoint of the interval and let $z_0 = a$ be the left endpoint of the interval. Given x_n , define $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, which is just Newton's method as before. Then define $z_{n+1} = z_n - \frac{f(z_n)}{f'(x_n)}$ and note that the denominator has $f'(x_n)$ and not $f'(z_n)$. The iterates x_n will be strictly decreasing to the root while the iterates z_n will be strictly increasing to the root. Also, $\lim_{n\to\infty} \frac{x_{n+1}-z_{n+1}}{(x_n-z_n)^2} = \frac{f''(\alpha)}{2f'(\alpha)}$ so that distance between x_n and z_n decreases quadratically.

17.6.6 Quasi-Newton methods

When the Jacobian is unavailable or too expensive to compute at every iteration, a Quasi-Newton method can be used.

17.7 Applications

17.7.1 Minimization and maximization problems

Main article: Newton's method in optimization

Newton's method can be used to find a minimum or maximum of a function. The derivative is zero at a minimum or maximum, so minima and maxima can be found by applying Newton's method to the derivative. The iteration becomes:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

17.7.2 Multiplicative inverses of numbers and power series

An important application is Newton–Raphson division, which can be used to quickly find the reciprocal of a number, using only multiplication and subtraction.

Finding the reciprocal of *a* amounts to finding the root of the function

$$f(x) = a - \frac{1}{x}$$

Newton's iteration is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ &= x_n - \frac{a - \frac{1}{x_n}}{\frac{1}{x_n^2}} \\ &= x_n\,(2 - ax_n) \end{aligned}$$

Therefore, Newton's iteration needs only two multiplications and one subtraction.

This method is also very efficient to compute the multiplicative inverse of a power series.

17.7.3 Solving transcendental equations

Many transcendental equations can be solved using Newton's method. Given the equation

$$g(x) = h(x),$$

with *g(x)* and/or *h(x)* a transcendental function, one writes

$$f(x) = g(x) - h(x).$$

The values of *x* that solves the original equation are then the roots of *f(x)*, which may be found via Newton's method.

17.8 Examples

17.8.1 Square root of a number

Consider the problem of finding the square root of a number. Newton's method is one of many methods of computing square roots.

For example, if one wishes to find the square root of 612, this is equivalent to finding the solution to

$$x^2 = 612$$

The function to use in Newton's method is then,

$$f(x) = x^2 - 612$$

with derivative,

$$f'(x) = 2x.$$

With an initial guess of 10, the sequence given by Newton's method is

$$\begin{array}{rclclcl} x_1 & = & x_0 - \dfrac{f(x_0)}{f'(x_0)} & = & 10 - \dfrac{10^2 - 612}{2 \cdot 10} & = & 35.6 \\ x_2 & = & x_1 - \dfrac{f(x_1)}{f'(x_1)} & = & 35.6 - \dfrac{35.6^2 - 612}{2 \cdot 35.6} & = & \underline{2}6.39550561797 \\ x_3 & = & \vdots & = & \vdots & = & \underline{24.7}9063549245 \\ x_4 & = & \vdots & = & \vdots & = & \underline{24.7386}8829407 \\ x_5 & = & \vdots & = & \vdots & = & \underline{24.7386337537}6 \end{array}$$

where the correct digits are underlined. With only a few iterations one can obtain a solution accurate to many decimal places.

17.8.2 Solution of cos(*x*) = *x*³

Consider the problem of finding the positive number x with $\cos(x) = x^3$. We can rephrase that as finding the zero of $f(x) = \cos(x) - x^3$. We have $f'(x) = -\sin(x) - 3x^2$. Since $\cos(x) \le 1$ for all x and $x^3 > 1$ for $x > 1$, we know that our solution lies between 0 and 1. We try a starting value of $x_0 = 0.5$. (Note that a starting value of 0 will lead to an undefined result, showing the importance of using a starting point that is close to the solution.)

$$\begin{array}{rclclcl} x_1 & = & x_0 - \dfrac{f(x_0)}{f'(x_0)} & = & 0.5 - \dfrac{\cos(0.5) - (0.5)^3}{-\sin(0.5) - 3(0.5)^2} & = & 1.112141 \\ x_2 & = & x_1 - \dfrac{f(x_1)}{f'(x_1)} & = & \vdots & = & \underline{0.}909672 \\ x_3 & = & \vdots & = & \vdots & = & \underline{0.86}7263 \\ x_4 & = & \vdots & = & \vdots & = & \underline{0.86547}7 \\ x_5 & = & \vdots & = & \vdots & = & \underline{0.865474} \\ x_6 & = & \vdots & = & \vdots & = & \underline{0.865474} \end{array}$$

The correct digits are underlined in the above example. In particular, x_6 is correct to the number of decimal places given. We see that the number of correct digits after the decimal point increases from 2 (for x_3) to 5 and 10, illustrating the quadratic convergence.

17.9 Pseudocode

The following is an example of using the Newton's Method to help find a root of a function f which has derivative fprime.

The initial guess will be $x_0 = 1$ and the function will be $f(x) = x^2 - 2$ so that $f'(x) = 2x$.

Each new iterative of Newton's method will be denoted by x1. We will check during the computation whether the denominator (yprime) becomes too small (smaller than epsilon), which would be the case if $f'(x_n) \approx 0$, since otherwise a large amount of error could be introduced.

%These choices depend on the problem being solved x0 = 1 %The initial value f = @(x) x^2 - 2 %The function whose root we are trying to find fprime = @(x) 2*x %The derivative of f(x) tolerance = 10^(−7) %7 digit accuracy is desired epsilon = 10^(−14) %Don't want to divide by a number smaller than this maxIterations = 20 %Don't allow the iterations to continue indefinitely haveWeFoundSolution = false %Have not converged to a solution yet for i = 1 : maxIterations y = f(x0) yprime = fprime(x0) if(abs(yprime) < epsilon) %Don't want to divide by too small of a number % denominator is too small break; %Leave the loop end x1 = x0 - y/yprime %Do Newton's computation if(abs(x1 - x0)/abs(x1) < tolerance) %If the result is within the desired tolerance haveWeFoundSolution = true break; %Done, so leave the loop end x0 = x1 %Update x0 to start the process again end if (haveWeFoundSolution) ... % x1 is a solution within tolerance and maximum number of iterations else ... % did not converge end

17.10 See also

- Aitken's delta-squared process
- Bisection method
- Euler method
- Fast inverse square root
- Fisher scoring
- Gradient descent
- Integer square root
- Laguerre's method
- Leonid Kantorovich, who initiated the convergence analysis of Newton's method in Banach spaces.
- Methods of computing square roots
- Newton's method in optimization
- Richardson extrapolation
- Root-finding algorithm
- Secant method
- Steffensen's method
- Subgradient method

17.11 References

[1] "Accelerated and Modified Newton Methods".

[2] Ryaben'kii, Victor S.; Tsynkov, Semyon V. (2006), *A Theoretical Introduction to Numerical Analysis*, CRC Press, p. 243, ISBN 9781584886075.

[3] Dence, Thomas, "Cubics, chaos and Newton's method", *Mathematical Gazette* 81, November 1997, 403-408.

[4] Strang, Gilbert, "A chaotic search for *i*", "*The College Mathematics Journal* 22, January 1991, pp. 3-12 (esp. p. 6).

- Kendall E. Atkinson, *An Introduction to Numerical Analysis*, (1989) John Wiley & Sons, Inc, ISBN 0-471-62489-6
- Tjalling J. Ypma, Historical development of the Newton-Raphson method, *SIAM Review* **37** (4), 531–551, 1995. doi:10.1137/1037125.
- Bonnans, J. Frédéric; Gilbert, J. Charles; Lemaréchal, Claude; Sagastizábal, Claudia A. (2006). *Numerical optimization: Theoretical and practical aspects*. Universitext (Second revised ed. of translation of 1997 French ed.). Berlin: Springer-Verlag. pp. xiv+490. doi:10.1007/978-3-540-35447-5. ISBN 3-540-35445-X. MR 2265882.
- P. Deuflhard, *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms.* Springer Series in Computational Mathematics, Vol. 35. Springer, Berlin, 2004. ISBN 3-540-21099-7.
- C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, no 1 in Fundamentals of Algorithms, SIAM, 2003. ISBN 0-89871-546-6.
- J. M. Ortega, W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables.* Classics in Applied Mathematics, SIAM, 2000. ISBN 0-89871-461-3.
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Chapter 9. Root Finding and Nonlinear Sets of Equations Importance Sampling". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.. See especially Sections 9.4, 9.6, and 9.7.

- Endre Süli and David Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003. ISBN 0-521-00794-1.
- Kaw, Autar; Kalu, Egwu (2008). "Numerical Methods with Applications" (1st ed.).

17.12 External links

- Hazewinkel, Michiel, ed. (2001), "Newton method", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Weisstein, Eric W., "Newton's Method", *MathWorld*.

Chapter 18

Supervised learning

See also: Unsupervised learning

Supervised learning is the machine learning task of inferring a function from labeled training data.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias).

The parallel task in human and animal psychology is often referred to as concept learning.

18.1 Overview

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.

2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.

4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.

5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.

6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

A wide range of supervised learning algorithms is available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems (see the No free lunch theorem).

There are four major issues to consider in supervised learning:

18.1.1 Bias-variance tradeoff

Main article: Bias-variance dilemma

A first issue is the tradeoff between *bias* and *variance*.[2] Imagine that we have available several different, but equally good, training data sets. A learning algorithm is biased for a particular input x if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for x . A learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets. The prediction error of a learned classifier is related to the sum of the bias and the variance of the learning algorithm.[3] Generally, there is a tradeoff between bias and variance. A learning algorithm with low bias must be "flexible" so that it can fit the data well. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance. A key aspect of many supervised learning methods is that they are able to adjust this tradeoff between bias and variance (either automatically or by providing a bias/variance parameter that the user can adjust).

18.1.2 Function complexity and amount of training data

The second issue is the amount of training data available relative to the complexity of the "true" function (classifier or regression function). If the true function is simple, then an "inflexible" learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex (e.g., because it involves complex interactions among many different input features and behaves differently in different parts of the input space), then the function will only be learnable from a very large amount of training data and using a "flexible" learning algorithm with low bias and high variance. Good learning algorithms therefore automatically adjust the bias/variance tradeoff based on the amount of data available and the apparent complexity of the function to be learned.

18.1.3 Dimensionality of the input space

A third issue is the dimensionality of the input space. If the input feature vectors have very high dimension, the learning problem can be difficult even if the true function only depends on a small number of those features. This is because the many "extra" dimensions can confuse the learning algorithm and cause it to have high variance. Hence, high input dimensionality typically requires tuning the classifier to have low variance and high bias. In practice, if the engineer can manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, there are many algorithms for feature selection that seek to identify the relevant features and discard the irrelevant ones. This is an instance of the more general strategy of dimensionality reduction, which seeks to map the input data into a lower-dimensional space prior to running the supervised learning algorithm.

18.1.4 Noise in the output values

A fourth issue is the degree of noise in the desired output values (the supervisory target variables). If the desired output values are often incorrect (because of human error or sensor errors), then the learning algorithm should not attempt to find a function that exactly matches the training examples. Attempting to fit the data too carefully leads to overfitting. You can overfit even when there are no measurement errors (stochastic noise) if the function you are trying to learn is too complex for your learning model. In such a situation that part of the target function that cannot be modeled "corrupts" your training data - this phenomenon has been called deterministic noise. When either type of noise is present, it is better to go with a higher bias, lower variance estimator.

In practice, there are several approaches to alleviate noise in the output values such as early stopping to prevent overfitting as well as detecting and removing the noisy training examples prior to training the supervised learning algorithm. There are several algorithms that identify noisy training examples and removing the suspected noisy training examples prior to training has decreased generalization error with statistical significance.[4][5]

18.1.5 Other factors to consider

Other factors to consider when choosing and applying a learning algorithm include the following:

1. Heterogeneity of the data. If the feature vectors include features of many different kinds (discrete, discrete ordered, counts, continuous values), some algorithms are easier to apply than others. Many algorithms, including Support Vector Machines, linear regression, logistic regression, neural networks, and nearest neighbor methods, require that the input features be numerical and scaled to similar ranges (e.g., to the $[-1,1]$ interval). Methods that employ a distance function, such as nearest neighbor methods and support vector machines with Gaussian kernels, are particularly sensitive to this. An advantage of decision trees is that they easily handle heterogeneous data.

2. Redundancy in the data. If the input features contain redundant information (e.g., highly correlated features), some learning algorithms (e.g., linear regression, logistic regression, and distance based methods) will perform poorly because of numerical instabilities. These problems can often be solved by imposing some form of regularization.

3. Presence of interactions and non-linearities. If each of the features makes an independent contribution to the output, then algorithms based on linear functions (e.g., linear regression, logistic regression, Support Vector Machines, naive Bayes) and distance functions (e.g., nearest neighbor methods, support vector machines with Gaussian kernels) generally perform well. However, if there are complex interactions among features, then algorithms such as decision trees and neural networks work better, because they are specifically designed to discover these interactions. Linear methods can also be applied, but the engineer must manually specify the interactions when using them.

When considering a new application, the engineer can compare multiple learning algorithms and experimentally determine which one works best on the problem at hand (see cross validation). Tuning the performance of a learning algorithm can be very time-consuming. Given fixed resources, it is often better to spend more time collecting additional training data and more informative features than it is to spend extra time tuning the learning algorithms.

The most widely used learning algorithms are Support Vector Machines, linear regression, logistic regression, naive Bayes, linear discriminant analysis, decision trees, k-nearest neighbor algorithm, and Neural Networks (Multilayer perceptron).

18.2 How supervised learning algorithms work

Given a set of N training examples of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$ such that x_i is the feature vector of the i-th example and y_i is its label (i.e., class), a learning algorithm seeks a function $g : X \to Y$, where X is the input space and Y is the output space. The function g is an element of some space of possible functions G , usually called the *hypothesis space*. It is sometimes convenient to represent g using a scoring function $f : X \times Y \to \mathbb{R}$ such that g is defined as returning the y value that gives the highest score: $g(x) = \arg\max_y \ f(x, y)$. Let F denote the space of scoring functions.

Although G and F can be any space of functions, many learning algorithms are probabilistic models where g takes the form of a conditional probability model $g(x) = P(y|x)$, or f takes the form of a joint probability model $f(x, y) = P(x, y)$. For example, naive Bayes and linear discriminant analysis are joint probability models, whereas logistic regression is a conditional probability model.

There are two basic approaches to choosing f or g : empirical risk minimization and structural risk minimization.[6] Empirical risk minimization seeks the function that best fits the training data. Structural risk minimize includes a *penalty function* that controls the bias/variance tradeoff.

In both cases, it is assumed that the training set consists of a sample of independent and identically distributed pairs, $(x_i,\ y_i)$. In order to measure how well a function fits the training data, a loss function $L : Y \times Y \to \mathbb{R}^{\geq 0}$ is defined. For training example $(x_i,\ y_i)$, the loss of predicting the value $\hat{y}$ is $L(y_i, \hat{y})$.

The *risk* $R(g)$ of function g is defined as the expected loss of g . This can be estimated from the training data as

$$R_{emp}(g) = \frac{1}{N} \sum_i L(y_i, g(x_i))$$

18.2.1 Empirical risk minimization

Main article: Empirical risk minimization

In empirical risk minimization, the supervised learning algorithm seeks the function g that minimizes $R(g)$. Hence, a supervised learning algorithm can be constructed by applying an optimization algorithm to find g .

When g is a conditional probability distribution $P(y|x)$ and the loss function is the negative log likelihood: $L(y, \hat{y}) = -\log P(y|x)$, then empirical risk minimization is equivalent to maximum likelihood estimation.

When G contains many candidate functions or the training set is not sufficiently large, empirical risk minimization leads to high variance and poor generalization. The learning algorithm is able to memorize the training examples without generalizing well. This is called overfitting.

18.2.2 Structural risk minimization

Structural risk minimization seeks to prevent overfitting by incorporating a regularization penalty into the optimization. The regularization penalty can be viewed as implementing a form of Occam's razor that prefers simpler functions over more complex ones.

A wide variety of penalties have been employed that correspond to different definitions of complexity. For example, consider the case where the function g is a linear function of the form

$$g(x) = \sum_{j=1}^{d} \beta_j x_j$$

A popular regularization penalty is $\sum_j \beta_j^2$, which is the squared Euclidean norm of the weights, also known as the L_2 norm. Other norms include the L_1 norm, $\sum_j |\beta_j|$, and the L_0 norm, which is the number of non-zero β_j s. The penalty will be denoted by $C(g)$.

The supervised learning optimization problem is to find the function g that minimizes

$$J(g) = R_{emp}(g) + \lambda C(g).$$

The parameter λ controls the bias-variance tradeoff. When $\lambda = 0$, this gives empirical risk minimization with low bias and high variance. When λ is large, the learning algorithm will have high bias and low variance. The value of λ can be chosen empirically via cross validation.

The complexity penalty has a Bayesian interpretation as the negative log prior probability of g , $-\log P(g)$, in which case $J(g)$ is the posterior probabability of g .

18.3 Generative training

The training methods described above are *discriminative training* methods, because they seek to find a function g that discriminates well between the different output values (see discriminative model). For the special case where $f(x, y) = P(x, y)$ is a joint probability distribution and the loss function is the negative log likelihood $-\sum_i \log P(x_i, y_i)$, a risk minimization algorithm is said to perform *generative training*, because f can be regarded as a generative model that explains how the data were generated. Generative training algorithms are often simpler and more computationally efficient than discriminative training algorithms. In some cases, the solution can be computed in closed form as in naive Bayes and linear discriminant analysis.

18.4 Generalizations of supervised learning

There are several ways in which the standard supervised learning problem can be generalized:

1. Semi-supervised learning: In this setting, the desired output values are provided only for a subset of the training data. The remaining data is unlabeled.
2. Active learning: Instead of assuming that all of the training examples are given at the start, active learning algorithms interactively collect new examples, typically by making queries to a human user. Often, the queries are based on unlabeled data, which is a scenario that combines semi-supervised learning with active learning.
3. Structured prediction: When the desired output value is a complex object, such as a parse tree or a labeled graph, then standard methods must be extended.
4. Learning to rank: When the input is a set of objects and the desired output is a ranking of those objects, then again the standard methods must be extended.

18.5 Approaches and algorithms

- Analytical learning
- Artificial neural network
- Backpropagation
- Boosting (meta-algorithm)
- Bayesian statistics
- Case-based reasoning
- Decision tree learning
- Inductive logic programming
- Gaussian process regression
- Group method of data handling
- Kernel estimators
- Learning Automata
- Minimum message length (decision trees, decision graphs, etc.)
- Multilinear subspace learning
- Naive bayes classifier
- Nearest Neighbor Algorithm
- Probably approximately correct learning (PAC) learning
- Ripple down rules, a knowledge acquisition methodology
- Symbolic machine learning algorithms
- Subsymbolic machine learning algorithms
- Support vector machines
- Random Forests
- Ensembles of Classifiers
- Ordinal classification
- Data Pre-processing
- Handling imbalanced datasets
- Statistical relational learning
- Proaftn, a multicriteria classification algorithm

18.6 Applications

- Bioinformatics
- Cheminformatics
    - Quantitative structure–activity relationship
- Database marketing
- Handwriting recognition
- Information retrieval
    - Learning to rank
- Object recognition in computer vision
- Optical character recognition
- Spam detection
- Pattern recognition
- Speech recognition

18.7 General issues

- Computational learning theory
- Inductive bias
- Overfitting (machine learning)
- (Uncalibrated) Class membership probabilities
- Version spaces

18.8 References

[1] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) *Foundations of Machine Learning*, The MIT Press ISBN 9780262018258.

[2] S. Geman, E. Bienenstock, and R. Doursat (1992). Neural networks and the bias/variance dilemma. Neural Computation 4, 1–58.

[3] G. James (2003) Variance and Bias for General Loss Functions, Machine Learning 51, 115-135. (http://www-bcf.usc.edu/~{}gareth/research/bv.pdf)

[4] C.E. Brodely and M.A. Friedl (1999). Identifying and Eliminating Mislabeled Training Instances, Journal of Artificial Intelligence Research 11, 131-167. (http://jair.org/media/606/live-606-1803-jair.pdf)

[5] M.R. Smith and T. Martinez (2011). "Improving Classification Accuracy by Identifying and Removing Instances that Should Be Misclassified". *Proceedings of International Joint Conference on Neural Networks (IJCNN 2011)*. pp. 2690–2697.

[6] Vapnik, V. N. The Nature of Statistical Learning Theory (2nd Ed.), Springer Verlag, 2000.

18.9 External links

- mloss.org: a directory of open source machine learning software.

Chapter 19

Linear regression

In statistics, **linear regression** is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variable) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.[1] (This term should be distinguished from *multivariate linear regression*, where multiple correlated dependent variables are predicted, rather than a single scalar variable.)[2]

In linear regression, data are modeled using linear predictor functions, and unknown model parameters are estimated from the data. Such models are called *linear models*.[3] Most commonly, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X. Less commonly, linear regression could refer to a model in which the median, or some other quantile of the conditional distribution of y given X is expressed as a linear function of X. Like all forms of regression analysis, *linear regression* focuses on the conditional probability distribution of y given X, rather than on the joint probability distribution of y and X, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications.[4] This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

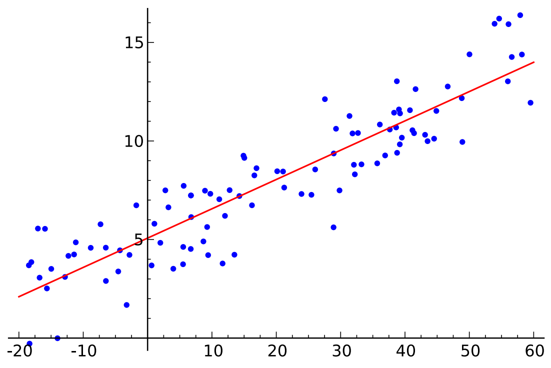
Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- If the goal is prediction, or forecasting, or reduction, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y.

- Given a variable y and a number of variables X_1, ..., Xp that may be related to y, linear regression analysis can be applied to quantify the strength of the relationship between y and the Xj, to assess which Xj may have no relationship with y at all, and to identify which subsets of the Xj contain redundant information about y.

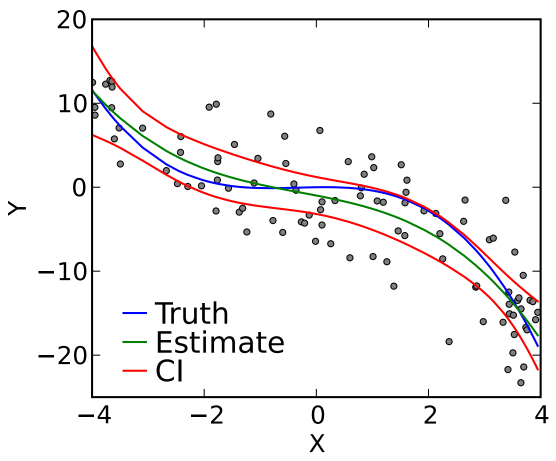
Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the “lack of fit” in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares loss function as in ridge regression (L2-norm penalty) and lasso (L1-norm penalty). Conversely, the least squares approach can be used to fit models that are not linear models. Thus, although the terms “least squares” and “linear model” are closely linked, they are not synonymous.

19.1 Introduction to linear regression



Example of simple linear regression, which has one independent variable

Given a data set $\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable yi and the p-vector of regressors xi is linear. This relationship is modeled through a *disturbance term* or *error variable* εi — an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form



Example of a cubic polynomial regression, which is a type of linear regression.

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm T}\boldsymbol\beta + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $^{\mathrm T}$ denotes the transpose, so that $\boldsymbol{xi}^{\mathrm T}\boldsymbol\beta$ is the inner product between vectors $\boldsymbol{xi}$ and $\boldsymbol\beta$.

Often these n equations are stacked together and written in vector form as

$$\mathbf{y} = \mathbf{X}\boldsymbol\beta + \boldsymbol\varepsilon,$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm T} \\ \mathbf{x}_2^{\mathrm T} \\ \vdots \\ \mathbf{x}_n^{\mathrm T} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Some remarks on terminology and general use:

- y_i is called the *regressand*, *endogenous variable*, *response variable*, *measured variable*, *criterion variable*, or *dependent variable* (see dependent and independent variables.) The decision as to which variable in a data set is modeled as the dependent variable and which are modeled as the independent variables may be based on a presumption that the value of one of the variables is caused by, or directly influenced by the other variables. Alternatively, there may be an operational reason to model one of the variables in terms of the others, in which case there need be no presumption of causality.

- $x_{i1}, x_{i2}, \ldots, x_{ip}$ are called *regressors*, *exogenous variables*, *explanatory variables*, *covariates*, *input variables*, *predictor variables*, or *independent variables* (see dependent and independent variables, but not to be confused with independent random variables). The matrix **X** is sometimes called the design matrix.

  - Usually a constant is included as one of the regressors. For example we can take $xi_1 = 1$ for $i = 1, \ldots, n$. The corresponding element of $\boldsymbol\beta$ is called the *intercept*. Many statistical inference procedures for linear models require an intercept to be present, so it is often included even if theoretical considerations suggest that its value should be zero.

  - Sometimes one of the regressors can be a non-linear function of another regressor or of the data, as in polynomial regression and segmented regression. The model remains linear as long as it is linear in the parameter vector $\boldsymbol\beta$.

  - The regressors *xij* may be viewed either as random variables, which we simply observe, or they can be considered as predetermined fixed values which we can choose. Both interpretations may be appropriate in different cases, and they generally lead to the same estimation procedures; however different approaches to asymptotic analysis are used in these two situations.

- $\boldsymbol\beta$ is a p-dimensional *parameter vector*. Its elements are also called *effects*, or *regression coefficients*. Statistical estimation and inference in linear regression focuses on $\boldsymbol\beta$. The elements of this parameter vector are interpreted as the partial derivatives of the dependent variable with respect to the various independent variables.

- ε_i is called the *error term*, *disturbance term*, or *noise*. This variable captures all other factors which influence the dependent variable *yi* other than the regressors $\boldsymbol{xi}$. The relationship between the error term and the regressors, for example whether they are correlated, is a crucial step in formulating a linear regression model, as it will determine the method to use for estimation.

Example. Consider a situation where a small ball is being tossed up in the air and then we measure its heights of ascent *hi* at various moments in time *ti*. Physics tells us that, ignoring the drag, the relationship can be modeled as

$$h_i = \beta_1 t_i + \beta_2 t_i^2 + \varepsilon_i,$$

where β_1 determines the initial velocity of the ball, β_2 is proportional to the standard gravity, and εi is due to measurement errors. Linear regression can be used to estimate the values of β_1 and β_2 from the measured data.

This model is non-linear in the time variable, but it is linear in the parameters β_1 and β_2; if we take regressors ***xi*** = $(xi_1, xi_2) = (ti, ti^2)$, the model takes on the standard form

$$h_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i.$$

19.1.1 Assumptions

Standard linear regression models with standard estimation techniques make a number of assumptions about the predictor variables, the response variables and their relationship. Numerous extensions have been developed that allow each of these assumptions to be relaxed (i.e. reduced to a weaker form), and in some cases eliminated entirely. Some methods are general enough that they can relax multiple assumptions at once, and in other cases this can be achieved by combining different extensions. Generally these extensions make the estimation procedure more complex and time-consuming, and may also require more data in order to produce an equally precise model.

The following are the major assumptions made by standard linear regression models with standard estimation techniques (e.g. ordinary least squares):

- **Weak exogeneity**. This essentially means that the predictor variables *x* can be treated as fixed values, rather than random variables. This means, for example, that the predictor variables are assumed to be error-free—that is, not contaminated with measurement errors. Although this assumption is not realistic in many settings, dropping it leads to significantly more difficult errors-in-variables models.

- **Linearity**. This means that the mean of the response variable is a linear combination of the parameters (regression coefficients) and the predictor variables. Note that this assumption is much less restrictive than it may at first seem. Because the predictor variables are treated as fixed values (see above), linearity is really only a restriction on the parameters. The predictor variables themselves can be arbitrarily transformed, and in fact multiple copies of the same underlying predictor variable can be added, each one transformed differently. This trick is used, for example, in polynomial regression, which uses linear regression to fit the response variable as an arbitrary polynomial function (up to a given rank) of a predictor variable. This makes linear regression an extremely powerful inference method. In fact, models such as polynomial regression are often "too powerful", in that they tend to overfit the data. As a result, some kind of regularization must typically be used to prevent unreasonable solutions coming out of the estimation process. Common examples are ridge regression and lasso regression. Bayesian linear regression can also be used, which by its nature is more or less immune to the problem of overfitting. (In fact, ridge regression and lasso regression can both be viewed as special cases of Bayesian linear regression, with particular types of prior distributions placed on the regression coefficients.)

- **Constant variance** (a.k.a. **homoscedasticity**). This means that different response variables have the same variance in their errors, regardless of the values of the predictor variables. In practice this assumption is invalid (i.e. the errors are heteroscedastic) if the response variables can vary over a wide scale. In order to determine for heterogeneous error variance, or when a pattern of residuals violates model assumptions of homoscedasticity (error is equally variable around the 'best-fitting line' for all points of x), it is prudent to look for a "fanning effect" between residual error and predicted values. This is to say there will be a systematic change in the absolute or squared residuals when plotted against the predicting outcome. Error will not be evenly distributed across the regression line. Heteroscedasticity will result in the averaging over of distinguishable variances around the points to get a single variance that is inaccurately representing all the variances of the line. In effect, residuals appear clustered and spread apart on their predicted plots for larger and smaller values for points along the linear regression line, and the mean squared error for the model will be wrong. Typically, for example, a response variable whose mean is large will have a greater variance than one whose mean is small. For example, a given person whose income is predicted to be \$100,000 may easily have an actual income of \$80,000 or \$120,000 (a standard deviation of around \$20,000), while another person with a predicted income of \$10,000 is unlikely to have the same \$20,000 standard deviation, which would imply their actual income would vary anywhere between -\$10,000 and \$30,000. (In fact, as this shows, in many cases—often the same cases where the assumption of normally distributed errors fails—the variance or standard deviation should be predicted to be proportional to the mean, rather than constant.) Simple linear regression estimation methods give less precise parameter estimates and misleading inferential quantities such as standard errors when substantial heteroscedasticity is present. However, various estimation techniques (e.g. weighted least squares and heteroscedasticity-consistent standard errors) can handle heteroscedasticity in a quite general way. Bayesian linear regression techniques can also be used when the variance is assumed to be a function of the mean. It is also possible in some cases to fix the problem by applying a transformation to the response variable (e.g. fit the logarithm of the response variable using a linear regression model, which implies that the response variable has a log-normal distribution rather than a normal distribution).

- **Independence** of errors. This assumes that the errors of the response variables are uncorrelated with each other. (Actual statistical independence is a stronger condition than mere lack of correlation and is often not needed, although it can be exploited if it is known to hold.) Some methods (e.g. generalized least squares) are capable of handling correlated errors, although they typically require significantly more data unless some sort of regularization is used to bias the model towards assuming uncorrelated errors. Bayesian linear regression is a general way of handling this issue.

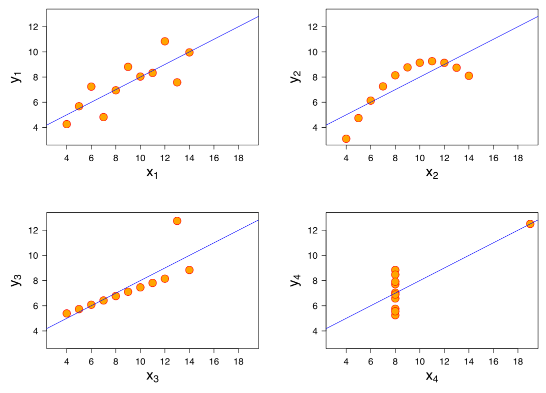
- **Lack of multicollinearity** in the predictors. For standard least squares estimation methods, the design matrix X must have full column rank p,; otherwise, we have a condition known as multicollinearity in the predictor variables. This can be triggered by having two or more perfectly correlated predictor variables (e.g. if the same predictor variable is mistakenly given twice, either without transforming one of the copies or by transforming one of the copies linearly). It can also happen if there is too little data available compared to the number of parameters to be estimated (e.g. fewer data points than regression coefficients). In the case of multicollinearity, the parameter vector β will be non-identifiable—it has no unique solution. At most we will be able to identify some of the parameters, i.e. narrow down its value to some linear subspace of $\mathbf{R}^p$. See partial least squares regression. Methods for fitting linear models with multicollinearity have been developed;[5][6][7][8] some require additional assumptions such as "effect sparsity"—that a large fraction of the effects are exactly zero. Note that the more computationally expensive iterated algorithms for parameter estimation, such as those used in generalized linear models, do not suffer from this problem—and in fact it's quite normal to when handling categorically valued predictors to introduce a separate indicator variable predictor for each possible category, which inevitably introduces multicollinearity.

Beyond these assumptions, several other statistical properties of the data strongly influence the performance of different estimation methods:

- The statistical relationship between the error terms and the regressors plays an important role in determining whether an estimation procedure has desirable sampling properties such as being unbiased and consistent.

- The arrangement, or probability distribution of the predictor variables x has a major influence on the precision of estimates of β. Sampling and design of experiments are highly developed subfields of statistics that provide guidance for collecting data in such a way to achieve a precise estimate of β.

19.1.2 Interpretation



The sets in the Anscombe's quartet have the same linear regression line but are themselves very different.

A fitted linear regression model can be used to identify the relationship between a single predictor variable xj and the response variable y when all the other predictor variables in the model are "held fixed". Specifically, the interpretation of βj is the expected change in y for a one-unit change in xj when the other covariates are held fixed—that is, the expected value of the partial derivative of y with respect to xj. This is sometimes called the *unique effect* of xj on y. In contrast, the *marginal effect* of xj on y can be assessed using a correlation coefficient or simple linear regression model relating xj to y; this effect is the total derivative of y with respect to xj.

Care must be taken when interpreting regression results, as some of the regressors may not allow for marginal changes (such as dummy variables, or the intercept term), while others cannot be held fixed (recall the example from the introduction: it would be impossible to "hold ti fixed" and at the same time change the value of ti^2).

It is possible that the unique effect can be nearly zero even when the marginal effect is large. This may imply that some other covariate captures all the information in xj, so that once that variable is in the model, there is no contribution of xj to the variation in y. Conversely, the unique effect of xj can be large while its marginal effect is nearly zero. This would happen if the other covariates explained a great deal of the variation of y, but they mainly explain variation in a way that is complementary to what is captured by xj. In this case, including the other variables in the model reduces the part of the variability of y that is unrelated to xj, thereby strengthening the apparent relationship with xj.

The meaning of the expression "held fixed" may depend on how the values of the predictor variables arise. If the experimenter directly sets the values of the predictor variables according to a study design, the comparisons of interest may literally correspond to comparisons among units whose predictor variables have been "held fixed" by the experimenter. Alternatively, the expression "held fixed" can refer to a selection that takes place in the context of data analysis. In this case, we "hold a variable fixed" by restricting our attention to the subsets of the data that happen to have a common value for the given predictor variable. This is the only interpretation of "held fixed" that can be used in an observational study.

The notion of a "unique effect" is appealing when studying a complex system where multiple interrelated components influence the response variable. In some cases, it can literally be interpreted as the causal effect of an intervention that is linked to the value of a predictor variable. However, it has been argued that in many cases multiple regression analysis fails to clarify the relationships between the predictor variables and the response variable when the predictors are correlated with each other and are not assigned following a study design.[9] A commonality analysis may be helpful in disentangling the shared and unique impacts of correlated independent variables.[10]

19.2 Extensions

Numerous extensions of linear regression have been developed, which allow some or all of the assumptions underlying the basic model to be relaxed.

19.2.1 Simple and multiple regression

The very simplest case of a single scalar predictor variable x and a single scalar response variable y is known as *simple linear regression*. The extension to multiple and/or vector-valued predictor variables (denoted with a capital X) is known as *multiple linear regression*, also known as *multivariable linear regression*. Nearly all real-world regression models involve multiple predictors, and basic descriptions of linear regression are often phrased in terms of the multiple regression model. Note, however, that in these cases the response variable y is still a scalar. Another term *multivariate linear regression* refers to cases where y is a vector, i.e., the same as *general linear regression*. The difference between *multivariate* linear regression and *multivariable* linear regression should be emphasized as it causes much confusion and misunderstanding in the literature.

19.2.2 General linear models

The general linear model considers the situation when the response variable Y is not a scalar but a vector. Conditional linearity of $E(y|x) = Bx$ is still assumed, with a matrix B replacing the vector β of the classical linear regression model. Multivariate analogues of OLS and GLS have been developed. The term "general linear models" is equivalent to "multivariate linear models". It should be noted the difference of "multivariate linear models" and "multivariable linear models," where the former is the same as "general linear models" and the latter is the same as "multiple linear models."

19.2.3 Heteroscedastic models

Various models have been created that allow for heteroscedasticity, i.e. the errors for different response variables may have different variances. For example, weighted least squares is a method for estimating linear regression models when the response variables may have different error variances, possibly with correlated errors. (See also Weighted linear least squares, and generalized least squares.) Heteroscedasticity-consistent standard errors is an improved method for use with uncorrelated but potentially heteroscedastic errors.

19.2.4 Generalized linear models

Generalized linear models (GLMs) are a framework for modeling a response variable y that is bounded or discrete. This is used, for example:

- when modeling positive quantities (e.g. prices or populations) that vary over a large scale—which are better described using a skewed distribution such as the log-normal distribution or Poisson distribution (although GLMs are not used for log-normal data, instead the response variable is simply transformed using the logarithm function);
- when modeling categorical data, such as the choice of a given candidate in an election (which is better described using a Bernoulli distribution/binomial distribution for binary choices, or a categorical distribution/multinomial distribution for multi-way choices), where there are a fixed number of choices that cannot be meaningfully ordered;
- when modeling ordinal data, e.g. ratings on a scale from 0 to 5, where the different outcomes can be ordered but where the quantity itself may not have any absolute meaning (e.g. a rating of 4 may not be "twice as good" in any objective sense as a rating of 2, but simply indicates that it is better than 2 or 3 but not as good as 5).

Generalized linear models allow for an arbitrary *link function* g that relates the mean of the response variable to the predictors, i.e. $E(y) = g(\beta' x)$. The link function is often related to the distribution of the response, and in particular it typically has the effect of transforming between the $(-\infty, \infty)$ range of the linear predictor and the range of the response variable.

Some common examples of GLMs are:

- Poisson regression for count data.
- Logistic regression and probit regression for binary data.
- Multinomial logistic regression and multinomial probit regression for categorical data.
- Ordered probit regression for ordinal data.

Single index models allow some degree of nonlinearity in the relationship between x and y, while preserving the central role of the linear predictor $\beta' x$ as in the classical linear regression model. Under certain conditions, simply applying OLS to data from a single-index model will consistently estimate β up to a proportionality constant.[11]

19.2.5 Hierarchical linear models

Hierarchical linear models (or *multilevel regression*) organizes the data into a hierarchy of regressions, for example where A is regressed on B, and B is regressed on C. It is often used where the data have a natural hierarchical structure such as in educational statistics, where students are nested in classrooms, classrooms are nested in schools, and schools are nested in some administrative grouping, such as a school district. The response variable might be a measure of student achievement such as a test score, and different covariates would be collected at the classroom, school, and school district levels.

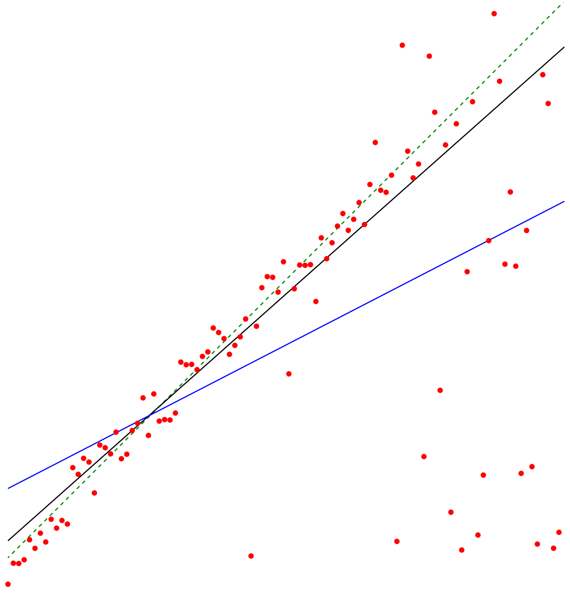
19.2.6 Errors-in-variables

Errors-in-variables models (or "measurement error models") extend the traditional linear regression model to allow the predictor variables X to be observed with error. This error causes standard estimators of β to become biased. Generally, the form of bias is an attenuation, meaning that the effects are biased toward zero.

19.2.7 Others

- In Dempster–Shafer theory, or a linear belief function in particular, a linear regression model may be represented as a partially swept matrix, which can be combined with similar matrices representing observations and other assumed normal distributions and state equations. The combination of swept or unswept matrices provides an alternative method for estimating linear regression models.

19.3 Estimation methods



Comparison of the Theil–Sen estimator (black) and simple linear regression (blue) for a set of points with outliers.

A large number of procedures have been developed for parameter estimation and inference in linear regression. These methods differ in computational simplicity of algorithms, presence of a closed-form solution, robustness with respect to heavy-tailed distributions, and theoretical assumptions needed to validate desirable statistical properties such as consistency and asymptotic efficiency.

Some of the more common estimation techniques for linear regression are summarized below.

19.3.1 Least-squares estimation and related techniques

- **Ordinary least squares** (OLS) is the simplest and thus most common estimator. It is conceptually simple and computationally straightforward. OLS estimates are commonly used to analyze both experimental and observational data.

  The OLS method minimizes the sum of squared residuals, and leads to a closed-form expression for the estimated value of the unknown parameter β:

  $$\hat{\boldsymbol\beta} = (\mathbf{X}^{\mathrm T}\mathbf{X})^{-1}\mathbf{X}^{\mathrm T}\mathbf{y} = \big(\textstyle\sum \mathbf{x}_i \mathbf{x}_i^{\mathrm T}\big)^{-1}\big(\textstyle\sum \mathbf{x}_i y_i\big).$$

  The estimator is unbiased and consistent if the errors have finite variance and are uncorrelated with the regressors[12]

  $$\mathrm{E}[\,\mathbf{x}_i\varepsilon_i\,] = 0.$$

It is also efficient under the assumption that the errors have finite variance and are homoscedastic, meaning that E[εi^2|**x**i] does not depend on i. The condition that the errors are uncorrelated with the regressors will generally be satisfied in an experiment, but in the case of observational data, it is difficult to exclude the possibility of an omitted covariate z that is related to both the observed covariates and the response variable. The existence of such a covariate will generally lead to a correlation between the regressors and the response variable, and hence to an inconsistent estimator of **β**. The condition of homoscedasticity can fail with either experimental or observational data. If the goal is either inference or predictive modeling, the performance of OLS estimates can be poor if multicollinearity is present, unless the sample size is large.

In simple linear regression, where there is only one regressor (with a constant), the OLS coefficient estimates have a simple form that is closely related to the correlation coefficient between the covariate and the response.

- **Generalized least squares** (GLS) is an extension of the OLS method, that allows efficient estimation of β when either heteroscedasticity, or correlations, or both are present among the error terms of the model, as long as the form of heteroscedasticity and correlation is known independently of the data. To handle heteroscedasticity when the error terms are uncorrelated with each other, GLS minimizes a weighted analogue to the sum of squared residuals from OLS regression, where the weight for the ith case is inversely proportional to var(εi). This special case of GLS is called "weighted least squares". The GLS solution to estimation problem is

$$\hat{\beta} = (\mathbf{X}^{\mathrm{T}}\mathbf{\Omega}^{-1}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{\Omega}^{-1}\mathbf{y},$$

  where **Ω** is the covariance matrix of the errors. GLS can be viewed as applying a linear transformation to the data so that the assumptions of OLS are met for the transformed data. For GLS to be applied, the covariance structure of the errors must be known up to a multiplicative constant.

- **Percentage least squares** focuses on reducing percentage errors, which is useful in the field of forecasting or time series analysis. It is also useful in situations where the dependent variable has a wide range without constant variance, as here the larger residuals at the upper end of the range would dominate if OLS were used. When the percentage or relative error is normally distributed, least squares percentage regression provides maximum likelihood estimates. Percentage regression is linked to a multiplicative error model, whereas OLS is linked to models containing an additive error term.[13]

- **Iteratively reweighted least squares** (IRLS) is used when heteroscedasticity, or correlations, or both are present among the error terms of the model, but where little is known about the covariance structure of the errors independently of the data.[14] In the first iteration, OLS, or GLS with a provisional covariance structure is carried out, and the residuals are obtained from the fit. Based on the residuals, an improved estimate of the covariance structure of the errors can usually be obtained. A subsequent GLS iteration is then performed using this estimate of the error structure to define the weights. The process can be iterated to convergence, but in many cases, only one iteration is sufficient to achieve an efficient estimate of β.[15][16]

- **Instrumental variables** regression (IV) can be performed when the regressors are correlated with the errors. In this case, we need the existence of some auxiliary *instrumental variables* **z**i such that E[**z**$i\varepsilon i$] = 0. If **Z** is the matrix of instruments, then the estimator can be given in closed form as

$$\hat{\beta} = (\mathbf{X}^{\mathrm{T}}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{y}.$$

- **Optimal instruments** regression is an extension of classical IV regression to the situation where E[εi|**z**i] = 0.

- **Total least squares** (TLS)[17] is an approach to least squares estimation of the linear regression model that treats the covariates and response variable in a more geometrically symmetric manner than OLS. It is one approach to handling the "errors in variables" problem, and is also sometimes used even when the covariates are assumed to be error-free.

19.3.2 Maximum-likelihood estimation and related techniques

- **Maximum likelihood estimation** can be performed when the distribution of the error terms is known to belong to a certain parametric family $f\theta$ of probability distributions.[18] When $f\theta$ is a normal distribution with zero mean and variance θ, the resulting estimate is identical to the OLS estimate. GLS estimates are maximum likelihood estimates when ε follows a multivariate normal distribution with a known covariance matrix.

- **Ridge regression**,[19][20][21] and other forms of penalized estimation such as **Lasso regression**,[5] deliberately introduce bias into the estimation of β in order to reduce the variability of the estimate. The resulting estimators generally have lower mean squared error than the OLS estimates, particularly when multicollinearity is present. They are generally used when the goal is to predict the value of the response variable y for values of the predictors x that have not yet been observed. These methods are not as commonly used when the goal is inference, since it is difficult to account for the bias.

- **Least absolute deviation** (LAD) regression is a robust estimation technique in that it is less sensitive to the presence of outliers than OLS (but is less efficient than OLS when no outliers are present). It is equivalent to maximum likelihood estimation under a Laplace distribution model for ε.[22]
- **Adaptive estimation**. If we assume that error terms are independent from the regressors $\varepsilon_i \perp \mathbf{x}_i$, the optimal estimator is the 2-step MLE, where the first step is used to non-parametrically estimate the distribution of the error term.[23]

19.3.3 Other estimation techniques

- **Bayesian linear regression** applies the framework of Bayesian statistics to linear regression. (See also Bayesian multivariate linear regression.) In particular, the regression coefficients β are assumed to be random variables with a specified prior distribution. The prior distribution can bias the solutions for the regression coefficients, in a way similar to (but more general than) ridge regression or lasso regression. In addition, the Bayesian estimation process produces not a single point estimate for the "best" values of the regression coefficients but an entire posterior distribution, completely describing the uncertainty surrounding the quantity. This can be used to estimate the "best" coefficients using the mean, mode, median, any quantile (see quantile regression), or any other function of the posterior distribution.
- **Quantile regression** focuses on the conditional quantiles of y given X rather than the conditional mean of y given X. Linear quantile regression models a particular conditional quantile, for example the conditional median, as a linear function $\beta^T x$ of the predictors.
- **Mixed models** are widely used to analyze linear regression relationships involving dependent data when the dependencies have a known structure. Common applications of mixed models include analysis of data involving repeated measurements, such as longitudinal data, or data obtained from cluster sampling. They are generally fit as parametric models, using maximum likelihood or Bayesian estimation. In the case where the errors are modeled as normal random variables, there is a close connection between mixed models and generalized least squares.[24] Fixed effects estimation is an alternative approach to analyzing this type of data.
- **Principal component regression** (PCR)[7][8] is used when the number of predictor variables is large, or when strong correlations exist among the predictor variables. This two-stage procedure first reduces the predictor variables using principal component analysis then uses the reduced variables in an OLS regression fit. While it often works well in practice, there is no general theoretical reason that the most informative linear function of the predictor variables should lie among the dominant principal components of the multivariate distribution of the predictor variables. The partial least squares regression is the extension of the PCR method which does not suffer from the mentioned deficiency.
- **Least-angle regression**[6] is an estimation procedure for linear regression models that was developed to handle high-dimensional covariate vectors, potentially with more covariates than observations.
- The **Theil–Sen estimator** is a simple robust estimation technique that chooses the slope of the fit line to be the median of the slopes of the lines through pairs of sample points. It has similar statistical efficiency properties to simple linear regression but is much less sensitive to outliers.[25]
- Other robust estimation techniques, including the **α-trimmed mean** approach, and **L-, M-, S-, and R-estimators** have been introduced.

19.3.4 Further discussion

In statistics and numerical analysis, the problem of **numerical methods for linear least squares** is an important one because linear regression models are one of the most important types of model, both as formal statistical models and for exploration of data sets. The majority of statistical computer packages contain facilities for regression analysis that make use of linear least squares computations. Hence it is appropriate that considerable effort has been devoted to the task of ensuring that these computations are undertaken efficiently and with due regard to numerical precision.

Individual statistical analyses are seldom undertaken in isolation, but rather are part of a sequence of investigatory steps. Some of the topics involved in considering numerical methods for linear least squares relate to this point. Thus important topics can be

- Computations where a number of similar, and often nested, models are considered for the same data set. That is, where models with the same dependent variable but different sets of independent variables are to be considered, for essentially the same set of data points.
- Computations for analyses that occur in a sequence, as the number of data points increases.
- Special considerations for very extensive data sets.

Fitting of linear models by least squares often, but not always, arises in the context of statistical analysis. It can therefore be important that considerations of computational efficiency for such problems extend to all of the auxiliary quantities required for such analyses, and are not restricted to the formal solution of the linear least squares problem.

Matrix calculations, like any others, are affected by rounding errors. An early summary of these effects, regarding the choice of computational methods for matrix inversion, was provided by Wilkinson.[26]

19.3.5 Using Linear Algebra

It follows that one can find a "best" approximation of another function by minimizing the area between two functions, a continuous function f on $[a,b]$ and a function $g \in W$ where W is a subspace of $C[a,b]$:

$$\text{Area} = \int_a^b |f(x) - g(x)|\, dx$$

within the subspace W . Due to the frequent difficulty of evaluating integrands involving absolute value, one can instead define

$$\int_a^b [f(x) - g(x)]^2\, dx$$

an adequate criterion for obtaining the least squares approximation, function g , of f with respect to the inner product space W .

As such, $\|f-g\|^2$ or, equivalently, $\|f-g\|$, can thus be written in vector form:

$$\int_a^b [f(x) - g(x)]^2\, dx = \langle f-g, f-g\rangle = \|f-g\|^2$$

other words, the least squares approximation of f is the function $g \in$ subspace W closest to f in terms of the inner product $\langle f, g\rangle$. Furthermore, this can be applied with a theorem:

> Let f be continuous on $[a,b]$, and let W be a finite-dimensional subspace of $C[a,b]$. The least squares approximating function of f with respect to W is given by
>
> $$g = \langle f, \vec{w}_1\rangle \vec{w}_1 + \langle f, \vec{w}_2\rangle \vec{w}_2 + \cdots + \langle f, \vec{w}_n\rangle \vec{w}_n$$
>
> where $B = \{\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_n\}$ is an orthonormal basis for W .

19.4 Applications of linear regression

Linear regression is widely used in biological, behavioral and social sciences to describe possible relationships between variables. It ranks as one of the most important tools used in these disciplines.

19.4.1 Trend line

Main article: Trend estimation

A **trend line** represents a trend, the long-term movement in time series data after other components have been accounted for. It tells whether a particular data set (say GDP, oil prices or stock prices) have increased or decreased over the period of time. A trend line could simply be drawn by eye through a set of data points, but more properly their position and slope is calculated using statistical techniques like linear regression. Trend lines typically are straight lines, although some variations use higher degree polynomials depending on the degree of curvature desired in the line.

Trend lines are sometimes used in business analytics to show changes in data over time. This has the advantage of being simple. Trend lines are often used to argue that a particular action or event (such as training, or an advertising campaign) caused observed changes at a point in time. This is a simple technique, and does not require a control group, experimental design, or a sophisticated analysis technique. However, it suffers from a lack of scientific validity in cases where other potential changes can affect the data.

19.4.2 Epidemiology

Early evidence relating tobacco smoking to mortality and morbidity came from observational studies employing regression analysis. In order to reduce spurious correlations when analyzing observational data, researchers usually include several variables in their regression models in addition to the variable of primary interest. For example, suppose we have a regression model in which cigarette smoking is the independent variable of interest, and the dependent variable is lifespan measured in years. Researchers might include socio-economic status as an additional independent variable, to ensure that any observed effect of smoking on lifespan is not due to some effect of education or income. However, it is never possible to include all possible confounding variables in an empirical analysis. For example, a hypothetical gene might increase mortality and also cause people to smoke more. For this reason, randomized controlled trials are often able to generate more compelling evidence of causal relationships than can be obtained using regression analyses of observational data. When controlled experiments are not feasible, variants of regression analysis such as instrumental variables regression may be used to attempt to estimate causal relationships from observational data.

19.4.3 Finance

The capital asset pricing model uses linear regression as well as the concept of beta for analyzing and quantifying the systematic risk of an investment. This comes directly from the beta coefficient of the linear regression model that relates the return on the investment to the return on all risky assets.

19.4.4 Economics

Main article: Econometrics

Linear regression is the predominant empirical tool in economics. For example, it is used to predict consumption spending,[27] fixed investment spending, inventory investment, purchases of a country's exports,[28] spending on imports,[28] the demand to hold liquid assets,[29] labor demand,[30] and labor supply.[30]

19.4.5 Environmental science

Linear regression finds application in a wide range of environmental science applications. In Canada, the Environmental Effects Monitoring Program uses statistical analyses on fish and benthic surveys to measure the effects of pulp mill or metal mine effluent on the aquatic ecosystem.[31]

19.5 See also

- Analysis of variance
- Censored regression model
- Cross-sectional regression
- Curve fitting
- Empirical Bayes methods
- Errors and residuals
- Lack-of-fit sum of squares
- Linear classifier
- Logistic regression
- M-estimator
- MLPACK contains a C++ implementation of linear regression
- Multivariate adaptive regression splines
- Nonlinear regression
- Nonparametric regression
- Normal equations
- Projection pursuit regression
- Segmented linear regression
- Stepwise regression
- Support vector machine
- Truncated regression model

19.6 Notes

[1] David A. Freedman (2009). *Statistical Models: Theory and Practice*. Cambridge University Press. p. 26. A simple regression equation has on the right hand side an intercept and an explanatory variable with a slope coefficient. A multiple regression equation has several explanatory variables on the right hand side, each with its own slope coefficient

[2] Rencher, Alvin C.; Christensen, William F. (2012), "Chapter 10, Multivariate regression – Section 10.1, Introduction", *Methods of Multivariate Analysis*, Wiley Series in Probability and Statistics **709** (3rd ed.), John Wiley & Sons, p. 19, ISBN 9781118391679.

[3] Hilary L. Seal (1967). "The historical development of the Gauss linear model". *Biometrika* **54** (1/2): 1–24. doi:10.1093/biomet/54.1-2.1.

[4] Yan, Xin (2009), *Linear Regression Analysis: Theory and Computing*, World Scientific, pp. 1–2, ISBN 9789812834119, Regression analysis ... is probably one of the oldest topics in mathematical statistics dating back to about two hundred years ago. The earliest form of the linear regression was the least squares method, which was published by Legendre in 1805, and by Gauss in 1809 ... Legendre and Gauss both applied the method to the problem of determining, from astronomical observations, the orbits of bodies about the sun.

[5] Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". *Journal of the Royal Statistical Society, Series B* **58** (1): 267–288. JSTOR 2346178.

[6] Efron, Bradley; Hastie, Trevor; Johnstone,Iain; Tibshirani,Robert (2004). "Least Angle Regression". *The Annals of Statistics* **32** (2): 407–451. doi:10.1214/009053604000000067. JSTOR 3448465.

[7] Hawkins, Douglas M. (1973). "On the Investigation of Alternative Regressions by Principal Component Analysis". *Journal of the Royal Statistical Society, Series C* **22** (3): 275–286. JSTOR 2346776.

[8] Jolliffe, Ian T. (1982). "A Note on the Use of Principal Components in Regression". *Journal of the Royal Statistical Society, Series C* **31** (3): 300–303. JSTOR 2348005.

[9] Berk, Richard A. *Regression Analysis: A Constructive Critique*. Sage. doi:10.1177/0734016807304871.

[10] Warne, R. T. (2011). Beyond multiple regression: Using commonality analysis to better understand R2 results. *Gifted Child Quarterly, 55*, 313-318. doi:10.1177/0016986211422217

[11] Brillinger, David R. (1977). "The Identification of a Particular Nonlinear Time Series System". *Biometrika* **64** (3): 509–515. doi:10.1093/biomet/64.3.509. JSTOR 2345326.

[12] Lai, T.L.; Robbins, H.; Wei, C.Z. (1978). "Strong consistency of least squares estimates in multiple regression". *PNAS* **75** (7): 3034–3036. Bibcode:1978PNAS...75.3034L. doi:10.1073/pnas.75.7.3034. JSTOR 68164.

[13] Tofallis, C (2009). "Least Squares Percentage Regression". *Journal of Modern Applied Statistical Methods* **7**: 526–534. doi:10.2139/ssrn.1406472.

[14] del Pino, Guido (1989). "The Unifying Role of Iterative Generalized Least Squares in Statistical Algorithms". *Statistical Science* **4** (4): 394–403. doi:10.1214/ss/1177012408. JSTOR 2245853.

[15] Carroll, Raymond J. (1982). "Adapting for Heteroscedasticity in Linear Models". *The Annals of Statistics* **10** (4): 1224–1233. doi:10.1214/aos/1176345987. JSTOR 2240725.

[16] Cohen, Michael; Dalal, Siddhartha R.; Tukey,John W. (1993). "Robust, Smoothly Heterogeneous Variance Regression". *Journal of the Royal Statistical Society, Series C* **42** (2): 339–353. JSTOR 2986237.

[17] Nievergelt, Yves (1994). "Total Least Squares: State-of-the-Art Regression in Numerical Analysis". *SIAM Review* **36** (2): 258–264. doi:10.1137/1036055. JSTOR 2132463.

[18] Lange, Kenneth L.; Little, Roderick J. A.; Taylor,Jeremy M. G. (1989). "Robust Statistical Modeling Using the t Distribution". *Journal of the American Statistical Association* **84** (408): 881–896. doi:10.2307/2290063. JSTOR 2290063.

[19] Swindel, Benee F. (1981). "Geometry of Ridge Regression Illustrated". *The American Statistician* **35** (1): 12–15. doi:10.2307/2683577. JSTOR 2683577.

[20] Draper, Norman R.; van Nostrand; R. Craig (1979). "Ridge Regression and James-Stein Estimation: Review and Comments". *Technometrics* **21** (4): 451–466. doi:10.2307/1268284. JSTOR 1268284.

[21] Hoerl, Arthur E.; Kennard,Robert W.; Hoerl,Roger W. (1985). "Practical Use of Ridge Regression: A Challenge Met". *Journal of the Royal Statistical Society, Series C* **34** (2): 114–120. JSTOR 2347363.

[22] Narula, Subhash C.; Wellington, John F. (1982). "The Minimum Sum of Absolute Errors Regression: A State of the Art Survey". *International Statistical Review* **50** (3): 317–326. doi:10.2307/1402501. JSTOR 1402501.

[23] Stone, C. J. (1975). "Adaptive maximum likelihood estimators of a location parameter". *The Annals of Statistics* **3** (2): 267–284. doi:10.1214/aos/1176343056. JSTOR 2958945.

[24] Goldstein, H. (1986). "Multilevel Mixed Linear Model Analysis Using Iterative Generalized Least Squares". *Biometrika* **73** (1): 43–56. doi:10.1093/biomet/73.1.43. JSTOR 2336270.

[25] Theil, H. (1950). "A rank-invariant method of linear and polynomial regression analysis. I, II, III". *Nederl. Akad. Wetensch., Proc.* **53**: 386–392, 521–525, 1397–1412. MR 0036489; Sen, Pranab Kumar (1968). "Estimates of the regression coefficient based on Kendall's tau". *Journal of the American Statistical Association* **63** (324): 1379–1389. doi:10.2307/2285891. JSTOR 2285891. MR 0258201.

[26] Wilkinson, J.H. (1963) "Chapter 3: Matrix Computations", *Rounding Errors in Algebraic Processes*, London: Her Majesty's Stationery Office (National Physical Laboratory, Notes in Applied Science, No.32)

[27] Deaton, Angus (1992). *Understanding Consumption*. Oxford University Press. ISBN 0-19-828824-7.

[28] Krugman, Paul R.; Obstfeld, M.; Melitz, Marc J. (2012). *International Economics: Theory and Policy* (9th global ed.). Harlow: Pearson. ISBN 9780273754091.

[29] Laidler, David E. W. (1993). "The Demand for Money: Theories, Evidence, and Problems" (4th ed.). New York: Harper Collins. ISBN 0065010981.

[30] Ehrenberg; Smith (2008). *Modern Labor Economics* (10th international ed.). London: Addison-Wesley. ISBN 9780321538963.

[31] EEMP webpage

19.7 References

- Cohen, J., Cohen P., West, S.G., & Aiken, L.S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences.* (2nd ed.) Hillsdale, NJ: Lawrence Erlbaum Associates

- Charles Darwin. *The Variation of Animals and Plants under Domestication.* (1868) *(Chapter XIII describes what was known about reversion in Galton's time. Darwin uses the term "reversion".)*

- Draper, N.R.; Smith, H. (1998). *Applied Regression Analysis* (3rd ed.). John Wiley. ISBN 0-471-17082-8.

- Francis Galton. "Regression Towards Mediocrity in Hereditary Stature," *Journal of the Anthropological Institute*, 15:246-263 (1886). *(Facsimile at:)*

- Robert S. Pindyck and Daniel L. Rubinfeld (1998, 4h ed.). *Econometric Models and Economic Forecasts*, ch. 1 (Intro, incl. appendices on Σ operators & derivation of parameter est.) & Appendix 4.3 (mult. regression in matrix form).

19.8 Further reading

- Barlow, Jesse L. (1993). “Chapter 9: Numerical aspects of Solving Linear Least Squares Problems”. In Rao, C.R. *Computational Statistics*. Handbook of Statistics **9**. North-Holland. ISBN 0-444-88096-8
- Björck, Åke (1996). *Numerical methods for least squares problems*. Philadelphia: SIAM. ISBN 0-89871-360-9.
- Goodall, Colin R. (1993). “Chapter 13: Computation using the QR decomposition”. In Rao, C.R. *Computational Statistics*. Handbook of Statistics **9**. North-Holland. ISBN 0-444-88096-8
- Pedhazur, Elazar J (1982). “Multiple regression in behavioral research: Explanation and prediction” (2nd ed.). New York: Holt, Rinehart and Winston. ISBN 0-03-041760-0.
- National Physical Laboratory (1961). “Chapter 1: Linear Equations and Matrices: Direct Methods”. *Modern Computing Methods*. Notes on Applied Science **16** (2nd ed.). Her Majesty’s Stationery Office
- National Physical Laboratory (1961). “Chapter 2: Linear Equations and Matrices: Direct Methods on Automatic Computers”. *Modern Computing Methods*. Notes on Applied Science **16** (2nd ed.). Her Majesty’s Stationery Office

19.9 External links

- Online Linear Regression Calculator & Trend Line Graphing Tool
- Using gradient descent in C++, Boost, Ublas for linear regression
- Lecture notes on linear regression analysis (Robert Nau, Duke University)

Chapter 20

Tikhonov regularization

Tikhonov regularization, named for Andrey Tikhonov, is the most commonly used method of regularization of ill-posed problems. In statistics, the method is known as **ridge regression**, and with multiple independent discoveries, it is also variously known as the **Tikhonov–Miller method**, the **Phillips–Twomey method**, the **constrained linear inversion** method, and the method of **linear regularization**. It is related to the Levenberg–Marquardt algorithm for non-linear least-squares problems.

When the following problem is not well posed (either because of non-existence or non-uniqueness of x)

$$A\mathbf{x} = \mathbf{b},$$

then the standard approach (known as ordinary least squares) leads to an overdetermined, or more often an underdetermined system of equations. Most real-world phenomena operate as low-pass filters in the forward direction where A maps $\mathbf{x}$ to $\mathbf{b}$. Therefore in solving the inverse-problem, the inverse-mapping operates as a high-pass filter that has the undesirable tendency of amplifying noise (eigenvalues / singular values are largest in the reverse mapping where they were smallest in the forward mapping). In addition, ordinary least squares implicitly nullifies every element of the reconstructed version of $\mathbf{x}$ that is in the null-space of A , rather than allowing for a model to be used as a prior for $\mathbf{x}$. Ordinary least squares seeks to minimize the sum of squared residuals, which can be compactly written as

$$\|A\mathbf{x} - \mathbf{b}\|^2$$

where $\|\cdot\|$ is the Euclidean norm. In order to give preference to a particular solution with desirable properties, a regularization term can be included in this minimization:

$$\|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2$$

for some suitably chosen **Tikhonov matrix**, Γ . In many cases, this matrix is chosen as a multiple of the identity matrix ($\Gamma = \alpha I$), giving preference to solutions with smaller norms; this is known as L_2 **regularization**.[1] In other cases, lowpass operators (e.g., a difference operator or a weighted Fourier operator) may be used to enforce smoothness if the underlying vector is believed to be mostly continuous. This regularization improves the conditioning of the problem, thus enabling a direct numerical solution. An explicit solution, denoted by $\hat{x}$, is given by:

$$\hat{x} = (A^T A + \Gamma^T \Gamma)^{-1} A^T \mathbf{b}$$

The effect of regularization may be varied via the scale of matrix Γ . For $\Gamma = 0$ this reduces to the unregularized least squares solution provided that $(\mathrm{A}^{\mathrm{T}}\mathrm{A})^{-1}$ exists.

L_2 regularization is used in many contexts aside from linear regression, such as classification with logistic regression or support vector machines,[2] and matrix factorization.[3]

20.1 History

Tikhonov regularization has been invented independently in many different contexts. It became widely known from its application to integral equations from the work of Andrey Tikhonov and David L. Phillips. Some authors use the term **Tikhonov–Phillips regularization**. The finite-dimensional case was expounded by Arthur E. Hoerl, who took a statistical approach, and by Manus Foster, who interpreted this method as a Wiener–Kolmogorov filter. Following Hoerl, it is known in the statistical literature as **ridge regression**.

20.2 Generalized Tikhonov regularization

For general multivariate normal distributions for x and the data error, one can apply a transformation of the variables to reduce to the case above. Equivalently, one can seek an x to minimize

$$\|Ax-b\|_P^2+\|x-x_0\|_Q^2$$

where we have used $\|x\|_Q^2$ to stand for the weighted norm x^TQx (compare with the Mahalanobis distance). In the Bayesian interpretation P is the inverse covariance matrix of b , x_0 is the expected value of x , and Q is the inverse covariance matrix of x . The Tikhonov matrix is then given as a factorization of the matrix $Q=\Gamma^T\Gamma$ (e.g. the Cholesky factorization), and is considered a whitening filter.

This generalized problem has an optimal solution x^* which can be solved explicitly using the formula

$$x^*=(A^TPA+Q)^{-1}(A^TPb+Qx_0).$$

or equivalently

$$x^*=x_0+(A^TPA+Q)^{-1}(A^TP(b-Ax_0)).$$

20.3 Regularization in Hilbert space

Typically discrete linear ill-conditioned problems result from discretization of integral equations, and one can formulate a Tikhonov regularization in the original infinite-dimensional context. In the above we can interpret A as a compact operator on Hilbert spaces, and x and b as elements in the domain and range of A . The operator $A^*A+\Gamma^T\Gamma$ is then a self-adjoint bounded invertible operator.

20.4 Relation to singular value decomposition and Wiener filter

With $\Gamma=\alpha I$, this least squares solution can be analyzed in a special way via the singular value decomposition. Given the singular value decomposition of A

$$A=U\Sigma V^T$$

with singular values σ_i , the Tikhonov regularized solution can be expressed as

$$\hat{x}=VDU^Tb$$

where D has diagonal values

$$D_{ii}=\frac{\sigma_i}{\sigma_i^2+\alpha^2}$$

and is zero elsewhere. This demonstrates the effect of the Tikhonov parameter on the condition number of the regularized problem. For the generalized case a similar representation can be derived using a generalized singular value decomposition.

Finally, it is related to the Wiener filter:

$$\hat{x}=\sum_{i=1}^q f_i\frac{u_i^Tb}{\sigma_i}v_i$$

where the Wiener weights are $f_i=\frac{\sigma_i^2}{\sigma_i^2+\alpha^2}$ and q is the rank of A .

20.5 Determination of the Tikhonov factor

The optimal regularization parameter α is usually unknown and often in practical problems is determined by an *ad hoc* method. A possible approach relies on the Bayesian interpretation described below. Other approaches include the discrepancy principle, cross-validation, L-curve method, restricted maximum likelihood and unbiased predictive risk estimator. Grace Wahba proved that the optimal parameter, in the sense of leave-one-out cross-validation minimizes:

$$G=\frac{\mathrm{RSS}}{\tau^2}=\frac{\left\|X\hat{\beta}-y\right\|^2}{[\mathrm{Tr}\,(I-X(X^TX+\alpha^2I)^{-1}X^T)]^2}$$

where RSS is the residual sum of squares and τ is the effective number of degrees of freedom.

Using the previous SVD decomposition, we can simplify the above expression:

$$\mathrm{RSS}=\left\|y-\sum_{i=1}^q(u_i'b)u_i\right\|^2+\left\|\sum_{i=1}^q\frac{\alpha^2}{\sigma_i^2+\alpha^2}(u_i'b)u_i\right\|^2$$

$$\mathrm{RSS}=\mathrm{RSS}_0+\left\|\sum_{i=1}^q\frac{\alpha^2}{\sigma_i^2+\alpha^2}(u_i'b)u_i\right\|^2$$

and

$$\tau=m-\sum_{i=1}^q\frac{\sigma_i^2}{\sigma_i^2+\alpha^2}=m-q+\sum_{i=1}^q\frac{\alpha^2}{\sigma_i^2+\alpha^2}$$

20.6 Relation to probabilistic formulation

The probabilistic formulation of an inverse problem introduces (when all uncertainties are Gaussian) a covariance matrix C_M representing the *a priori* uncertainties on the model parameters, and a covariance matrix C_D representing the uncertainties on the observed parameters (see, for instance, Tarantola, 2005). In the special case when these two matrices are diagonal and isotropic, $C_M = \sigma_M^2 I$ and $C_D = \sigma_D^2 I$, and, in this case, the equations of inverse theory reduce to the equations above, with $\alpha = \sigma_D / \sigma_M$.

20.7 Bayesian interpretation

Further information: Minimum mean square error § Linear MMSE estimator for linear observation process

Although at first the choice of the solution to this regularized problem may look artificial, and indeed the matrix Γ seems rather arbitrary, the process can be justified from a Bayesian point of view. Note that for an ill-posed problem one must necessarily introduce some additional assumptions in order to get a unique solution. Statistically, the prior probability distribution of x is sometimes taken to be a multivariate normal distribution. For simplicity here, the following assumptions are made: the means are zero; their components are independent; the components have the same standard deviation σ_x . The data are also subject to errors, and the errors in b are also assumed to be independent with zero mean and standard deviation σ_b . Under these assumptions the Tikhonov-regularized solution is the most probable solution given the data and the *a priori* distribution of x , according to Bayes' theorem.[4]

If the assumption of normality is replaced by assumptions of homoskedasticity and uncorrelatedness of errors, and if one still assumes zero mean, then the Gauss–Markov theorem entails that the solution is the minimal unbiased estimator.

20.8 See also

- LASSO estimator is another regularization method in statistics.

20.9 References

[1] Ng, Andrew Y. (2004). Proc. ICML (PDF) http://www.machinelearning.org/proceedings/icml2004/papers/354.pdf. Missing or empty |title= (help)

[2] R.-E. Fan; K.-W. Chang; C.-J. Hsieh; X.-R. Wang; C.-J. Lin (2008). "LIBLINEAR: A library for large linear classification". *Journal of Machine Learning Research* **9**: 1871–1874.

[3] Guan, Naiyang; Tao, Dacheng; Luo, Zhigang; Yuan, Bo (2012). "Online nonnegative matrix factorization with robust stochastic approximation". *IEEE Trans. Neural Networks and Learning Systems* **23** (7): 1087–1099.

[4] Vogel, Curtis R. (2002). *Computational methods for inverse problems*. Philadelphia: Society for Industrial and Applied Mathematics. ISBN 0-89871-550-4.

- Amemiya, Takeshi (1985). *Advanced Econometrics*. Harvard University Press. pp. 60–61. ISBN 0-674-00560-0.
- Tikhonov, Andrey Nikolayevich (1943). "Об устойчивости обратных задач" [On the stability of inverse problems]. *Doklady Akademii Nauk SSSR* **39** (5): 195–198.
- Tikhonov, A. N. (1963). "О решении некорректно поставленных задач и методе регуляризации" [Solution of incorrectly formulated problems and the regularization method]. *Doklady Akademii Nauk SSSR* **151**: 501–504.. Translated in *Soviet Mathematics* **4**: 1035–1038. Missing or empty |title= (help)
- Tikhonov, A. N.; V. Y. Arsenin (1977). *Solution of Ill-posed Problems*. Washington: Winston & Sons. ISBN 0-470-99124-0.
- Tikhonov A.N., Goncharsky A.V., Stepanov V.V., Yagola A.G., 1995, *Numerical Methods for the Solution of Ill-Posed Problems*, Kluwer Academic Publishers.
- Tikhonov A.N., Leonov A.S., Yagola A.G., 1998, *Nonlinear Ill-Posed Problems*, V. 1, V. 2, Chapman and Hall.
- Hansen, P.C., 1998, *Rank-deficient and Discrete ill-posed problems*, SIAM
- Hoerl AE, 1962, *Application of ridge analysis to regression problems*, Chemical Engineering Progress, 1958, 54–59.
- Hoerl, A.E.; R.W. Kennard (1970). "Ridge regression: Biased estimation for nonorthogonal problems". *Technometrics* **12** (1): 55–67. doi:10.2307/1267351. JSTOR 1271436.
- Foster, M. (1961). "An Application of the Wiener-Kolmogorov Smoothing Theory to Matrix Inversion". *Journal of the Society for Industrial and Applied Mathematics* **9** (3): 387. doi:10.1137/0109031.
- Phillips, D. L. (1962). "A Technique for the Numerical Solution of Certain Integral Equations of the First Kind". *Journal of the ACM* **9**: 84. doi:10.1145/321105.321114.

- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 19.5. Linear Regularization Methods". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.
- Tarantola A, 2005, *Inverse Problem Theory* (free PDF version), Society for Industrial and Applied Mathematics, ISBN 0-89871-572-5
- Wahba, G. (1990). "Spline Models for Observational Data". Society for Industrial and Applied Mathematics.
- Golub, G.; Heath, M.; Wahba, G. (1979). "Generalized cross-validation as a method for choosing a good ridge parameter" (PDF). *Technometrics* **21**: 215–223. doi:10.1080/00401706.1979.10489751.

Chapter 21

Regression analysis

In statistics, **regression analysis** is a statistical process for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed. Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables – that is, the average value of the dependent variable when the independent variables are fixed. Less commonly, the focus is on a quantile, or other location parameter of the conditional distribution of the dependent variable given the independent variables. In all cases, the estimation target is a function of the independent variables called the **regression function**. In regression analysis, it is also of interest to characterize the variation of the dependent variable around the regression function which can be described by a probability distribution.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable;[1] for example, correlation does not imply causation.

Many techniques for carrying out regression analysis have been developed. Familiar methods such as linear regression and ordinary least squares regression are parametric, in that the regression function is defined in terms of a finite number of unknown parameters that are estimated from the data. Nonparametric regression refers to techniques that allow the regression function to lie in a specified set of functions, which may be infinite-dimensional.

The performance of regression analysis methods in practice depends on the form of the data generating process, and how it relates to the regression approach being used. Since the true form of the data-generating process is generally not known, regression analysis often depends to some extent on making assumptions about this process. These assumptions are sometimes testable if a sufficient quantity of data is available. Regression models for prediction are often useful even when the assumptions are moderately violated, although they may not perform optimally. However, in many applications, especially with small effects or questions of causality based on observational data, regression methods can give misleading results.[2][3]

21.1 History

The earliest form of regression was the method of least squares, which was published by Legendre in 1805,[4] and by Gauss in 1809.[5] Legendre and Gauss both applied the method to the problem of determining, from astronomical observations, the orbits of bodies about the Sun (mostly comets, but also later the then newly discovered minor planets). Gauss published a further development of the theory of least squares in 1821,[6] including a version of the Gauss–Markov theorem.

The term "regression" was coined by Francis Galton in the nineteenth century to describe a biological phenomenon. The phenomenon was that the heights of descendants of tall ancestors tend to regress down towards a normal average (a phenomenon also known as regression toward the mean).[7][8] For Galton, regression had only this biological meaning,[9][10] but his work was later extended by Udny Yule and Karl Pearson to a more general statistical context.[11][12] In the work of Yule and Pearson, the joint distribution of the response and explanatory variables is assumed to be Gaussian. This assumption was weakened by R.A. Fisher in his works of 1922 and 1925.[13][14][15] Fisher assumed that the conditional distribution of the response variable is Gaussian, but the joint distribution need not be. In this respect, Fisher's assumption is closer to Gauss's formulation of 1821.

In the 1950s and 1960s, economists used electromechanical desk calculators to calculate regressions. Before 1970, it sometimes took up to 24 hours to receive the result from one regression.[16]

Regression methods continue to be an area of active research. In recent decades, new methods have been developed for robust regression, regression involving correlated responses such as time series and growth curves, regression in which the predictor (independent variable) or response variables are curves, images, graphs, or other complex data objects, regression methods accommodating various types of missing data, nonparametric regression, Bayesian methods for regression, regression in which the predictor variables are measured with error, regression with more predictor variables than observations, and causal inference with regression.

21.2 Regression models

Regression models involve the following variables:

- The **unknown parameters**, denoted as $\boldsymbol{\beta}$, which may represent a scalar or a vector.
- The **independent variables**, $\mathbf{X}$.
- The **dependent variable**, Y.

In various fields of application, different terminologies are used in place of dependent and independent variables.

A regression model relates Y to a function of $\mathbf{X}$ and $\boldsymbol{\beta}$.

$$Y \approx f(\mathbf{X}, \boldsymbol{\beta})$$

The approximation is usually formalized as $E(Y \mid \mathbf{X}) = f(\mathbf{X}, \boldsymbol{\beta})$. To carry out regression analysis, the form of the function f must be specified. Sometimes the form of this function is based on knowledge about the relationship between Y and $\mathbf{X}$ that does not rely on the data. If no such knowledge is available, a flexible or convenient form for f is chosen.

Assume now that the vector of unknown parameters $\boldsymbol{\beta}$ is of length k. In order to perform a regression analysis the user must provide information about the dependent variable Y:

- If N data points of the form $(Y, \mathbf{X})$ are observed, where $N < k$, most classical approaches to regression analysis cannot be performed: since the system of equations defining the regression model is underdetermined, there are not enough data to recover $\boldsymbol{\beta}$.
- If exactly $N = k$ data points are observed, and the function f is linear, the equations $Y = f(\mathbf{X}, \boldsymbol{\beta})$ can be solved exactly rather than approximately. This reduces to solving a set of N equations with N unknowns (the elements of $\boldsymbol{\beta}$), which has a unique solution as long as the $\mathbf{X}$ are linearly independent. If f is nonlinear, a solution may not exist, or many solutions may exist.
- The most common situation is where $N > k$ data points are observed. In this case, there is enough information in the data to estimate a unique value for $\boldsymbol{\beta}$ that best fits the data in some sense, and the regression model when applied to the data can be viewed as an overdetermined system in $\boldsymbol{\beta}$.

In the last case, the regression analysis provides the tools for:

1. Finding a solution for unknown parameters $\boldsymbol{\beta}$ that will, for example, minimize the distance between the measured and predicted values of the dependent variable Y (also known as method of least squares).
2. Under certain statistical assumptions, the regression analysis uses the surplus of information to provide statistical information about the unknown parameters $\boldsymbol{\beta}$ and predicted values of the dependent variable Y.

21.2.1 Necessary number of independent measurements

Consider a regression model which has three unknown parameters, β_0, β_1, and β_2. Suppose an experimenter performs 10 measurements all at exactly the same value of independent variable vector $\mathbf{X}$ (which contains the independent variables X_1, X_2, and X_3). In this case, regression analysis fails to give a unique set of estimated values for the three unknown parameters; the experimenter did not provide enough information. The best one can do is to estimate the average value and the standard deviation of the dependent variable Y. Similarly, measuring at two different values of $\mathbf{X}$ would give enough data for a regression with two unknowns, but not for three or more unknowns.

If the experimenter had performed measurements at three different values of the independent variable vector $\mathbf{X}$, then regression analysis would provide a unique set of estimates for the three unknown parameters in $\boldsymbol{\beta}$.

In the case of general linear regression, the above statement is equivalent to the requirement that the matrix $\mathbf{X}^T\mathbf{X}$ is invertible.

21.2.2 Statistical assumptions

When the number of measurements, N, is larger than the number of unknown parameters, k, and the measurement errors ε_i are normally distributed then *the excess of information* contained in $(N - k)$ measurements is used to make statistical predictions about the unknown parameters. This excess of information is referred to as the degrees of freedom of the regression.

21.3 Underlying assumptions

Classical assumptions for regression analysis include:

- The sample is representative of the population for the inference prediction.
- The error is a random variable with a mean of zero conditional on the explanatory variables.
- The independent variables are measured with no error. (Note: If this is not so, modeling may be done instead using errors-in-variables model techniques).
- The independent variables (predictors) are linearly independent, i.e. it is not possible to express any predictor as a linear combination of the others.
- The errors are uncorrelated, that is, the variance–covariance matrix of the errors is diagonal and each non-zero element is the variance of the error.
- The variance of the error is constant across observations (homoscedasticity). If not, weighted least squares or other methods might instead be used.

These are sufficient conditions for the least-squares estimator to possess desirable properties; in particular, these assumptions imply that the parameter estimates will be unbiased, consistent, and efficient in the class of linear unbiased estimators. It is important to note that actual data rarely satisfies the assumptions. That is, the method is used even though the assumptions are not true. Variation from the assumptions can sometimes be used as a measure of how far the model is from being useful. Many of these assumptions may be relaxed in more advanced treatments. Reports of statistical analyses usually include analyses of tests on the sample data and methodology for the fit and usefulness of the model.

Assumptions include the geometrical support of the variables.[17] Independent and dependent variables often refer to values measured at point locations. There may be spatial trends and spatial autocorrelation in the variables that violate statistical assumptions of regression. Geographic weighted regression is one technique to deal with such data.[18] Also, variables may include values aggregated by areas. With aggregated data the modifiable areal unit problem can cause extreme variation in regression parameters.[19] When analyzing data aggregated by political boundaries, postal codes or census areas results may be very distinct with a different choice of units.

21.4 Linear regression

Main article: Linear regression
See simple linear regression for a derivation of these formulas and a numerical example

In linear regression, the model specification is that the dependent variable, y_i is a linear combination of the *parameters* (but need not be linear in the *independent variables*). For example, in simple linear regression for modeling n data points there is one independent variable: x_i , and two parameters, β_0 and β_1 :

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = 1, \dots, n.$$

In multiple linear regression, there are several independent variables or functions of independent variables.

Adding a term in xi^2 to the preceding regression gives:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i,\ i = 1, \dots, n.$$

This is still linear regression; although the expression on the right hand side is quadratic in the independent variable x_i , it is linear in the parameters β_0 , β_1 and β_2.

In both cases, ε_i is an error term and the subscript i indexes a particular observation.

Returning our attention to the straight line case: Given a random sample from the population, we estimate the population parameters and obtain the sample linear regression model:

$$\widehat{y_i} = \widehat{\beta}_0 + \widehat{\beta}_1 x_i.$$

The residual, $e_i = y_i - \widehat{y}_i$, is the difference between the value of the dependent variable predicted by the model, $\widehat{y_i}$, and the true value of the dependent variable, y_i . One method of estimation is ordinary least squares. This method obtains parameter estimates that minimize the sum of squared residuals, SSE,[20][21] also sometimes denoted RSS:

$$SSE = \sum_{i=1}^{n} e_i^2.$$

Minimization of this function results in a set of normal equations, a set of simultaneous linear equations in the parameters, which are solved to yield the parameter estimators, $\widehat{\beta}_0, \widehat{\beta}_1$.

In the case of simple regression, the formulas for the least squares estimates are

$$\widehat{\beta_1} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \text{ and } \hat{\beta}_0 = \bar{y} - \widehat{\beta_1}\bar{x}$$

where $\bar{x}$ is the mean (average) of the x values and $\bar{y}$ is the mean of the y values.

Under the assumption that the population error term has a constant variance, the estimate of that variance is given by:

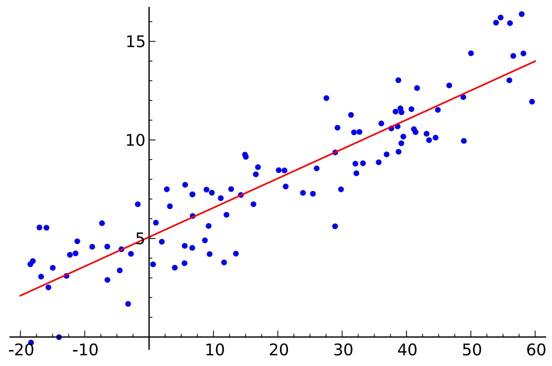


Illustration of linear regression on a data set.

$$\hat{\sigma}^2_\varepsilon = \frac{SSE}{n-2}.$$

This is called the mean square error (MSE) of the regression. The denominator is the sample size reduced by the number of model parameters estimated from the same data, (*n-p*) for *p* regressors or (n-p−1) if an intercept is used.[22] In this case, *p*=1 so the denominator is n−2.

The standard errors of the parameter estimates are given by

$$\hat{\sigma}_{\beta_0} = \hat{\sigma}_\varepsilon \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum(x_i - \bar{x})^2}}$$

$$\hat{\sigma}_{\beta_1} = \hat{\sigma}_\varepsilon \sqrt{\frac{1}{\sum(x_i - \bar{x})^2}}.$$

Under the further assumption that the population error term is normally distributed, the researcher can use these estimated standard errors to create confidence intervals and conduct hypothesis tests about the population parameters.

21.4.1 General linear model

For a derivation, see linear least squares
For a numerical example, see linear regression

In the more general multiple regression model, there are *p* independent variables:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i,$$

where *xij* is the *i*th observation on the *j*th independent variable, and where the first independent variable takes the value 1 for all *i* (so β_1 is the regression intercept).

The least squares parameter estimates are obtained from *p* normal equations. The residual can be written as

$$\varepsilon_i = y_i - \hat{\beta}_1 x_{i1} - \cdots - \hat{\beta}_p x_{ip}.$$

The **normal equations** are

$$\sum_{i=1}^{n} \sum_{k=1}^{p} X_{ij} X_{ik} \hat{\beta}_k = \sum_{i=1}^{n} X_{ij} y_i,\ j = 1, \ldots, p.$$

In matrix notation, the normal equations are written as

$$(\mathbf{X}^\top \mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}^\top \mathbf{Y},$$

where the *ij* element of *X* is *xij*, the *i* element of the column vector *Y* is *yi*, and the *j* element of $\hat{\beta}$ is $\hat{\beta}_j$. Thus *X* is *n*×*p*, *Y* is *n*×1, and $\hat{\beta}$ is *p*×1. The solution is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

21.4.2 Diagnostics

See also: Category:Regression diagnostics.

Once a regression model has been constructed, it may be important to confirm the goodness of fit of the model and the statistical significance of the estimated parameters. Commonly used checks of goodness of fit include the R-squared, analyses of the pattern of residuals and hypothesis testing. Statistical significance can be checked by an F-test of the overall fit, followed by t-tests of individual parameters.

Interpretations of these diagnostic tests rest heavily on the model assumptions. Although examination of the residuals can be used to invalidate a model, the results of a t-test or F-test are sometimes more difficult to interpret if the model's assumptions are violated. For example, if the error term does not have a normal distribution, in small samples the estimated parameters will not follow normal distributions and complicate inference. With relatively large samples, however, a central limit theorem can be invoked such that hypothesis testing may proceed using asymptotic approximations.

21.4.3 "Limited dependent" variables

The phrase "limited dependent" is used in econometric statistics for categorical and constrained variables.

The response variable may be non-continuous ("limited" to lie on some subset of the real line). For binary (zero or one) variables, if analysis proceeds with least-squares linear regression, the model is called the linear probability model. Nonlinear models for binary dependent variables include the probit and logit model. The multivariate probit model is a standard method of estimating a joint relationship between several binary dependent variables and some independent variables. For categorical variables with more than two values there is the multinomial logit. For ordinal variables with more than two values, there are the ordered logit and ordered probit models. Censored regression models may be used when the dependent variable is only sometimes observed, and Heckman correction type models may be used when the sample is not randomly selected from the population of interest. An alternative to such procedures is linear regression based on polychoric correlation (or polyserial correlations) between the categorical variables. Such procedures differ in the assumptions made about the distribution of the variables in the population. If the variable is positive with low values and represents the repetition of the occurrence of an event, then count models like the Poisson regression or the negative binomial model may be used instead.

21.5 Interpolation and extrapolation

Regression models predict a value of the *Y* variable given known values of the *X* variables. Prediction *within* the range of values in the dataset used for model-fitting is known informally as interpolation. Prediction *outside* this range of the data is known as extrapolation. Performing extrapolation relies strongly on the regression assumptions. The further the extrapolation goes outside the data, the more room there is for the model to fail due to differences between the assumptions and the sample data or the true values.

It is generally advised that when performing extrapolation, one should accompany the estimated value of the dependent variable with a prediction interval that represents the uncertainty. Such intervals tend to expand rapidly as the values of the independent variable(s) moved outside the range covered by the observed data.

For such reasons and others, some tend to say that it might be unwise to undertake extrapolation.[23]

However, this does not cover the full set of modelling errors that may be being made: in particular, the assumption of a particular form for the relation between *Y* and *X*. A properly conducted regression analysis will include an assessment of how well the assumed form is matched by the observed data, but it can only do so within the range of values of the independent variables actually available. This means that any extrapolation is particularly reliant on the assumptions being made about the structural form of the regression relationship. Best-practice advice here is that a linear-in-variables and linear-in-parameters relationship should not be chosen simply for computational convenience, but that all available knowledge should be deployed in constructing a regression model. If this knowledge includes the fact that the dependent variable cannot go outside a certain range of values, this can be made use of in selecting the model – even if the observed dataset has no values particularly near such bounds. The implications of this step of choosing an appropriate functional form for the regression can be great when extrapolation is considered. At a minimum, it can ensure that any extrapolation arising from a fitted model is “realistic” (or in accord with what is known).

21.6 Nonlinear regression

Main article: Nonlinear regression

When the model function is not linear in the parameters, the sum of squares must be minimized by an iterative procedure. This introduces many complications which are summarized in Differences between linear and non-linear least squares

21.7 Power and sample size calculations

There are no generally agreed methods for relating the number of observations versus the number of independent variables in the model. One rule of thumb suggested by Good and Hardin is $N = m^n$, where N is the sample size, n is the number of independent variables and m is the number of observations needed to reach the desired precision if the model had only one independent variable.[24] For example, a researcher is building a linear regression model using a dataset that contains 1000 patients (N). If the researcher decides that five observations are needed to precisely define a straight line (m), then the maximum number of independent variables the model can support is 4, because

$\frac{\log 1000}{\log 5} = 4.29$.

21.8 Other methods

Although the parameters of a regression model are usually estimated using the method of least squares, other methods which have been used include:

- Bayesian methods, e.g. Bayesian linear regression
- Percentage regression, for situations where reducing *percentage* errors is deemed more appropriate.[25]
- Least absolute deviations, which is more robust in the presence of outliers, leading to quantile regression

- Nonparametric regression, requires a large number of observations and is computationally intensive
- Distance metric learning, which is learned by the search of a meaningful distance metric in a given input space.[26]

21.9 Software

Main article: List of statistical packages

All major statistical software packages perform least squares regression analysis and inference. Simple linear regression and multiple regression using least squares can be done in some spreadsheet applications and on some calculators. While many statistical software packages can perform various types of nonparametric and robust regression, these methods are less standardized; different software packages implement different methods, and a method with a given name may be implemented differently in different packages. Specialized regression software has been developed for use in fields such as survey analysis and neuroimaging.

21.10 See also

- Confidence Interval for Maximin Effects in Inhomogeneous Data
- Curve fitting
- Estimation Theory
- Forecasting
- Fraction of variance unexplained
- Function approximation
- Kriging (a linear least squares estimation algorithm)
- Local regression
- Modifiable areal unit problem
- Multivariate adaptive regression splines
- Multivariate normal distribution
- Pearson product-moment correlation coefficient
- Prediction interval
- Robust regression
- Segmented regression
- Signal processing
- Stepwise regression
- Trend estimation

21.11 References

[1] Armstrong, J. Scott (2012). "Illusions in Regression Analysis". *International Journal of Forecasting (forthcoming)* **28** (3): 689. doi:10.1016/j.ijforecast.2012.02.001.

[2] David A. Freedman, *Statistical Models: Theory and Practice*, Cambridge University Press (2005)

[3] R. Dennis Cook; Sanford Weisberg Criticism and Influence Analysis in Regression, *Sociological Methodology*, Vol. 13. (1982), pp. 313–361

[4] A.M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*, Firmin Didot, Paris, 1805. "Sur la Méthode des moindres quarrés" appears as an appendix.

[5] C.F. Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum*. (1809)

[6] C.F. Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae*. (1821/1823)

[7] Mogull, Robert G. (2004). *Second-Semester Applied Statistics*. Kendall/Hunt Publishing Company. p. 59. ISBN 0-7575-1181-3.

[8] Galton, Francis (1989). "Kinship and Correlation (reprinted 1989)". *Statistical Science* (Institute of Mathematical Statistics) **4** (2): 80–86. doi:10.1214/ss/1177012581. JSTOR 2245330.

[9] Francis Galton. "Typical laws of heredity", Nature 15 (1877), 492–495, 512–514, 532–533. *(Galton uses the term "reversion" in this paper, which discusses the size of peas.)*

[10] Francis Galton. Presidential address, Section H, Anthropology. (1885) *(Galton uses the term "regression" in this paper, which discusses the height of humans.)*

[11] Yule, G. Udny (1897). "On the Theory of Correlation". *Journal of the Royal Statistical Society* (Blackwell Publishing) **60** (4): 812–54. doi:10.2307/2979746. JSTOR 2979746.

[12] Pearson, Karl; Yule, G.U.; Blanchard, Norman; Lee,Alice (1903). "The Law of Ancestral Heredity". *Biometrika* (Biometrika Trust) **2** (2): 211–236. doi:10.1093/biomet/2.2.211. JSTOR 2331683.

[13] Fisher, R.A. (1922). "The goodness of fit of regression formulae, and the distribution of regression coefficients". *Journal of the Royal Statistical Society* (Blackwell Publishing) **85** (4): 597–612. doi:10.2307/2341124. JSTOR 2341124.

[14] Ronald A. Fisher (1954). *Statistical Methods for Research Workers* (Twelfth ed.). Edinburgh: Oliver and Boyd. ISBN 0-05-002170-2.

[15] Aldrich, John (2005). "Fisher and Regression". *Statistical Science* **20** (4): 401–417. doi:10.1214/088342305000000331. JSTOR 20061201.

[16] Rodney Ramcharan. Regressions: Why Are Economists Obessessed with Them? March 2006. Accessed 2011-12-03.

[17] N. Cressie (1996) Change of Support and the Modiable Areal Unit Problem. Geographical Systems 3:159–180.

[18] Fotheringham, A. Stewart; Brunsdon, Chris; Charlton, Martin (2002). *Geographically weighted regression: the analysis of spatially varying relationships* (Reprint ed.). Chichester, England: John Wiley. ISBN 978-0-471-49616-8.

[19] Fotheringham, AS; Wong, DWS (1 January 1991). "The modifiable areal unit problem in multivariate statistical analysis". *Environment and Planning A* **23** (7): 1025–1044. doi:10.1068/a231025.

[20] M. H. Kutner, C. J. Nachtsheim, and J. Neter (2004), "Applied Linear Regression Models", 4th ed., McGraw-Hill/Irwin, Boston (p. 25)

[21] N. Ravishankar and D. K. Dey (2002), "A First Course in Linear Model Theory", Chapman and Hall/CRC, Boca Raton (p. 101)

[22] Steel, R.G.D, and Torrie, J. H., *Principles and Procedures of Statistics with Special Reference to the Biological Sciences.*, McGraw Hill, 1960, page 288.

[23] Chiang, C.L, (2003) *Statistical methods of analysis*, World Scientific. ISBN 981-238-310-7 - page 274 section 9.7.4 "interpolation vs extrapolation"

[24] Good, P. I.; Hardin, J. W. (2009). *Common Errors in Statistics (And How to Avoid Them)* (3rd ed.). Hoboken, New Jersey: Wiley. p. 211. ISBN 978-0-470-45798-6.

[25] Tofallis, C. (2009). "Least Squares Percentage Regression". *Journal of Modern Applied Statistical Methods* **7**: 526–534. doi:10.2139/ssrn.1406472.

[26] YangJing Long (2009). "Human age estimation by metric learning for regression problems" (PDF). *Proc. International Conference on Computer Analysis of Images and Patterns*: 74–82.

21.12 Further reading

- William H. Kruskal and Judith M. Tanur, ed. (1978), "Linear Hypotheses," *International Encyclopedia of Statistics*. Free Press, v. 1,

  Evan J. Williams, "I. Regression," pp. 523–41.

  Julian C. Stanley, "II. Analysis of Variance," pp. 541–554.

- Lindley, D.V. (1987). "Regression and correlation analysis," New Palgrave: A Dictionary of Economics, v. 4, pp. 120–23.
- Birkes, David and Dodge, Y., *Alternative Methods of Regression*. ISBN 0-471-56881-3
- Chatfield, C. (1993) "Calculating Interval Forecasts," *Journal of Business and Economic Statistics,* **11**. pp. 121–135.
- Draper, N.R.; Smith, H. (1998). *Applied Regression Analysis* (3rd ed.). John Wiley. ISBN 0-471-17082-8.
- Fox, J. (1997). *Applied Regression Analysis, Linear Models and Related Methods.* Sage
- Hardle, W., *Applied Nonparametric Regression* (1990), ISBN 0-521-42950-1
- Meade, N. and T. Islam (1995) "Prediction Intervals for Growth Curve Forecasts" *Journal of Forecasting,* **14**, pp. 413–430.
- A. Sen, M. Srivastava, *Regression Analysis — Theory, Methods, and Applications*, Springer-Verlag, Berlin, 2011 (4th printing).
- T. Strutz: *Data Fitting and Uncertainty (A practical introduction to weighted least squares and beyond).* Vieweg+Teubner, ISBN 978-3-8348-1022-9.
- Malakooti, B. (2013). Operations and Production Systems with Multiple Objectives. John Wiley & Sons.

21.13 External links

- Hazewinkel, Michiel, ed. (2001), "Regression analysis", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Earliest Uses: Regression – basic history and references
- Regression of Weakly Correlated Data – how linear regression mistakes can appear when Y-range is much smaller than X-range

Chapter 22

Statistical learning theory

See also: Computational learning theory
This article is about statistical learning in machine learning. For its use in psychology, see Statistical learning in language acquisition.

Statistical learning theory is a framework for machine learning drawing from the fields of statistics and functional analysis.[1] Statistical learning theory deals with the problem of finding a predictive function based on data. Statistical learning theory has led to successful applications in fields such as computer vision, speech recognition, bioinformatics and baseball.[2] It is the theoretical framework underlying support vector machines.

22.1 Introduction

The goal of learning is prediction. Learning falls into many categories, including supervised learning, unsupervised learning, online learning, and reinforcement learning. From the perspective of statistical learning theory, supervised learning is best understood.[3] Supervised learning involves learning from a training set of data. Every point in the training is an input-output pair, where the input maps to an output. The learning problem consists of inferring the function that maps between the input and the output in a predictive fashion, such that the learned function can be used to predict output from future input.

Depending of the type of output, supervised learning problems are either problems of regression or problems of classification. If the output takes a continuous range of values, it is a regression problem. Using Ohm's Law as an example, a regression could be performed with voltage as input and current as output. The regression would find the functional relationship between voltage and current to be $\frac{1}{R}$, such that

$$I = \frac{1}{R}V$$

Classification problems are those for which the output will be an element from a discrete set of labels. Classification is very common for machine learning applications. In facial recognition, for instance, a picture of a person's face would be the input, and the output label would be that person's name. The input would be represented by a large multidimensional vector, in which each dimension represents the value of one of the pixels.

After learning a function based on the training set data, that function is validated on a test set of data, data that did not appear in the training set.

22.2 Formal Description

Take X to be the vector space of all possible inputs, and Y to be the vector space of all possible outputs. Statistical learning theory takes the perspective that there is some unknown probability distribution over the product space $Z = X \otimes Y$, i.e. there exists some unknown $p(z) = p(\vec{x}, y)$. The training set is made up of n samples from this probability distribution, and is notated

$$S = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\} = \{\vec{z}_1, \dots, \vec{z}_n\}$$

Every $\vec{x}_i$ is an input vector from the training data, and y_i is the output that corresponds to it.

In this formalism, the inference problem consists of finding a function $f : X \mapsto Y$ such that $f(\vec{x}) \sim y$. Let $\mathcal{H}$ be a space of functions $f : X \mapsto Y$ called the hypothesis space. The hypothesis space is the space of functions the algorithm will search through. Let $V(f(\vec{x}), y)$ be the loss functional, a metric for the difference between the predicted value $f(\vec{x})$ and the actual value y . The expected risk is defined to be

$$I[f] = \int_{X \otimes Y} V(f(\vec{x}), y) p(\vec{x}, y) d\vec{x} dy$$

The target function, the best possible function f that can be chosen, is given by the f that satisfies

$$\inf_{f \in \mathcal{H}} I[f]$$

Because the probability distribution $p(\vec{x}, y)$ is unknown, a proxy measure for the expected risk must be used. This measure is based on the training set, a sample from this unknown probability distribution. It is called the empirical risk

$$I_S[f] = \frac{1}{n} \sum_{i=1}^{n} V(f(\vec{x}_i), y_i)$$

A learning algorithm that chooses the function f_S that minimizes the empirical risk is called empirical risk minimization.

22.3 Loss Functions

The choice of loss function is a determining factor on the function f_S that will be chosen by the learning algorithm. The loss function also affects the convergence rate for an algorithm. It is important for the loss function to be convex.[4]

Different loss functions are used depending on whether the problem is one of regression or one of classification.

22.3.1 Regression

The most common loss function for regression is the square loss function. This familiar loss function is used in ordinary least squares regression. The form is:

$$V(f(\vec{x}), y) = (y - f(\vec{x}))^2$$

The absolute value loss is also sometimes used:

$$V(f(\vec{x}), y) = |y - f(\vec{x})|$$

22.3.2 Classification

Main article: Statistical classification

In some sense the 0-1 indicator function is the most natural loss function for classification. It takes the value 0 if the predicted output is the same as the actual output, and it takes the value 1 if the predicted output is different from the actual output. For binary classification, this is:

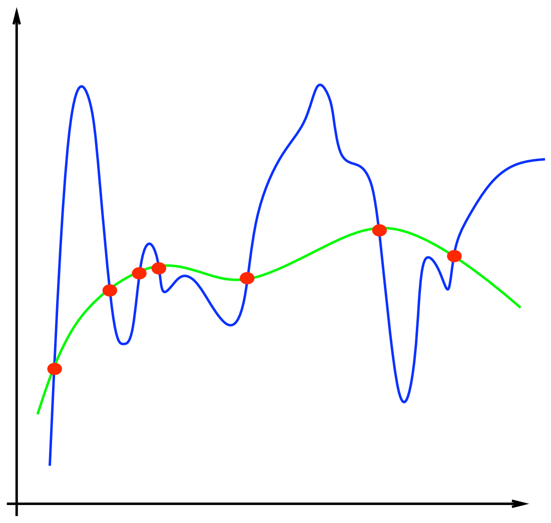
$$V(f(\vec{x}, y)) = \theta(-yf(\vec{x}))$$

where θ is the Heaviside step function.

The 0-1 loss function, however, is not convex. The hinge loss is thus often used:

$$V(f(\vec{x}, y)) = (-yf(\vec{x}))_+$$

22.4 Regularization



This image represents an example of overfitting in machine learning. The red dots represent training set data. The green line represents the true functional relationship, while the blue line shows the learned function, which has fallen victim to overfitting.

In machine learning problems, a major problem that arises is that of overfitting. Because learning is a prediction problem, the goal is not to find a function that most closely fits the (previously observed) data, but to find one that will most accurately predict output from future input. Empirical risk minimization runs this risk of overfitting: finding a function that matches the data exactly but does not predict future output well.

Overfitting is symptomatic of unstable solutions; a small perturbation in the training set data would cause a large variation in the learned function. It can be shown that if the stability for the solution can be guaranteed, generalization and consistency are guaranteed as well.[5][6] Regularization can solve the overfitting problem and give the problem stability.

Regularization can be accomplished by restricting the hypothesis space $\mathcal{H}$. A common example would be restricting $\mathcal{H}$ to linear functions: this can be seen as a reduction to the standard problem of linear regression. $\mathcal{H}$ could also be restricted to polynomial of degree p , exponentials, or bounded functions on L1. Restriction of the hypothesis space avoids overfitting because the form of the potential functions are limited, and so does not allow for the choice of a function that gives empirical risk arbitrarily close to zero.

One example of regularization is Tikhonov regularization. This consists of minimizing

$$\frac{1}{n}\sum_{i=1}^{n} V(f(\vec{x}_i, y_i)) + \gamma \|f\|_{\mathcal{H}}^2$$

where γ is a fixed and positive parameter, the regularization parameter. Tikhonov regularization ensures existence, uniqueness, and stability of the solution.[7]

22.5 See also

- Reproducing kernel Hilbert spaces are a useful choice for $\mathcal{H}$.
- Proximal gradient methods for learning

22.6 References

[1] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) *Foundations of Machine Learning*, The MIT Press ISBN 9780262018258.

[2] Gagan Sidhu, Brian Caffo. Exploiting pitcher decision-making using Reinforcement Learning. *Annals of Applied Statistics*

[3] Tomaso Poggio, Lorenzo Rosasco, et al. *Statistical Learning Theory and Applications*, 2012, Class 1

[4] Rosasco, L., Vito, E.D., Caponnetto, A., Fiana, M., and Verri A. 2004. *Neural computation* Vol 16, pp 1063-1076

[5] Vapnik, V.N. and Chervonenkis, A.Y. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* Vol 16, pp 264-280.

[6] Mukherjee, S., Niyogi, P. Poggio, T., and Rifkin, R. 2006. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*. Vol 25, pp 161-193.

[7] Tomaso Poggio, Lorenzo Rosasco, et al. *Statistical Learning Theory and Applications*, 2012, Class 2

Chapter 23

Vapnik–Chervonenkis theory

Vapnik–Chervonenkis theory (also known as **VC theory**) was developed during 1960–1990 by Vladimir Vapnik and Alexey Chervonenkis. The theory is a form of computational learning theory, which attempts to explain the learning process from a statistical point of view.

VC theory is related to statistical learning theory and to empirical processes. Richard M. Dudley and Vladimir Vapnik himself, among others, apply VC-theory to empirical processes.

23.1 Introduction

VC theory covers at least four parts (as explained in *The Nature of Statistical Learning Theory*):

- Theory of consistency of learning processes
    - What are (necessary and sufficient) conditions for consistency of a learning process based on the empirical risk minimization principle?
- Nonasymptotic theory of the rate of convergence of learning processes
    - How fast is the rate of convergence of the learning process?
- Theory of controlling the generalization ability of learning processes
    - How can one control the rate of convergence (the generalization ability) of the learning process?
- Theory of constructing learning machines
    - How can one construct algorithms that can control the generalization ability?

VC Theory is a major subbranch of statistical learning theory. One of its main applications in statistical learning theory is to provide generalization conditions for learning algorithms. From this point of view, VC theory is related to **stability**, which is an alternative approach for characterizing generalization.

In addition, VC theory and VC dimension are instrumental in the theory of empirical processes, in the case of processes indexed by VC classes. Arguably these are the most important applications of the VC theory, and are employed in proving generalization. Several techniques will be introduced that are widely used in the empirical process and VC theory. The discussion is mainly based on the book "Weak Convergence and Empirical Processes: With Applications to Statistics".

23.2 Overview of VC theory in Empirical Processes

23.2.1 Background on Empirical Processes

Let $X_1, \ldots, X_n$ be random elements defined on a measurable space $(\mathcal{X}, \mathcal{A})$. For a measure Q set:

$$Qf = \int f dQ$$

Measurability issues, will be ignored here, for more technical detail see . Let $\mathcal{F}$ be a class of measurable functions $f : \mathcal{X} \to \mathbf{R}$ and define:

$$\|Q\|_{\mathcal{F}} = \sup\{|Qf| \ : \ f \in \mathcal{F}\}.$$

Define the empirical measure

$$\mathbb{P}_n = n^{-1} \sum_{i=1}^{n} \delta_{X_i},$$

where δ here stands for the Dirac measure. The empirical measure induces a map $\mathcal{F} \to \mathbf{R}$ given by:

$$f \mapsto \mathbb{P}_n f$$

Now suppose P is the underlying true distribution of the data, which is unknown. Empirical Processes theory aims at identifying classes $\mathcal{F}$ for which statements such as the following hold:

- uniform law of large numbers:

$$\|\mathbb{P}_n - P\|_{\mathcal{F}} \to 0,$$

- uniform central limit theorem:

$$\mathbb{G}_n = \sqrt{n}(\mathbb{P}_n - P) \rightsquigarrow \mathbb{G}, \quad \text{in} \ell^\infty(\mathcal{F})$$

In the former case $\mathcal{F}$ is called *Glivenko-Cantelli* class, and in the latter case (under the assumption $\forall x, \sup_{f\in\mathcal{F}} |f(x) - Pf| < \infty$) the class $\mathcal{F}$ is called *Donsker* or P-Donsker. Obviously, a Donsker class is Glivenko-Cantelli in probability by an application of Slutsky's theorem .

These statements are true for a single f , by standard LLN, CLT arguments under regularity conditions, and the difficulty in the Empirical Processes comes in because joint statements are being made for all $f \in \mathcal{F}$. Intuitively then, the set $\mathcal{F}$ cannot be too large, and as it turns out that the geometry of $\mathcal{F}$ plays a very important role.

One way of measuring how big the function set $\mathcal{F}$ is to use the so-called covering numbers. The covering number

$$N(\varepsilon, \mathcal{F}, \|\cdot\|)$$

is the minimal number of balls $\{g : \|g - f\| < \varepsilon\}$ needed to cover the set $\mathcal{F}$ (here it is obviously assumed that there is an underlying norm on $\mathcal{F}$). The entropy is the logarithm of the covering number.

Two sufficient conditions are provided below, under which it can be proved that the set $\mathcal{F}$ is Glivenko-Cantelli or Donsker.

A class $\mathcal{F}$ is P-Glivenko-Cantelli if it is P-measurable with envelope F such that $P^*F < \infty$ and satisfies:

$$\forall\varepsilon > 0 \quad \sup_Q N(\varepsilon\|F\|_Q, \mathcal{F}, L_1(Q)) < \infty.$$

The next condition is a version of the celebrated Dudley's theorem. If $\mathcal{F}$ is a class of functions such that

$$\int_0^\infty \sup_Q \sqrt{\log N\left(\varepsilon\|F\|_{Q,2}, \mathcal{F}, L_2(Q)\right)}d\varepsilon < \infty$$

then $\mathcal{F}$ is P-Donsker for every probability measure P such that $P^*F^2 < \infty$. In the last integral, the notation means

$$\|f\|_{Q,2} = \left(\int |f|^2 dQ\right)^{\frac{1}{2}}$$

23.2.2 Symmetrization

The majority of the arguments of how to bound the empirical process, rely on symmetrization, maximal and concentration inequalities and chaining. Symmetrization is usually the first step of the proofs, and since it is used in many machine learning proofs on bounding empirical loss functions (including the proof of the VC inequality which is discussed in the next section) it is presented here.

Consider the empirical process:

$$f \mapsto (\mathbb{P}_n - P)f = \frac{1}{n}\sum_{i=1}^n (f(X_i) - Pf)$$

Turns out that there is a connection between the empirical and the following symmetrized process:

$$f \mapsto \mathbb{P}_n^0 = \frac{1}{n}\sum_{i=1}^n \varepsilon_i f(X_i)$$

The symmetrized process is a Rademacher process, conditionally on the data X_i . Therefore it is a sub-Gaussian process by Hoeffding's inequality.

Lemma (Symmetrization). For every nondecreasing, convex Φ: **R** → **R** and class of measurable functions $\mathcal{F}$,

$$\mathbb{E}\Phi(\|\mathbb{P}_n - P\|_{\mathcal{F}}) \leq \mathbb{E}\Phi\left(2\left\|\mathbb{P}_n^0\right\|_{\mathcal{F}}\right)$$

The proof of the Symmetrization lemma relies on introducing independent copies of the original variables X_i (sometimes referred to as a *ghost sample*) and replacing the inner expectation of the LHS by these copies. After an application of Jensen's inequality different signs could be introduced (hence the name symmetrization) without changing the expectation. The proof can be found below because of its instructive nature.

[Proof]

Introduce the "ghost sample" $Y_1, \ldots, Y_n$ to be independent copies of $X_1, \ldots, X_n$. For fixed values of $X_1, \ldots, X_n$ one has:

$$\|\mathbb{P}_n - P\|_{\mathcal{F}} = \sup_{f\in\mathcal{F}} \frac{1}{n}\left|\sum_{i=1}^n f(X_i) - \mathbb{E}f(Y_i)\right| \leq \mathbb{E}_Y \sup_{f\in\mathcal{F}} \frac{1}{n}\left|\sum_{i=1}^n f(X_i) - f\right.$$

Therefore by Jensen's inequality:

$$\Phi(\|\mathbb{P}_n - P\|_{\mathcal{F}}) \leq \mathbb{E}_Y \Phi\left(\left\|\frac{1}{n}\sum_{i=1}^n f(X_i) - f(Y_i)\right\|_{\mathcal{F}}\right)$$

Taking expectation with respect to X gives:

$$\mathbb{E}\Phi(\|\mathbb{P}_n - P\|_{\mathcal{F}}) \le \mathbb{E}_X\mathbb{E}_Y\Phi\left(\left\|\frac{1}{n}\sum_{i=1}^{n} f(X_i) - f(Y_i)\right\|_{\mathcal{F}}\right)$$

Note that adding a minus sign in front of a term $f(X_i) - f(Y_i)$ doesn't change the RHS, because it's a symmetric function of X and Y . Therefore the RHS remains the same under “sign perturbation":

$$\mathbb{E}\Phi\left(\left\|\frac{1}{n}\sum_{i=1}^{n} e_i f(X_i) - f(Y_i)\right\|_{\mathcal{F}}\right)$$

for any $(e_1, e_2, \dots, e_n) \in \{-1,1\}^n$. Therefore:

$$\mathbb{E}\Phi(\|\mathbb{P}_n - P\|_{\mathcal{F}}) \le \mathbb{E}_\varepsilon\mathbb{E}\Phi\left(\left\|\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i f(X_i) - f(Y_i)\right\|_{\mathcal{F}}\right)$$

Finally using first triangle inequality and then convexity of Φ gives:

$$\mathbb{E}\Phi(\|\mathbb{P}_n - P\|_{\mathcal{F}}) \le \frac{1}{2}\mathbb{E}_\varepsilon\mathbb{E}\Phi\left(2\left\|\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i f(X_i)\right\|_{\mathcal{F}}\right) + \frac{1}{2}\mathbb{E}_Y\mathbb{E}\Phi\left(2\left\|\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i f(Y_i)\right\|_{\mathcal{F}}\right)$$

Where the last two expressions on the RHS are the same, which concludes the proof.

A typical way of proving empirical CLTs, first uses symmetrization to pass the empirical process to $\mathbb{P}_n^0$ and then argue conditionally on the data, using the fact that Rademacher processes are simple processes with nice properties.

23.2.3 VC Connection

It turns out that there is a fascinating connection between certain combinatorial properties of the set $\mathcal{F}$ and the entropy numbers. Uniform covering numbers can be controlled by the notion of *Vapnik-Cervonenkis classes of sets* - or shortly *VC sets*.

Take a collection of subsets of the sample space $\mathcal{X}$ - $\mathcal{C}$. A collection of sets $\mathcal{C}$ is said to *pick out* a certain subset of the finite set $S = \{x_1, \dots, x_n\} \subset \mathcal{X}$ if $S = S \cap C$ for some $C \in \mathcal{C}$. $\mathcal{C}$ is said to *shatter* S if it picks out each of its 2^n subsets. The *VC-index* (similar to VC dimension + 1 for an appropriately chosen classifier set) $V(\mathcal{C})$ of $\mathcal{C}$ is the smallest n for which no set of size n is shattered by $\mathcal{C}$.

Sauer's lemma then states that the number $\Delta_n(\mathcal{C}, x_1, \dots, x_n)$ of subsets picked out by a VC-class $\mathcal{C}$ satisfies:

$$\max_{x_1,\dots,x_n} \Delta_n(\mathcal{C}, x_1, \dots, x_n) \le \sum_{j=0}^{V(\mathcal{C})-1}\binom{n}{j} \le \left(\frac{ne}{V(\mathcal{C})-1}\right)^{V(\mathcal{C})-1}$$

Which is a polynomial number $O(n^{V(\mathcal{C})-1})$ of subsets rather than an exponential number. Intuitively this means that a finite VC-index implies that $\mathcal{C}$ has an apparent simplistic structure.

A similar bound can be shown (with a different constant, same rate) for the so-called *VC subgraph classes*. For a function $f : \mathcal{X} \to \mathbf{R}$ the *subgraph* is a subset of $\mathcal{X} \times \mathbf{R}$ such that: $\{(x,t) : t < f(x)\}$. A collection of $\mathcal{F}$ is called a VC subgraph class if all subgraphs form a VC-class.

Consider a set of indicator functions $\mathcal{I}_{\mathcal{C}} = \{1_C : C \in \mathcal{C}\}$ in $L_1(Q)$ for discrete empirical type of measure Q (or equivalently for any probability measure Q). It can then be shown that quite remarkably, for $r \ge 1$:

$$N(\varepsilon, \mathcal{I}_{\mathcal{C}}, L_r(Q)) \le KV(\mathcal{C})(4e)^{V(\mathcal{C})}\varepsilon^{-r(V(\mathcal{C})-1)}$$

Further consider the *symmetric convex hull* of a set $\mathcal{F}$: sconv $\mathcal{F}$ being the collection of functions of the form $\sum_{i=1}^m \alpha_i f_i$ with $\sum_{i=1}^m |\alpha_i| \le 1$. Then if

$$N(\varepsilon\|F\|_{Q,2}, \mathcal{F}, L_2(Q)) \le C\varepsilon^{-V}$$

the following is valid for the convex hull of $\mathcal{F}$:

$$\log N\left(\varepsilon\|F\|_{Q,2}, \text{sconv}\,\mathcal{F}, L_2(Q)\right) \le K\varepsilon^{-\frac{2V}{V+2}}$$

The important consequence of this fact is that

$$\frac{2V}{V+2} > 2,$$

which is just enough so that the entropy integral is going to converge, and therefore the class sconv $\mathcal{F}$ is going to be P-Donsker.

Finally an example of a VC-subgraph class is considered. Any finite-dimensional vector space $\mathcal{F}$ of measurable functions $f : \mathcal{X} \to \mathbf{R}$ is VC-subgraph of index smaller than or equal to $\dim(\mathcal{F}) + 2$.

[Proof]

Take $n = \dim(\mathcal{F}) + 2$ points $(x_1, t_1), \dots, (x_n, t_n)$. The vectors:

$$(f(x_1), \dots, f(x_n)) - (t_1, \dots, t_n)$$

are in a $n-1$ dimensional subspace of $\mathbf{R}^n$. Take $a \neq 0$, a vector that is orthogonal to this subspace. Therefore:

$$\sum_{a_i > 0} a_i(f(x_i) - t_i) = \sum_{a_i < 0}(-a_i)(f(x_i) - t_i), \quad \forall f \in \mathcal{F}$$

Consider the set $S = \{(x_i, t_i) : a_i > 0\}$. This set cannot be picked out since if there is some f such that $S = \{(x_i, t_i) : f(x_i) > t_i\}$ that would imply that the LHS is strictly positive but the RHS is non-negative.

There are generalizations of the notion VC subgraph class, e.g. there is the notion of pseudo-dimension. The interested reader can look into.

23.3 VC Inequality

A similar setting is considered, which is more common to machine learning. Let $\mathcal{X}$ is a feature space and $\mathcal{Y} = \{0, 1\}$. A function $f : \mathcal{X} \to \mathcal{Y}$ is called a classifier. Let $\mathcal{F}$ be a set of classifiers. Similarly to the previous section, define the *shattering coefficient* (also known as growth function):

$$S(\mathcal{F}, n) = \max_{x_1,\dots,x_n} |\{(f(x_1), \dots, f(x_n)), f \in \mathcal{F}\}|$$

Note here that there is a 1:1 mapping between each of the functions in $\mathcal{F}$ and the set on which the function is 1. Therefore in terms of the previous section the shattering coefficient is precisely

$$\max_{x_1,\dots,x_n} \Delta_n(\mathcal{C}, x_1, \dots, x_n)$$

for $\mathcal{C}$ being the collection of all sets described above. Now for the same reasoning as before, namely using Sauer's Lemma it can be shown that $S(\mathcal{F}, n)$ is going to be polynomial in n provided that the class $\mathcal{F}$ has a finite VC-dimension or equivalently the collection $\mathcal{C}$ has finite VC-index.

Let $D_n = \{(X_1, Y_1), \dots, (X_n, Y_m)\}$ is an observed dataset. Assume that the data is generated by an unknown probability distribution P_{XY} . Define $R(f) = P(f(X) \neq Y)$ to be the expected 0/1 loss. Of course since P_{XY} is unknown in general, one has no access to $R(f)$. However the *empirical risk*, given by:

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(f(X_n) \neq Y_n)$$

can certainly be evaluated. Then one has the following Theorem:

23.3.1 Theorem (VC Inequality)

For binary classification and the 0/1 loss function we have the following generalization bounds:

$$P\left(\sup_{f\in\mathcal{F}} \left|\hat{R}_n(f) - R(f)\right| > \varepsilon\right) \leq 8S(\mathcal{F}, n)e^{-n\varepsilon^2/32}$$

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}} \left|\hat{R}_n(f) - R(f)\right|\right] \leq 2\sqrt{\frac{\log S(\mathcal{F}, n) + \log 2}{n}}$$

In words the VC inequality is saying that as the sample increases, provided that $\mathcal{F}$ has a finite VC dimension, the empirical 0/1 risk becomes a good proxy for the expected 0/1 risk. Note that both RHS of the two inequalities will converge to 0, provided that $S(\mathcal{F}, n)$ grows polynomially in n.

The connection between this framework and the Empirical Process framework is evident. Here one is dealing with a modified empirical process

$$\left|\hat{R}_n - R\right|_{\mathcal{F}}$$

but not surprisingly the ideas are the same. The proof of the (first part of) VC inequality, relies on symmetrization, and then argue conditionally on the data using concentration inequalities (in particular Hoeffding's inequality). The interested reader can check the book Theorems 12.4 and 12.5.

23.4 References

- ^ Vapnik, Vladimir N (2000). *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer-Verlag. ISBN 978-0-387-98780-4.
- Vapnik, Vladimir N (1989). Statistical Learning Theory. Wiley-Interscience. ISBN 0-471-03003-1.
- ^ van der Vaart, Aad W.; Wellner, Jon A. (2000). *Weak Convergence and Empirical Processes: With Applications to Statistics* (2nd ed.). Springer. ISBN 978-0-387-94640-5.
- ^ Gyorfi, L.; Devroye, L.; Lugosi, G. (1996). *A probabilistic theory of pattern recognition.* (1st ed.). Springer. ISBN 978-0387946184.
- See references in articles: Richard M. Dudley, empirical processes, Shattered set.
- ^ Pollard, David (1990). Empirical Processes: Theory and Applications. NSF-CBMS Regional Conference Series in Probability and Statistics Volume 2. ISBN 0-940600-16-1.
- Bousquet, O.; Boucheron, S.; Lugosi, G. (2004). "Introduction to Statistical Learning Theory". *Advanced Lectures on Machine Learning Lecture Notes in Artificial Intelligence 3176, 169-207. (Eds.)*

Bousquet, O., U. von Luxburg and G. Ratsch, Springer.

- Vapnik, V.; Chervonenkis, A. (2004). "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities". *Theory Probab. Appl., 16(2), 264–280*.

Chapter 24

Probably approximately correct learning

In computational learning theory, **probably approximately correct learning** (**PAC learning**) is a framework for mathematical analysis of machine learning. It was proposed in 1984 by Leslie Valiant.[1]

In this framework, the learner receives samples and must select a generalization function (called the *hypothesis*) from a certain class of possible functions. The goal is that, with high probability (the "probably" part), the selected function will have low generalization error (the "approximately correct" part). The learner must be able to learn the concept given any arbitrary approximation ratio, probability of success, or distribution of the samples.

The model was later extended to treat noise (misclassified samples).

An important innovation of the PAC framework is the introduction of computational complexity theory concepts to machine learning. In particular, the learner is expected to find efficient functions (time and space requirements bounded to a polynomial of the example size), and the learner itself must implement an efficient procedure (requiring an example count bounded to a polynomial of the concept size, modified by the approximation and likelihood bounds).

24.1 Definitions and terminology

In order to give the definition for something that is PAC-learnable, we first have to introduce some terminology.[2] [3]

For the following definitions, two examples will be used. The first is the problem of character recognition given an array of n bits encoding a binary-valued image. The other example is the problem of finding an interval that will correctly classify points within the interval as positive and the points outside of the range as negative.

Let X be a set called the *instance space* or the encoding of all the samples, and each *instance* have length assigned. In the character recognition problem, the instance space is $X = \{0,1\}^n$. In the interval problem the instance space is $X = \mathbb{R}$, where $\mathbb{R}$ denotes the set of all real numbers.

A *concept* is a subset $c \subset X$. One concept is the set of all patterns of bits in $X = \{0,1\}^n$ that encode a picture of the letter "P". An example concept from the second example is the set of all of the numbers between $\pi/2$ and $\sqrt{10}$. A *concept class* C is a set of concepts over X . This could be the set of all subsets of the array of bits that are skeletonized 4-connected (width of the font is 1).

Let $EX(c, D)$ be a procedure that draws an example, x , using a probability distribution D and gives the correct label $c(x)$, that is 1 if $x \in c$ and 0 otherwise.

Say that there is an algorithm A that given access to $EX(c, D)$ and inputs ϵ and δ that, with probability of at least $1 - \delta$, A outputs a hypothesis $h \in C$ that has error less than or equal to ϵ with examples drawn from X with the distribution D . If there is such an algorithm for every concept $c \in C$, for every distribution D over X , and for all $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$ then C is **PAC learnable** (or *distribution-free PAC learnable*). We can also say that A is a **PAC learning algorithm** for C .

An algorithm runs in time t if it draws at most t examples and requires at most t time steps. A concept class is **efficiently PAC learnable** if it is *PAC learnable* by an algorithm that runs in time polynomial in $1/\epsilon$, $1/\delta$ and *instance* length.

24.2 Equivalence

Under some regularity conditions these three conditions are equivalent:

1. The concept class C is **PAC learnable**.
2. The VC dimension of C is finite.
3. C is a uniform Glivenko-Cantelli class.

24.3 References

[1] L. Valiant. *A theory of the learnable.* Communications of the ACM, 27, 1984.

[2] Kearns and Vazirani, pg. 1-12,

[3] Balas Kausik Natarajan, Machine Learning , A Theoretical Approach, Morgan Kaufmann Publishers, 1991

24.4 Further reading

- M. Kearns, U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. A textbook.
- D. Haussler. Overview of the Probably Approximately Correct (PAC) Learning Framework. An introduction to the topic.
- L. Valiant. *Probably Approximately Correct.* Basic Books, 2013. In which Valiant argues that PAC learning describes how organisms evolve and learn.

Chapter 25

Algorithmic learning theory

Algorithmic learning theory is a mathematical framework for analyzing machine learning problems and algorithms. Synonyms include **formal learning theory** and **algorithmic inductive inference**. Algorithmic learning theory is different from statistical learning theory in that it does not make use of statistical assumptions and analysis. Both algorithmic and statistical learning theory are concerned with machine learning and can thus be viewed as branches of computational learning theory.

25.1 Distinguishing Characteristics

Unlike statistical learning theory and most statistical theory in general, algorithmic learning theory does not assume that data are random samples, that is, that data points are independent of each other. This makes the theory suitable for domains where observations are (relatively) noise-free but not random, such as language learning [1] and automated scientific discovery.[2][3]

The fundamental concept of algorithmic learning theory is learning in the limit: as the number of data points increases, a learning algorithm should converge to a correct hypothesis on *every* possible data sequence consistent with the problem space. This is a non-probabilistic version of statistical consistency, which also requires convergence to a correct model in the limit, but allows a learner to fail on data sequences with probability measure 0.

Algorithmic learning theory investigates the learning power of Turing machines. Other frameworks consider a much more restricted class of learning algorithms than Turing machines, for example learners that compute hypotheses more quickly, for instance in polynomial time. An example of such a framework is probably approximately correct learning.

25.2 Learning in the limit

The concept was introduced in E. Mark Gold's seminal paper "Language identification in the limit".[4] The objective of language identification is for a machine running one program to be capable of developing another program by which any given sentence can be tested to determine whether it is “grammatical” or “ungrammatical”. The language being learned need not be English or any other natural language - in fact the definition of “grammatical” can be absolutely anything known to the tester.

In Gold’s learning model, the tester gives the learner an example sentence at each step, and the learner responds with a hypothesis, which is a suggested program to determine grammatical correctness. It is required of the tester that every possible sentence (grammatical or not) appears in the list eventually, but no particular order is required. It is required of the learner that at each step the hypothesis must be correct for all the sentences so far.

A particular learner is said to be able to “learn a language in the limit” if there is a certain number of steps beyond which its hypothesis no longer changes. At this point it has indeed learned the language, because every possible sentence appears somewhere in the sequence of inputs (past or future), and the hypothesis is correct for all inputs (past or future), so the hypothesis is correct for every sentence. The learner is not required to be able to tell when it has reached a correct hypothesis, all that is required is that it be true.

Gold showed that any language which is defined by a Turing machine program can be learned in the limit by another Turing-complete machine using enumeration. This is done by the learner testing all possible Turing machine programs in turn until one is found which is correct so far - this forms the hypothesis for the current step. Eventually, the correct program will be reached, after which the hypothesis will never change again (but note that the learner does not know that it won't need to change).

Gold also showed that if the learner is given only positive examples (that is, only grammatical sentences appear in the input, not ungrammatical sentences), then the language can only be guaranteed to be learned in the limit if there are only a finite number of possible sentences in the language (this is possible if, for example, sentences are known to be of limited length).

Language identification in the limit is a highly abstract model. It does not allow for limits of runtime or computer memory which can occur in practice, and the enumeration method may fail if there are errors in the input. However the framework is very powerful, because if these strict conditions are maintained, it allows the learning of any program known to be computable. This is because a Turing machine program can be written to mimic any program in any conventional programming language. See Church-Turing thesis.

25.3 Other Identification Criteria

Learning theorists have investigated other learning criteria,[5] such as the following.

- *Efficiency*: minimizing the number of data points required before convergence to a correct hypothesis.
- *Mind Changes*: minimizing the number of hypothesis changes that occur before convergence.[6]

Mind change bounds are closely related to mistake bounds that are studied in statistical learning theory.[7] Kevin Kelly has suggested that minimizing mind changes is closely related to choosing maximally simple hypotheses in the sense of Occam's Razor.[8]

25.4 See also

- Sample exclusion dimension

25.5 References

[1] Jain, S. et al (1999): *Systems That Learn*, 2nd ed. Cambridge, MA: MIT Press.

[2] Langley, P.; Simon, H.; Bradshaw, G. & Zytkow, J. (1987), *Scientific Discovery: Computational Explorations of the Creative Processes*, MIT Press, Cambridge

[3] Schulte, O. (2009), *Simultaneous Discovery of Conservation Laws and Hidden Particles With Smith Matrix Decomposition*, in Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09), pp. 1481-1487

[4] Gold, E. Mark (1967), *Language Identification in the Limit* (PDF) **10**, Information and Control, pp. 447–474 The original paper.

[5] Jain, S. et al (1999): *Systems That Learn*, 2nd ed. Cambridge, MA: MIT Press.

[6] Luo, W. & Schulte, O. (2005), *Mind Change Efficient Learning*, in Peter Auer & Ron Meir, ed., Proceedings of the Conference on Learning Theory (COLT), pp. 398-412

[7] Jain, S. and Sharma, A. (1999), *On a generalized notion of mistake bounds*, Proceedings of the Conference on Learning Theory (COLT), pp.249-256.

[8] Kevin T. Kelly (2007), *Ockham's Razor, Empirical Complexity, and Truth-finding Efficiency*, Theoretical Computer Science, 383: 270-289.

25.6 External links

- Learning Theory in Computer Science.
- The Stanford Encyclopaedia of Philosophy provides a highly accessible introduction to key concepts in algorithmic learning theory, especially as they apply to the philosophical problems of inductive inference.

Chapter 26

Statistical hypothesis testing

"Critical region" redirects here. For the computer science notion of a "critical section", sometimes called a "critical region", see critical section.

A **statistical hypothesis** is a scientific hypothesis that is testable on the basis of observing a process that is modeled via a set of random variables.[1] A **statistical hypothesis test** is a method of statistical inference used for testing a statistical hypothesis.

A test result is called *statistically significant* if it has been predicted as unlikely to have occurred by sampling error alone, according to a threshold probability—the significance level. Hypothesis tests are used in determining what outcomes of a study would lead to a rejection of the null hypothesis for a pre-specified level of significance. In the Neyman-Pearson framework (see below), the process of distinguishing between the null hypothesis and the alternative hypothesis is aided by identifying two conceptual types of errors (type 1 & type 2), and by specifying parametric limits on e.g. how much type 1 error will be permitted.

An alternative framework for statistical hypothesis testing is to specify a set of statistical models, one for each candidate hypothesis, and then use model selection techniques to choose the most appropriate model.[2] The most common selection techniques are based on either Akaike information criterion or Bayes factor.

Statistical hypothesis testing is sometimes called **confirmatory data analysis**. It can be contrasted with exploratory data analysis, which may not have pre-specified hypotheses.

26.1 Variations and sub-classes

Statistical hypothesis testing is a key technique of both Frequentist inference and Bayesian inference, although the two types of inference have notable differences. Statistical hypothesis tests define a procedure that controls (fixes) the probability of incorrectly *deciding* that a default position (null hypothesis) is incorrect. The procedure is based on how likely it would be for a set of observations to occur if the null hypothesis were true. Note that this probability of making an incorrect decision is *not* the probability that the null hypothesis is true, nor whether any specific alternative hypothesis is true. This contrasts with other possible techniques of decision theory in which the null and alternative hypothesis are treated on a more equal basis.

One naive Bayesian approach to hypothesis testing is to base decisions on the posterior probability,[3][4] but this fails when comparing point and continuous hypotheses. Other approaches to decision making, such as Bayesian decision theory, attempt to balance the consequences of incorrect decisions across all possibilities, rather than concentrating on a single null hypothesis. A number of other approaches to reaching a decision based on data are available via decision theory and optimal decisions, some of which have desirable properties. Hypothesis testing, though, is a dominant approach to data analysis in many fields of science. Extensions to the theory of hypothesis testing include the study of the power of tests, i.e. the probability of correctly rejecting the null hypothesis given that it is false. Such considerations can be used for the purpose of sample size determination prior to the collection of data.

26.2 The testing process

In the statistics literature, statistical hypothesis testing plays a fundamental role.[5] The usual line of reasoning is as follows:

1. There is an initial research hypothesis of which the truth is unknown.
2. The first step is to state the relevant **null and alternative hypotheses**. This is important as mis-stating the hypotheses will muddy the rest of the process.
3. The second step is to consider the statistical assumptions being made about the sample in doing the test; for example, assumptions about the statistical independence or about the form of the distributions of the observations. This is equally important as invalid assumptions will mean that the results of the test are invalid.

4. Decide which test is appropriate, and state the relevant **test statistic** T.

5. Derive the distribution of the test statistic under the null hypothesis from the assumptions. In standard cases this will be a well-known result. For example the test statistic might follow a Student's t distribution or a normal distribution.

6. Select a significance level (α), a probability threshold below which the null hypothesis will be rejected. Common values are 5% and 1%.

7. The distribution of the test statistic under the null hypothesis partitions the possible values of T into those for which the null hypothesis is rejected—the so-called *critical region*—and those for which it is not. The probability of the critical region is α.

8. Compute from the observations the observed value t_{obs} of the test statistic T.

9. Decide to either reject the null hypothesis in favor of the alternative or not reject it. The decision rule is to reject the null hypothesis H_0 if the observed value t_{obs} is in the critical region, and to accept or "fail to reject" the hypothesis otherwise.

An alternative process is commonly used:

1. Compute from the observations the observed value t_{obs} of the test statistic T.

2. Calculate the p-value. This is the probability, under the null hypothesis, of sampling a test statistic at least as extreme as that which was observed.

3. Reject the null hypothesis, in favor of the alternative hypothesis, if and only if the p-value is less than the significance level (the selected probability) threshold.

The two processes are equivalent.[6] The former process was advantageous in the past when only tables of test statistics at common probability thresholds were available. It allowed a decision to be made without the calculation of a probability. It was adequate for classwork and for operational use, but it was deficient for reporting results.

The latter process relied on extensive tables or on computational support not always available. The explicit calculation of a probability is useful for reporting. The calculations are now trivially performed with appropriate software.

The difference in the two processes applied to the Radioactive suitcase example (below):

- "The Geiger-counter reading is 10. The limit is 9. Check the suitcase."
- "The Geiger-counter reading is high; 97% of safe suitcases have lower readings. The limit is 95%. Check the suitcase."

The former report is adequate, the latter gives a more detailed explanation of the data and the reason why the suitcase is being checked.

It is important to note the difference between accepting the null hypothesis and simply failing to reject it. The "fail to reject" terminology highlights the fact that the null hypothesis is assumed to be true from the start of the test; if there is a lack of evidence against it, it simply continues to be assumed true. The phrase "accept the null hypothesis" may suggest it has been proved simply because it has not been disproved, a logical fallacy known as the argument from ignorance. Unless a test with particularly high power is used, the idea of "accepting" the null hypothesis may be dangerous. Nonetheless the terminology is prevalent throughout statistics, where its meaning is well understood.

The processes described here are perfectly adequate for computation. They seriously neglect the design of experiments considerations.[7][8]

It is particularly critical that appropriate sample sizes be estimated before conducting the experiment.

The phrase "test of significance" was coined by statistician Ronald Fisher.[9]

26.2.1 Interpretation

If the *p*-value is less than the required significance level (equivalently, if the observed test statistic is in the critical region), then we say the null hypothesis is rejected at the given level of significance. Rejection of the null hypothesis is a conclusion. This is like a "guilty" verdict in a criminal trial: the evidence is sufficient to reject innocence, thus proving guilt. We might accept the alternative hypothesis (and the research hypothesis).

If the *p*-value is *not* less than the required significance level (equivalently, if the observed test statistic is outside the critical region), then the test has no result. The evidence is insufficient to support a conclusion. (This is like a jury that fails to reach a verdict.) The researcher typically gives extra consideration to those cases where the *p*-value is close to the significance level.

In the Lady tasting tea example (below), Fisher required the Lady to properly categorize all of the cups of tea to justify the conclusion that the result was unlikely to result from chance. He defined the critical region as that case alone. The region was defined by a probability (that the null hypothesis was correct) of less than 5%.

Whether rejection of the null hypothesis truly justifies acceptance of the research hypothesis depends on the structure of the hypotheses. Rejecting the hypothesis that a large paw print originated from a bear does not immediately prove the existence of Bigfoot. Hypothesis testing emphasizes the rejection, which is based on a probability, rather than the acceptance, which requires extra steps of logic.

"The probability of rejecting the null hypothesis is a function of five factors: whether the test is one- or two tailed, the level of significance, the standard deviation, the amount of deviation from the null hypothesis, and the number of observations."[10] These factors are a source of criticism; factors under the control of the experimenter/analyst give the results an appearance of subjectivity.

26.2.2 Use and importance

Statistics are helpful in analyzing most collections of data. This is equally true of hypothesis testing which can justify conclusions even when no scientific theory exists. In the Lady tasting tea example, it was "obvious" that no difference existed between (milk poured into tea) and (tea poured into milk). The data contradicted the "obvious".

Real world applications of hypothesis testing include:[11]

- Testing whether more men than women suffer from nightmares
- Establishing authorship of documents
- Evaluating the effect of the full moon on behavior
- Determining the range at which a bat can detect an insect by echo
- Deciding whether hospital carpeting results in more infections
- Selecting the best means to stop smoking
- Checking whether bumper stickers reflect car owner behavior
- Testing the claims of handwriting analysts

Statistical hypothesis testing plays an important role in the whole of statistics and in statistical inference. For example, Lehmann (1992) in a review of the fundamental paper by Neyman and Pearson (1933) says: "Nevertheless, despite their shortcomings, the new paradigm formulated in the 1933 paper, and the many developments carried out within its framework continue to play a central role in both the theory and practice of statistics and can be expected to do so in the foreseeable future".

Significance testing has been the favored statistical tool in some experimental social sciences (over 90% of articles in the Journal of Applied Psychology during the early 1990s).[12] Other fields have favored the estimation of parameters (e.g., effect size). Significance testing is used as a substitute for the traditional comparison of predicted value and experimental result at the core of the scientific method. When theory is only capable of predicting the sign of a relationship, a directional (one-sided) hypothesis test can be configured so that only a statistically significant result supports theory. This form of theory appraisal is the most heavily criticized application of hypothesis testing.

26.2.3 Cautions

"If the government required statistical procedures to carry warning labels like those on drugs, most inference methods would have long labels indeed."[13] This caution applies to hypothesis tests and alternatives to them.

The successful hypothesis test is associated with a probability and a type-I error rate. The conclusion *might* be wrong.

The conclusion of the test is only as solid as the sample upon which it is based. The design of the experiment is critical. A number of unexpected effects have been observed including:

- The Clever Hans effect. A horse appeared to be capable of doing simple arithmetic.
- The Hawthorne effect. Industrial workers were more productive in better illumination, and most productive in worse.
- The Placebo effect. Pills with no medically active ingredients were remarkably effective.

A statistical analysis of misleading data produces misleading conclusions. The issue of data quality can be more subtle. In forecasting for example, there is no agreement on a measure of forecast accuracy. In the absence of a consensus measurement, no decision based on measurements will be without controversy.

The book *How to Lie with Statistics*[14][15] is the most popular book on statistics ever published.[16] It does not much consider hypothesis testing, but its cautions are applicable, including: Many claims are made on the basis of samples too small to convince. If a report does not mention sample size, be doubtful.

Hypothesis testing acts as a filter of statistical conclusions; only those results meeting a probability threshold are publishable. Economics also acts as a publication filter; only those results favorable to the author and funding source may be submitted for publication. The impact of filtering on publication is termed publication bias. A related problem is that of multiple testing (sometimes linked to data mining), in which a variety of tests for a variety of possible effects are applied to a single data set and only those yielding a significant result are reported. These are often dealt with by using multiplicity correction procedures that control the family wise error rate (FWER) or the false discovery rate (FDR).

Those making critical decisions based on the results of a hypothesis test are prudent to look at the details rather than the conclusion alone. In the physical sciences most results are fully accepted only when independently confirmed. The general advice concerning statistics is, “Figures never lie, but liars figure” (anonymous).

26.3 Example

26.3.1 Lady tasting tea

Main article: Lady tasting tea

In a famous example of hypothesis testing, known as the *Lady tasting tea*,[17] a female colleague of Fisher claimed to be able to tell whether the tea or the milk was added first to a cup. Fisher proposed to give her eight cups, four of each variety, in random order. One could then ask what the probability was for her getting the number she got correct, but just by chance. The null hypothesis was that the Lady had no such ability. The test statistic was a simple count of the number of successes in selecting the 4 cups. The critical region was the single case of 4 successes of 4 possible based on a conventional probability criterion ($< 5\%$; 1 of 70 $\approx 1.4\%$). Fisher asserted that no alternative hypothesis was (ever) required. The lady correctly identified every cup,[18] which would be considered a statistically significant result.

26.3.2 Analogy – Courtroom trial

A statistical test procedure is comparable to a criminal trial; a defendant is considered not guilty as long as his or her guilt is not proven. The prosecutor tries to prove the guilt of the defendant. Only when there is enough charging evidence the defendant is convicted.

In the start of the procedure, there are two hypotheses H_0 : “the defendant is not guilty”, and H_1 : “the defendant is guilty”. The first one is called *null hypothesis*, and is for the time being accepted. The second one is called *alternative (hypothesis)*. It is the hypothesis one hopes to support.

The hypothesis of innocence is only rejected when an error is very unlikely, because one doesn't want to convict an innocent defendant. Such an error is called *error of the first kind* (i.e., the conviction of an innocent person), and the occurrence of this error is controlled to be rare. As a consequence of this asymmetric behaviour, the *error of the second kind* (acquitting a person who committed the crime), is often rather large.

A criminal trial can be regarded as either or both of two decision processes: guilty vs not guilty or evidence vs a threshold (“beyond a reasonable doubt”). In one view, the defendant is judged; in the other view the performance of the prosecution (which bears the burden of proof) is judged. A hypothesis test can be regarded as either a judgment of a hypothesis or as a judgment of evidence.

26.3.3 Example 1 – Philosopher’s beans

The following example was produced by a philosopher describing scientific methods generations before hypothesis testing was formalized and popularized.[19]

> Few beans of this handful are white.
> Most beans in this bag are white.
> Therefore: Probably, these beans were taken from another bag.
> This is an hypothetical inference.

The beans in the bag are the population. The handful are the sample. The null hypothesis is that the sample originated from the population. The criterion for rejecting the null-hypothesis is the “obvious” difference in appearance (an informal difference in the mean). The interesting result is that consideration of a real population and a real sample produced an imaginary bag. The philosopher was considering logic rather than probability. To be a real statistical hypothesis test, this example requires the formalities of a probability calculation and a comparison of that probability to a standard.

A simple generalization of the example considers a mixed bag of beans and a handful that contain either very few or very many white beans. The generalization considers both extremes. It requires more calculations and more comparisons to arrive at a formal answer, but the core philosophy is unchanged; If the composition of the handful is greatly different from that of the bag, then the sample probably originated from another bag. The original example is termed a one-sided or a one-tailed test while the generalization is termed a two-sided or two-tailed test.

The statement also relies on the inference that the sampling was random. If someone had been picking through the bag to find white beans, then it would explain why the handful had so many white beans, and also explain why the number of white beans in the bag was depleted (although the bag is probably intended to be assumed much larger than one’s hand).

26.3.4 Example 2 – Clairvoyant card game[20]

A person (the subject) is tested for clairvoyance. He is shown the reverse of a randomly chosen playing card 25 times and asked which of the four suits it belongs to. The number of hits, or correct answers, is called X.

As we try to find evidence of his clairvoyance, for the time being the null hypothesis is that the person is not clairvoyant. The alternative is, of course: the person is (more or less) clairvoyant.

If the null hypothesis is valid, the only thing the test person can do is guess. For every card, the probability (relative frequency) of any single suit appearing is 1/4. If the alternative is valid, the test subject will predict the suit correctly with probability greater than 1/4. We will call the probability of guessing correctly *p*. The hypotheses, then, are:

- null hypothesis : $H_0 : p = \frac{1}{4}$ (just guessing)

and

- alternative hypothesis :$H_1 \, : \, p \neq \frac{1}{4}$ (true clairvoyant).

When the test subject correctly predicts all 25 cards, we will consider him clairvoyant, and reject the null hypothesis. Thus also with 24 or 23 hits. With only 5 or 6 hits, on the other hand, there is no cause to consider him so. But what about 12 hits, or 17 hits? What is the critical number, *c*, of hits, at which point we consider the subject to be clairvoyant? How do we determine the critical value *c*? It is obvious that with the choice *c*=25 (i.e. we only accept clairvoyance when all cards are predicted correctly) we're more critical than with *c*=10. In the first case almost no test subjects will be recognized to be clairvoyant, in the second case, a certain number will pass the test. In practice, one decides how critical one will be. That is, one decides how often one accepts an error of the first kind – a false positive, or Type I error. With *c* = 25 the probability of such an error is:

$$P(\text{ reject}H_0|H_0\text{valid is }) = P(X = 25|p = \tfrac{1}{4}) = \left(\tfrac{1}{4}\right)^{25} \approx 10^{-15},$$

and hence, very small. The probability of a false positive is the probability of randomly guessing correctly all 25 times.

Being less critical, with *c*=10, gives:

$$P(\text{ reject}H_0|H_0\text{valid is }) = P(X \geq 10|p = \tfrac{1}{4}) = \sum_{k=10}^{25} P(X = k|p = \tfrac{1}{4}) \approx 0.07.$$

Thus, *c* = 10 yields a much greater probability of false positive.

Before the test is actually performed, the maximum acceptable probability of a Type I error (α) is determined. Typically, values in the range of 1% to 5% are selected. (If the maximum acceptable error rate is zero, an infinite number of correct guesses is required.) Depending on this Type 1 error rate, the critical value *c* is calculated. For example, if we select an error rate of 1%, *c* is calculated thus:

$$P(\text{ reject}H_0|H_0\text{valid is }) = P(X \geq c|p = \tfrac{1}{4}) \leq 0.01.$$

From all the numbers c, with this property, we choose the smallest, in order to minimize the probability of a Type II error, a false negative. For the above example, we select: $c = 13$.

26.3.5 Example 3 – Radioactive suitcase

As an example, consider determining whether a suitcase contains some radioactive material. Placed under a Geiger counter, it produces 10 counts per minute. The null hypothesis is that no radioactive material is in the suitcase and that all measured counts are due to ambient radioactivity typical of the surrounding air and harmless objects. We can then calculate how likely it is that we would observe 10 counts per minute if the null hypothesis were true. If the null hypothesis predicts (say) on average 9 counts per minute, then according to the Poisson distribution typical for radioactive decay there is about 41% chance of recording 10 or more counts. Thus we can say that the suitcase is compatible with the null hypothesis (this does not guarantee that there is no radioactive material, just that we don't have enough evidence to suggest there is). On the other hand, if the null hypothesis predicts 3 counts per minute (for which the Poisson distribution predicts only 0.1% chance of recording 10 or more counts) then the suitcase is not compatible with the null hypothesis, and there are likely other factors responsible to produce the measurements.

The test does not directly assert the presence of radioactive material. A *successful* test asserts that the claim of no radioactive material present is unlikely given the reading (and therefore ...). The double negative (disproving the null hypothesis) of the method is confusing, but using a counter-example to disprove is standard mathematical practice. The attraction of the method is its practicality. We know (from experience) the expected range of counts with only ambient radioactivity present, so we can say that a measurement is *unusually* large. Statistics just formalizes the intuitive by using numbers instead of adjectives. We probably do not know the characteristics of the radioactive suitcases; We just assume that they produce larger readings.

To slightly formalize intuition: Radioactivity is suspected if the Geiger-count with the suitcase is among or exceeds the greatest (5% or 1%) of the Geiger-counts made with ambient radiation alone. This makes no assumptions about the distribution of counts. Many ambient radiation observations are required to obtain good probability estimates for rare events.

The test described here is more fully the null-hypothesis statistical significance test. The null hypothesis represents what we would believe by default, before seeing any evidence. Statistical significance is a possible finding of the test, declared when the observed sample is unlikely to have occurred by chance if the null hypothesis were true. The name of the test describes its formulation and its possible outcome. One characteristic of the test is its crisp decision: to reject or not reject the null hypothesis. A calculated value is compared to a threshold, which is determined from the tolerable risk of error.

26.4 Definition of terms

The following definitions are mainly based on the exposition in the book by Lehmann and Romano:[5]

Statistical hypothesis A statement about the parameters describing a population (not a sample).

Statistic A value calculated from a sample, often to summarize the sample for comparison purposes.

Simple hypothesis Any hypothesis which specifies the population distribution completely.

Composite hypothesis Any hypothesis which does *not* specify the population distribution completely.

Null hypothesis (H_0) A simple hypothesis associated with a contradiction to a theory one would like to prove.

Alternative hypothesis (H_1) A hypothesis (often composite) associated with a theory one would like to prove.

Statistical test A procedure whose inputs are samples and whose result is a hypothesis.

Region of acceptance The set of values of the test statistic for which we fail to reject the null hypothesis.

Region of rejection / Critical region The set of values of the test statistic for which the null hypothesis is rejected.

Critical value The threshold value delimiting the regions of acceptance and rejection for the test statistic.

Power of a test ($1 - \beta$) The test's probability of correctly rejecting the null hypothesis. The complement of the false negative rate, β. Power is termed **sensitivity** in biostatistics. ("This is a sensitive test. Because the result is negative, we can confidently say that the patient does not have the condition.") See sensitivity and specificity and Type I and type II errors for exhaustive definitions.

Size For simple hypotheses, this is the test's probability of *incorrectly* rejecting the null hypothesis. The false positive rate. For composite hypotheses this is the supremum of the probability of rejecting the null hypothesis over all cases covered by the null hypothesis. The complement of the false positive rate is termed **specificity** in biostatistics. ("This is a specific test. Because the result is positive, we can confidently say that the patient has the condition.") See sensitivity and specificity and Type I and type II errors for exhaustive definitions.

Significance level of a test (α) It is the upper bound imposed on the size of a test. Its value is chosen by the statistician prior to looking at the data or choosing any particular test to be used. It is the maximum exposure to erroneously rejecting H_0 he/she is ready to accept. Testing H_0 at significance level α means testing H_0 with a test whose size does not exceed α. In most cases, one uses tests whose size is equal to the significance level.

p-value The probability, assuming the null hypothesis is true, of observing a result at least as extreme as the test statistic.

Statistical significance test A predecessor to the statistical hypothesis test (see the Origins section). An experimental result was said to be statistically significant if a sample was sufficiently inconsistent with the (null) hypothesis. This was variously considered common sense, a pragmatic heuristic for identifying meaningful experimental results, a convention establishing a threshold of statistical evidence or a method for drawing conclusions from data. The statistical hypothesis test added mathematical rigor and philosophical consistency to the concept by making the alternative hypothesis explicit. The term is loosely used to describe the modern version which is now part of statistical hypothesis testing.

Conservative test A test is conservative if, when constructed for a given nominal significance level, the true probability of *incorrectly* rejecting the null hypothesis is never greater than the nominal level.

Exact test A test in which the significance level or critical value can be computed exactly, i.e., without any approximation. In some contexts this term is restricted to tests applied to categorical data and to permutation tests, in which computations are carried out by complete enumeration of all possible outcomes and their probabilities.

A statistical hypothesis test compares a test statistic (z or t for examples) to a threshold. The test statistic (the formula found in the table below) is based on optimality. For a fixed level of Type I error rate, use of these statistics minimizes Type II error rates (equivalent to maximizing power). The following terms describe tests in terms of such optimality:

Most powerful test For a given *size* or *significance level*, the test with the greatest power (probability of rejection) for a given value of the parameter(s) being tested, contained in the alternative hypothesis.

Uniformly most powerful test (UMP) A test with the greatest *power* for all values of the parameter(s) being tested, contained in the alternative hypothesis.

26.5 Common test statistics

Main article: Test statistic

One-sample tests are appropriate when a sample is being compared to the population from a hypothesis. The population characteristics are known from theory or are calculated from the population.

Two-sample tests are appropriate for comparing two samples, typically experimental and control samples from a scientifically controlled experiment.

Paired tests are appropriate for comparing two samples where it is impossible to control important variables. Rather than comparing two sets, members are paired between samples so the difference between the members becomes the sample. Typically the mean of the differences is then compared to zero. The common example scenario for when a paired difference test is appropriate is when a single set of test subjects has something applied to them and the test is intended to check for an effect.

Z-tests are appropriate for comparing means under stringent conditions regarding normality and a known standard deviation.

A *t*-test is appropriate for comparing means under relaxed conditions (less is assumed).

Tests of proportions are analogous to tests of means (the 50% proportion).

Chi-squared tests use the same calculations and the same probability distribution for different applications:

- Chi-squared tests for variance are used to determine whether a normal population has a specified variance. The null hypothesis is that it does.
- Chi-squared tests of independence are used for deciding whether two variables are associated or are independent. The variables are categorical rather than numeric. It can be used to decide whether left-handedness is correlated with libertarian politics (or not). The null hypothesis is that the variables are independent. The numbers used in the calculation are the observed and expected frequencies of occurrence (from contingency tables).
- Chi-squared goodness of fit tests are used to determine the adequacy of curves fit to data. The null hypothesis is that the curve fit is adequate. It is common to determine curve shapes to minimize the mean square error, so it is appropriate that the goodness-of-fit calculation sums the squared errors.

F-tests (analysis of variance, ANOVA) are commonly used when deciding whether groupings of data by category are meaningful. If the variance of test scores of the left-handed in a class is much smaller than the variance of the whole class, then it may be useful to study lefties as a group. The null hypothesis is that two variances are the same – so the proposed grouping is not meaningful.

In the table below, the symbols used are defined at the bottom of the table. Many other tests can be found in other articles. Proofs exist that the test statistics are appropriate.[21]

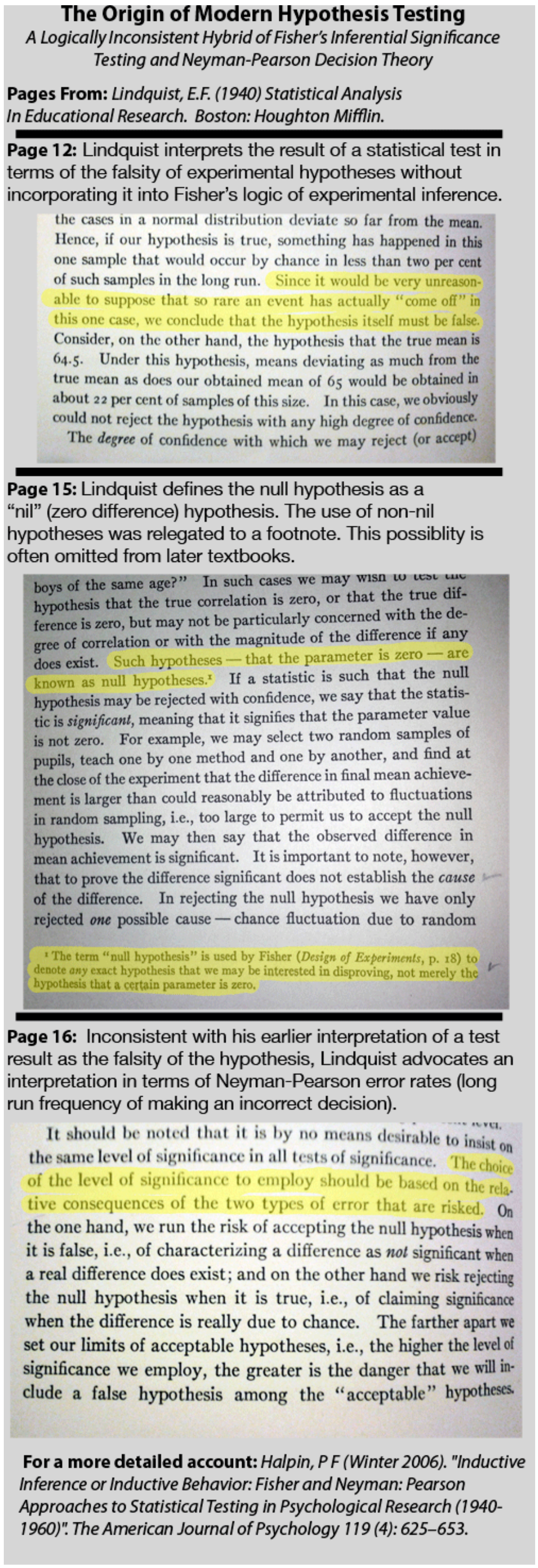
26.6 Origins and early controversy

Significance testing is largely the product of Karl Pearson (p-value, Pearson's chi-squared test), William Sealy Gosset (Student's t-distribution), and Ronald Fisher ("null hypothesis", analysis of variance, "significance test"), while hypothesis testing was developed by Jerzy Neyman and Egon Pearson (son of Karl). Ronald Fisher, mathematician and biologist described by Richard Dawkins as "the greatest biologist since Darwin", began his life in statistics as a Bayesian (Zabell 1992), but Fisher soon grew disenchanted with the subjectivity involved (namely use of the principle of indifference when determining prior probabilities), and sought to provide a more "objective" approach to inductive inference.[27]

Fisher was an agricultural statistician who emphasized rigorous experimental design and methods to extract a result from few samples assuming Gaussian distributions. Neyman (who teamed with the younger Pearson) emphasized mathematical rigor and methods to obtain more results from many samples and a wider range of distributions. Modern hypothesis testing is an inconsistent hybrid of the Fisher vs Neyman/Pearson formulation, methods and terminology developed in the early 20th century. While hypothesis testing was popularized early in the 20th century, evidence of its use can be found much earlier. In the 1770s Laplace considered the statistics of almost half a million births. The statistics showed an excess of boys compared to girls.[28] He concluded by calculation of a p-value that the excess was a real, but unexplained, effect.[29]

Fisher popularized the "significance test". He required a null-hypothesis (corresponding to a population frequency distribution) and a sample. His (now familiar) calculations determined whether to reject the null-hypothesis or not. Significance testing did not utilize an alternative hypothesis so there was no concept of a Type II error.

The p-value was devised as an informal, but objective,



The Origin of Modern Hypothesis Testing

A Logically Inconsistent Hybrid of Fisher's Inferential Significance Testing and Neyman-Pearson Decision Theory

Pages From: *Lindquist, E.F. (1940) Statistical Analysis In Educational Research. Boston: Houghton Mifflin.*

Page 12: Lindquist interprets the result of a statistical test in terms of the falsity of experimental hypotheses without incorporating it into Fisher's logic of experimental inference.

the cases in a normal distribution deviate so far from the mean. Hence, if our hypothesis is true, something has happened in this one sample that would occur by chance in less than two per cent of such samples in the long run. Since it would be very unreasonable to suppose that so rare an event has actually "come off" in this one case, we conclude that the hypothesis itself must be false. Consider, on the other hand, the hypothesis that the true mean is 64.5. Under this hypothesis, means deviating as much from the true mean as does our obtained mean of 65 would be obtained in about 22 per cent of samples of this size. In this case, we obviously could not reject the hypothesis with any high degree of confidence.
The *degree* of confidence with which we may reject (or accept)

Page 15: Lindquist defines the null hypothesis as a "nil" (zero difference) hypothesis. The use of non-nil hypotheses was relegated to a footnote. This possiblity is often omitted from later textbooks.

boys of the same age?" In such cases we may wish to test the hypothesis that the true correlation is zero, or that the true difference is zero, but may not be particularly concerned with the degree of correlation or with the magnitude of the difference if any does exist. Such hypotheses — that the parameter is zero — are known as null hypotheses.[1] If a statistic is such that the null hypothesis may be rejected with confidence, we say that the statistic is *significant*, meaning that it signifies that the parameter value is not zero. For example, we may select two random samples of pupils, teach one by one method and one by another, and find at the close of the experiment that the difference in final mean achievement is larger than could reasonably be attributed to fluctuations in random sampling, i.e., too large to permit us to accept the null hypothesis. We may then say that the observed difference in mean achievement is significant. It is important to note, however, that to prove the difference significant does not establish the *cause* of the difference. In rejecting the null hypothesis we have only rejected *one* possible cause — chance fluctuation due to random

[1] The term "null hypothesis" is used by Fisher (*Design of Experiments*, p. 18) to denote *any* exact hypothesis that we may be interested in disproving, not merely the hypothesis that a certain parameter is zero.

Page 16: Inconsistent with his earlier interpretation of a test result as the falsity of the hypothesis, Lindquist advocates an interpretation in terms of Neyman-Pearson error rates (long run frequency of making an incorrect decision).

It should be noted that it is by no means desirable to insist on the same level of significance in all tests of significance. The choice of the level of significance to employ should be based on the relative consequences of the two types of error that are risked. On the one hand, we run the risk of accepting the null hypothesis when it is false, i.e., of characterizing a difference as *not* significant when a real difference does exist; and on the other hand we risk rejecting the null hypothesis when it is true, i.e., of claiming significance when the difference is really due to chance. The farther apart we set our limits of acceptable hypotheses, i.e., the higher the level of significance we employ, the greater is the danger that we will include a false hypothesis among the "acceptable" hypotheses.

For a more detailed account: *Halpin, P F (Winter 2006). "Inductive Inference or Inductive Behavior: Fisher and Neyman: Pearson Approaches to Statistical Testing in Psychological Research (1940-1960)". The American Journal of Psychology 119 (4): 625–653.*

A likely originator of the "hybrid" method of hypothesis testing, as well as the use of "nil" null hypotheses, is E.F. Lindquist in his statistics textbook: Lindquist, E.F. (1940) Statistical Analysis In Educational Research. Boston: Houghton Mifflin.

index meant to help a researcher determine (based on other knowledge) whether to modify future experiments or strengthen one's faith in the null hypothesis.[30] Hypothesis testing (and Type I/II errors) was devised by Neyman and Pearson as a more objective alternative to Fisher's p-value, also meant to determine researcher behaviour, but without requiring any inductive inference by the researcher.[31][32]

Neyman & Pearson considered a different problem (which they called "hypothesis testing"). They initially considered two simple hypotheses (both with frequency distributions). They calculated two probabilities and typically selected the hypothesis associated with the higher probability (the hypothesis more likely to have generated the sample). Their method always selected a hypothesis. It also allowed the calculation of both types of error probabilities.

Fisher and Neyman/Pearson clashed bitterly. Neyman/Pearson considered their formulation to be an improved generalization of significance testing.(The defining paper[31] was abstract. Mathematicians have generalized and refined the theory for decades.[33]) Fisher thought that it was not applicable to scientific research because often, during the course of the experiment, it is discovered that the initial assumptions about the null hypothesis are questionable due to unexpected sources of error. He believed that the use of rigid reject/accept decisions based on models formulated before data is collected was incompatible with this common scenario faced by scientists and attempts to apply this method to scientific research would lead to mass confusion.[34]

The dispute between Fisher and Neyman-Pearson was waged on philosophical grounds, characterized by a philosopher as a dispute over the proper role of models in statistical inference.[35]

Events intervened: Neyman accepted a position in the western hemisphere, breaking his partnership with Pearson and separating disputants (who had occupied the same building) by much of the planetary diameter. World War II provided an intermission in the debate. The dispute between Fisher and Neyman terminated (unresolved after 27 years) with Fisher's death in 1962. Neyman wrote a well-regarded eulogy.[36] Some of Neyman's later publications reported p-values and significance levels.[37]

The modern version of hypothesis testing is a hybrid of the two approaches that resulted from confusion by writers of statistical textbooks (as predicted by Fisher) beginning in the 1940s.[38] (But signal detection, for example, still uses the Neyman/Pearson formulation.) Great conceptual differences and many caveats in addition to those mentioned above were ignored. Neyman and Pearson provided the stronger terminology, the more rigorous mathematics and the more consistent philosophy, but the subject taught today in introductory statistics has more similarities with Fisher's method than theirs.[39] This history explains the inconsistent terminology (example: the null hypothesis is never accepted, but there is a region of acceptance).

Sometime around 1940,[38] in an apparent effort to provide researchers with a "non-controversial"[40] way to have their cake and eat it too, the authors of statistical text books began anonymously combining these two strategies by using the p-value in place of the test statistic (or data) to test against the Neyman-Pearson "significance level".[38] Thus, researchers were encouraged to infer the strength of their data against some null hypothesis using p-values, while also thinking they are retaining the post-data collection objectivity provided by hypothesis testing. It then became customary for the null hypothesis, which was originally some realistic research hypothesis, to be used almost solely as a strawman "nil" hypothesis (one where a treatment has no effect, regardless of the context).[41]

A comparison between Fisherian, frequentist (Neyman-Pearson)

26.6.1 Early choices of null hypothesis

Paul Meehl has argued that the epistemological importance of the choice of null hypothesis has gone largely unacknowledged. When the null hypothesis is predicted by theory, a more precise experiment will be a more severe test of the underlying theory. When the null hypothesis defaults to "no difference" or "no effect", a more precise experiment is a less severe test of the theory that motivated performing the experiment.[42] An examination of the origins of the latter practice may therefore be useful:

1778: Pierre Laplace compares the birthrates of boys and girls in multiple European cities. He states: "it is natural to conclude that these possibilities are very nearly in the same ratio". Thus Laplace's null hypothesis that the birthrates of boys and girls should be equal given "conventional wisdom".[28]

1900: Karl Pearson develops the chi squared test to determine "whether a given form of frequency curve will effectively describe the samples drawn from a given population." Thus the null hypothesis is that a population is described by some distribution predicted by theory. He uses as an example the numbers of five and sixes in the Weldon dice throw data.[43]

1904: Karl Pearson develops the concept of "contingency" in order to determine whether outcomes are independent of a given categorical factor. Here the null hypothesis is by default that two things are unrelated (e.g. scar formation and death rates from smallpox).[44] The null hypothesis in this case is no longer predicted by theory or conventional wisdom, but is instead the principle of indifference that lead Fisher and others to dismiss the use of "inverse probabilities".[45]

26.7 Null hypothesis statistical significance testing vs hypothesis testing

An example of Neyman-Pearson hypothesis testing can be made by a change to the radioactive suitcase example. If the "suitcase" is actually a shielded container for the transportation of radioactive material, then a test might be used to select among three hypotheses: no radioactive source present, one present, two (all) present. The test could be required for safety, with actions required in each case. The Neyman-Pearson lemma of hypothesis testing says that a good criterion for the selection of hypotheses is the ratio of their probabilities (a likelihood ratio). A simple method of solution is to select the hypothesis with the highest probability for the Geiger counts observed. The typical result matches intuition: few counts imply no source, many counts imply two sources and intermediate counts imply one source.

Neyman-Pearson theory can accommodate both prior probabilities and the costs of actions resulting from decisions.[46] The former allows each test to consider the results of earlier tests (unlike Fisher's significance tests). The latter allows the consideration of economic issues (for example) as well as probabilities. A likelihood ratio remains a good criterion for selecting among hypotheses.

The two forms of hypothesis testing are based on different problem formulations. The original test is analogous to a true/false question; the Neyman-Pearson test is more like multiple choice. In the view of Tukey[47] the former produces a conclusion on the basis of only strong evidence while the latter produces a decision on the basis of available evidence. While the two tests seem quite different both mathematically and philosophically, later developments lead to the opposite claim. Consider many tiny radioactive sources. The hypotheses become 0,1,2,3... grains of radioactive sand. There is little distinction between none or some radiation (Fisher) and 0 grains of radioactive sand versus all of the alternatives (Neyman-Pearson). The major Neyman-Pearson paper of 1933[31] also considered composite hypotheses (ones whose distribution includes an unknown parameter). An example proved the optimality of the (Student's) t-test, "there can be no better test for the hypothesis under consideration" (p 321). Neyman-Pearson theory was proving the optimality of Fisherian methods from its inception.

Fisher's significance testing has proven a popular flexible statistical tool in application with little mathematical growth potential. Neyman-Pearson hypothesis testing is claimed as a pillar of mathematical statistics,[48] creating a new paradigm for the field. It also stimulated new applications in Statistical process control, detection theory, decision theory and game theory. Both formulations have been successful, but the successes have been of a different character.

The dispute over formulations is unresolved. Science primarily uses Fisher's (slightly modified) formulation as taught in introductory statistics. Statisticians study Neyman-Pearson theory in graduate school. Mathematicians are proud of uniting the formulations. Philosophers consider them separately. Learned opinions deem the formulations variously competitive (Fisher vs Neyman), incompatible[27] or complementary.[33] The dispute has become more complex since Bayesian inference has achieved respectability.

The terminology is inconsistent. Hypothesis testing can mean any mixture of two formulations that both changed with time. Any discussion of significance testing vs hypothesis testing is doubly vulnerable to confusion.

Fisher thought that hypothesis testing was a useful strategy for performing industrial quality control, however, he strongly disagreed that hypothesis testing could be useful for scientists.[30] Hypothesis testing provides a means of finding test statistics used in significance testing.[33] The concept of power is useful in explaining the consequences of adjusting the significance level and is heavily used in sample size determination. The two methods remain philosophically distinct.[35] They usually (but *not always*) produce the same mathematical answer. The preferred answer is context dependent.[33] While the existing merger of Fisher and Neyman-Pearson theories has been heavily criticized, modifying the merger to achieve Bayesian goals has been considered.[49]

26.8 Criticism

See also: p-value § Criticisms

Criticism of statistical hypothesis testing fills volumes[50][51][52][53][54][55] citing 300–400 primary references. Much of the criticism can be summarized by the following issues:

- The interpretation of a *p*-value is dependent upon stopping rule and definition of multiple comparison. The former often changes during the course of a study and the latter is unavoidably ambiguous. (i.e. “p values depend on both the (data) observed and on the other possible (data) that might have been observed but weren't”).[56]
- Confusion resulting (in part) from combining the methods of Fisher and Neyman-Pearson which are conceptually distinct.[47]
- Emphasis on statistical significance to the exclusion of estimation and confirmation by repeated experiments.[57]
- Rigidly requiring statistical significance as a criterion for publication, resulting in publication bias.[58]

  Most of the criticism is indirect. Rather than being wrong, statistical hypothesis testing is misunderstood, overused and misused.

- When used to detect whether a difference exists between groups, a paradox arises. As improvements are made to experimental design (e.g., increased precision of measurement and sample size), the test becomes more lenient. Unless one accepts the absurd assumption that all sources of noise in the data cancel out completely, the chance of finding statistical significance in either direction approaches 100%.[59]
- Layers of philosophical concerns. The probability of statistical significance is a function of decisions made by experimenters/analysts.[10] If the decisions are based on convention they are termed arbitrary or mindless[40] while those not so based may be termed subjective. To minimize type II errors, large samples are recommended. In psychology practically all null hypotheses are claimed to be false for sufficiently large samples so "...it is usually nonsensical to perform an experiment with the *sole* aim of rejecting the null hypothesis.".[60] “Statistically significant findings are often misleading” in psychology.[61] Statistical significance does not imply practical significance and correlation does not imply causation. Casting doubt on the null hypothesis is thus far from directly supporting the research hypothesis.
- "[I]t does not tell us what we want to know”.[62] Lists of dozens of complaints are available.[54][63]

Critics and supporters are largely in factual agreement regarding the characteristics of null hypothesis significance testing (NHST): While it can provide critical information, it is *inadequate as the sole tool for statistical analysis. Successfully rejecting the null hypothesis may offer no support for the research hypothesis.* The continuing controversy concerns the selection of the best statistical practices for the near-term future given the (often poor) existing practices. Critics would prefer to ban NHST completely, forcing a complete departure from those practices, while supporters suggest a less absolute change.

Controversy over significance testing, and its effects on publication bias in particular, has produced several results. The American Psychological Association has strengthened its statistical reporting requirements after review,[64] medical journal publishers have recognized the obligation to publish some results that are not statistically significant to combat publication bias[65] and a journal (*Journal of Articles in Support of the Null Hypothesis*) has been created to publish such results exclusively.[66] Textbooks have added some cautions[67] and increased coverage of the tools necessary to estimate the size of the sample required to produce significant results. Major organizations have not abandoned use of significance tests although some have discussed doing so.[64]

26.9 Alternatives

Main article: Estimation statistics
See also: Confidence interval § Statistical hypothesis testing

The numerous criticisms of significance testing do not lead to a single alternative. A unifying position of critics is that statistics should not lead to a conclusion or a decision but to a probability or to an estimated value with a confidence interval rather than to an accept-reject decision regarding a particular hypothesis. It is unlikely that the controversy surrounding significance testing will be resolved in the near future. Its supposed flaws and unpopularity do not eliminate the need for an objective and transparent means of reaching conclusions regarding studies that produce statistical results. Critics have not unified around an alternative. Other forms of reporting confidence or uncertainty could probably grow in popularity. One strong critic of significance testing suggested a list of reporting alternatives:[68] effect sizes for importance, prediction intervals for confidence, replications and extensions for replicability, meta-analyses for generality. None of these suggested alternatives produces a conclusion/decision. Lehmann said that hypothesis testing theory can be presented in terms of conclusions/decisions, probabilities, or confidence intervals. “The distinction between the ... approaches is largely one of reporting and interpretation.”[69]

On one “alternative” there is no disagreement: Fisher himself said,[17] “In relation to the test of significance, we may say that a phenomenon is experimentally demonstrable when we know how to conduct an experiment which will rarely fail to give us a statistically significant result.” Cohen, an influential critic of significance testing, concurred,[62] "... don't look for a magic alternative to NHST *[null hypothesis significance testing]* ... It doesn't exist.” "... given the problems of statistical induction, we must finally rely, as have the older sciences, on replication.” The “alternative” to significance testing is repeated testing. The easiest way to decrease statistical uncertainty is by obtaining more data, whether by increased sample size or by repeated tests. Nickerson claimed to have never seen the publication of a literally replicated experiment in psychology.[63] An indirect approach to replication is meta-analysis.

Bayesian inference is one proposed alternative to significance testing. (Nickerson cited 10 sources suggesting it, including Rozeboom (1960)).[63] For example, Bayesian parameter estimation can provide rich information about the data from which researchers can draw inferences, while using uncertain priors that exert only minimal influence on the results when enough data is available. Psychologist Kruschke, John K. has suggested Bayesian estimation as an alternative for the *t*-test.[70] Alternatively two competing models/hypothesis can be compared using Bayes factors.[71] Bayesian methods could be criticized for requiring information that is seldom available in the cases where significance testing is most heavily used. Neither the prior probabilities nor the probability distribution of the test statistic under the alternative hypothesis are often available in the social sciences.[63]

Advocates of a Bayesian approach sometimes claim that the goal of a researcher is most often to objectively assess the probability that a hypothesis is true based on the data they have collected.[72][73] Neither Fisher's significance testing, nor Neyman-Pearson hypothesis testing can provide this information, and do not claim to. The probability a hypothesis is true can only be derived from use of Bayes’ Theorem, which was unsatisfactory to both the Fisher and Neyman-Pearson camps due to the explicit use of subjectivity in the form of the prior probability.[31][74] Fisher’s strategy is to sidestep this with the p-value (an objective *index* based on the data alone) followed by *inductive inference*, while Neyman-Pearson devised their approach of *inductive behaviour*.

26.10 Philosophy

Hypothesis testing and philosophy intersect. Inferential statistics, which includes hypothesis testing, is applied probability. Both probability and its application are intertwined with philosophy. Philosopher David Hume wrote, “All knowledge degenerates into probability.” Competing practical definitions of probability reflect philosophical differences. The most common application of hypothesis testing is in the scientific interpretation of experimental data, which is naturally studied by the philosophy of science.

Fisher and Neyman opposed the subjectivity of probability. Their views contributed to the objective definitions. The core of their historical disagreement was philosophical.

Many of the philosophical criticisms of hypothesis testing are discussed by statisticians in other contexts, particularly correlation does not imply causation and the design of experiments. Hypothesis testing is of continuing interest to philosophers.[35][75]

26.11 Education

Main article: Statistics education

Statistics is increasingly being taught in schools with hypothesis testing being one of the elements taught.[76][77] Many conclusions reported in the popular press (political opinion polls to medical studies) are based on statistics. An informed public should understand the limitations of statistical conclusions[78][79] and many college fields of study require a course in statistics for the same reason.[78][79] An introductory college statistics class places much emphasis on hypothesis testing – perhaps half of the course. Such fields as literature and divinity now include findings based on statistical analysis (see the Bible Analyzer). An introductory statistics class teaches hypothesis testing as a cookbook process. Hypothesis testing is also taught at the postgraduate level. Statisticians learn how to create good statistical test procedures (like z, Student's t, F and chi-squared). Statistical hypothesis testing is considered a mature area within statistics,[69] but a limited amount of development continues.

The cookbook method of teaching introductory statistics leaves no time for history, philosophy or controversy. Hypothesis testing has been taught as received unified method. Surveys showed that graduates of the class were filled with philosophical misconceptions (on all aspects of statistical inference) that persisted among instructors.[80] While the problem was addressed more than a decade ago,[81] and calls for educational reform continue,[82] students still graduate from statistics classes holding fundamental misconceptions about hypothesis testing.[83] Ideas for improving the teaching of hypothesis testing include encouraging students to search for statistical errors in published papers, teaching the history of statistics and emphasizing the controversy in a generally dry subject.[84]

26.12 See also

- Behrens–Fisher problem
- Bootstrapping (statistics)
- Checking if a coin is fair
- Comparing means test decision tree
- Complete spatial randomness
- Counternull
- Falsifiability
- Fisher's method for combining independent tests of significance
- Granger causality
- Look-elsewhere effect
- Modifiable areal unit problem
- Omnibus test

26.13 References

[1] Stuart A., Ord K., Arnold S. (1999), *Kendall's Advanced Theory of Statistics: Volume 2A—Classical Inference & the Linear Model* (Arnold) §20.2.

[2] Burnham, K. P.; Anderson, D. R. (2002), *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach* (2nd ed.), Springer-Verlag, ISBN 0-387-95364-7.

[3] Schervish, M (1996) *Theory of Statistics*, p. 218. Springer ISBN 0-387-94546-6

[4] Kaye, David H.; Freedman, David A. (2011). "Reference Guide on Statistics". *Reference Manual on Scientific Evidence* (3rd ed.). Eagan, MN Washington, D.C: West National Academies Press. p. 259. ISBN 978-0-309-21421-6.

[5] Lehmann, E.L.; Romano, Joseph P. (2005). *Testing Statistical Hypotheses* (3E ed.). New York: Springer. ISBN 0-387-98864-5.

[6] Triola, Mario (2001). *Elementary statistics* (8 ed.). Boston: Addison-Wesley. p. 388. ISBN 0-201-61477-4.

[7] Hinkelmann, Klaus and Kempthorne, Oscar (2008). *Design and Analysis of Experiments*. I and II (Second ed.). Wiley. ISBN 978-0-470-38551-7.

[8] Montgomery, Douglas (2009). *Design and analysis of experiments*. Hoboken, NJ: Wiley. ISBN 978-0-470-12866-4.

[9] R. A. Fisher (1925).*Statistical Methods for Research Workers*, Edinburgh: Oliver and Boyd, 1925, p.43.

[10] Bakan, David (1966). "The test of significance in psychological research". *Psychological Bulletin* **66** (6): 423–437. doi:10.1037/h0020412.

[11] Richard J. Larsen, Donna Fox Stroup (1976). *Statistics in the Real World: a book of examples*. Macmillan. ISBN 978-0023677205.

[12] Hubbard, R.; Parsa, A. R.; Luthy, M. R. (1997). "The Spread of Statistical Significance Testing in Psychology: The Case of the Journal of Applied Psychology". *Theory and Psychology* **7** (4): 545–554. doi:10.1177/0959354397074006.

[13] Moore, David (2003). *Introduction to the Practice of Statistics*. New York: W.H. Freeman and Co. p. 426. ISBN 9780716796572.

[14] Huff, Darrell (1993). *How to lie with statistics*. New York: Norton. ISBN 0-393-31072-8.

[15] Huff, Darrell (1991). *How to Lie with Statistics*. London: Penguin Books. ISBN 0-14-013629-0.

[16] "Over the last fifty years, How to Lie with Statistics has sold more copies than any other statistical text." J. M. Steele. "Darrell Huff and Fifty Years of *How to Lie with Statistics*. *Statistical Science*, 20 (3), 2005, 205–209.

[17] Fisher, Sir Ronald A. (1956) [1935]. "Mathematics of a Lady Tasting Tea". In James Roy Newman. *The World of Mathematics, volume 3* [*Design of Experiments*]. Courier Dover Publications. ISBN 978-0-486-41151-4. Originally from Fisher's book *Design of Experiments*.

[18] Box, Joan Fisher (1978). *R.A. Fisher, The Life of a Scientist*. New York: Wiley. p. 134. ISBN 0-471-09300-9.

[19] C. S. Peirce (August 1878). "Illustrations of the Logic of Science VI: Deduction, Induction, and Hypothesis". *Popular Science Monthly* **13**. Retrieved 30 March 2012.

[20] Jaynes, E.T. (2007). *Probability theory : the logic of science* (5. print. ed.). Cambridge [u.a.]: Cambridge Univ. Press. ISBN 978-0-521-59271-0.

[21] Loveland, Jennifer L. (2011). *Mathematical Justification of Introductory Hypothesis Tests and Development of Reference Materials* (M.Sc. (Mathematics)). Utah State University. Retrieved April 2013. Abstract: "The focus was on the Neyman-Pearson approach to hypothesis testing. A brief historical development of the Neyman-Pearson approach is followed by mathematical proofs of each of the hypothesis tests covered in the reference material." The proofs do not reference the concepts introduced by Neyman and Pearson, instead they show that traditional test statistics have the probability distributions ascribed to them, so that significance calculations assuming those distributions are correct. The thesis information is also posted at mathnstats.com as of April 2013.

[22] NIST handbook: Two-Sample *t*-test for Equal Means

[23] Steel, R.G.D, and Torrie, J. H., *Principles and Procedures of Statistics with Special Reference to the Biological Sciences.*, McGraw Hill, 1960, page 350.

[24] Weiss, Neil A. (1999). *Introductory Statistics* (5th ed.). p. 802. ISBN 0-201-59877-9.

[25] NIST handbook: F-Test for Equality of Two Standard Deviations (Testing standard deviations the same as testing variances)

[26] Steel, R.G.D, and Torrie, J. H., *Principles and Procedures of Statistics with Special Reference to the Biological Sciences.*, McGraw Hill, 1960, page 288.)

[27] Raymond Hubbard, M.J. Bayarri, *P Values are not Error Probabilities*. A working paper that explains the difference between Fisher's evidential p-value and the Neyman–Pearson Type I error rate α .

[28] Laplace, P (1778). "Memoire Sur Les Probabilities" (PDF). *Memoirs de l'Academie royale des Sciences de Paris* **9**: 227–332.

[29] Stigler, Stephen M. (1986). *The History of Statistics: The Measurement of Uncertainty before 1900*. Cambridge, Mass: Belknap Press of Harvard University Press. p. 134. ISBN 0-674-40340-1.

[30] Fisher, R (1955). "Statistical Methods and Scientific Induction" (PDF). *Journal of the Royal Statistical Society, Series B* **17** (1): 69–78.

[31] Neyman, J; Pearson, E. S. (January 1, 1933). "On the Problem of the most Efficient Tests of Statistical Hypotheses". *Philosophical Transactions of the Royal Society A* **231** (694–706): 289–337. doi:10.1098/rsta.1933.0009.

[32] Goodman, S N (June 15, 1999). "Toward evidence-based medical statistics. 1: The P Value Fallacy". *Ann Intern Med* **130** (12): 995–1004. doi:10.7326/0003-4819-130-12-199906150-00008. PMID 10383371.

[33] Lehmann, E. L. (December 1993). "The Fisher, Neyman-Pearson Theories of Testing Hypotheses: One Theory or Two?". *Journal of the American Statistical Association* **88** (424): 1242–1249. doi:10.1080/01621459.1993.10476404.

[34] Fisher, R N (1958). "The Nature of Probability" (PDF). *Centennial Review* **2**: 261–274."We are quite in danger of sending highly trained and highly intelligent young men out into the world with tables of erroneous numbers under their arms, and with a dense fog in the place where their brains ought to be. In this century, of course, they will be working on guided missiles and advising the medical profession on the control of disease, and there is no limit to the extent to which they could impede every sort of national effort."

[35] Lenhard, Johannes (2006). "Models and Statistical Inference: The Controversy between Fisher and Neyman–Pearson". *Brit. J. Phil. Sci.* **57**: 69–91. doi:10.1093/bjps/axi152.

[36] Neyman, Jerzy (1967). "RA Fisher (1890—1962): An Appreciation.". *Science*. 156.3781: 1456–1460. doi:10.1126/science.156.3781.1456.

[37] Losavich, J. L.; Neyman, J.; Scott, E. L.; Wells, M. A. (1971). "Hypothetical explanations of the negative apparent effects of cloud seeding in the Whitetop Experiment.". *Proceedings of the U.S. National Academy of Sciences* **68**: 2643–2646. doi:10.1073/pnas.68.11.2643.

[38] Halpin, P F; Stam, HJ (Winter 2006). "Inductive Inference or Inductive Behavior: Fisher and Neyman: Pearson Approaches to Statistical Testing in Psychological Research (1940–1960)". *The American Journal of Psychology* **119** (4): 625–653. doi:10.2307/20445367. JSTOR 20445367. PMID 17286092.

[39] Gigerenzer, Gerd; Zeno Swijtink; Theodore Porter; Lorraine Daston; John Beatty; Lorenz Kruger (1989). "Part 3: The Inference Experts". *The Empire of Chance: How Probability Changed Science and Everyday Life*. Cambridge University Press. pp. 70–122. ISBN 978-0-521-39838-1.

[40] Gigerenzer, G (November 2004). "Mindless statistics". *The Journal of Socio-Economics* **33** (5): 587–606. doi:10.1016/j.socec.2004.09.033.

[41] Loftus, G R (1991). "On the Tyranny of Hypothesis Testing in the Social Sciences" (PDF). *Contemporary Psychology* **36** (2): 102–105. doi:10.1037/029395.

[42] Meehl, P (1990). "Appraising and Amending Theories: The Strategy of Lakatosian Defense and Two Principles That Warrant It" (PDF). *Psychological Inquiry* **1** (2): 108–141. doi:10.1207/s15327965pli0102_1.

[43] Pearson, K (1900). "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling"

(PDF). *Philosophical Magazine Series* **5** (50): 157–175. doi:10.1080/14786440009463897.

[44] Pearson, K (1904). "On the Theory of Contingency and Its Relation to Association and Normal Correlation" (PDF). *Drapers' Company Research Memoirs Biometric Series* **1**: 1–35.

[45] Zabell, S (1989). "R. A. Fisher on the History of Inverse Probability". *Statistical Science* **4** (3): 247–256. doi:10.1214/ss/1177012488. JSTOR 2245634.

[46] Ash, Robert (1970). *Basic probability theory*. New York: Wiley. ISBN 978-0471034506.Section 8.2

[47] Tukey, John W. (1960). "Conclusions vs decisions". *Technometrics* **26** (4): 423–433. doi:10.1080/00401706.1960.10489909. "Until we go through the accounts of testing hypotheses, separating [Neyman-Pearson] decision elements from [Fisher] conclusion elements, the intimate mixture of disparate elements will be a continual source of confusion." ... "There is a place for both "doing one's best" and "saying only what is certain," but it is important to know, in each instance, both which one is being done, and which one ought to be done."

[48] Stigler, Stephen M. (Aug 1996). "The History of Statistics in 1933". *Statistical Science* **11** (3): 244–252. doi:10.1214/ss/1032280216. JSTOR 2246117.

[49] Berger, James O. (2003). "Could Fisher, Jeffreys and Neyman Have Agreed on Testing?". *Statistical Science* **18** (1): 1–32. doi:10.1214/ss/1056397485.

[50] Morrison, Denton; Henkel, Ramon, ed. (2006) [1970]. *The Significance Test Controversy*. AldineTransaction. ISBN 0-202-30879-0.

[51] Oakes, Michael (1986). *Statistical Inference: A Commentary for the Social and Behavioural Sciences*. Chichester New York: Wiley. ISBN 0471104434.

[52] Chow, Siu L. (1997). *Statistical Significance: Rationale, Validity and Utility*. ISBN 0-7619-5205-5.

[53] Harlow, Lisa Lavoie; Stanley A. Mulaik; James H. Steiger, ed. (1997). *What If There Were No Significance Tests?*. Lawrence Erlbaum Associates. ISBN 978-0-8058-2634-0.

[54] Kline, Rex (2004). *Beyond Significance Testing: Reforming Data Analysis Methods in Behavioral Research*. Washington, DC: American Psychological Association. ISBN 9781591471189.

[55] McCloskey, Deirdre N.; Stephen T. Ziliak (2008). *The Cult of Statistical Significance: How the Standard Error Costs Us Jobs, Justice, and Lives*. University of Michigan Press. ISBN 0-472-05007-9.

[56] Cornfield, Jerome (1976). "Recent Methodological Contributions to Clinical Trials" (PDF). *American Journal of Epidemiology* **104** (4): 408–421.

[57] Yates, Frank (1951). "The Influence of Statistical Methods for Research Workers on the Development of the Science of Statistics". *Journal of the American Statistical Association* **46**: 19–34. doi:10.1080/01621459.1951.10500764. "The emphasis given to formal tests of significance throughout [R.A. Fisher's] Statistical Methods ... has caused scientific research workers to pay undue attention to the results of the tests of significance they perform on their data, particularly data derived from experiments, and too little to the estimates of the magnitude of the effects they are investigating." ... "The emphasis on tests of significance and the consideration of the results of each experiment in isolation, have had the unfortunate consequence that scientific workers have often regarded the execution of a test of significance on an experiment as the ultimate objective."

[58] Begg, Colin B.; Berlin, Jesse A. (1988). "Publication bias: a problem in interpreting medical data". *Journal of the Royal Statistical Society, Series A*: 419–463.

[59] Meehl, Paul E. (1967). "Theory-Testing in Psychology and Physics: A Methodological Paradox" (PDF). *Philosophy of Science* **34** (2): 103–115. doi:10.1086/288135. Thirty years later, Meehl acknowledged statistical significance theory to be mathematically sound while continuing to question the default choice of null hypothesis, blaming instead the "social scientists' poor understanding of the logical relation between theory and fact" in "The Problem Is Epistemology, Not Statistics: Replace Significance Tests by Confidence Intervals and Quantify Accuracy of Risky Numerical Predictions" (Chapter 14 in Harlow (1997)).

[60] Nunnally, Jum (1960). "The place of statistics in psychology". *Educational and Psychological Measurement* **20** (4): 641–650. doi:10.1177/001316446002000401.

[61] Lykken, David T. (1991). "What's wrong with psychology, anyway?". *Thinking Clearly About Psychology* **1**: 3–39.

[62] Jacob Cohen (December 1994). "The Earth Is Round ($p < .05$)". *American Psychologist* **49** (12): 997–1003. doi:10.1037/0003-066X.49.12.997. This paper lead to the review of statistical practices by the APA. Cohen was a member of the Task Force that did the review.

[63] Nickerson, Raymond S. (2000). "Null Hypothesis Significance Tests: A Review of an Old and Continuing Controversy". *Psychological Methods* **5** (2): 241–301. doi:10.1037/1082-989X.5.2.241. PMID 10937333.

[64] Wilkinson, Leland (1999). "Statistical Methods in Psychology Journals; Guidelines and Explanations". *American Psychologist* **54** (8): 594–604. doi:10.1037/0003-066X.54.8.594. "Hypothesis tests. It is hard to imagine a situation in which a dichotomous accept-reject decision is better than reporting an actual p value or, better still, a confidence interval." (p 599). The committee used the cautionary term "forbearance" in describing its decision against a ban of hypothesis testing in psychology reporting. (p 603)

[65] "ICMJE: Obligation to Publish Negative Studies". Retrieved 3 September 2012. Editors should seriously consider for publication any carefully done study of an important question, relevant to their readers, whether the results for the primary or any additional outcome are statistically significant. Failure to submit or publish findings because of lack of statistical significance is an important cause of publication bias.

[66] *Journal of Articles in Support of the Null Hypothesis* website: JASNH homepage. Volume 1 number 1 was published in 2002, and all articles are on psychology-related subjects.

[67] Howell, David (2002). *Statistical Methods for Psychology* (5 ed.). Duxbury. p. 94. ISBN 0-534-37770-X.

[68] Armstrong, J. Scott (2007). "Significance tests harm progress in forecasting". *International Journal of Forecasting* **23** (2): 321–327. doi:10.1016/j.ijforecast.2007.03.004.

[69] E. L. Lehmann (1997). "Testing Statistical Hypotheses: The Story of a Book". *Statistical Science* **12** (1): 48–52. doi:10.1214/ss/1029963261.

[70] Kruschke, J K (July 9, 2012). "Bayesian Estimation Supersedes the T Test". *Journal of Experimental Psychology: General* **N/A** (N/A): N/A. doi:10.1037/a0029146.

[71] Kass, R E (1993). "Bayes factors and model uncertainty" (PDF).Department of Statistics, University of Washington Technical Paper

[72] Rozeboom, William W (1960), "The fallacy of the null-hypothesis significance test" (PDF), *Psychological Bulletin* **57** (5): 416–428, doi:10.1037/h0042040 "...the proper application of statistics to scientific inference is irrevocably committed to extensive consideration of inverse [AKA Bayesian] probabilities..." It was acknowledged, with regret, that a priori probability distributions were available "only as a subjective feel, differing from one person to the next" "in the more immediate future, at least".

[73] Berger, James (2006), "The Case for Objective Bayesian Analysis", *Bayesian Analysis* **1** (3): 385–402, doi:10.1214/06-ba115 In listing the competing definitions of "objective" Bayesian analysis, "A major goal of statistics (indeed science) is to find a completely coherent objective Bayesian methodology for learning from data." The author expressed the view that this goal "is not attainable".

[74] Aldrich, J (2008). "R. A. Fisher on Bayes and Bayes' theorem" (PDF). *Bayesian Analysis* **3** (1): 161–170. doi:10.1214/08-BA306.

[75] Mayo, D. G.; Spanos, A. (2006). "Severe Testing as a Basic Concept in a Neyman-Pearson Philosophy of Induction". *The British Journal for the Philosophy of Science* **57** (2): 323. doi:10.1093/bjps/axl003.

[76] Mathematics > High School: Statistics & Probability > Introduction Common Core State Standards Initiative (relates to USA students)

[77] College Board Tests > AP: Subjects > Statistics The College Board (relates to USA students)

[78] Huff, Darrell (1993). *How to lie with statistics*. New York: Norton. p. 8. ISBN 0-393-31072-8.'Statistical methods and statistical terms are necessary in reporting the mass data of social and economic trends, business conditions, "opinion" polls, the census. But without writers who use the words with honesty and readers who know what they mean, the result can only be semantic nonsense.'

[79] Snedecor, George W.; Cochran, William G. (1967). *Statistical Methods* (6 ed.). Ames, Iowa: Iowa State University Press. p. 3. "...the basic ideas in statistics assist us in thinking clearly about the problem, provide some guidance about the conditions that must be satisfied if sound inferences are to be made, and enable us to detect many inferences that have no good logical foundation."

[80] Sotos, Ana Elisa Castro; Vanhoof, Stijn; Noortgate, Wim Van den; Onghena, Patrick (2007). "Students' Misconceptions of Statistical Inference: A Review of the Empirical Evidence from Research on Statistics Education". *Educational Research Review* **2**: 98–113. doi:10.1016/j.edurev.2007.04.001.

[81] Moore, David S. (1997). "New Pedagogy and New Content: The Case of Statistics". *International Statistical Review* **65**: 123–165. doi:10.2307/1403333.

[82] Hubbard, Raymond; Armstrong, J. Scott (2006). "Why We Don't Really Know What Statistical Significance Means: Implications for Educators". *Journal of Marketing Education* **28** (2): 114. doi:10.1177/0273475306288399. Preprint

[83] Sotos, Ana Elisa Castro; Vanhoof, Stijn; Noortgate, Wim Van den; Onghena, Patrick (2009). "How Confident Are Students in Their Misconceptions about Hypothesis Tests?". *Journal of Statistics Education* **17** (2).

[84] Gigerenzer, G (2004). "The Null Ritual What You Always Wanted to Know About Significant Testing but Were Afraid to Ask" (PDF). *The SAGE Handbook of Quantitative Methodology for the Social Sciences*: 391–408. doi:10.4135/9781412986311.

26.14 Further reading

- Lehmann E.L. (1992) "Introduction to Neyman and Pearson (1933) On the Problem of the Most Efficient Tests of Statistical Hypotheses". In: *Breakthroughs in Statistics, Volume 1*, (Eds Kotz, S., Johnson, N.L.), Springer-Verlag. ISBN 0-387-94037-5 (followed by reprinting of the paper)

- Neyman, J.; Pearson, E.S. (1933). "On the Problem of the Most Efficient Tests of Statistical Hypotheses". *Philosophical Transactions of the Royal Society A* **231** (694–706): 289–337. doi:10.1098/rsta.1933.0009.

26.15 External links

- Hazewinkel, Michiel, ed. (2001), "Statistical hypotheses, verification of", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Wilson González, Georgina; Kay Sankaran (September 10, 1997). "Hypothesis Testing". *Environmental Sampling & Monitoring Primer*. Virginia Tech.
- Bayesian critique of classical hypothesis testing
- Critique of classical hypothesis testing highlighting long-standing qualms of statisticians
- Dallal GE (2007) The Little Handbook of Statistical Practice (A good tutorial)
- References for arguments for and against hypothesis testing
- Statistical Tests Overview: How to choose the correct statistical test
- An Interactive Online Tool to Encourage Understanding Hypothesis Testing
- A non mathematical way to understand Hypothesis Testing

26.15.1 Online calculators

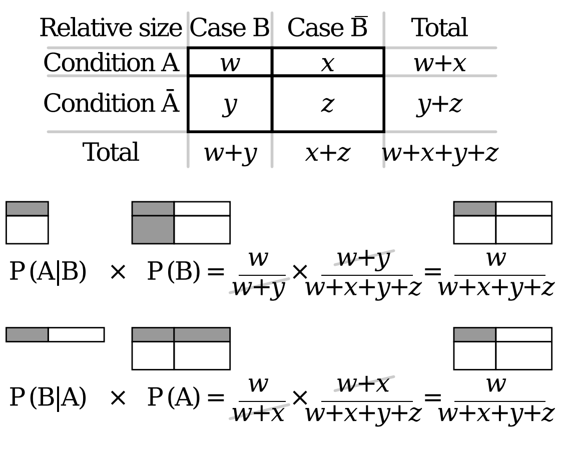
- MBAStats confidence interval and hypothesis test calculators

Chapter 27

Bayesian inference

Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as evidence is acquired. Bayesian inference is an important technique in statistics, and especially in mathematical statistics. Bayesian updating is particularly important in the dynamic analysis of a sequence of data. Bayesian inference has found application in a wide range of activities, including science, engineering, philosophy, medicine, and law. In the philosophy of decision theory, Bayesian inference is closely related to subjective probability, often called "Bayesian probability". Bayesian probability provides a rational method for updating beliefs.

27.1 Introduction to Bayes' rule



Relative size	Case B	Case B̄	Total
Condition A	w	x	$w+x$
Condition Ā	y	z	$y+z$
Total	$w+y$	$x+z$	$w+x+y+z$

$$P(A|B) \times P(B) = \frac{w}{w+y} \times \frac{w+y}{w+x+y+z} = \frac{w}{w+x+y+z}$$

$$P(B|A) \times P(A) = \frac{w}{w+x} \times \frac{w+x}{w+x+y+z} = \frac{w}{w+x+y+z}$$

Ā) P(Ā)/P(B) etc.

Main article: Bayes' rule
See also: Bayesian probability

27.1.1 Formal

Bayesian inference derives the posterior probability as a consequence of two antecedents, a prior probability and a "likelihood function" derived from a statistical model for the observed data. Bayesian inference computes the posterior probability according to Bayes' theorem:

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)}$$

where

- $|$ denotes a conditional probability; more specifically, it means *given*.
- H stands for any *hypothesis* whose probability may be affected by data (called *evidence* below). Often there are competing hypotheses, from which one chooses the most probable.
- the *evidence* E corresponds to new data that were not used in computing the prior probability.
- $P(H)$, the *prior probability*, is the probability of H *before* E is observed. This indicates one's previous estimate of the probability that a hypothesis is true, before gaining the current evidence.
- $P(H \mid E)$, the *posterior probability*, is the probability of H *given* E , i.e., *after* E is observed. This tells us what we want to know: the probability of a hypothesis *given* the observed evidence.
- $P(E \mid H)$ is the probability of observing E *given* H . As a function of H with E fixed, this is the *likelihood*. The likelihood function should **not** be confused with $P(H \mid E)$ as a function of H rather than of E . It indicates the compatibility of the evidence with the given hypothesis.
- $P(E)$ is sometimes termed the marginal likelihood or "model evidence". This factor is the same for all possible hypotheses being considered. (This can be seen by the fact that the hypothesis H does not appear anywhere in the symbol, unlike for all the other factors.) This means that this factor does not enter into determining the relative probabilities of different hypotheses.

Note that, for different values of H , only the factors $P(H)$ and $P(E \mid H)$ affect the value of $P(H \mid E)$

. As both of these factors appear in the numerator, the posterior probability is proportional to both. In words:

- (more precisely) *The posterior probability of a hypothesis is determined by a combination of the inherent likeliness of a hypothesis (the prior) and the compatibility of the observed evidence with the hypothesis (the likelihood).*
- (more concisely) *Posterior is proportional to likelihood times prior.*

Note that Bayes' rule can also be written as follows:

$$P(H \mid E) = \frac{P(E \mid H)}{P(E)} \cdot P(H)$$

where the factor $\frac{P(E|H)}{P(E)}$ represents the impact of E on the probability of H .

27.1.2 Informal

If the evidence does not match up with a hypothesis, one should reject the hypothesis. But if a hypothesis is extremely unlikely *a priori*, one should also reject it, even if the evidence does appear to match up.

For example, imagine that I have various hypotheses about the nature of a newborn baby of a friend, including:

- H_1 : the baby is a brown-haired boy.
- H_2 : the baby is a blond-haired girl.
- H_3 : the baby is a dog.

Then consider two scenarios:

1. I'm presented with evidence in the form of a picture of a blond-haired baby girl. I find this evidence supports H_2 and opposes H_1 and H_3 .
2. I'm presented with evidence in the form of a picture of a baby dog. Although this evidence, treated in isolation, supports H_3 , my prior belief in this hypothesis (that a human can give birth to a dog) is extremely small, so the posterior probability is nevertheless small.

The critical point about Bayesian inference, then, is that it provides a principled way of combining new evidence with prior beliefs, through the application of Bayes' rule. (Contrast this with frequentist inference, which relies only on the evidence as a whole, with no reference to prior beliefs.) Furthermore, Bayes' rule can be applied iteratively: after observing some evidence, the resulting posterior probability can then be treated as a prior probability, and a new posterior probability computed from new evidence. This allows for Bayesian principles to be applied to various kinds of evidence, whether viewed all at once or over time. This procedure is termed "Bayesian updating".

27.1.3 Bayesian updating

Bayesian updating is widely used and computationally convenient. However, it is not the only updating rule that might be considered "rational".

Ian Hacking noted that traditional "Dutch book" arguments did not specify Bayesian updating: they left open the possibility that non-Bayesian updating rules could avoid Dutch books. Hacking wrote[1] "And neither the Dutch book argument, nor any other in the personalist arsenal of proofs of the probability axioms, entails the dynamic assumption. Not one entails Bayesianism. So the personalist requires the dynamic assumption to be Bayesian. It is true that in consistency a personalist could abandon the Bayesian model of learning from experience. Salt could lose its savour."

Indeed, there are non-Bayesian updating rules that also avoid Dutch books (as discussed in the literature on "probability kinematics" following the publication of Richard C. Jeffrey's rule, which applies Bayes' rule to the case where the evidence itself is assigned a probability.[2] The additional hypotheses needed to uniquely require Bayesian updating have been deemed to be substantial, complicated, and unsatisfactory.[3]

27.2 Formal description of Bayesian inference

27.2.1 Definitions

- x , a data point in general. This may in fact be a vector of values.
- θ , the parameter of the data point's distribution, i.e., $x \sim p(x \mid \theta)$. This may in fact be a vector of parameters.
- α , the hyperparameter of the parameter, i.e., $\theta \sim p(\theta \mid \alpha)$. This may in fact be a vector of hyperparameters.
- $\mathbf{X}$, a set of n observed data points, i.e., $x_1, \ldots, x_n$.
- $\tilde{x}$, a new data point whose distribution is to be predicted.

27.2.2 Bayesian inference

- The prior distribution is the distribution of the parameter(s) before any data is observed, i.e. $p(\theta \mid \alpha)$

.

- The prior distribution might not be easily determined. In this case, we can use the Jeffreys prior to obtain the posterior distribution before updating them with newer observations.

- The sampling distribution is the distribution of the observed data conditional on its parameters, i.e. $p(\mathbf{X} \mid \theta)$. This is also termed the likelihood, especially when viewed as a function of the parameter(s), sometimes written $\operatorname{L}(\theta \mid \mathbf{X}) = p(\mathbf{X} \mid \theta)$.

- The marginal likelihood (sometimes also termed the *evidence*) is the distribution of the observed data marginalized over the parameter(s), i.e. $p(\mathbf{X} \mid \alpha) = \int_\theta p(\mathbf{X} \mid \theta) p(\theta \mid \alpha) \operatorname{d}\!\theta$.

- The posterior distribution is the distribution of the parameter(s) after taking into account the observed data. This is determined by Bayes' rule, which forms the heart of Bayesian inference:

$$p(\theta \mid \mathbf{X},\alpha) = \frac{p(\mathbf{X} \mid \theta) p(\theta \mid \alpha)}{p(\mathbf{X} \mid \alpha)} \propto p(\mathbf{X} \mid \theta) p(\theta \mid \alpha)$$

Note that this is expressed in words as "posterior is proportional to likelihood times prior", or sometimes as "posterior = likelihood times prior, over evidence".

27.2.3 Bayesian prediction

- The posterior predictive distribution is the distribution of a new data point, marginalized over the posterior:

$$p(\tilde{x} \mid \mathbf{X},\alpha) = \int_{\theta} p(\tilde{x} \mid \theta) p(\theta \mid \mathbf{X},\alpha) \operatorname{d}\!\theta$$

- The prior predictive distribution is the distribution of a new data point, marginalized over the prior:

$$p(\tilde{x} \mid \alpha) = \int_{\theta} p(\tilde{x} \mid \theta) p(\theta \mid \alpha) \operatorname{d}\!\theta$$

Bayesian theory calls for the use of the posterior predictive distribution to do predictive inference, i.e., to predict the distribution of a new, unobserved data point. That is, instead of a fixed point as a prediction, a distribution over possible points is returned. Only this way is the entire posterior distribution of the parameter(s) used. By comparison, prediction in frequentist statistics often involves finding an optimum point estimate of the parameter(s)—e.g., by maximum likelihood or maximum a posteriori estimation (MAP)—and then plugging this estimate into the formula for the distribution of a data point. This has the disadvantage that it does not account for any uncertainty in the value of the parameter, and hence will underestimate the variance of the predictive distribution.

(In some instances, frequentist statistics can work around this problem. For example, confidence intervals and prediction intervals in frequentist statistics when constructed from a normal distribution with unknown mean and variance are constructed using a Student's t-distribution. This correctly estimates the variance, due to the fact that (1) the average of normally distributed random variables is also normally distributed; (2) the predictive distribution of a normally distributed data point with unknown mean and variance, using conjugate or uninformative priors, has a student's t-distribution. In Bayesian statistics, however, the posterior predictive distribution can always be determined exactly—or at least, to an arbitrary level of precision, when numerical methods are used.)

Note that both types of predictive distributions have the form of a compound probability distribution (as does the marginal likelihood). In fact, if the prior distribution is a conjugate prior, and hence the prior and posterior distributions come from the same family, it can easily be seen that both prior and posterior predictive distributions also come from the same family of compound distributions. The only difference is that the posterior predictive distribution uses the updated values of the hyperparameters (applying the Bayesian update rules given in the conjugate prior article), while the prior predictive distribution uses the values of the hyperparameters that appear in the prior distribution.

27.3 Inference over exclusive and exhaustive possibilities

If evidence is simultaneously used to update belief over a set of exclusive and exhaustive propositions, Bayesian inference may be thought of as acting on this belief distribution as a whole.

27.3.1 General formulation

Suppose a process is generating independent and identically distributed events E_n , but the probability distribution is unknown. Let the event space Ω represent the current state of belief for this process. Each model is represented by event M_m . The conditional probabilities $P(E_n \mid M_m)$ are specified to define the models. $P(M_m)$ is the degree of belief in M_m . Before the first inference step, $\{P(M_m)\}$ is a set of *initial prior probabilities*. These must sum to 1, but are otherwise arbitrary.

Suppose that the process is observed to generate $E \in \{E_n\}$. For each $M \in \{M_m\}$, the prior $P(M)$ is updated to the posterior $P(M \mid E)$. From Bayes' theorem:[4]

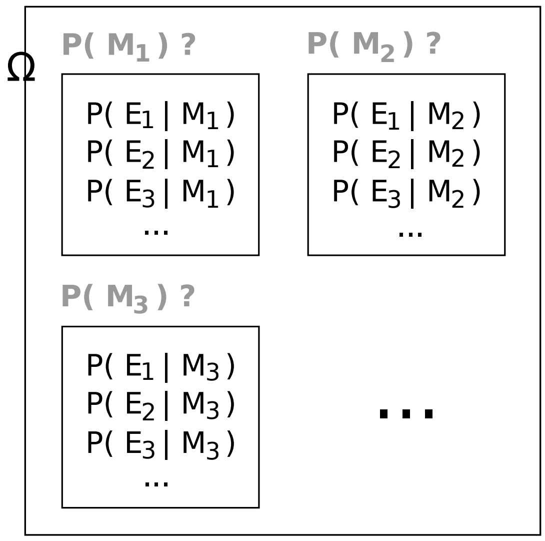


Diagram illustrating event space Ω in general formulation of Bayesian inference. Although this diagram shows discrete models and events, the continuous case may be visualized similarly using probability densities.

$$P(M \mid E) = \frac{P(E \mid M)}{\sum_m P(E \mid M_m)P(M_m)} \cdot P(M)$$

Upon observation of further evidence, this procedure may be repeated.

27.3.2 Multiple observations

For a set of independent and identically distributed observations $\mathbf{E} = \{e_1, \ldots, e_n\}$, it may be shown that repeated application of the above is equivalent to

$$P(M \mid \mathbf{E}) = \frac{P(\mathbf{E} \mid M)}{\sum_m P(\mathbf{E} \mid M_m)P(M_m)} \cdot P(M)$$

Where

$$P(\mathbf{E} \mid M) = \prod_k P(e_k \mid M).$$

This may be used to optimize practical calculations.

27.3.3 Parametric formulation

By parameterizing the space of models, the belief in all models may be updated in a single step. The distribution of belief over the model space may then be thought of as a distribution of belief over the parameter space. The distributions in this section are expressed as continuous, represented by probability densities, as this is the usual situation. The technique is however equally applicable to discrete distributions.

Let the vector θ span the parameter space. Let the initial prior distribution over θ be $p(\theta \mid \alpha)$, where α is a set of parameters to the prior itself, or *hyperparameters*. Let $\mathbf{E} = \{e_1, \ldots, e_n\}$ be a set of independent and identically distributed event observations, where all e_i are distributed as $p(e \mid \theta)$ for some θ . Bayes' theorem is applied to find the posterior distribution over θ :

$$\begin{aligned} p(\theta \mid \mathbf{E}, \alpha) &= \frac{p(\mathbf{E} \mid \theta, \alpha)}{p(\mathbf{E} \mid \alpha)} \cdot p(\theta \mid \alpha) \\ &= \frac{p(\mathbf{E} \mid \theta, \alpha)}{\int_\theta p(\mathbf{E}|\theta, \alpha)p(\theta \mid \alpha)\, d\theta} \cdot p(\theta \mid \alpha) \end{aligned}$$

Where

$$p(\mathbf{E} \mid \theta, \alpha) = \prod_k p(e_k \mid \theta)$$

27.4 Mathematical properties

27.4.1 Interpretation of factor

$\frac{P(E|M)}{P(E)} > 1 \Rightarrow P(E \mid M) > P(E)$. That is, if the model were true, the evidence would be more likely than is predicted by the current state of belief. The reverse applies for a decrease in belief. If the belief does not change, $\frac{P(E|M)}{P(E)} = 1 \Rightarrow P(E \mid M) = P(E)$. That is, the evidence is independent of the model. If the model were true, the evidence would be exactly as likely as predicted by the current state of belief.

27.4.2 Cromwell's rule

Main article: Cromwell's rule

If $P(M) = 0$ then $P(M \mid E) = 0$. If $P(M) = 1$, then $P(M|E) = 1$. This can be interpreted to mean that hard convictions are insensitive to counter-evidence.

The former follows directly from Bayes' theorem. The latter can be derived by applying the first rule to the event “not M " in place of " M ", yielding “if $1 - P(M) = 0$, then $1 - P(M \mid E) = 0$ ", from which the result immediately follows.

27.4.3 Asymptotic behaviour of posterior

Consider the behaviour of a belief distribution as it is updated a large number of times with independent and identically distributed trials. For sufficiently nice prior probabilities, the Bernstein-von Mises theorem gives that in the limit of infinite trials, the posterior converges to a Gaussian distribution independent of the initial prior under some conditions firstly outlined and rigorously proven by Joseph L. Doob in 1948, namely if the random variable in consideration has a finite probability space. The more general results were obtained later by the statistician David A. Freedman who published in two seminal research papers in 1963 and 1965 when and under what circumstances the asymptotic behaviour of posterior is guaranteed. His 1963 paper treats, like Doob (1949), the finite case and comes to a satisfactory conclusion. However, if the random variable has an infinite but countable probability space (i.e., corresponding to a die with infinite many faces) the 1965 paper demonstrates that for a dense subset of priors the Bernstein-von Mises theorem is not applicable. In this case there is almost surely no asymptotic convergence. Later in the 1980s and 1990s Freedman and Persi Diaconis continued to work on the case of infinite countable probability spaces.[5] To summarise, there may be insufficient trials to suppress the effects of the initial choice, and especially for large (but finite) systems the convergence might be very slow.

27.4.4 Conjugate priors

Main article: Conjugate prior

In parameterized form, the prior distribution is often assumed to come from a family of distributions called conjugate priors. The usefulness of a conjugate prior is that the corresponding posterior distribution will be in the same family, and the calculation may be expressed in closed form.

27.4.5 Estimates of parameters and predictions

It is often desired to use a posterior distribution to estimate a parameter or variable. Several methods of Bayesian estimation select measurements of central tendency from the posterior distribution.

For one-dimensional problems, a unique median exists for practical continuous problems. The posterior median is attractive as a robust estimator.[6]

If there exists a finite mean for the posterior distribution, then the posterior mean is a method of estimation.

$$\tilde{\theta} = \operatorname{E}[\theta] = \int_\theta \theta\, p(\theta \mid \mathbf{X}, \alpha)\, d\theta$$

Taking a value with the greatest probability defines maximum *a posteriori* (MAP) estimates:

$$\{\theta_{\text{MAP}}\} \subset \arg\max_\theta p(\theta \mid \mathbf{X}, \alpha).$$

There are examples where no maximum is attained, in which case the set of MAP estimates is empty.

There are other methods of estimation that minimize the posterior *risk* (expected-posterior loss) with respect to a loss function, and these are of interest to statistical decision theory using the sampling distribution (“frequentist statistics”).

The posterior predictive distribution of a new observation $\tilde{x}$ (that is independent of previous observations) is determined by

$$p(\tilde{x}|\mathbf{X}, \alpha) = \int_\theta p(\tilde{x}, \theta \mid \mathbf{X}, \alpha)\, d\theta = \int_\theta p(\tilde{x} \mid \theta) p(\theta \mid \mathbf{X}, \alpha)\, d\theta.$$

27.5 Examples

27.5.1 Probability of a hypothesis

Suppose there are two full bowls of cookies. Bowl #1 has 10 chocolate chip and 30 plain cookies, while bowl #2 has 20 of each. Our friend Fred picks a bowl at random, and then picks a cookie at random. We may assume there is no reason to believe Fred treats one bowl differently from another, likewise for the cookies. The cookie turns out to be a plain one. How probable is it that Fred picked it out of bowl #1?

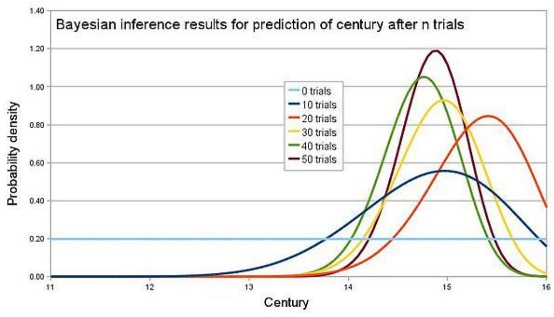
Intuitively, it seems clear that the answer should be more than a half, since there are more plain cookies in bowl #1. The precise answer is given by Bayes' theorem. Let H_1 correspond to bowl #1, and H_2 to bowl #2. It is given that the bowls are identical from Fred's point of view, thus $P(H_1) = P(H_2)$, and the two must add up to 1, so both are equal to 0.5. The event E is the observation of a plain cookie. From the contents of the bowls, we know that $P(E \mid H_1) = 30/40 = 0.75$ and $P(E \mid H_2) = 20/40 = 0.5$. Bayes' formula then yields

$$P(H_1 \mid E) = \frac{P(E \mid H_1)\, P(H_1)}{P(E \mid H_1)\, P(H_1)\; +\; P(E \mid H_2)\, P(H_2)}$$

$$= \frac{0.75 \times 0.5}{0.75 \times 0.5 + 0.5 \times 0.5}$$

$$= 0.6$$

Before we observed the cookie, the probability we assigned for Fred having chosen bowl #1 was the prior probability, $P(H_1)$, which was 0.5. After observing the cookie, we must revise the probability to $P(H_1 \mid E)$, which is 0.6.



Example results for archaeology example. This simulation was generated using c=15.2.

27.5.2 Making a prediction

An archaeologist is working at a site thought to be from the medieval period, between the 11th century to the 16th century. However, it is uncertain exactly when in this period the site was inhabited. Fragments of pottery are found, some of which are glazed and some of which are decorated. It is expected that if the site were inhabited during the early medieval period, then 1% of the pottery would be glazed and 50% of its area decorated, whereas if it had been inhabited in the late medieval period then 81% would be glazed and 5% of its area decorated. How confident can the archaeologist be in the date of inhabitation as fragments are unearthed?

The degree of belief in the continuous variable C (century) is to be calculated, with the discrete set of events $\{GD, G\bar{D}, \bar{G}D, \bar{G}\bar{D}\}$ as evidence. Assuming linear variation of glaze and decoration with time, and that these variables are independent,

$$P(E = GD \mid C = c) = (0.01{+}0.16(c{-}11))(0.5{-}0.09(c{-}11))$$

$$P(E = G\bar{D} \mid C = c) = (0.01{+}0.16(c{-}11))(0.5{+}0.09(c{-}11))$$

$$P(E = \bar{G}D \mid C = c) = (0.99{-}0.16(c{-}11))(0.5{-}0.09(c{-}11))$$

$$P(E = \bar{G}\bar{D} \mid C = c) = (0.99{-}0.16(c{-}11))(0.5{+}0.09(c{-}11))$$

Assume a uniform prior of $f_C(c) = 0.2$, and that trials are independent and identically distributed. When a new fragment of type e is discovered, Bayes' theorem is applied to update the degree of belief for each c :

$$f_C(c \mid E = e) = \frac{P(E=e|C=c)}{P(E=e)} f_C(c) = \frac{P(E=e|C=c)}{\int_{11}^{16} P(E=e|C=c) f_C(c) dc} f_C(c)$$

A computer simulation of the changing belief as 50 fragments are unearthed is shown on the graph. In the simulation, the site was inhabited around 1420, or $c = 15.2$. By calculating the area under the relevant portion of the graph for 50 trials, the archaeologist can say that there is practically no chance the site was inhabited in the 11th and 12th centuries, about 1% chance that it was inhabited during the 13th century, 63% chance during the 14th century and 36% during the 15th century. Note that the Bernstein-von Mises theorem asserts here the asymptotic convergence to the "true" distribution because the probability space corresponding to the discrete set of events $\{GD, G\bar{D}, \bar{G}D, \bar{G}\bar{D}\}$ is finite (see above section on asymptotic behaviour of the posterior).

27.6 In frequentist statistics and decision theory

A decision-theoretic justification of the use of Bayesian inference was given by Abraham Wald, who proved that every unique Bayesian procedure is admissible. Conversely, every admissible statistical procedure is either a Bayesian procedure or a limit of Bayesian procedures.[7]

Wald characterized admissible procedures as Bayesian procedures (and limits of Bayesian procedures), making the Bayesian formalism a central technique in such areas of frequentist inference as parameter estimation, hypothesis testing, and computing confidence intervals.[8] For example:

- "Under some conditions, all admissible procedures are either Bayes procedures or limits of Bayes procedures (in various senses). These remarkable results, at least in their original form, are due essentially to Wald. They are useful because the property of being Bayes is easier to analyze than admissibility."[7]

- "In decision theory, a quite general method for proving admissibility consists in exhibiting a procedure as a unique Bayes solution."[9]

- "In the first chapters of this work, prior distributions with finite support and the corresponding Bayes procedures were used to establish some of the main theorems relating to the comparison of experiments. Bayes procedures with respect to more general prior distributions have played a very important role in the development of statistics, including its asymptotic theory." "There are many problems where a glance at posterior distributions, for suitable priors, yields immediately interesting information. Also, this technique can hardly be avoided in sequential analysis."[10]

- "A useful fact is that any Bayes decision rule obtained by taking a proper prior over the whole parameter space must be admissible"[11]

- "An important area of investigation in the development of admissibility ideas has been that of conventional sampling-theory procedures, and many interesting results have been obtained."[12]

27.6.1 Model selection

See Bayesian model selection

27.7 Applications

27.7.1 Computer applications

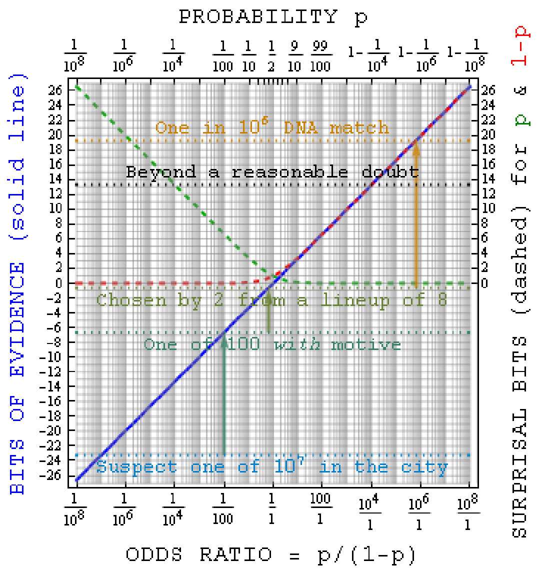
Bayesian inference has applications in artificial intelligence and expert systems. Bayesian inference techniques have been a fundamental part of computerized pattern recognition techniques since the late 1950s. There is also an ever growing connection between Bayesian methods and simulation-based Monte Carlo techniques since complex models cannot be processed in closed form by a Bayesian analysis, while a graphical model structure *may* allow for efficient simulation algorithms like the Gibbs sampling and other Metropolis–Hastings algorithm schemes.[13] Recently Bayesian inference has gained popularity amongst the phylogenetics community for these reasons; a number of applications allow many demographic and evolutionary parameters to be estimated simultaneously.

As applied to statistical classification, Bayesian inference has been used in recent years to develop algorithms for identifying e-mail spam. Applications which make use of Bayesian inference for spam filtering include CRM114, DSPAM, Bogofilter, SpamAssassin, SpamBayes, Mozilla, XEAMS, and others. Spam classification is treated in more detail in the article on the naive Bayes classifier.

Solomonoff's Inductive inference is the theory of prediction based on observations; for example, predicting the next symbol based upon a given series of symbols. The only assumption is that the environment follows some unknown but computable probability distribution. It is a formal inductive framework that combines two well-studied principles of inductive inference: Bayesian statistics and Occam's Razor.[14] Solomonoff's universal prior probability of any prefix p of a computable sequence x is the sum of the probabilities of all programs (for a universal computer) that compute something starting with p. Given some p and any computable but unknown probability distribution from which x is sampled, the universal prior and Bayes' theorem can be used to predict the yet unseen parts of x in optimal fashion.[15][16]

27.7.2 In the courtroom

Bayesian inference can be used by jurors to coherently accumulate the evidence for and against a defendant, and to see whether, in totality, it meets their personal threshold for 'beyond a reasonable doubt'.[17][18][19] Bayes' theorem is applied successively to all evidence presented, with the posterior from one stage becoming the prior for the next. The benefit of a Bayesian approach is that it gives the juror an unbiased, rational mechanism for combining evidence. It may be appropriate to explain Bayes' theorem to jurors in odds form, as betting odds are more widely understood than probabilities. Alternatively, a logarithmic approach, replacing multiplication with addition, might be easier for a jury to handle.



Adding up evidence.

If the existence of the crime is not in doubt, only the identity of the culprit, it has been suggested that the prior should be uniform over the qualifying population.[20] For example, if 1,000 people could have committed the crime, the prior probability of guilt would be 1/1000.

The use of Bayes' theorem by jurors is controversial. In the United Kingdom, a defence expert witness explained Bayes' theorem to the jury in *R v Adams*. The jury convicted, but the case went to appeal on the basis that no means of accumulating evidence had been provided for jurors who did not wish to use Bayes' theorem. The Court of Appeal upheld the conviction, but it also gave the opinion that "To introduce Bayes' Theorem, or any similar method, into a criminal trial plunges the jury into inappropriate and unnecessary realms of theory and complexity, deflecting them from their proper task."

Gardner-Medwin[21] argues that the criterion on which a verdict in a criminal trial should be based is *not* the probability of guilt, but rather the *probability of the evidence, given that the defendant is innocent* (akin to a frequentist p-value). He argues that if the posterior probability of guilt is to be computed by Bayes' theorem, the prior probability of guilt must be known. This will depend on the incidence of the crime, which is an unusual piece of evidence to consider in a criminal trial. Consider the following three propositions:

A The known facts and testimony could have arisen if the defendant is guilty

B The known facts and testimony could have arisen if the defendant is innocent

C The defendant is guilty.

Gardner-Medwin argues that the jury should believe both A and not-B in order to convict. A and not-B implies the truth of C, but the reverse is not true. It is possible that B and C are both true, but in this case he argues that a jury should acquit, even though they know that they will be letting some guilty people go free. See also Lindley's paradox.

27.7.3 Bayesian epistemology

Bayesian epistemology is a movement that advocates for Bayesian inference as a means of justifying the rules of inductive logic.

Karl Popper and David Miller have rejected the alleged rationality of Bayesianism, i.e. using Bayes rule to make epistemological inferences:[22] It is prone to the same vicious circle as any other justificationist epistemology, because it presupposes what it attempts to justify. According to this view, a rational interpretation of Bayesian inference would see it merely as a probabilistic version of falsification, rejecting the belief, commonly held by Bayesians, that high likelihood achieved by a series of Bayesian updates would prove the hypothesis beyond any reasonable doubt, or even with likelihood greater than 0.

27.7.4 Other

- The scientific method is sometimes interpreted as an application of Bayesian inference. In this view, Bayes' rule guides (or should guide) the updating of probabilities about hypotheses conditional on new observations or experiments.[23]
- Bayesian search theory is used to search for lost objects.
- Bayesian inference in phylogeny
- Bayesian tool for methylation analysis

27.8 Bayes and Bayesian inference

The problem considered by Bayes in Proposition 9 of his essay, "An Essay towards solving a Problem in the Doctrine of Chances", is the posterior distribution for the parameter *a* (the success rate) of the binomial distribution.

27.9 History

Main article: History of statistics § Bayesian statistics

The term *Bayesian* refers to Thomas Bayes (1702–1761), who proved a special case of what is now called Bayes' theorem. However, it was Pierre-Simon Laplace (1749–1827) who introduced a general version of the theorem and used it to approach problems in celestial mechanics, medical statistics, reliability, and jurisprudence.[24] Early Bayesian inference, which used uniform priors following Laplace's principle of insufficient reason, was called "inverse probability" (because it infers backwards from observations to parameters, or from effects to causes[25]). After the 1920s, "inverse probability" was largely supplanted by a collection of methods that came to be called frequentist statistics.[25]

In the 20th century, the ideas of Laplace were further developed in two different directions, giving rise to *objective* and *subjective* currents in Bayesian practice. In the objective or "non-informative" current, the statistical analysis depends on only the model assumed, the data analyzed,[26] and the method assigning the prior, which differs from one objective Bayesian to another objective Bayesian. In the subjective or "informative" current, the specification of the prior depends on the belief (that is, propositions on which the analysis is prepared to act), which can summarize information from experts, previous studies, etc.

In the 1980s, there was a dramatic growth in research and applications of Bayesian methods, mostly attributed to the discovery of Markov chain Monte Carlo methods, which removed many of the computational problems, and an increasing interest in nonstandard, complex applications.[27] Despite growth of Bayesian research, most undergraduate teaching is still based on frequentist statistics.[28] Nonetheless, Bayesian methods are widely accepted and used, such as for example in the field of machine learning.[29]

27.10 See also

- Bayes' theorem
- Bayesian hierarchical modeling
- Bayesian Analysis, the journal of the ISBA
- Inductive probability
- International Society for Bayesian Analysis (ISBA)
- Jeffreys prior

27.11 Notes

[1] Hacking (1967, Section 3, p. 316), Hacking (1988, p. 124)

[2] "Bayes' Theorem (Stanford Encyclopedia of Philosophy)". Plato.stanford.edu. Retrieved 2014-01-05.

[3] van Fraassen, B. (1989) *Laws and Symmetry*, Oxford University Press. ISBN 0-19-824860-1

[4] Gelman, Andrew; Carlin, John B.; Stern, Hal S.; Dunson, David B.;Vehtari, Aki; Rubin, Donald B. (2013). *Bayesian Data Analysis*, Third Edition. Chapman and Hall/CRC. ISBN 978-1-4398-4095-5.

[5] Larry Wasserman et alia, JASA 2000.

[6] Sen, Pranab K.; Keating, J. P.; Mason, R. L. (1993). *Pitman's measure of closeness: A comparison of statistical estimators*. Philadelphia: SIAM.

[7] Bickel & Doksum (2001, p. 32)

[8]
- Kiefer, J. and Schwartz, R. (1965). "Admissible Bayes Character of T^2-, R^2-, and Other Fully Invariant Tests for Multivariate Normal Problems". *Annals of Mathematical Statistics* **36**: 747–770. doi:10.1214/aoms/1177700051.
- Schwartz, R. (1969). "Invariant Proper Bayes Tests for Exponential Families". *Annals of Mathematical Statistics* **40**: 270–283. doi:10.1214/aoms/1177697822.
- Hwang, J. T. and Casella, George (1982). "Minimax Confidence Sets for the Mean of a Multivariate Normal Distribution". *Annals of Statistics* **10**: 868–881. doi:10.1214/aos/1176345877.

[9] Lehmann, Erich (1986). *Testing Statistical Hypotheses* (Second ed.). (see p. 309 of Chapter 6.7 "Admissibilty", and pp. 17–18 of Chapter 1.8 "Complete Classes"

[10] Le Cam, Lucien (1986). *Asymptotic Methods in Statistical Decision Theory*. Springer-Verlag. ISBN 0-387-96307-3. (From "Chapter 12 Posterior Distributions and Bayes Solutions", p. 324)

[11] Cox, D. R. and Hinkley, D.V (1974). *Theoretical Statistics*. Chapman and Hall. ISBN 0-04-121537-0. page 432

[12] Cox, D. R. and Hinkley, D. V. (1974). *Theoretical Statistics*. Chapman and Hall. ISBN 0-04-121537-0. p. 433)

[13] Jim Albert (2009). *Bayesian Computation with R, Second edition*. New York, Dordrecht, etc.: Springer. ISBN 978-0-387-92297-3.

[14] Samuel Rathmanner and Marcus Hutter. "A Philosophical Treatise of Universal Induction". *Entropy*, 13(6):1076–1136, 2011.

[15] "The Problem of Old Evidence", in §5 of "On Universal Prediction and Bayesian Confirmation", M. Hutter - Theoretical Computer Science, 2007 - Elsevier

[16] "Raymond J. Solomonoff", Peter Gacs, Paul M. B. Vitanyi, 2011 cs.bu.edu

[17] Dawid, A. P. and Mortera, J. (1996) "Coherent Analysis of Forensic Identification Evidence". *Journal of the Royal Statistical Society*, Series B, 58, 425–443.

[18] Foreman, L. A.; Smith, A. F. M., and Evett, I. W. (1997). "Bayesian analysis of deoxyribonucleic acid profiling data in forensic identification applications (with discussion)". *Journal of the Royal Statistical Society*, Series A, 160, 429–469.

[19] Robertson, B. and Vignaux, G. A. (1995) *Interpreting Evidence: Evaluating Forensic Science in the Courtroom*. John Wiley and Sons. Chichester. ISBN 978-0-471-96026-3

[20] Dawid, A. P. (2001) "Bayes' Theorem and Weighing Evidence by Juries"; http://128.40.111.250/evidence/content/dawid-paper.pdf

[21] Gardner-Medwin, A. (2005) "What Probability Should the Jury Address?". *Significance*, 2 (1), March 2005

[22] David Miller: *Critical Rationalism*

[23] Howson & Urbach (2005), Jaynes (2003)

[24] Stigler, Stephen M. (1986). "Chapter 3". *The History of Statistics*. Harvard University Press.

[25] Fienberg, Stephen E. (2006). "When did Bayesian Inference Become 'Bayesian'?" (PDF). *Bayesian Analysis* **1** (1): 1–40 [p. 5]. doi:10.1214/06-ba101.

[26] Bernardo, José-Miguel (2005). "Reference analysis". *Handbook of statistics* **25**. pp. 17–90.

[27] Wolpert, R. L. (2004). "A Conversation with James O. Berger". *Statistical Science* **19** (1): 205–218. doi:10.1214/088342304000000053. MR 2082155.

[28] Bernardo, José M. (2006). "A Bayesian mathematical statistics primer" (PDF). *ICOTS-7*.

[29] Bishop, C. M. (2007). *Pattern Recognition and Machine Learning*. New York: Springer. ISBN 0387310738.

27.12 References

- Aster, Richard; Borchers, Brian, and Thurber, Clifford (2012). *Parameter Estimation and Inverse Problems*, Second Edition, Elsevier. ISBN 0123850487, ISBN 978-0123850485
- Bickel, Peter J. and Doksum, Kjell A. (2001). *Mathematical Statistics, Volume 1: Basic and Selected Topics* (Second (updated printing 2007) ed.). Pearson Prentice–Hall. ISBN 0-13-850363-X.
- Box, G. E. P. and Tiao, G. C. (1973) *Bayesian Inference in Statistical Analysis*, Wiley, ISBN 0-471-57428-7
- Edwards, Ward (1968). "Conservatism in Human Information Processing". In Kleinmuntz, B. *Formal Representation of Human Judgment*. Wiley.

- Edwards, Ward (1982). “Conservatism in Human Information Processing (excerpted)". In Daniel Kahneman, Paul Slovic and Amos Tversky. *Judgment under uncertainty: Heuristics and biases*. Cambridge University Press.
- Jaynes E. T. (2003) *Probability Theory: The Logic of Science*, CUP. ISBN 978-0-521-59271-0 (Link to Fragmentary Edition of March 1996).
- Howson, C. and Urbach, P. (2005). *Scientific Reasoning: the Bayesian Approach* (3rd ed.). Open Court Publishing Company. ISBN 978-0-8126-9578-6.
- Phillips, L. D.; Edwards, Ward (October 2008). “Chapter 6: Conservatism in a Simple Probability Inference Task (*Journal of Experimental Psychology* (1966) 72: 346-354)". In Jie W. Weiss and David J. Weiss. *A Science of Decision Making:The Legacy of Ward Edwards*. Oxford University Press. p. 536. ISBN 978-0-19-532298-9.

27.13 Further reading

27.13.1 Elementary

The following books are listed in ascending order of probabilistic sophistication:

- Stone, JV (2013), “Bayes’ Rule: A Tutorial Introduction to Bayesian Analysis”, Download first chapter here, Sebtel Press, England.
- Dennis V. Lindley (2013). *Understanding Uncertainty, Revised Edition* (2nd ed.). John Wiley. ISBN 978-1-118-65012-7.
- Colin Howson and Peter Urbach (2005). *Scientific Reasoning: The Bayesian Approach* (3rd ed.). Open Court Publishing Company. ISBN 978-0-8126-9578-6.
- Berry, Donald A. (1996). *Statistics: A Bayesian Perspective*. Duxbury. ISBN 0-534-23476-3.
- Morris H. DeGroot and Mark J. Schervish (2002). *Probability and Statistics* (third ed.). Addison-Wesley. ISBN 978-0-201-52488-8.
- Bolstad, William M. (2007) *Introduction to Bayesian Statistics*: Second Edition, John Wiley ISBN 0-471-27020-2
- Winkler, Robert L (2003). *Introduction to Bayesian Inference and Decision* (2nd ed.). Probabilistic. ISBN 0-9647938-4-9. Updated classic textbook. Bayesian theory clearly presented.
- Lee, Peter M. *Bayesian Statistics: An Introduction*. Fourth Edition (2012), John Wiley ISBN 978-1-1183-3257-3
- Carlin, Bradley P. and Louis, Thomas A. (2008). *Bayesian Methods for Data Analysis, Third Edition*. Boca Raton, FL: Chapman and Hall/CRC. ISBN 1-58488-697-8.
- Gelman, Andrew; Carlin, John B.; Stern, Hal S.; Dunson, David B.; Vehtari, Aki; Rubin, Donald B. (2013). *Bayesian Data Analysis, Third Edition*. Chapman and Hall/CRC. ISBN 978-1-4398-4095-5.

27.13.2 Intermediate or advanced

- Berger, James O (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics (Second ed.). Springer-Verlag. ISBN 0-387-96098-8.
- Bernardo, José M.; Smith, Adrian F. M. (1994). *Bayesian Theory*. Wiley.
- DeGroot, Morris H., *Optimal Statistical Decisions*. Wiley Classics Library. 2004. (Originally published (1970) by McGraw-Hill.) ISBN 0-471-68029-X.
- Schervish, Mark J. (1995). *Theory of statistics*. Springer-Verlag. ISBN 0-387-94546-6.
- Jaynes, E. T. (1998) *Probability Theory: The Logic of Science*.
- O'Hagan, A. and Forster, J. (2003) *Kendall’s Advanced Theory of Statistics*, Volume 2B: *Bayesian Inference*. Arnold, New York. ISBN 0-340-52922-9.
- Robert, Christian P (2001). *The Bayesian Choice – A Decision-Theoretic Motivation* (second ed.). Springer. ISBN 0-387-94296-3.
- Glenn Shafer and Pearl, Judea, eds. (1988) *Probabilistic Reasoning in Intelligent Systems*, San Mateo, CA: Morgan Kaufmann.
- Pierre Bessière et al. (2013), "Bayesian Programming", CRC Press. ISBN 9781439880326

27.14 External links

- Hazewinkel, Michiel, ed. (2001), “Bayesian approach to statistical problems”, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Bayesian Statistics from Scholarpedia.
- Introduction to Bayesian probability from Queen Mary University of London

- Mathematical Notes on Bayesian Statistics and Markov Chain Monte Carlo
- Bayesian reading list, categorized and annotated by Tom Griffiths
- A. Hajek and S. Hartmann: Bayesian Epistemology, in: J. Dancy et al. (eds.), A Companion to Epistemology. Oxford: Blackwell 2010, 93-106.
- S. Hartmann and J. Sprenger: Bayesian Epistemology, in: S. Bernecker and D. Pritchard (eds.), Routledge Companion to Epistemology. London: Routledge 2010, 609-620.
- *Stanford Encyclopedia of Philosophy*: "Inductive Logic"
- Bayesian Confirmation Theory
- What Is Bayesian Learning?

Chapter 28

Chi-squared distribution

This article is about the mathematics of the chi-squared distribution. For its uses in statistics, see chi-squared test. For the music group, see Chi2 (band).

In probability theory and statistics, the **chi-squared distribution** (also **chi-square** or **χ^2-distribution**) with *k* degrees of freedom is the distribution of a sum of the squares of *k* independent standard normal random variables. It is a special case of the gamma distribution and is one of the most widely used probability distributions in inferential statistics, e.g., in hypothesis testing or in construction of confidence intervals.[2][3][4][5] When it is being distinguished from the more general noncentral chi-squared distribution, this distribution is sometimes called the **central chi-squared distribution**.

The chi-squared distribution is used in the common chi-squared tests for goodness of fit of an observed distribution to a theoretical one, the independence of two criteria of classification of qualitative data, and in confidence interval estimation for a population standard deviation of a normal distribution from a sample standard deviation. Many other statistical tests also use this distribution, like Friedman's analysis of variance by ranks.

28.1 History and name

This distribution was first described by the German statistician Friedrich Robert Helmert in papers of 1875-6,[6][7] where he computed the sampling distribution of the sample variance of a normal population. Thus in German this was traditionally known as the *Helmert'sche* ("Helmertian") or "Helmert distribution".

The distribution was independently rediscovered by the English mathematician Karl Pearson in the context of goodness of fit, for which he developed his Pearson's chi-squared test, published in 1900, with computed table of values published in (Elderton 1902), collected in (Pearson 1914, pp. xxxi–xxxiii, 26–28, Table XII). The name "chi-squared" ultimately derives from Pearson's shorthand for the exponent in a multivariate normal distribution with the Greek letter Chi, writing $-½\chi^2$ for what would appear in modern notation as $-½\mathbf{x}^T\Sigma^{-1}\mathbf{x}$ (Σ being the covariance matrix).[8] The idea of a family of "chi-squared distributions", however, is not due to Pearson but arose as a further development due to Fisher in the 1920s.[6]

28.2 Definition

If Z_1, ..., Zk are independent, standard normal random variables, then the sum of their squares,

$$Q = \sum_{i=1}^{k} Z_i^2,$$

is distributed according to the **chi-squared distribution** with *k* degrees of freedom. This is usually denoted as

$$Q \sim \chi^2(k) \text{ or } Q \sim \chi^2_k.$$

The chi-squared distribution has one parameter: *k* — a positive integer that specifies the number of degrees of freedom (i.e. the number of *Zi*'s)

28.3 Characteristics

Further properties of the chi-squared distribution can be found in the box at the upper right corner of this article.

28.3.1 Probability density function

The probability density function (pdf) of the chi-squared distribution is

$$f(x;\,k) = \begin{cases} \frac{x^{(k/2-1)}e^{-x/2}}{2^{k/2}\Gamma\left(\frac{k}{2}\right)}, & x \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

where $\Gamma(k/2)$ denotes the Gamma function, which has closed-form values for integer *k*.

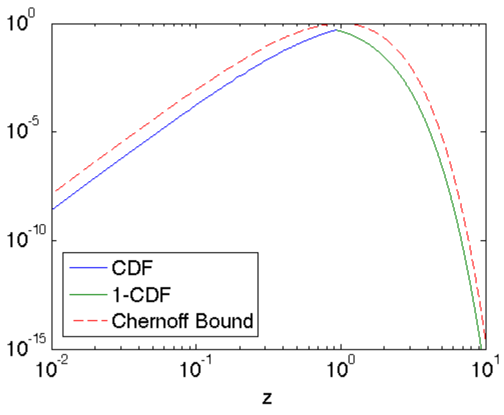
For derivations of the pdf in the cases of one, two and k degrees of freedom, see Proofs related to chi-squared distribution.

28.3.2 Differential equation

The pdf of the chi-squared distribution is a solution to the following differential equation:

$$\left\{ \begin{array}{l} 2xf'(x)+f(x)(-k+x+2)=0, \\ f(1)=\frac{2^{-k/2}}{\sqrt{e}\Gamma\left(\frac{k}{2}\right)} \end{array} \right\}$$

28.3.3 Cumulative distribution function



Chernoff bound for the CDF and tail (1-CDF) of a chi-squared random variable with ten degrees of freedom (k = *10)*

Its cumulative distribution function is:

$$F(x;\,k)=\frac{\gamma(\frac{k}{2},\,\frac{x}{2})}{\Gamma(\frac{k}{2})}=P\left(\frac{k}{2},\,\frac{x}{2}\right),$$

where γ(*s*,*t*) is the lower incomplete Gamma function and *P*(*s*,*t*) is the regularized Gamma function.

In a special case of *k* = 2 this function has a simple form:

$$F(x;\,2)=1-e^{-\frac{x}{2}}$$

and the form is not much more complicated for other small even *k*.

Tables of the chi-squared cumulative distribution function are widely available and the function is included in many spreadsheets and all statistical packages.

Letting $z\equiv x/k$, Chernoff bounds on the lower and upper tails of the CDF may be obtained.[9] For the cases when $0<z<1$ (which include all of the cases when this CDF is less than half):

$$F(zk;\,k)\leq(ze^{1-z})^{k/2}.$$

The tail bound for the cases when $z>1$, similarly, is

$$1-F(zk;\,k)\leq(ze^{1-z})^{k/2}.$$

For another approximation for the CDF modeled after the cube of a Gaussian, see under Noncentral chi-squared distribution.

28.3.4 Additivity

It follows from the definition of the chi-squared distribution that the sum of independent chi-squared variables is also chi-squared distributed. Specifically, if $\{Xi\}_{i=1}^{n}$ are independent chi-squared variables with $\{ki\}_{i=1}^{n}$ degrees of freedom, respectively, then $Y = X_1 + \cdots + Xn$ is chi-squared distributed with $k_1 + \cdots + kn$ degrees of freedom.

28.3.5 Sample mean

The sample mean of n i.i.d. chi-squared variables of degree k is distributed according to a gamma distribution with shape α and scale θ parameters: $\bar{X}=\frac{1}{n}\sum_{i=1}^{n}X_i\sim$ Gamma $(\alpha=n\,k/2,\theta=2/n)$ where $X_i\sim\chi^2(k)$

Asymptotically, given that for a scale parameter α going to infinity, a Gamma distribution converges towards a Normal distribution with expectation $\mu=\alpha\cdot\theta$ and variance $\sigma^2=\alpha\,\theta^2$, the sample mean converges towards: $\bar{X}\xrightarrow{n\to\infty}N(\mu=k,\sigma^2=2\,k/n)$

Note that we would have obtained the same result invoking instead the central limit theorem, noting that for each chi-squared variable of degree k the expectation is k , and its variance $2\,k$ (and hence the variance of the sample mean $\bar{X}$ being $\sigma^2=2\,k/n$).

28.3.6 Entropy

The differential entropy is given by

$$h=\int_{-\infty}^{\infty}f(x;\,k)\ln f(x;\,k)\,dx=\frac{k}{2}+\ln\left[2\,\Gamma\left(\frac{k}{2}\right)\right]+\left(1-\frac{k}{2}\right)\,\psi\left[\frac{k}{2}\right],$$

where $\psi(x)$ is the Digamma function.

The chi-squared distribution is the maximum entropy probability distribution for a random variate *X* for which $E(X)=k$ and $E(\ln(X))=\psi\,(k/2)+log(2)$ are fixed. Since the chi-squared is in the family of gamma distributions, this can be derived by substituting appropriate values in the Expectation of the Log moment of Gamma. For derivation from more basic principles, see the derivation in moment generating function of the sufficient statistic.

28.3.7 Noncentral moments

The moments about zero of a chi-squared distribution with k degrees of freedom are given by[10][11]

$$\mathrm{E}(X^m) = k(k+2)(k+4)\cdots(k+2m-2) = 2^m \frac{\Gamma(m+\frac{k}{2})}{\Gamma(\frac{k}{2})}.$$

28.3.8 Cumulants

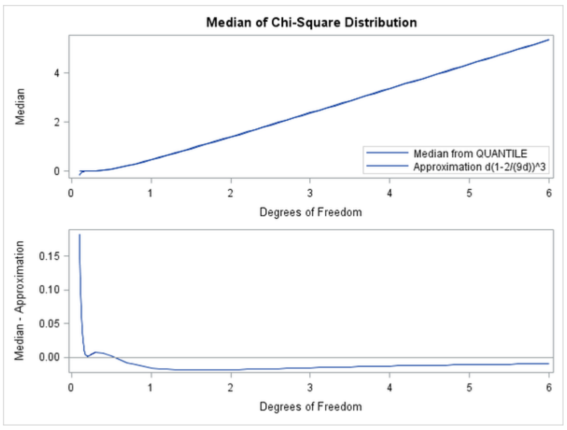
The cumulants are readily obtained by a (formal) power series expansion of the logarithm of the characteristic function:

$$\kappa_n = 2^{n-1}(n-1)!\,k$$

28.3.9 Asymptotic properties

By the central limit theorem, because the chi-squared distribution is the sum of k independent random variables with finite mean and variance, it converges to a normal distribution for large k. For many practical purposes, for $k > 50$ the distribution is sufficiently close to a normal distribution for the difference to be ignored.[12] Specifically, if $X \sim \chi^2(k)$, then as k tends to infinity, the distribution of $(X-k)/\sqrt{2k}$ tends to a standard normal distribution. However, convergence is slow as the skewness is $\sqrt{8/k}$ and the excess kurtosis is $12/k$.

- The sampling distribution of $\ln(\chi^2)$ converges to normality much faster than the sampling distribution of χ^2,[13] as the logarithm removes much of the asymmetry.[14] Other functions of the chi-squared distribution converge more rapidly to a normal distribution. Some examples are:
- If $X \sim \chi^2(k)$ then $\sqrt{2X}$ is approximately normally distributed with mean $\sqrt{2k-1}$ and unit variance (result credited to R. A. Fisher).
- If $X \sim \chi^2(k)$ then $\sqrt[3]{X/k}$ is approximately normally distributed with mean $1-2/(9k)$ and variance $2/(9k)$.[15] This is known as the Wilson–Hilferty transformation.



Approximate formula for median compared with numerical quantile (top). Difference between numerical quantile and approximate formula (bottom).

28.4 Relation to other distributions

- As $k \to \infty$, $(\chi^2_k - k)/\sqrt{2k} \xrightarrow{d} N(0,1)$ (normal distribution)
- $\chi^2_k \sim {\chi'}^2_k(0)$ (Noncentral chi-squared distribution with non-centrality parameter $\lambda = 0$)
- If $X \sim \mathrm{F}(\nu_1, \nu_2)$ then $Y = \lim_{\nu_2\to\infty} \nu_1 X$ has the chi-squared distribution $\chi^2_{\nu_1}$
- As a special case, if $X \sim \mathrm{F}(1, \nu_2)$ then $Y = \lim_{\nu_2\to\infty} X$ has the chi-squared distribution χ^2_1
- $\|\boldsymbol{N}_{i=1,\ldots,k}(0,1)\|^2 \sim \chi^2_k$ (The squared norm of **k** standard normally distributed variables is a chi-squared distribution with **k** degrees of freedom)
- If $X \sim \chi^2(\nu)$ and $c > 0$, then $cX \sim \Gamma(k = \nu/2, \theta = 2c)$. (gamma distribution)
- If $X \sim \chi^2_k$ then $\sqrt{X} \sim \chi_k$ (chi distribution)
- If $X \sim \chi^2(2)$, then $X \sim \mathrm{Exp}(1/2)$ is an exponential distribution. (See Gamma distribution for more.)
- If $X \sim \mathrm{Rayleigh}(1)$ (Rayleigh distribution) then $X^2 \sim \chi^2(2)$
- If $X \sim \mathrm{Maxwell}(1)$ (Maxwell distribution) then $X^2 \sim \chi^2(3)$
- If $X \sim \chi^2(\nu)$ then $\frac{1}{X} \sim \text{Inv-}\chi^2(\nu)$ (Inverse-chi-squared distribution)
- The chi-squared distribution is a special case of type 3 Pearson distribution

- If $X \sim \chi^2(\nu_1)$ and $Y \sim \chi^2(\nu_2)$ are independent then $\frac{X}{X+Y} \sim \text{Beta}(\frac{\nu_1}{2}, \frac{\nu_2}{2})$ (beta distribution)
- If $X \sim \text{U}(0,1)$ (uniform distribution) then $-2\log(X) \sim \chi^2(2)$
- $\chi^2(6)$ is a transformation of Laplace distribution
- If $X_i \sim \text{Laplace}(\mu, \beta)$ then $\sum_{i=1}^n \frac{2|X_i-\mu|}{\beta} \sim \chi^2(2n)$
- chi-squared distribution is a transformation of Pareto distribution
- Student's t-distribution is a transformation of chi-squared distribution
- Student's t-distribution can be obtained from chi-squared distribution and normal distribution
- Noncentral beta distribution can be obtained as a transformation of chi-squared distribution and Noncentral chi-squared distribution
- Noncentral t-distribution can be obtained from normal distribution and chi-squared distribution

A chi-squared variable with *k* degrees of freedom is defined as the sum of the squares of *k* independent standard normal random variables.

If *Y* is a *k*-dimensional Gaussian random vector with mean vector *μ* and rank *k* covariance matrix *C*, then $X = (Y-\mu)^{\mathrm{T}}C^{-1}(Y-\mu)$ is chi-squared distributed with *k* degrees of freedom.

The sum of squares of statistically independent unit-variance Gaussian variables which do *not* have mean zero yields a generalization of the chi-squared distribution called the noncentral chi-squared distribution.

If *Y* is a vector of *k* i.i.d. standard normal random variables and *A* is a $k \times k$ symmetric, idempotent matrix with rank $k-n$ then the quadratic form Y^TAY is chi-squared distributed with $k-n$ degrees of freedom.

The chi-squared distribution is also naturally related to other distributions arising from the Gaussian. In particular,

- *Y* is F-distributed, $Y \sim F(k_1,k_2)$ if $Y=\frac{X_1/k_1}{X_2/k_2}$ where $X_1 \sim \chi^2(k_1)$ and $X_2 \sim \chi^2(k_2)$ are statistically independent.
- If *X* is chi-squared distributed, then $\sqrt{X}$ is chi distributed.
- If $X_1 \sim \chi^2 k_1$ and $X_2 \sim \chi^2 k_2$ are statistically independent, then $X_1 + X_2 \sim \chi^2 k_1{+}k_2$. If X_1 and X_2 are not independent, then $X_1 + X_2$ is not chi-squared distributed.

28.5 Generalizations

The chi-squared distribution is obtained as the sum of the squares of *k* independent, zero-mean, unit-variance Gaussian random variables. Generalizations of this distribution can be obtained by summing the squares of other types of Gaussian random variables. Several such distributions are described below.

28.5.1 Linear combination

If $X_1, ..., X_n$ are chi square random variables and $a_1, ..., a_n \in \mathbb{R}_{>0}$, then a closed expression for the distribution of $X = \sum_{i=1}^n a_i X_i$ is not known. It may be, however, calculated using the property of characteristic functions of the chi-squared random variable.[16]

28.5.2 Chi-squared distributions

Noncentral chi-squared distribution

Main article: Noncentral chi-squared distribution

The noncentral chi-squared distribution is obtained from the sum of the squares of independent Gaussian random variables having unit variance and *nonzero* means.

Generalized chi-squared distribution

Main article: Generalized chi-squared distribution

The generalized chi-squared distribution is obtained from the quadratic form *z′Az* where *z* is a zero-mean Gaussian vector having an arbitrary covariance matrix, and *A* is an arbitrary matrix.

28.5.3 Gamma, exponential, and related distributions

The chi-squared distribution $X \sim \boldsymbol{\chi^2(k)}$ is a special case of the gamma distribution, in that $X \sim \Gamma(k/2, 1/2)$ using the rate parameterization of the gamma distribution (or $X \sim \boldsymbol{\Gamma(k/2, 2)}$ using the scale parameterization of the gamma distribution) where *k* is an integer.

Because the exponential distribution is also a special case of the Gamma distribution, we also have that if $X \sim \chi^2(2)$, then $X \sim \text{Exp}(1/2)$ is an exponential distribution.

The Erlang distribution is also a special case of the Gamma distribution and thus we also have that if $X \sim \chi^2(k)$ with even *k*, then *X* is Erlang distributed with shape parameter *k*/2 and scale parameter 1/2.

28.6 Applications

The chi-squared distribution has numerous applications in inferential statistics, for instance in chi-squared tests and in estimating variances. It enters the problem of estimating the mean of a normally distributed population and the problem of estimating the slope of a regression line via its role in Student's t-distribution. It enters all analysis of variance problems via its role in the F-distribution, which is the distribution of the ratio of two independent chi-squared random variables, each divided by their respective degrees of freedom.

Following are some of the most common situations in which the chi-squared distribution arises from a Gaussian-distributed sample.

- if X_1, ..., Xn are i.i.d. $N(\mu, \sigma^2)$ random variables, then $\sum_{i=1}^n (X_i - \bar{X})^2 \sim \sigma^2 \chi^2_{n-1}$ where $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$.
- The box below shows some statistics based on $Xi \sim$ Normal(μi, $\sigma^2 i$), $i = 1, \cdots, k$, independent random variables that have probability distributions related to the chi-squared distribution:

The chi-squared distribution is also often encountered in Magnetic Resonance Imaging .[17]

28.7 Table of χ^2 value vs p-value

The p-value is the probability of observing a test statistic *at least* as extreme in a chi-squared distribution. Accordingly, since the cumulative distribution function (CDF) for the appropriate degrees of freedom *(df)* gives the probability of having obtained a value *less extreme* than this point, subtracting the CDF value from 1 gives the p-value. The table below gives a number of p-values matching to χ^2 for the first 10 degrees of freedom.

A low p-value indicates greater statistical significance, i.e. greater confidence that the observed deviation from the null hypothesis is significant. A p-value of 0.05 is often used as a bright-line cutoff between significant and not-significant results.

28.8 See also

- Cochran's theorem
- F-distribution
- Fisher's method for combining independent tests of significance
- Gamma distribution
- Generalized chi-squared distribution
- Noncentral chi-squared distribution
- Hotelling's T-squared distribution
- Pearson's chi-squared test
- Student's t-distribution
- Wilks' lambda distribution
- Wishart distribution

28.9 References

[1] M.A. Sanders. "Characteristic function of the central chi-squared distribution" (PDF). Retrieved 2009-03-06.

[2] Abramowitz, Milton; Stegun, Irene A., eds. (1965), "Chapter 26", *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, New York: Dover, p. 940, ISBN 978-0486612720, MR 0167642.

[3] NIST (2006). Engineering Statistics Handbook - Chi-Squared Distribution

[4] Jonhson, N. L.; Kotz, S.; Balakrishnan, N. (1994). "Chi-Squared Distributions including Chi and Rayleigh". *Continuous Univariate Distributions* **1** (Second ed.). John Wiley and Sons. pp. 415–493. ISBN 0-471-58495-9.

[5] Mood, Alexander; Graybill, Franklin A.; Boes, Duane C. (1974). *Introduction to the Theory of Statistics* (Third ed.). McGraw-Hill. pp. 241–246. ISBN 0-07-042864-6.

[6] Hald 1998, pp. 633–692, 27. Sampling Distributions under Normality.

[7] F. R. Helmert, "Ueber die Wahrscheinlichkeit der Potenzsummen der Beobachtungsfehler und über einige damit im Zusammenhange stehende Fragen", *Zeitschrift für Mathematik und Physik* 21, 1876, S. 102–219

[8] R. L. Plackett, *Karl Pearson and the Chi-Squared Test*, International Statistical Review, 1983, 61f. See also Jeff Miller, Earliest Known Uses of Some of the Words of Mathematics.

[9] Dasgupta, Sanjoy D. A.; Gupta, Anupam K. (2002). "An Elementary Proof of a Theorem of Johnson and Lindenstrauss" (PDF). *Random Structures and Algorithms* **22**: 60–65. doi:10.1002/rsa.10073. Retrieved 2012-05-01.

[10] Chi-squared distribution, from MathWorld, retrieved Feb. 11, 2009

[11] M. K. Simon, *Probability Distributions Involving Gaussian Random Variables*, New York: Springer, 2002, eq. (2.35), ISBN 978-0-387-34657-1

[12] Box, Hunter and Hunter (1978). *Statistics for experimenters*. Wiley. p. 118. ISBN 0471093157.

[13] Bartlett, M. S.; Kendall, D. G. (1946). "The Statistical Analysis of Variance-Heterogeneity and the Logarithmic Transformation". *Supplement to the Journal of the Royal Statistical Society* **8** (1): 128–138. JSTOR 2983618.

[14] Shoemaker, Lewis H. (2003). "Fixing the F Test for Equal Variances". *The American Statistician* **57** (2): 105–114. doi:10.1198/0003130031441. JSTOR 30037243.

[15] Wilson, E. B.; Hilferty, M. M. (1931). "The distribution of chi-squared" (PDF). *Proc. Natl. Acad. Sci. USA* **17** (12): 684–688.

[16] Davies, R.B. (1980). "Algorithm AS155: The Distributions of a Linear Combination of χ^2 Random Variables". *Journal of the Royal Statistical Society* **29** (3): 323–333. doi:10.2307/2346911.

[17] den Dekker A. J., Sijbers J., (2014) "Data distributions in magnetic resonance images: a review", *Physica Medica*,

[18] Chi-Squared Test Table B.2. Dr. Jacqueline S. McLaughlin at The Pennsylvania State University. In turn citing: R.A. Fisher and F. Yates, Statistical Tables for Biological Agricultural and Medical Research, 6th ed., Table IV

28.10 Further reading

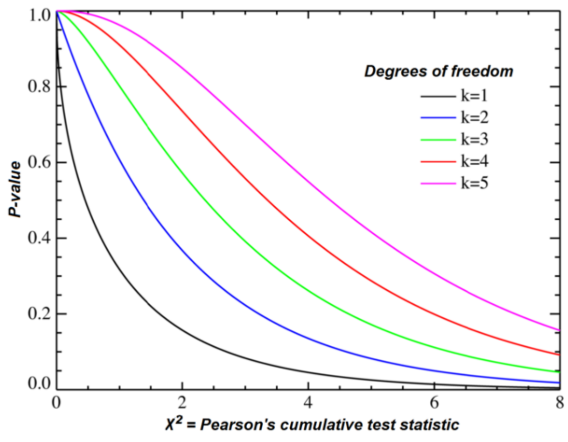
- Hald, Anders (1998). *A history of mathematical statistics from 1750 to 1930*. New York: Wiley. ISBN 0-471-17912-4.
- Elderton, William Palin (1902). "Tables for Testing the Goodness of Fit of Theory to Observation". *Biometrika* **1** (2): 155–163. doi:10.1093/biomet/1.2.155.

28.11 External links

- Hazewinkel, Michiel, ed. (2001), "Chi-squared distribution", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Calculator for the pdf, cdf and quantiles of the chi-squared distribution
- Earliest Uses of Some of the Words of Mathematics: entry on Chi squared has a brief history
- Course notes on Chi-Squared Goodness of Fit Testing from Yale University Stats 101 class.
- *Mathematica* demonstration showing the chi-squared sampling distribution of various statistics, e.g. Σx^2, for a normal population
- Simple algorithm for approximating cdf and inverse cdf for the chi-squared distribution with a pocket calculator

Chapter 29

Chi-squared test



Chi-square distribution, showing X^2 *on the x-axis and P-value on the y-axis.*

A **chi-squared test**, also referred to as χ^2 **test** (or **chi-square test**), is any statistical hypothesis test in which the sampling distribution of the test statistic is a chi-square distribution when the null hypothesis is true. Chi-squared tests are often constructed from a sum of squared errors, or through the sample variance. Test statistics that follow a chi-squared distribution arise from an assumption of independent normally distributed data, which is valid in many cases due to the central limit theorem. A chi-squared test can then be used to reject the hypothesis that the data are independent.

Also considered a chi-square test is a test in which this is *asymptotically* true, meaning that the sampling distribution (if the null hypothesis is true) can be made to approximate a chi-square distribution as closely as desired by making the sample size large enough. The chi-squared test is used to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more categories. Does the number of individuals or objects that fall in each category differ significantly from the number you would expect? Is this difference between the expected and observed due to sampling variation, or is it a real difference?

29.1 Examples of chi-square tests with samples

One test statistic that follows a chi-square distribution exactly is the test that the variance of a normally distributed population has a given value based on a sample variance. Such tests are uncommon in practice because the true variance of the population is usually unknown. However, there are several statistical tests where the chi-square distribution is approximately valid:

29.1.1 Pearson's chi-square test

Main article: Pearson's chi-square test

Pearson's chi-square test, also known as the chi-square goodness-of-fit test or chi-square test for independence. When the chi-square test is mentioned without any modifiers or without other precluding context, this test is often meant (for an exact test used in place of χ^2 , see Fisher's exact test).

29.1.2 Yates's correction for continuity

Main article: Yates's correction for continuity

Using the chi-square distribution to interpret Pearson's chi-square statistic requires one to assume that the discrete probability of observed binomial frequencies in the table can be approximated by the continuous chi-square distribution. This assumption is not quite correct, and introduces some error.

To reduce the error in approximation, Frank Yates suggested a correction for continuity that adjusts the formula for Pearson's chi-square test by subtracting 0.5 from the difference between each observed value and its expected value in a 2×2 contingency table.[1] This reduces the chi-square value obtained and thus increases its p-value.

29.1.3 Other chi-square tests

- Cochran–Mantel–Haenszel chi-squared test.
- McNemar's test, used in certain 2 × 2 tables with pairing
- Tukey's test of additivity
- The portmanteau test in time-series analysis, testing for the presence of autocorrelation
- Likelihood-ratio tests in general statistical modelling, for testing whether there is evidence of the need to move from a simple model to a more complicated one (where the simple model is nested within the complicated one).

29.2 Chi-squared test for variance in a normal population

If a sample of size *n* is taken from a population having a normal distribution, then there is a result (see distribution of the sample variance) which allows a test to be made of whether the variance of the population has a predetermined value. For example, a manufacturing process might have been in stable condition for a long period, allowing a value for the variance to be determined essentially without error. Suppose that a variant of the process is being tested, giving rise to a small sample of *n* product items whose variation is to be tested. The test statistic T in this instance could be set to be the sum of squares about the sample mean, divided by the nominal value for the variance (i.e. the value to be tested as holding). Then T has a chi-square distribution with $n-1$ degrees of freedom. For example if the sample size is 21, the acceptance region for T for a significance level of 5% is the interval 9.59 to 34.17.

29.3 Example chi-squared test for categorical data

Suppose there is a city of 1 million residents with four neighborhoods: A, B, C, and D. A random sample of 650 residents of the city is taken and their occupation is recorded as "blue collar", "white collar", or "service". The null hypothesis is that each person's neighborhood of residence is independent of the person's occupational classification. The data are tabulated as:

Let us take the sample living in neighborhood A, 150/650, to estimate what proportion of the whole 1 million people live in neighborhood A. Similarly we take 349/650 to estimate what proportion of the 1 million people are blue-collar workers. By the assumption of independence under the hypothesis we should "expect" the number of blue-collar workers in neighborhood A to be

$$\frac{150}{650} \times \frac{349}{650} \times 650 \approx 80.54.$$

Then in that "cell" of the table, we have

$$\frac{(\text{observed} - \text{expected})^2}{\text{expected}} = \frac{(90 - 80.54)^2}{80.54}.$$

The sum of these quantities over all of the cells is the test statistic. Under the null hypothesis, it has approximately a chi-square distribution whose number of degrees of freedom is

$$(\text{rows of number}-1)(\text{columns of number}-1) = (3-1)(4-1) = 6.$$

If the test statistic is improbably large according to that chi-square distribution, then one rejects the null hypothesis of **independence**.

A related issue is a test of **homogeneity**. Suppose that instead of giving every resident of each of the four neighborhoods an equal chance of inclusion in the sample, we decide in advance how many residents of each neighborhood to include. Then each resident has the same chance of being chosen as do all residents of the same neighborhood, but residents of different neighborhoods would have different probabilities of being chosen if the four sample sizes are not proportional to the populations of the four neighborhoods. In such a case, we would be testing "homogeneity" rather than "independence". The question is whether the proportions of blue-collar, white-collar, and service workers in the four neighborhoods are the same. However, the test is done in the same way.

29.4 Applications

In cryptanalysis, chi-square test is used to compare the distribution of plaintext and (possibly) decrypted ciphertext. The lowest value of the test means that the decryption was successful with high probability.[2][3] This method can be generalized for solving modern cryptographic problems.[4]

29.5 See also

- Chi-square test nomogram
- *G*-test
- Minimum chi-square estimation
- The Wald test can be evaluated against a chi-square distribution.

29.6 References

[1] Yates, F (1934). "Contingency table involving small numbers and the χ^2 test". *Supplement to the Journal of the Royal Statistical Society* **1**(2): 217–235. JSTOR 2983604

[2] "Chi-squared Statistic". *Practical Cryptography*. Retrieved 18 February 2015.

[3] "Using Chi Squared to Crack Codes". *IB Maths Resources*. British International School Phuket.

[4] Ryabko, B.Ya.; Stognienko, V.S.; Shokin, Yu.I. (2004). "A new test for randomness and its application to some cryptographic problems" (PDF). *Journal of Statistical Planning and Inference* **123**: 365–376. Retrieved 18 February 2015.

- Weisstein, Eric W., "Chi-Squared Test", *MathWorld*.
- Corder, G.W. & Foreman, D.I. (2014). *Nonparametric Statistics: A Step-by-Step Approach*. Wiley, New York. ISBN 978-1118840313
- Greenwood, P.E., Nikulin, M.S. (1996) *A guide to chi-squared testing*. Wiley, New York. ISBN 0-471-55779-X
- Nikulin, M.S. (1973). "Chi-squared test for normality". In: *Proceedings of the International Vilnius Conference on Probability Theory and Mathematical Statistics*, v.2, pp. 119–122.
- Bagdonavicius, V., Nikulin, M.S. (2011) "Chi-squared goodness-of-fit test for right censored data". *The International Journal of Applied Mathematics and Statistics*, p. 30-50.

Chapter 30

Goodness of fit

The **goodness of fit** of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question. Such measures can be used in statistical hypothesis testing, e.g. to test for normality of residuals, to test whether two samples are drawn from identical distributions (see Kolmogorov–Smirnov test), or whether outcome frequencies follow a specified distribution (see Pearson's chi-squared test). In the analysis of variance, one of the components into which the variance is partitioned may be a lack-of-fit sum of squares.

30.1 Fit of distributions

In assessing whether a given distribution is suited to a data-set, the following tests and their underlying measures of fit can be used:

- Kolmogorov–Smirnov test;
- Cramér–von Mises criterion;
- Anderson–Darling test;
- Shapiro–Wilk test;
- Chi Square test;
- Akaike information criterion;
- Hosmer–Lemeshow test;

30.2 Regression analysis

In regression analysis, the following topics relate to goodness of fit:

- Coefficient of determination (The R squared measure of goodness of fit);
- Lack-of-fit sum of squares.

30.2.1 Example

One way in which a measure of goodness of fit statistic can be constructed, in the case where the variance of the measurement error is known, is to construct a weighted sum of squared errors:

$$\chi^2 = \sum \frac{(O-E)^2}{\sigma^2}$$

where σ^2 is the known variance of the observation, O is the observed data and E is the theoretical data.[1] This definition is only useful when one has estimates for the error on the measurements, but it leads to a situation where a chi-squared distribution can be used to test goodness of fit, provided that the errors can be assumed to have a normal distribution.

The reduced chi-squared statistic is simply the chi-squared divided by the number of degrees of freedom:[1][2][3][4]

$$\chi^2_{\text{red}} = \frac{\chi^2}{\nu} = \frac{1}{\nu} \sum \frac{(O-E)^2}{\sigma^2}$$

where ν is the number of degrees of freedom, usually given by $N - n - 1$, where N is the number of observations, and n is the number of fitted parameters, assuming that the mean value is an additional fitted parameter. The advantage of the reduced chi-squared is that it already normalizes for the number of data points and model complexity. This is also known as the mean square weighted deviation.

As a rule of thumb (again valid only when the variance of the measurement error is known *a priori* rather than estimated from the data), a $\chi^2_{\text{red}} \gg 1$ indicates a poor model fit. A $\chi^2_{\text{red}} > 1$ indicates that the fit has not fully captured the data (or that the error variance has been underestimated). In principle, a value of $\chi^2_{\text{red}} = 1$ indicates that the extent of the match between observations and estimates is in accord with the error variance. A $\chi^2_{\text{red}} < 1$ indicates that the model is 'over-fitting' the data: either the model is improperly fitting noise, or the error variance has been overestimated.[5]

30.3 Categorical data

The following are examples that arise in the context of categorical data.

30.3.1 Pearson's chi-squared test

Pearson's chi-squared test uses a measure of goodness of fit which is the sum of differences between observed and expected outcome frequencies (that is, counts of observations), each squared and divided by the expectation:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)}{E_i}^2$$

where:

Oi = an observed frequency (i.e. count) for bin *i*

Ei = an expected (theoretical) frequency for bin *i*, asserted by the null hypothesis.

The expected frequency is calculated by:

$$E_i = \Big(F(Y_u) - F(Y_l)\Big) N$$

where:

F = the cumulative Distribution function for the distribution being tested.

Yu = the upper limit for class *i*,

Yl = the lower limit for class *i*, and

N = the sample size

The resulting value can be compared to the chi-squared distribution to determine the goodness of fit. In order to determine the degrees of freedom of the chi-squared distribution, one takes the total number of observed frequencies and subtracts the number of estimated parameters. The test statistic follows, approximately, a chi-square distribution with (k − c) degrees of freedom where k is the number of non-empty cells and c is the number of estimated parameters (including location and scale parameters and shape parameters) for the distribution.

Example: equal frequencies of men and women

For example, to test the hypothesis that a random sample of 100 people has been drawn from a population in which men and women are equal in frequency, the observed number of men and women would be compared to the theoretical frequencies of 50 men and 50 women. If there were 44 men in the sample and 56 women, then

$$\chi^2 = \frac{(44-50)^2}{50} + \frac{(56-50)^2}{50} = 1.44$$

If the null hypothesis is true (i.e., men and women are chosen with equal probability in the sample), the test statistic will be drawn from a chi-squared distribution with one degree of freedom. Though one might expect two degrees of freedom (one each for the men and women), we must take into account that the total number of men and women is constrained (100), and thus there is only one degree of freedom (2 − 1). Alternatively, if the male count is known the female count is determined, and vice versa.

Consultation of the chi-squared distribution for 1 degree of freedom shows that the probability of observing this difference (or a more extreme difference than this) if men and women are equally numerous in the population is approximately 0.23. This probability is higher than conventional criteria for statistical significance (.001-.05), so normally we would not reject the null hypothesis that the number of men in the population is the same as the number of women (i.e. we would consider our sample within the range of what we'd expect for a 50/50 male/female ratio.)

30.3.2 Binomial case

A binomial experiment is a sequence of independent trials in which the trials can result in one of two outcomes, success or failure. There are n trials each with probability of success, denoted by p. Provided that $np_i \gg 1$ for every i (where i = 1, 2, ..., k), then

$$\chi^2 = \sum_{i=1}^{k} \frac{(N_i - np_i)^2}{np_i} = \sum_{\text{all cells}} \frac{(\mathrm{O}-\mathrm{E})^2}{\mathrm{E}}.$$

This has approximately a chi-squared distribution with k − 1 df. The fact that df = k − 1 is a consequence of the restriction $\sum N_i = n$. We know there are k observed cell counts, however, once any k − 1 are known, the remaining one is uniquely determined. Basically, one can say, there are only k − 1 freely determined cell counts, thus df = k − 1.

30.4 Other measures of fit

The likelihood ratio test statistic is a measure of the goodness of fit of a model, judged by whether an expanded form of the model provides a substantially improved fit.

30.5 See also

- Deviance (statistics) (related to GLM)

- Overfitting

30.6 References

[1] Laub, Charlie; Kuhl, Tonya L. (n.d.), *How Bad is Good? A Critical Look at the Fitting of Reflectivity Models using the Reduced Chi-Square Statistic* (PDF), University California, Davis, retrieved 30 May 2015

[2] Taylor, John Robert (1997), *An introduction to error analysis*, University Science Books, p. 268

[3] Kirkman, T. W. (n.d.), *Chi-Squared Curve Fitting*, retrieved 30 May 2015

[4] Glover, David M.; Jenkins, William J.; Doney, Scott C. (2008), *Least Squares and regression techniques, goodness of fit and tests, non-linear least squares techniques*, Woods Hole Oceanographic Institute

[5] Bevington, Philip R. (1969), *Data Reduction and Error Analysis for the Physical Sciences*, New York: McGraw-Hill, p. 89, For χ^2 tests, $\chi\nu^2$ should be approximately equal to one.

Chapter 31

Likelihood-ratio test

Not to be confused with the use of likelihood ratios in diagnostic testing.

In statistics, a **likelihood ratio test** is a statistical test used to compare the goodness of fit of two models, one of which (the *null model*) is a special case of the other (the *alternative model*). The test is based on the likelihood ratio, which expresses how many times more likely the data are under one model than the other. This likelihood ratio, or equivalently its logarithm, can then be used to compute a *p*-value, or compared to a critical value to decide whether to reject the null model in favour of the alternative model. When the logarithm of the likelihood ratio is used, the statistic is known as a **log-likelihood ratio statistic**, and the probability distribution of this test statistic, assuming that the null model is true, can be approximated using **Wilks's theorem**.

In the case of distinguishing between two models, each of which has no unknown parameters, use of the likelihood ratio test can be justified by the Neyman–Pearson lemma, which demonstrates that such a test has the highest power among all competitors.[1]

31.1 Use

Each of the two competing models, the null model and the alternative model, is separately fitted to the data and the log-likelihood recorded. The test statistic (often denoted by *D*) is twice the difference in these log-likelihoods:

$$\begin{aligned} D &= -2\ln\left(\frac{\text{model null for likelihood}}{\text{model alternative for likelihood}}\right) \\ &= -2\ln(\text{model null for likelihood}) + 2\ln(\text{model alternative for likelihood}) \end{aligned}$$

The model with more parameters will always fit at least as well (have an equal or greater log-likelihood). Whether it fits significantly better and should thus be preferred is determined by deriving the probability or *p*-value of the difference *D*. Where the null hypothesis represents a special case of the alternative hypothesis, the probability distribution of the test statistic is approximately a chi-squared distribution with degrees of freedom equal to $df2 - df1$.[2] Symbols $df1$ and $df2$ represent the number of free parameters of models 1 and 2, the null model and the alternative model, respectively.

Here is an example of use. If the null model has 1 parameter and a log-likelihood of −8024 and the alternative model has 3 parameters and a log-likelihood of −8012, then the probability of this difference is that of chi-squared value of +2·(8024 − 8012) = 24 with 3 − 1 = 2 degrees of freedom. Certain assumptions[3] must be met for the statistic to follow a chi-squared distribution, and often empirical *p*-values are computed.

The likelihood-ratio test requires nested models, i.e. models in which the more complex one can be transformed into the simpler model by imposing a set of constraints on the parameters. If the models are not nested, then a generalization of the likelihood-ratio test can usually be used instead: the relative likelihood.

31.2 Simple-vs-simple hypotheses

Main article: Neyman–Pearson lemma

A statistical model is often a parametrized family of probability density functions or probability mass functions $f(x|\theta)$. A simple-vs-simple hypothesis test has completely specified models under both the null and alternative hypotheses, which for convenience are written in terms of fixed values of a notional parameter θ :

$$\begin{aligned} H_0 &: \quad \theta = \theta_0, \\ H_1 &: \quad \theta = \theta_1. \end{aligned}$$

Note that under either hypothesis, the distribution of the data is fully specified; there are no unknown parameters to estimate. The likelihood ratio test is based on the **likelihood ratio**, which is often denoted by Λ (the capital Greek letter lambda). The likelihood ratio is defined as follows:[4][5]

$$\Lambda(x) = \frac{L(\theta_0|x)}{L(\theta_1|x)} = \frac{f(\cup_i x_i|\theta_0)}{f(\cup_i x_i|\theta_1)}$$

or

$$\Lambda(x) = \frac{L(\theta_0 \mid x)}{\sup\{\, L(\theta \mid x) : \theta \in \{\theta_0, \theta_1\}\}},$$

where $L(\theta|x)$ is the likelihood function, and sup is the supremum function. Note that some references may use the reciprocal as the definition.[6] In the form stated here, the likelihood ratio is small if the alternative model is better than the null model and the likelihood ratio test provides the decision rule as follows:

If $\Lambda > c$, do not reject H_0 ;

If $\Lambda < c$, reject H_0 ;

Reject with probability q if $\Lambda = c.$

The values $c, \;\; q$ are usually chosen to obtain a specified significance level α , through the relation $q \cdot P(\Lambda = c \mid H_0) + P(\Lambda < c \mid H_0) = \alpha$. The Neyman-Pearson lemma states that this likelihood ratio test is the most powerful among all level α tests for this problem.[1]

31.3 Definition (likelihood ratio test for composite hypotheses)

A null hypothesis is often stated by saying the parameter θ is in a specified subset Θ_0 of the parameter space Θ .

$$\begin{aligned} H_0 &: \quad \theta \in \Theta_0 \\ H_1 &: \quad \theta \in \Theta_0^{\complement} \end{aligned}$$

The likelihood function is $L(\theta|x) = f(x|\theta)$ (with $f(x|\theta)$ being the pdf or pmf), which is a function of the parameter θ with x held fixed at the value that was actually observed, *i.e.*, the data. The **likelihood ratio test statistic** is [7]

$$\Lambda(x) = \frac{\sup\{\, L(\theta \mid x) : \theta \in \Theta_0 \,\}}{\sup\{\, L(\theta \mid x) : \theta \in \Theta \,\}}.$$

Here, the sup notation refers to the supremum function.

A **likelihood ratio test** is any test with critical region (or rejection region) of the form $\{x|\Lambda \leq c\}$ where c is any number satisfying $0 \leq c \leq 1$. Many common test statistics such as the *Z*-test, the *F*-test, Pearson's chi-squared test and the *G*-test are tests for nested models and can be phrased as log-likelihood ratios or approximations thereof.

31.3.1 Interpretation

Being a function of the data x , the likelihood ratio is therefore a statistic. The **likelihood ratio test** rejects the null hypothesis if the value of this statistic is too small. How small is too small depends on the significance level of the test, *i.e.*, on what probability of Type I error is considered tolerable ("Type I" errors consist of the rejection of a null hypothesis that is true).

The numerator corresponds to the maximum likelihood of an observed outcome under the null hypothesis. The denominator corresponds to the maximum likelihood of an observed outcome varying parameters over the whole parameter space. The numerator of this ratio is less than the denominator. The likelihood ratio hence is between 0 and 1. Low values of the likelihood ratio mean that the observed result was less likely to occur under the null hypothesis as compared to the alternative. High values of the statistic mean that the observed outcome was nearly as likely to occur under the null hypothesis as the alternative, and the null hypothesis cannot be rejected.

31.3.2 Distribution: Wilks's theorem

If the distribution of the likelihood ratio corresponding to a particular null and alternative hypothesis can be explicitly determined then it can directly be used to form decision regions (to accept/reject the null hypothesis). In most cases, however, the exact distribution of the likelihood ratio corresponding to specific hypotheses is very difficult to determine. A convenient result, attributed to Samuel S. Wilks, says that as the sample size n approaches ∞ , the test statistic $-2\log(\Lambda)$ for a nested model will be asymptotically χ^2 -distributed with degrees of freedom equal to the difference in dimensionality of Θ and Θ_0 .[3] This means that for a great variety of hypotheses, a practitioner can compute the likelihood ratio Λ for the data and compare $-2\log(\Lambda)$ to the χ^2 value corresponding to a desired statistical significance as an approximate statistical test.

31.4 Examples

31.4.1 Coin tossing

An example, in the case of Pearson's test, we might try to compare two coins to determine whether they have the same probability of coming up heads. Our observation can be put into a contingency table with rows corresponding to the coin and columns corresponding to heads or tails. The elements of the contingency table will be the number of times the coin for that row came up heads or tails. The contents of this table are our observation X .

Here Θ consists of the possible combinations of values of the parameters p_{1H} , p_{1T} , p_{2H} , and p_{2T} , which are the probability that coins 1 and 2 come up heads or tails. In what follows, $i = 1, 2$ and $j = H, T$. The hypothesis space H is constrained by the usual constraints on a probability distribution, $0 \leq p_{ij} \leq 1$, and $p_{iH} + p_{iT} = 1$. The space of the null hypothesis H_0 is the subspace where $p_{1j} = p_{2j}$. Writing n_{ij} for the best values for p_{ij} under the hypothesis H , the maximum likelihood estimate is given by

$n_{ij} = \frac{k_{ij}}{k_{iH}+k_{iT}}$.

Similarly, the maximum likelihood estimates of p_{ij} under the null hypothesis H_0 are given by

$m_{ij} = \frac{k_{1j}+k_{2j}}{k_{1H}+k_{2H}+k_{1T}+k_{2T}}$,

which does not depend on the coin i .

The hypothesis and null hypothesis can be rewritten slightly so that they satisfy the constraints for the logarithm of the likelihood ratio to have the desired nice distribution. Since the constraint causes the two-dimensional H to be reduced to the one-dimensional H_0 , the asymptotic distribution for the test will be $\chi^2(1)$, the χ^2 distribution with one degree of freedom.

For the general contingency table, we can write the log-likelihood ratio statistic as

$$-2\log\Lambda = 2\sum_{i,j} k_{ij}\log\frac{n_{ij}}{m_{ij}}.$$

31.5 References

[1] Neyman, Jerzy; Pearson, Egon S. (1933). “On the Problem of the Most Efficient Tests of Statistical Hypotheses”. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **231** (694–706): 289–337. Bibcode:1933RSPTA.231..289N. doi:10.1098/rsta.1933.0009. JSTOR 91247.

[2] Huelsenbeck, J. P.; Crandall, K. A. (1997). “Phylogeny Estimation and Hypothesis Testing Using Maximum Likelihood”. *Annual Review of Ecology and Systematics* **28**: 437–466. doi:10.1146/annurev.ecolsys.28.1.437.

[3] Wilks, S. S. (1938). “The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses”. *The Annals of Mathematical Statistics* **9**: 60–62. doi:10.1214/aoms/1177732360.

[4] Mood, A.M.; Graybill, F.A. (1963) *Introduction to the Theory of Statistics*, 2nd edition. McGraw-Hill ISBN 978-0070428638 (page 286)

[5] Stuart, A.; Ord, K.; Arnold, S. (1999) *Kendall’s Advanced Theory of Statistics, Volume 2A*, Arnold. (Chapter 22).

[6] Cox, D. R.; Hinkley, D. V. (1974). *Theoretical Statistics*. Chapman and Hall. p. 92. ISBN 0-412-12420-3.

[7] Casella, George; Berger, Roger L. (2001). *Statistical Inference* (Second ed.). p. 375. ISBN 0-534-24312-6.

31.6 External links

- Practical application of likelihood ratio test described
- R Package: Wald’s Sequential Probability Ratio Test
- Richard Lowry’s Predictive Values and Likelihood Ratios Online Clinical Calculator

Chapter 32

Statistical classification

For the unsupervised learning approach, see Cluster analysis.

In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).

In the terminology of machine learning,[1] classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

Often, the individual observations are analyzed into a set of quantifiable properties, known variously explanatory variables, *features*, etc. These properties may variously be categorical (e.g. "A", "B", "AB" or "O", for blood type), ordinal (e.g. "large", "medium" or "small"), integer-valued (e.g. the number of occurrences of a part word in an email) or real-valued (e.g. a measurement of blood pressure). Other classifiers work by comparing observations to previous observations by means of a similarity or distance function.

An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

Terminology across fields is quite varied. In statistics, where classification is often done with logistic regression or a similar procedure, the properties of observations are termed explanatory variables (or independent variables, regressors, etc.), and the categories to be predicted are known as outcomes, which are considered to be possible values of the dependent variable. In machine learning, the observations are often known as *instances*, the explanatory variables are termed *features* (grouped into a feature vector), and the possible categories to be predicted are *classes*. There is also some argument over whether classification methods that do not involve a statistical model can be considered "statistical". Other fields may use different terminology: e.g. in community ecology, the term "classification" normally refers to cluster analysis, i.e. a type of unsupervised learning, rather than the supervised learning described in this article.

32.1 Relation to other problems

Classification and clustering are examples of the more general problem of pattern recognition, which is the assignment of some sort of output value to a given input value. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence; etc.

A common subclass of classification is probabilistic classification. Algorithms of this nature use statistical inference to find the best class for a given instance. Unlike other algorithms, which simply output a "best" class, probabilistic algorithms output a probability of the instance being a member of each of the possible classes. The best class is normally then selected as the one with the highest probability. However, such an algorithm has numerous advantages over non-probabilistic classifiers:

- It can output a confidence value associated with its choice (in general, a classifier that can do this is known as a *confidence-weighted classifier*).
- Correspondingly, it can *abstain* when its confidence of choosing any particular output is too low.
- Because of the probabilities which are generated, probabilistic classifiers can be more effectively incorporated into larger machine-learning tasks, in a

way that partially or completely avoids the problem of *error propagation*.

32.2 Frequentist procedures

Early work on statistical classification was undertaken by Fisher,[2][3] in the context of two-group problems, leading to Fisher's linear discriminant function as the rule for assigning a group to a new observation.[4] This early work assumed that data-values within each of the two groups had a multivariate normal distribution. The extension of this same context to more than two-groups has also been considered with a restriction imposed that the classification rule should be linear.[4][5] Later work for the multivariate normal distribution allowed the classifier to be nonlinear:[6] several classification rules can be derived based on slight different adjustments of the Mahalanobis distance, with a new observation being assigned to the group whose centre has the lowest adjusted distance from the observation.

32.3 Bayesian procedures

Unlike frequentist procedures, Bayesian classification procedures provide a natural way of taking into account any available information about the relative sizes of the sub-populations associated with the different groups within the overall population.[7] Bayesian procedures tend to be computationally expensive and, in the days before Markov chain Monte Carlo computations were developed, approximations for Bayesian clustering rules were devised.[8]

Some Bayesian procedures involve the calculation of group membership probabilities: these can be viewed as providing a more informative outcome of a data analysis than a simple attribution of a single group-label to each new observation.

32.4 Binary and multiclass classification

Classification can be thought of as two separate problems – binary classification and multiclass classification. In binary classification, a better understood task, only two classes are involved, whereas multiclass classification involves assigning an object to one of several classes.[9] Since many classification methods have been developed specifically for binary classification, multiclass classification often requires the combined use of multiple binary classifiers.

32.5 Feature vectors

Most algorithms describe an individual instance whose category is to be predicted using a feature vector of individual, measurable properties of the instance. Each property is termed a feature, also known in statistics as an explanatory variable (or independent variable, although in general different features may or may not be statistically independent). Features may variously be binary ("male" or "female"); categorical (e.g. "A", "B", "AB" or "O", for blood type); ordinal (e.g. "large", "medium" or "small"); integer-valued (e.g. the number of occurrences of a particular word in an email); or real-valued (e.g. a measurement of blood pressure). If the instance is an image, the feature values might correspond to the pixels of an image; if the instance is a piece of text, the feature values might be occurrence frequencies of different words. Some algorithms work only in terms of discrete data and require that real-valued or integer-valued data be *discretized* into groups (e.g. less than 5, between 5 and 10, or greater than 10).

The vector space associated with these vectors is often called the *feature space*. In order to reduce the dimensionality of the feature space, a number of dimensionality reduction techniques can be employed.

32.6 Linear classifiers

A large number of algorithms for classification can be phrased in terms of a linear function that assigns a score to each possible category k by combining the feature vector of an instance with a vector of weights, using a dot product. The predicted category is the one with the highest score. This type of score function is known as a linear predictor function and has the following general form:

$$\operatorname{score}(\mathbf{X}_i, k) = \boldsymbol{\beta}_k \cdot \mathbf{X}_i,$$

where $\mathbf{X}i$ is the feature vector for instance i, $\boldsymbol{\beta}k$ is the vector of weights corresponding to category k, and score($\mathbf{X}i$, k) is the score associated with assigning instance i to category k. In discrete choice theory, where instances represent people and categories represent choices, the score is considered the utility associated with person i choosing category k.

Algorithms with this basic setup are known as linear classifiers. What distinguishes them is the procedure for determining (training) the optimal weights/coefficients and the way that the score is interpreted.

Examples of such algorithms are

- Logistic regression and Multinomial logistic regression
- Probit regression

- The perceptron algorithm
- Support vector machines
- Linear discriminant analysis.

32.7 Algorithms

Examples of classification algorithms include:

- Linear classifiers
  - Fisher's linear discriminant
  - Logistic regression
  - Naive Bayes classifier
  - Perceptron
- Support vector machines
  - Least squares support vector machines
- Quadratic classifiers
- Kernel estimation
  - k-nearest neighbor
- Boosting (meta-algorithm)
- Decision trees
  - Random forests
- Neural networks
- Learning vector quantization

32.8 Evaluation

Classifier performance depends greatly on the characteristics of the data to be classified. There is no single classifier that works best on all given problems (a phenomenon that may be explained by the no-free-lunch theorem). Various empirical tests have been performed to compare classifier performance and to find the characteristics of data that determine classifier performance. Determining a suitable classifier for a given problem is however still more an art than a science.

The measures precision and recall are popular metrics used to evaluate the quality of a classification system. More recently, receiver operating characteristic (ROC) curves have been used to evaluate the tradeoff between true- and false-positive rates of classification algorithms.

As a performance metric, the uncertainty coefficient has the advantage over simple accuracy in that it is not affected by the relative sizes of the different classes. [10] Further, it will not penalize an algorithm for simply *rearranging* the classes.

32.9 Application domains

See also: Cluster analysis § Applications

Classification has many applications. In some of these it is employed as a data mining procedure, while in others more detailed statistical modeling is undertaken.

- Computer vision
  - Medical imaging and medical image analysis
  - Optical character recognition
  - Video tracking
- Drug discovery and development
  - Toxicogenomics
  - Quantitative structure-activity relationship
- Geostatistics
- Speech recognition
- Handwriting recognition
- Biometric identification
- Biological classification
- Statistical natural language processing
- Document classification
- Internet search engines
- Credit scoring
- Pattern recognition
- Micro-array classification

32.10 See also

- Class membership probabilities
- Classification rule
- Binary classification
- Compound term processing
- Data mining
- Fuzzy logic
- Data warehouse
- Information retrieval
- Artificial intelligence
- Machine learning
- Recommender system

32.11 References

[1] Alpaydin, Ethem (2010). *Introduction to Machine Learning*. MIT Press. p. 9. ISBN 978-0-262-01243-0.

[2] Fisher R.A. (1936) " The use of multiple measurements in taxonomic problems”, *Annals of Eugenics*, 7, 179–188

[3] Fisher R.A. (1938) " The statistical utilization of multiple measurements”, *Annals of Eugenics*, 8, 376–386

[4] Gnanadesikan, R. (1977) *Methods for Statistical Data Analysis of Multivariate Observations*, Wiley. ISBN 0-471-30845-5 (p. 83–86)

[5] Rao, C.R. (1952) *Advanced Statistical Methods in Multivariate Analysis*, Wiley. (Section 9c)

[6] Anderson,T.W. (1958) *An Introduction to Multivariate Statistical Analysis*, Wiley.

[7] Binder, D.A. (1978) “Bayesian cluster analysis”, *Biometrika*, 65, 31–38.

[8] Binder, D.A. (1981) “Approximations to Bayesian clustering rules”, *Biometrika*, 68, 275–285.

[9] Har-Peled, S., Roth, D., Zimak, D. (2003) “Constraint Classification for Multiclass Classification and Ranking.” In: Becker, B., Thrun, S., Obermayer, K. (Eds) *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, MIT Press. ISBN 0-262-02550-7

[10] Peter Mills (2011). “Efficient statistical classification of satellite measurements”. *International Journal of Remote Sensing*. doi:10.1080/01431161.2010.507795.

32.12 External links

- Classifier showdown A practical comparison of classification algorithms.
- Statistical Pattern Recognition Toolbox for Matlab.
- TOOLDIAG Pattern recognition toolbox.
- Statistical classification software based on adaptive kernel density estimation.
- PAL Classification Suite written in Java.
- kNN and Potential energy (Applet), University of Leicester
- scikit-learn a widely used package in python
- Weka A java based package with an extensive variety of algorithms.

Chapter 33

Binary classification

Binary or **binomial classification** is the task of classifying the elements of a given set into two groups on the basis of a classification rule. Some typical binary classification tasks are:

- medical testing to determine if a patient has certain disease or not – the classification property is the presence of the disease;
- A “pass or fail” test method or quality control in factories; i.e. deciding if a specification has or has not been met: a Go/no go classification.
- An item may have a qualitative property; it does or does not have a specified characteristic
- information retrieval, namely deciding whether a page or an article should be in the result set of a search or not – the classification property is the relevance of the article, or the usefulness to the user.

An important point is that in many practical binary classification problems, the two groups are not symmetric – rather than overall accuracy, the relative proportion of different types of errors is of interest. For example, in medical testing, a false positive (detecting a disease when it is not present) is considered differently from a false negative (not detecting a disease when it is present).

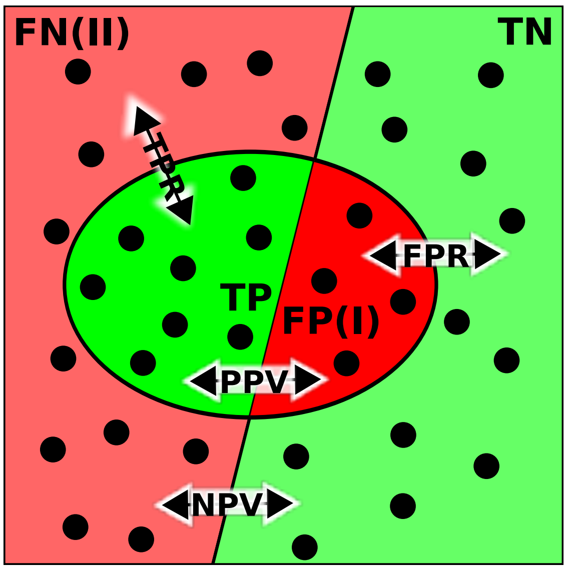
Statistical classification in general is one of the problems studied in computer science, in order to automatically learn classification systems; some methods suitable for learning binary classifiers include the decision trees, Bayesian networks, support vector machines, neural networks, probit regression, and logistic regression.

Sometimes, classification tasks are trivial. Given 100 balls, some of them red and some blue, a human with normal color vision can easily separate them into red ones and blue ones. However, some tasks, like those in practical medicine, and those interesting from the computer science point of view, are far from trivial, and may produce faulty results if executed imprecisely.

33.1 Evaluation of binary classifiers

Main article: Evaluation of binary classifiers

There are many metrics that can be used to measure the



From the contingency table you can derive four basic ratios

performance of a classifier or predictor; different fields have different preferences for specific metrics due to different goals. For example, in medicine sensitivity and specificity are often used, while in information retrieval precision and recall are preferred. An important distinction is between metrics that are independent on the prevalence (how often each category occurs in the population), and metrics that depend on the prevalence – both types are useful, but they have very different properties.

Given a classification of a specific data set, there are four basic data: the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These can be arranged into a 2×2 contingency table, with columns corresponding to actual value – condition positive (CP) or condition negative (CN) – and rows corresponding to classification value – test outcome positive or test outcome negative. There are eight basic ratios that one can compute from this table, which come in four complementary pairs (each pair summing to 1). These are obtained by dividing each of the four numbers by the sum of its row or column, yielding eight numbers, which can be referred to generically in the form "true positive row ratio" or "false negative column ratio", though there are conventional terms. There are thus two pairs of column ratios and two pairs of row ratios, and one can summarize these with four numbers by choosing one ratio from each pair – the other four numbers are the complements.

The column ratios are True Positive Rate (TPR, aka **Sensitivity** or recall), with complement the False Negative Rate (FNR); and True Negative Rate (TNR, aka **Specificity,** SPC), with complement False Positive Rate (FPR). These are the proportion of the *population with the condition* (resp., without the condition) for which the test is correct (or, complementarily, for which the test is incorrect); these are independent of prevalence.

The row ratios are Positive Predictive Value (PPV, aka precision), with complement the False Discovery Rate (FDR); and Negative Predictive Value (NPV), with complement the False Omission Rate (FOR). These are the proportion of the *population with a given test result* for which the test is correct (or, complementarily, for which the test is incorrect); these depend on prevalence.

In diagnostic testing, the main ratios used are the true column ratios – True Positive Rate and True Negative Rate – where they are known as sensitivity and specificity. In informational retrieval, the main ratios are the true positive ratios (row and column) – Positive Predictive Value and True Positive Rate – where they are known as precision and recall.

One can take ratios of a complementary pair of ratios, yielding four likelihood ratios (two column ratio of ratios, two row ratio of ratios). This is primarily done for the column (condition) ratios, yielding likelihood ratios in diagnostic testing. Taking the ratio of one of these groups of ratios yields a final ratio, the diagnostic odds ratio (DOR). This can also be defined directly as (TP×TN)/(FP×FN) = (TP/FN)/(FP/TN); this has a useful interpretation – as an odds ratio – and is prevalence-independent.

There are a number of other metrics, most simply the accuracy or Fraction Correct (FC), which measures the fraction of all instances that are correctly categorized; the complement is the Fraction Incorrect (FiC). The F-score combines precision and recall into one number via a choice of weighing, most simply equal weighing, as the balanced F-score (F1 score). Some metrics come from regression coefficients: the markedness and the informedness, and their geometric mean, the Matthews correlation coefficient. Other metrics include Youden's J statistic, the uncertainty coefficient, the Phi coefficient, and Cohen's kappa.

33.2 Converting continuous values to binary

Tests whose results are of continuous values, such as most blood values, can artificially be made binary by defining a cutoff value, with test results being designated as positive or negative depending on whether the resultant value is higher or lower than the cutoff.

However, such conversion causes a loss of information, as the resultant binary classification does not tell *how much* above or below the cutoff a value is. As a result, when converting a continuous value that is close to the cutoff to a binary one, the resultant positive or negative predictive value is generally higher than the predictive value given directly from the continuous value. In such cases, the designation of the test of being either positive or negative gives the appearance of an inappropriately high certainty, while the value is in fact in an interval of uncertainty. For example, with the urine concentration of hCG as a continuous value, a urine pregnancy test that measured 52 mIU/ml of hCG may show as "positive" with 50 mIU/ml as cutoff, but is in fact in an interval of uncertainty, which may be apparent only by knowing the original continuous value. On the other hand, a test result very far from the cutoff generally has a resultant positive or negative predictive value that is lower than the predictive value given from the continuous value. For example, a urine hCG value of 200,000 mIU/ml confers a very high probability of pregnancy, but conversion to binary values results in that it shows just as "positive" as the one of 52 mIU/ml.

33.3 See also

- Examples of Bayesian inference
- Classification rule
- Detection theory
- Kernel methods
- Matthews correlation coefficient
- Multiclass classification
- Multi-label classification
- One-class classification
- Prosecutor's fallacy
- Receiver operating characteristic
- Thresholding (image processing)
- Type I and type II errors
- Uncertainty coefficient, aka Proficiency
- Qualitative property

33.4 References

33.5 Bibliography

- Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000. ISBN 0-521-78019-5 *(SVM Book)*
- John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004. ISBN 0-521-81397-2 *(Kernel Methods Book)*
- Bernhard Schölkopf and A. J. Smola: *Learning with Kernels.* MIT Press, Cambridge, MA, 2002. *(Partly available on line: .)* ISBN 0-262-19475-9

Chapter 34

Maximum likelihood

This article is about the statistical techniques. For computer data storage, see Partial response maximum likelihood.

In statistics, **maximum-likelihood estimation** (**MLE**) is a method of estimating the parameters of a statistical model. When applied to a data set and given a statistical model, maximum-likelihood estimation provides estimates for the model's parameters.

The method of maximum likelihood corresponds to many well-known estimation methods in statistics. For example, one may be interested in the heights of adult female penguins, but be unable to measure the height of every single penguin in a population due to cost or time constraints. Assuming that the heights are normally distributed with some unknown mean and variance, the mean and variance can be estimated with MLE while only knowing the heights of some sample of the overall population. MLE would accomplish this by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable given the model.

In general, for a fixed set of data and underlying statistical model, the method of maximum likelihood selects the set of values of the model parameters that maximizes the likelihood function. Intuitively, this maximizes the "agreement" of the selected model with the observed data, and for discrete random variables it indeed maximizes the probability of the observed data under the resulting distribution. Maximum-likelihood estimation gives a unified approach to estimation, which is well-defined in the case of the normal distribution and many other problems. However, in some complicated problems, difficulties do occur: in such problems, maximum-likelihood estimators are unsuitable or do not exist.

34.1 Principles

Suppose there is a sample x_1, x_2, …, *xn* of *n* independent and identically distributed observations, coming from a distribution with an unknown probability density function $f_0(\cdot)$. It is however surmised that the function f_0 belongs to a certain family of distributions { $f(\cdot| \theta)$, $\theta \in \Theta$ } (where θ is a vector of parameters for this family), called the parametric model, so that $f_0 = f(\cdot| \theta_0)$. The value θ_0 is unknown and is referred to as the *true value* of the parameter vector. It is desirable to find an estimator $\hat{\theta}$ which would be as close to the true value θ_0 as possible. Either or both the observed variables *xi* and the parameter θ can be vectors.

To use the method of maximum likelihood, one first specifies the joint density function for all observations. For an independent and identically distributed sample, this joint density function is

$$f(x_1, x_2, \ldots, x_n \mid \theta) = f(x_1|\theta)\times f(x_2|\theta)\times\cdots\times f(x_n|\theta).$$

Now we look at this function from a different perspective by considering the observed values x_1, x_2, …, *xn* to be fixed "parameters" of this function, whereas θ will be the function's variable and allowed to vary freely; this function will be called the likelihood:

$$\mathcal{L}(\theta\,;\,x_1,\ldots,x_n) = f(x_1,x_2,\ldots,x_n \mid \theta) = \prod_{i=1}^{n} f(x_i \mid \theta).$$

Note ; denotes a separation between the two input arguments: θ and the vector-valued input $x_1,\ldots,x_n$.

In practice it is often more convenient to work with the logarithm of the likelihood function, called the **log-likelihood**:

$$\ln \mathcal{L}(\theta\,;\,x_1,\ldots,x_n) = \sum_{i=1}^{n} \ln f(x_i \mid \theta),$$

or the average log-likelihood:

$$\hat{\ell} = \frac{1}{n}\ln \mathcal{L}.$$

The hat over ℓ indicates that it is akin to some estimator. Indeed, $\hat{\ell}$ estimates the expected log-likelihood of a single observation in the model.

The method of maximum likelihood estimates θ_0 by finding a value of θ that maximizes $\hat{\ell}(\theta;x)$. This method of estimation defines a **maximum-likelihood estimator** (**MLE**) of θ_0 …

$$\{\hat{\theta}_{\text{mle}}\} \subseteq \{\underset{\theta\in\Theta}{\arg\max}\ \hat{\ell}(\theta\,;x_1,\ldots,x_n)\}.$$

… if any maximum exists. An MLE estimate is the same regardless of whether we maximize the likelihood or the log-likelihood function, since log is a strictly monotonically increasing function.

For many models, a maximum likelihood estimator can be found as an explicit function of the observed data x_1, …, xn. For many other models, however, no closed-form solution to the maximization problem is known or available, and an MLE has to be found numerically using optimization methods. For some problems, there may be multiple estimates that maximize the likelihood. For other problems, no maximum likelihood estimate exists (meaning that the log-likelihood function increases without attaining the supremum value).

In the exposition above, it is assumed that the data are independent and identically distributed. The method can be applied however to a broader setting, as long as it is possible to write the joint density function $f(x_1, \ldots, xn \mid \theta)$, and its parameter θ has a finite dimension which does not depend on the sample size n. In a simpler extension, an allowance can be made for data heterogeneity, so that the joint density is equal to $f_1(x_1|\theta) \cdot f_2(x_2|\theta) \cdot \cdots \cdot fn(xn \mid \theta)$. Put another way, we are now assuming that each observation x_i comes from a random variable that has its own distribution function f_i. In the more complicated case of time series models, the independence assumption may have to be dropped as well.

A maximum likelihood estimator coincides with the most probable Bayesian estimator given a uniform prior distribution on the parameters. Indeed, the maximum a posteriori estimate is the parameter θ that maximizes the probability of θ given the data, given by Bayes' theorem:

$$P(\theta \mid x_1, x_2, \ldots, x_n) = \frac{f(x_1, x_2, \ldots, x_n \mid \theta)P(\theta)}{P(x_1, x_2, \ldots, x_n)}$$

where $P(\theta)$ is the prior distribution for the parameter θ and where $P(x_1, x_2, \ldots, x_n)$ is the probability of the data averaged over all parameters. Since the denominator is independent of θ, the Bayesian estimator is obtained by maximizing $f(x_1, x_2, \ldots, x_n \mid \theta)P(\theta)$ with respect to θ. If we further assume that the prior $P(\theta)$ is a uniform distribution, the Bayesian estimator is obtained by maximizing the likelihood function $f(x_1, x_2, \ldots, x_n|\theta)$. Thus the Bayesian estimator coincides with the maximum-likelihood estimator for a uniform prior distribution $P(\theta)$.

34.2 Properties

A maximum-likelihood estimator is an extremum estimator obtained by maximizing, as a function of θ, the *objective function* (c.f., the loss function)

$$\hat{\ell}(\theta \mid x) = \frac{1}{n}\sum_{i=1}^{n} \ln f(x_i \mid \theta),$$

this being the sample analogue of the expected log-likelihood $\ell(\theta) = \operatorname{E}[\,\ln f(x_i \mid \theta)\,]$, where this expectation is taken with respect to the true density $f(\cdot \mid \theta_0)$.

Maximum-likelihood estimators have no optimum properties for finite samples, in the sense that (when evaluated on finite samples) other estimators may have greater concentration around the true parameter-value.[1] However, like other estimation methods, maximum-likelihood estimation possesses a number of attractive limiting properties: As the sample size increases to infinity, sequences of maximum-likelihood estimators have these properties:

- Consistency: the sequence of MLEs converges in probability to the value being estimated.
- Asymptotic normality: as the sample size increases, the distribution of the MLE tends to the Gaussian distribution with mean θ and covariance matrix equal to the inverse of the Fisher information matrix.
- Efficiency, i.e., it achieves the Cramér–Rao lower bound when the sample size tends to infinity. This means that no consistent estimator has lower asymptotic mean squared error than the MLE (or other estimators attaining this bound).
- Second-order efficiency after correction for bias.

34.2.1 Consistency

Under the conditions outlined below, the maximum likelihood estimator is **consistent**. The consistency means that having a sufficiently large number of observations n, it is possible to find the value of θ_0 with arbitrary precision. In mathematical terms this means that as n goes to infinity the estimator $\hat{\theta}$ converges in probability to its true value:

$$\hat{\theta}_{\text{mle}} \xrightarrow{p} \theta_0.$$

Under slightly stronger conditions, the estimator converges almost surely (or *strongly*) to:

$$\hat{\theta}_{\text{mle}} \xrightarrow{\text{a.s.}} \theta_0.$$

To establish consistency, the following conditions are sufficient:[2]

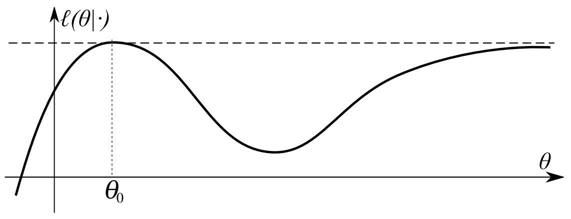
1. **Identification** of the model:

   $\theta \neq \theta_0 \quad \Leftrightarrow \quad f(\cdot \mid \theta) \neq f(\cdot \mid \theta_0).$

   In other words, different parameter values θ correspond to different distributions within the model. If this condition did not hold, there would be some value θ_1 such that θ_0 and θ_1 generate an identical distribution of the observable data. Then we wouldn't be able to distinguish between these two parameters even with an infinite amount of data — these parameters would have been *observationally equivalent*.

   The identification condition is absolutely necessary for the ML estimator to be consistent. When this condition holds, the limiting likelihood function $\ell(\theta|\cdot)$ has unique global maximum at θ_0.

2. **Compactness**: the parameter space Θ of the model is compact. The identification condition establishes



   that the log-likelihood has a unique global maximum. Compactness implies that the likelihood cannot approach the maximum value arbitrarily close at some other point (as demonstrated for example in the picture on the right).

   Compactness is only a sufficient condition and not a necessary condition. Compactness can be replaced by some other conditions, such as:

   - both concavity of the log-likelihood function and compactness of some (nonempty) upper level sets of the log-likelihood function, or
   - existence of a compact neighborhood N of θ_0 such that outside of N the log-likelihood function is less than the maximum by at least some $\varepsilon > 0$.

3. **Continuity**: the function ln $f(x|\theta)$ is continuous in θ for almost all values of x:

   $\Pr\Big[\ln f(x \mid \theta) \in \mathbb{C}^0(\Theta) \Big] = 1.$

   The continuity here can be replaced with a slightly weaker condition of upper semi-continuity.

4. **Dominance**: there exists $D(x)$ integrable with respect to the distribution $f(x|\theta_0)$ such that

   $\big|\ln f(x \mid \theta)\big| < D(x) \quad$ all for $\theta \in \Theta$.

   By the uniform law of large numbers, the dominance condition together with continuity establish the **uniform convergence in probability** of the log-likelihood:

$$\sup_{\theta\in\Theta} \big|\hat{\ell}(\theta \mid x) - \ell(\theta)\,\big| \xrightarrow{p} 0.$$

The dominance condition can be employed in the case of i.i.d. observations. In the non-i.i.d. case the uniform convergence in probability can be checked by showing that the sequence $\hat{\ell}(\theta|x)$ is stochastically equicontinuous.

If one wants to demonstrate that the ML estimator $\hat{\theta}$ converges to θ_0 almost surely, then a stronger condition of **uniform convergence almost surely** has to be imposed:

$$\sup_{\theta\in\Theta} \big\|\;\hat{\ell}(x \mid \theta) - \ell(\theta)\;\big\| \xrightarrow{\text{a.s.}} 0.$$

34.2.2 Asymptotic normality

Maximum-likelihood estimators can lack asymptotic normality and can be inconsistent if there is a failure of one (or more) of the below regularity conditions:

Estimate on boundary. Sometimes the maximum likelihood estimate lies on the boundary of the set of possible parameters, or (if the boundary is not, strictly speaking, allowed) the likelihood gets larger and larger as the parameter approaches the boundary. Standard asymptotic theory needs the assumption that the true parameter value lies away from the boundary. If we have enough data, the maximum likelihood estimate will keep away from the boundary too. But with smaller samples, the estimate can lie on the boundary. In such cases, the asymptotic theory clearly does not give a practically useful approximation. Examples here would be variance-component models, where each component of variance, σ^2, must satisfy the constraint $\sigma^2 \geq 0$.

Data boundary parameter-dependent. For the theory to apply in a simple way, the set of data values which has positive probability (or positive probability density) should not depend on the unknown parameter. A simple example where such parameter-dependence does hold is the case of estimating θ from a set of independent identically distributed when the common distribution is uniform on the range (0,θ). For estimation purposes the relevant range of θ is such that θ cannot be less than the largest observation. Because the interval (0,θ) is not compact, there exists no maximum for the likelihood function: For any estimate of theta, there exists a greater estimate that also has greater likelihood. In contrast, the interval [0,θ] includes the end-point θ and is compact, in which case the maximum-likelihood estimator exists. However, in this case, the maximum-likelihood estimator is biased. Asymptotically, this maximum-likelihood estimator is not normally distributed.[3]

Nuisance parameters. For maximum likelihood estimations, a model may have a number of nuisance parameters. For the asymptotic behaviour outlined to hold, the number of nuisance parameters should not increase with the number of observations (the sample size). A well-known example of this case is where observations occur as pairs, where the observations in each pair have a different (unknown) mean but otherwise the observations are independent and normally distributed with a common variance. Here for $2N$ observations, there are N+1 parameters. It is well known that the maximum likelihood estimate for the variance does not converge to the true value of the variance.

Increasing information. For the asymptotics to hold in cases where the assumption of independent identically distributed observations does not hold, a basic requirement is that the amount of information in the data increases indefinitely as the sample size increases. Such a requirement may not be met if either there is too much dependence in the data (for example, if new observations are essentially identical to existing observations), or if new independent observations are subject to an increasing observation error.

Some regularity conditions which ensure this behavior are:

1. The first and second derivatives of the log-likelihood function must be defined.
2. The Fisher information matrix must not be zero, and must be continuous as a function of the parameter.
3. The maximum likelihood estimator is consistent.

Suppose that conditions for consistency of maximum likelihood estimator are satisfied, and[4]

1. $\theta_0 \in$ interior(Θ);
2. $f(x \mid \theta) > 0$ and is twice continuously differentiable in θ in some neighborhood N of θ_0;
3. $\int$ sup$\theta{\in}N$||$\nabla\theta f(x \mid \theta)$||d$x < \infty$, and $\int$ sup$\theta{\in}N$||$\nabla\theta\theta f(x|\theta)$||d$x < \infty$;
4. I = E[$\nabla\theta$ln$f(x \mid \theta_0)$ $\nabla\theta$ln$f(x|\theta_0)'$] exists and is nonsingular;
5. E[sup$\theta{\in}N$||$\nabla\theta\theta$ln$f(x \mid \theta)$||] < ∞.

Then the maximum likelihood estimator has asymptotically normal distribution:

$$\sqrt{n}\big(\hat\theta_{\text{mle}} - \theta_0\big) \xrightarrow{d} \mathcal{N}(0,\, I^{-1}).$$

Proof, skipping the technicalities:

Since the log-likelihood function is differentiable, and θ_0 lies in the interior of the parameter set, in the maximum the first-order condition will be satisfied:

$$\nabla_\theta\, \hat\ell(\hat\theta \mid x) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ln f(x_i \mid \hat\theta) = 0.$$

When the log-likelihood is twice differentiable, this expression can be expanded into a Taylor series around the point $\theta = \theta_0$:

$$0 = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ln f(x_i \mid \theta_0) + \left[\frac{1}{n}\sum_{i=1}^{n} \nabla_{\theta\theta} \ln f(x_i \mid \tilde\theta)\right](\hat\theta - \theta_0),$$

where $\tilde\theta$ is some point intermediate between θ_0 and $\hat\theta$. From this expression we can derive that

$$\sqrt{n}(\hat\theta - \theta_0) = \left[-\frac{1}{n}\sum_{i=1}^{n} \nabla_{\theta\theta} \ln f(x_i \mid \tilde\theta)\right]^{-1} \frac{1}{\sqrt{n}}\sum_{i=1}^{n} \nabla_\theta \ln f(x_i \mid \theta_0)$$

Here the expression in square brackets converges in probability to H = E[−$\nabla\theta\theta$ln$f(x \mid \theta_0)$] by the law of large numbers. The continuous mapping theorem ensures that the inverse of this expression also converges in probability, to H^{-1}. The second sum, by the central limit theorem, converges in distribution to a multivariate normal with mean zero and variance matrix equal to the Fisher information I. Thus, applying Slutsky's theorem to the whole expression, we obtain that

$$\sqrt{n}(\hat\theta - \theta_0) \xrightarrow{d} \mathcal{N}\big(0,\ H^{-1}IH^{-1}\big).$$

Finally, the information equality guarantees that when the model is correctly specified, matrix H will be equal to the Fisher information I, so that the variance expression simplifies to just I^{-1}.

34.2.3 Functional invariance

The maximum likelihood estimator selects the parameter value which gives the observed data the largest possible probability (or probability density, in the continuous case). If the parameter consists of a number of components, then we define their separate maximum likelihood estimators, as the corresponding component of the MLE of the complete parameter. Consistent with this, if $\widehat{\theta}$ is the MLE for θ, and if g(θ) is any transformation of θ, then the MLE for $\alpha = g(\theta)$ is by definition

$\widehat{\alpha} = g(\widehat{\theta})$.

It maximizes the so-called profile likelihood:

$$\bar{L}(\alpha) = \sup_{\theta:\alpha=g(\theta)} L(\theta).$$

The MLE is also invariant with respect to certain transformations of the data. If $Y = g(X)$ where g is one to one and does not depend on the parameters to be estimated, then the density functions satisfy

$$f_Y(y) = f_X(x)/|g'(x)|$$

and hence the likelihood functions for X and Y differ only by a factor that does not depend on the model parameters.

For example, the MLE parameters of the log-normal distribution are the same as those of the normal distribution fitted to the logarithm of the data.

34.2.4 Higher-order properties

The standard asymptotics tells that the maximum-likelihood estimator is √n-consistent and asymptotically efficient, meaning that it reaches the Cramér–Rao bound:

$$\sqrt{n}(\hat{\theta}_{\text{mle}} - \theta_0) \xrightarrow{d} \mathcal{N}(0,\, I^{-1}),$$

where I is the Fisher information matrix:

$$I_{jk} = \mathrm{E}_X\left[-\frac{\partial^2 \ln f_{\theta_0}(X_t)}{\partial\theta_j\,\partial\theta_k} \right].$$

In particular, it means that the bias of the maximum-likelihood estimator is equal to zero up to the order $n^{-1/2}$. However when we consider the higher-order terms in the expansion of the distribution of this estimator, it turns out that θ_{mle} has bias of order n^{-1}. This bias is equal to (componentwise)[5]

$$b_s \equiv \mathrm{E}[(\hat{\theta}_{\text{mle}} - \theta_0)_s] = \frac{1}{n} \cdot I^{si} I^{jk} \big(\tfrac12 K_{ijk} + J_{j,ik}\big)$$

where Einstein's summation convention over the repeating indices has been adopted; I^{jk} denotes the j,k-th component of the *inverse* Fisher information matrix I^{-1}, and

$$\tfrac12 K_{ijk} + J_{j,ik} = \mathrm{E}\left[\frac12 \frac{\partial^3 \ln f_{\theta_0}(x_t)}{\partial\theta_i\,\partial\theta_j\,\partial\theta_k} + \frac{\partial \ln f_{\theta_0}(x_t)}{\partial\theta_j} \frac{\partial^2 \ln f_{\theta_0}(x_t)}{\partial\theta_i\,\partial\theta_k} \right]$$

Using these formulas it is possible to estimate the second-order bias of the maximum likelihood estimator, and *correct* for that bias by subtracting it:

$$\hat{\theta}^*_{\text{mle}} = \hat{\theta}_{\text{mle}} - \hat{b}.$$

This estimator is unbiased up to the terms of order n^{-1}, and is called the **bias-corrected maximum likelihood estimator**.

This bias-corrected estimator is *second-order efficient* (at least within the curved exponential family), meaning that it has minimal mean squared error among all second-order bias-corrected estimators, up to the terms of the order n^{-2}. It is possible to continue this process, that is to derive the third-order bias-correction term, and so on. However as was shown by Kano (1996), the maximum-likelihood estimator is **not** third-order efficient.

34.3 Examples

34.3.1 Discrete uniform distribution

Main article: German tank problem

Consider a case where n tickets numbered from 1 to n are placed in a box and one is selected at random (*see uniform distribution*); thus, the sample size is 1. If n is unknown, then the maximum-likelihood estimator $\hat{n}$ of n is the number m on the drawn ticket. (The likelihood is 0 for $n < m$, $1/n$ for $n \geq m$, and this is greatest when $n = m$. Note that the maximum likelihood estimate of n occurs at the lower extreme of possible values $\{m, m + 1, ...\}$, rather than somewhere in the "middle" of the range of possible values, which would result in less bias.) The expected value of the number m on the drawn ticket, and therefore the expected value of $\hat{n}$, is $(n + 1)/2$. As a result, with a sample size of 1, the maximum likelihood estimator for n will systematically underestimate n by $(n - 1)/2$.

34.3.2 Discrete distribution, finite parameter space

Suppose one wishes to determine just how biased an unfair coin is. Call the probability of tossing a HEAD p. The goal then becomes to determine p.

Suppose the coin is tossed 80 times: i.e., the sample might be something like x_1 = H, x_2 = T, ..., x_{80} = T, and the count of the number of HEADS "H" is observed.

The probability of tossing TAILS is $1 - p$ (so here p is θ above). Suppose the outcome is 49 HEADS and 31 TAILS, and suppose the coin was taken from a box containing three coins: one which gives HEADS with probability $p = 1/3$, one which gives HEADS with probability $p = 1/2$ and another which gives HEADS with probability $p = 2/3$. The coins have lost their labels, so which one it was is unknown. Using **maximum likelihood**

estimation the coin that has the largest likelihood can be found, given the data that were observed. By using the probability mass function of the binomial distribution with sample size equal to 80, number successes equal to 49 but different values of p (the "probability of success"), the likelihood function (defined below) takes one of three values:

$$\Pr(\mathrm{H}=49 \mid p=1/3) = \binom{80}{49}(1/3)^{49}(1-1/3)^{31} \approx 0.000,$$

$$\Pr(\mathrm{H}=49 \mid p=1/2) = \binom{80}{49}(1/2)^{49}(1-1/2)^{31} \approx 0.012,$$

$$\Pr(\mathrm{H}=49 \mid p=2/3) = \binom{80}{49}(2/3)^{49}(1-2/3)^{31} \approx 0.054.$$

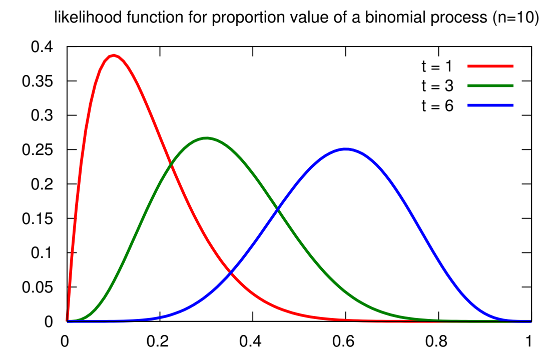
The likelihood is maximized when $p = 2/3$, and so this is the *maximum likelihood estimate* for p.

34.3.3 Discrete distribution, continuous parameter space

Now suppose that there was only one coin but its p could have been any value $0 \le p \le 1$. The likelihood function to be maximised is

$$L(p) = f_D(\mathrm{H}=49 \mid p) = \binom{80}{49} p^{49}(1-p)^{31},$$

and the maximisation is over all possible values $0 \le p \le 1$.



likelihood function for proportion value of a binomial process (n *= 10)*

One way to maximize this function is by differentiating with respect to p and setting to zero:

$$\begin{aligned} 0 &= \frac{\partial}{\partial p}\left(\binom{80}{49} p^{49}(1-p)^{31}\right) \\ &\propto 49p^{48}(1-p)^{31} - 31p^{49}(1-p)^{30} \\ &= p^{48}(1-p)^{30}\left[49(1-p) - 31p\right] \\ &= p^{48}(1-p)^{30}\left[49 - 80p\right] \end{aligned}$$

which has solutions $p = 0$, $p = 1$, and $p = 49/80$. The solution which maximizes the likelihood is clearly $p = 49/80$ (since $p = 0$ and $p = 1$ result in a likelihood of zero). Thus the *maximum likelihood estimator* for p is 49/80.

This result is easily generalized by substituting a letter such as t in the place of 49 to represent the observed number of 'successes' of our Bernoulli trials, and a letter such as n in the place of 80 to represent the number of Bernoulli trials. Exactly the same calculation yields the *maximum likelihood estimator t / n* for any sequence of n Bernoulli trials resulting in t 'successes'.

34.3.4 Continuous distribution, continuous parameter space

For the normal distribution $\mathcal{N}(\mu, \sigma^2)$ which has probability density function

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

the corresponding probability density function for a sample of n independent identically distributed normal random variables (the likelihood) is

$$f(x_1, \ldots, x_n \mid \mu, \sigma^2) = \prod_{i=1}^{n} f(x_i \mid \mu, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\frac{\sum_{i=1}^{n}(}{2}\right.$$

or more conveniently:

$$f(x_1, \ldots, x_n \mid \mu, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2 + n(\bar{x} - \mu}{2\sigma^2}\right.$$

where $\bar{x}$ is the sample mean.

This family of distributions has two parameters: $\theta = (\mu, \sigma)$, so we maximize the likelihood, $\mathcal{L}(\mu, \sigma) = f(x_1, \ldots, x_n \mid \mu, \sigma)$, over both parameters simultaneously, or if possible, individually.

Since the logarithm is a continuous strictly increasing function over the range of the likelihood, the values which maximize the likelihood will also maximize its logarithm. This log likelihood can be written as follows:

$$\log(\mathcal{L}(\mu,\sigma)) = (-n/2)\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2$$

(Note: the log-likelihood is closely related to information entropy and Fisher information.)

We now compute the derivatives of this log likelihood as follows.

$$0 = \frac{\partial}{\partial\mu}\log(\mathcal{L}(\mu,\sigma)) = 0 - \frac{-2n(\bar{x}-\mu)}{2\sigma^2}.$$

This is solved by

$$\hat{\mu} = \bar{x} = \sum_{i=1}^{n}\frac{x_i}{n}.$$

This is indeed the maximum of the function since it is the only turning point in μ and the second derivative is strictly less than zero. Its expectation value is equal to the parameter μ of the given distribution,

$$E\left[\widehat{\mu}\right] = \mu,$$

which means that the maximum-likelihood estimator $\widehat{\mu}$ is unbiased.

Similarly we differentiate the log likelihood with respect to σ and equate to zero:

$$\begin{aligned} 0 &= \frac{\partial}{\partial\sigma}\log\left(\left(\frac{1}{2\pi\sigma^2}\right)^{n/2}\exp\left(-\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2+n(\bar{x}-\mu)^2}{2\sigma^2}\right)\right) \\ &= \frac{\partial}{\partial\sigma}\left(\frac{n}{2}\log\left(\frac{1}{2\pi\sigma^2}\right) - \frac{\sum_{i=1}^{n}(x_i-\bar{x})^2+n(\bar{x}-\mu)^2}{2\sigma^2}\right) \\ &= -\frac{n}{\sigma} + \frac{\sum_{i=1}^{n}(x_i-\bar{x})^2+n(\bar{x}-\mu)^2}{\sigma^3} \end{aligned}$$

which is solved by

$$\widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2.$$

Inserting the estimate $\mu = \widehat{\mu}$ we obtain

$$\widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}x_ix_j.$$

To calculate its expected value, it is convenient to rewrite the expression in terms of zero-mean random variables (statistical error) $\delta_i \equiv \mu - x_i$. Expressing the estimate in these variables yields

$$\widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(\mu-\delta_i)^2 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\mu-\delta_i)(\mu-\delta_j).$$

Simplifying the expression above, utilizing the facts that $E\left[\delta_i\right] = 0$ and $E[\delta_i^2] = \sigma^2$, allows us to obtain

$$E\left[\widehat{\sigma}^2\right] = \frac{n-1}{n}\sigma^2.$$

This means that the estimator $\widehat{\sigma}$ is biased. However, $\widehat{\sigma}$ is consistent.

Formally we say that the *maximum likelihood estimator* for $\theta = (\mu,\sigma^2)$ is:

$$\widehat{\theta} = \left(\widehat{\mu},\widehat{\sigma}^2\right).$$

In this case the MLEs could be obtained individually. In general this may not be the case, and the MLEs would have to be obtained simultaneously.

The normal log likelihood at its maximum takes a particularly simple form:

$$\log(\mathcal{L}(\hat{\mu},\hat{\sigma})) = \frac{-n}{2}(\log(2\pi\hat{\sigma}^2)+1)$$

This maximum log likelihood can be shown to be the same for more general least squares, even for non-linear least squares. This is often used in determining likelihood-based approximate confidence intervals and confidence regions, which are generally more accurate than those using the asymptotic normality discussed above.

34.4 Non-independent variables

It may be the case that variables are correlated, that is, not independent. Two random variables *X* and *Y* are independent only if their joint probability density function is the product of the individual probability density functions, i.e.

$$f(x,y) = f(x)f(y)$$

Suppose one constructs an order-*n* Gaussian vector out of random variables $(x_1,\ldots,x_n)$, where each variable has means given by $(\mu_1,\ldots,\mu_n)$. Furthermore, let the covariance matrix be denoted by Σ .

The joint probability density function of these *n* random variables is then given by:

$$f(x_1,\ldots,x_n) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}\left[x_1-\mu_1,\ldots,x_n-\mu_n\right]\Sigma^{-1}\left[x_1-\mu_1,\ldots,x_n-\mu_n\right]^T\right)$$

In the two variable case, the joint probability density function is given by:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{\sigma_y^2}\right)\right]$$

In this and other cases where a joint density function exists, the likelihood function is defined as above, in the section Principles, using this density.

34.5 Iterative procedures

Consider problems where both states x_i and parameters such as σ^2 require to be estimated. Iterative procedures such as Expectation-maximization algorithms may be used to solve joint state-parameter estimation problems.

For example, suppose that n samples of state estimates $\hat{x}_i$ together with a sample mean $\bar{x}$ have been calculated by either a minimum-variance Kalman filter or a minimum-variance smoother using a previous variance estimate $\widehat{\sigma}^2$. Then the next variance iterate may be obtained from the maximum likelihood estimate calculation

$$\widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - \bar{x})^2.$$

The convergence of MLEs within filtering and smoothing EM algorithms are studied in[6] [7] .[8]

34.6 Applications

Maximum likelihood estimation is used for a wide range of statistical models, including:

- linear models and generalized linear models;
- exploratory and confirmatory factor analysis;
- structural equation modeling;
- many situations in the context of hypothesis testing and confidence interval \
- discrete choice models;

These uses arise across applications in widespread set of fields, including:

- communication systems;
- psychometrics;
- econometrics;
- time-delay of arrival (TDOA) in acoustic or electromagnetic detection;
- data modeling in nuclear and particle physics;
- magnetic resonance imaging;[9][10]
- computational phylogenetics;
- origin/destination and path-choice modeling in transport networks;
- geographical satellite-image classification.
- power system state estimation

34.7 History

Maximum-likelihood estimation was recommended, analyzed (with flawed attempts at proofs) and vastly popularized by R. A. Fisher between 1912 and 1922[11] (although it had been used earlier by Gauss, Laplace, T. N. Thiele, and F. Y. Edgeworth).[12] Reviews of the development of maximum likelihood have been provided by a number of authors.[13]

Much of the theory of maximum-likelihood estimation was first developed for Bayesian statistics, and then simplified by later authors.[11]

34.8 See also

- **Other estimation methods**
    - Generalized method of moments are methods related to the likelihood equation in maximum likelihood estimation.
    - M-estimator, an approach used in robust statistics.
    - Maximum a posteriori (MAP) estimator, for a contrast in the way to calculate estimators when prior knowledge is postulated.
    - Maximum spacing estimation, a related method that is more robust in many situations.
    - Method of moments (statistics), another popular method for finding parameters of distributions.
    - Method of support, a variation of the maximum likelihood technique.
    - Minimum distance estimation

    - Quasi-maximum likelihood estimator, an MLE estimator that is misspecified, but still consistent.
    - Restricted maximum likelihood, a variation using a likelihood function calculated from a transformed set of data.

- **Related concepts**:
    - The BHHH algorithm is a non-linear optimization algorithm that is popular for Maximum Likelihood estimations.
    - Extremum estimator, a more general class of estimators to which MLE belongs.
    - Fisher information, information matrix, its relationship to covariance matrix of ML estimates
    - Likelihood function, a description on what likelihood functions are.
    - Mean squared error, a measure of how 'good' an estimator of a distributional parameter is (be it the maximum likelihood estimator or some other estimator).
    - The Rao–Blackwell theorem, a result which yields a process for finding the best possible unbiased estimator (in the sense of having minimal mean squared error). The MLE is often a good starting place for the process.
    - Sufficient statistic, a function of the data through which the MLE (if it exists and is unique) will depend on the data.

34.9 References

[1] Pfanzagl (1994, p. 206)

[2] Newey & McFadden (1994, Theorem 2.5.)

[3] Lehmann & Casella (1998)

[4] Newey & McFadden (1994, Theorem 3.3.)

[5] Cox & Snell (1968, formula (20))

[6] Einicke, G.A.; Malos, J.T.; Reid, D.C.; Hainsworth, D.W. (January 2009). "Riccati Equation and EM Algorithm Convergence for Inertial Navigation Alignment". *IEEE Trans. Signal Processing* **57** (1): 370–375. doi:10.1109/TSP.2008.2007090.

[7] Einicke, G.A.; Falco, G.; Malos, J.T. (May 2010). "EM Algorithm State Matrix Estimation for Navigation". *IEEE Signal Processing Letters* **17** (5): 437–440. doi:10.1109/LSP.2010.2043151.

[8] Einicke, G.A.; Falco, G.; Dunn, M.T.; Reid, D.C. (May 2012). "Iterative Smoother-Based Variance Estimation". *IEEE Signal Processing Letters* **19** (5): 275–278. doi:10.1109/LSP.2012.2190278.

[9] Sijbers, Jan; den Dekker, A.J. (2004). "Maximum Likelihood estimation of signal amplitude and noise variance from MR data". *Magnetic Resonance in Medicine* **51** (3): 586–594. doi:10.1002/mrm.10728. PMID 15004801.

[10] Sijbers, Jan; den Dekker, A.J.; Scheunders, P.; Van Dyck, D. (1998). "Maximum Likelihood estimation of Rician distribution parameters". *IEEE Transactions on Medical Imaging* **17** (3): 357–361. doi:10.1109/42.712125. PMID 9735899.

[11] Pfanzagl (1994)

[12] Edgeworth (September 1908) and Edgeworth (December 1908)

[13] Savage (1976), Pratt (1976), Stigler (1978, 1986, 1999), Hald (1998, 1999), and Aldrich (1997)

34.10 Further reading

- Aldrich, John (1997). "R. A. Fisher and the making of maximum likelihood 1912–1922". *Statistical Science* **12** (3): 162–176. doi:10.1214/ss/1030037906. MR 1617519.
- Andersen, Erling B. (1970); "Asymptotic Properties of Conditional Maximum Likelihood Estimators", *Journal of the Royal Statistical Society* **B** 32, 283–301
- Andersen, Erling B. (1980); *Discrete Statistical Models with Social Science Applications*, North Holland, 1980
- Basu, Debabrata (1988); *Statistical Information and Likelihood : A Collection of Critical Essays by Dr. D. Basu*; in Ghosh, Jayanta K., editor; *Lecture Notes in Statistics*, Volume 45, Springer-Verlag, 1988
- Cox, David R.; Snell, E. Joyce (1968). "A general definition of residuals". *Journal of the Royal Statistical Society, Series B*: 248–275. JSTOR 2984505.
- Edgeworth, Francis Y. (Sep 1908). "On the probable errors of frequency-constants". *Journal of the Royal Statistical Society* **71** (3): 499–512. doi:10.2307/2339293. JSTOR 2339293.
- Edgeworth, Francis Y. (Dec 1908). "On the probable errors of frequency-constants". *Journal of the Royal Statistical Society* **71** (4): 651–678. doi:10.2307/2339378. JSTOR 2339378.
- Einicke, G.A. (2012). *Smoothing, Filtering and Prediction: Estimating the Past, Present and Future*. Rijeka, Croatia: Intech. ISBN 978-953-307-752-9.
- Ferguson, Thomas S. (1982). "An inconsistent maximum likelihood estimate". *Journal of the American Statistical Association* **77** (380): 831–834. doi:10.1080/01621459.1982.10477894. JSTOR 2287314.

- Ferguson, Thomas S. (1996). *A course in large sample theory*. Chapman & Hall. ISBN 0-412-04371-8.
- Hald, Anders (1998). *A history of mathematical statistics from 1750 to 1930*. New York, NY: Wiley. ISBN 0-471-17912-4.
- Hald, Anders (1999). “On the history of maximum likelihood in relation to inverse probability and least squares”. *Statistical Science* **14** (2): 214–222. doi:10.1214/ss/1009212248. JSTOR 2676741.
- Kano, Yutaka (1996). “Third-order efficiency implies fourth-order efficiency”. *Journal of the Japan Statistical Society* **26**: 101–117. doi:10.14490/jjss1995.26.101.
- Le Cam, Lucien (1990). “Maximum likelihood — an introduction”. *ISI Review* **58** (2): 153–171. doi:10.2307/1403464.
- Le Cam, Lucien; Lo Yang, Grace (2000). *Asymptotics in statistics: some basic concepts* (Second ed.). Springer. ISBN 0-387-95036-2.
- Le Cam, Lucien (1986). *Asymptotic methods in statistical decision theory*. Springer-Verlag. ISBN 0-387-96307-3.
- Lehmann, Erich L.; Casella, George (1998). *Theory of Point Estimation, 2nd ed*. Springer. ISBN 0-387-98502-6.
- Newey, Whitney K.; McFadden, Daniel (1994). “Chapter 35: Large sample estimation and hypothesis testing”. In Engle, Robert; McFadden, Dan. *Handbook of Econometrics, Vol.4*. Elsevier Science. pp. 2111–2245. ISBN 0-444-88766-0.
- Pfanzagl, Johann (1994). *Parametric statistical theory*. with the assistance of R. Hamböker. Berlin, DE: Walter de Gruyter. pp. 207–208. ISBN 3-11-013863-8.
- Pratt, John W. (1976). “F. Y. Edgeworth and R. A. Fisher on the efficiency of maximum likelihood estimation”. *The Annals of Statistics* **4** (3): 501–514. doi:10.1214/aos/1176343457. JSTOR 2958222.
- Ruppert, David (2010). *Statistics and Data Analysis for Financial Engineering*. Springer. p. 98. ISBN 978-1-4419-7786-1.
- Savage, Leonard J. (1976). “On rereading R. A. Fisher”. *The Annals of Statistics* **4** (3): 441–500. doi:10.1214/aos/1176343456. JSTOR 2958221.
- Stigler, Stephen M. (1978). “Francis Ysidro Edgeworth, statistician”. *Journal of the Royal Statistical Society, Series A* **141** (3): 287–322. doi:10.2307/2344804. JSTOR 2344804.
- Stigler, Stephen M. (1986). *The history of statistics: the measurement of uncertainty before 1900*. Harvard University Press. ISBN 0-674-40340-1.
- Stigler, Stephen M. (1999). *Statistics on the table: the history of statistical concepts and methods*. Harvard University Press. ISBN 0-674-83601-4.
- van der Vaart, Aad W. (1998). *Asymptotic Statistics*. ISBN 0-521-78450-6.

34.11 External links

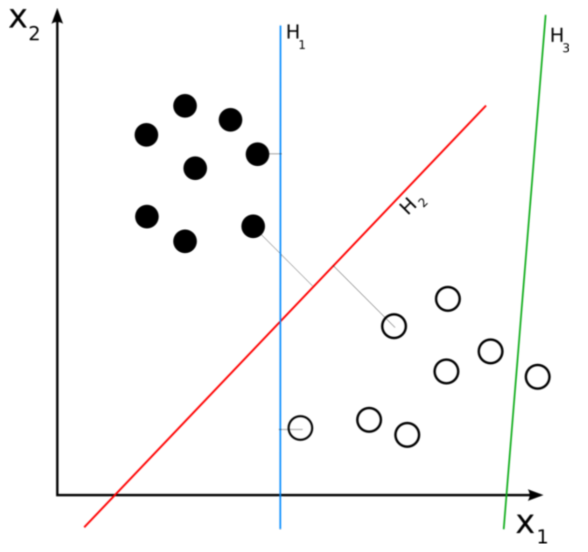
- Hazewinkel, Michiel, ed. (2001), “Maximum-likelihood method”, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Maximum Likelihood Estimation Primer (an excellent tutorial)
- Implementing MLE for your own likelihood function using R
- A selection of likelihood functions in R
- “Tutorial on maximum likelihood estimation”. *Journal of Mathematical Psychology*. CiteSeerX: 10.1.1.74.671.

Chapter 35

Linear classifier

In the field of machine learning, the goal of statistical classification is to use an object's characteristics to identify which class (or group) it belongs to. A **linear classifier** achieves this by making a classification decision based on the value of a linear combination of the characteristics. An object's characteristics are also known as feature values and are typically presented to the machine in a vector called a feature vector. Such classifiers work well for practical problems such as document classification, and more generally for problems with many variables (features), reaching accuracy levels comparable to non-linear classifiers while taking less time to train and use.[1]

35.1 Definition



In this case, the solid and empty dots can be correctly classified by any number of linear classifiers. H1 (blue) classifies them correctly, as does H2 (red). H2 could be considered "better" in the sense that it is also furthest from both groups. H3 (green) fails to correctly classify the dots.

If the input feature vector to the classifier is a real vector $\vec{x}$, then the output score is

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right),$$

where $\vec{w}$ is a real vector of weights and f is a function that converts the dot product of the two vectors into the desired output. (In other words, $\vec{w}$ is a one-form or linear functional mapping $\vec{x}$ onto **R**.) The weight vector $\vec{w}$ is learned from a set of labeled training samples. Often f is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex f might give the probability that an item belongs to a certain class.

For a two-class classification problem, one can visualize the operation of a linear classifier as splitting a high-dimensional input space with a hyperplane: all points on one side of the hyperplane are classified as "yes", while the others are classified as "no".

A linear classifier is often used in situations where the speed of classification is an issue, since it is often the fastest classifier, especially when $\vec{x}$ is sparse. Also, linear classifiers often work very well when the number of dimensions in $\vec{x}$ is large, as in document classification, where each element in $\vec{x}$ is typically the number of occurrences of a word in a document (see document-term matrix). In such cases, the classifier should be well-regularized.

35.2 Generative models vs. discriminative models

There are two broad classes of methods for determining the parameters of a linear classifier $\vec{w}$.[2][3] Methods of the first class model conditional density functions $P(\vec{x}|\text{class})$. Examples of such algorithms include:

- Linear Discriminant Analysis (or Fisher's linear discriminant) (LDA)—assumes Gaussian conditional density models
- Naive Bayes classifier with multinomial or multivariate Bernoulli event models.

The second set of methods includes discriminative models, which attempt to maximize the quality of the output on a training set. Additional terms in the training cost function can easily perform regularization of the final model. Examples of discriminative training of linear classifiers include

- Logistic regression—maximum likelihood estimation of $\vec{w}$ assuming that the observed training set was generated by a binomial model that depends on the output of the classifier.
- Perceptron—an algorithm that attempts to fix all errors encountered in the training set
- Support vector machine—an algorithm that maximizes the margin between the decision hyperplane and the examples in the training set.

Note: Despite its name, LDA does not belong to the class of discriminative models in this taxonomy. However, its name makes sense when we compare LDA to the other main linear dimensionality reduction algorithm: principal components analysis (PCA). LDA is a supervised learning algorithm that utilizes the labels of the data, while PCA is an unsupervised learning algorithm that ignores the labels. To summarize, the name is a historical artifact.[4]:117

Discriminative training often yields higher accuracy than modeling the conditional density functions. However, handling missing data is often easier with conditional density models.

All of the linear classifier algorithms listed above can be converted into non-linear algorithms operating on a different input space $\varphi(\vec{x})$, using the kernel trick.

35.2.1 Discriminative training

Discriminative training of linear classifiers usually proceeds in a supervised way, by means of an optimization algorithm that is given a training set with desired outputs and a loss function that measures the discrepancy between the classifier's outputs and the desired outputs. Thus, the learning algorithm solves an optimization problem of the form[1]

$$\arg\min_{\mathbf{w}} R(\mathbf{w}) + C \sum_{i=1}^{N} L(y_i, \mathbf{w}^\mathsf{T}\mathbf{x}_i)$$

where

- **w** are the classifier's parameters,
- $L(yi,\ \mathbf{w}^\mathsf{T}\mathbf{x}i)$ is the loss of the prediction given the desired output y_i for the i'th training example,
- $R(\mathbf{w})$ is a regularization term that prevents the parameters from getting too large (causing overfitting), and
- C is some constant (set by the user of the learning algorithm) that weighs the regularization against the loss.

Popular loss functions include the hinge loss (for linear SVMs) and the log loss (for linear logistic regression). If the regularization function R is convex, then the above is a convex problem.[1] Many algorithms exist for solving such problems; popular ones for linear classification include (stochastic) gradient descent, L-BFGS, coordinate descent and Newton methods.

35.3 See also

- Linear regression
- Winnow (algorithm)
- Quadratic classifier
- Support vector machines

35.4 Notes

[1] Guo-Xun Yuan; Chia-Hua Ho; Chih-Jen Lin (2012). "Recent Advances of Large-Scale Linear Classification". *Proc. IEEE* **100** (9).

[2] T. Mitchell, Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression. Draft Version, 2005 download

[3] A. Y. Ng and M. I. Jordan. On Discriminative vs. Generative Classifiers: A comparison of logistic regression and Naive Bayes. in NIPS 14, 2002. download

[4] R.O. Duda, P.E. Hart, D.G. Stork, "Pattern Classification", Wiley, (2001). ISBN 0-471-05669-3

See also:

1. Y. Yang, X. Liu, "A re-examination of text categorization", Proc. ACM SIGIR Conference, pp. 42–49, (1999). paper @ citeseer
2. R. Herbrich, "Learning Kernel Classifiers: Theory and Algorithms," MIT Press, (2001). ISBN 0-262-08306-X

Chapter 36

Logistic regression

In statistics, **logistic regression**, or **logit regression**, or **logit model**[1] is a direct probability model that was developed by statistician D. R. Cox in 1958[2] [3] although much work was done in the single independent variable case almost two decades earlier. The binary logistic model is used to predict a binary response based on one or more predictor variables (features). That is, it is used in estimating the parameters of a qualitative response model. The probabilities describing the possible outcomes of a single trial are modeled, as a function of the explanatory (predictor) variables, using a logistic function. Frequently (and hereafter in this article) "logistic regression" is used to refer specifically to the problem in which the dependent variable is binary—that is, the number of available categories is two—while problems with more than two categories are referred to as multinomial logistic regression, or, if the multiple categories are ordered, as ordinal logistic regression.[3]

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables, which are usually (but not necessarily) continuous, by estimating probabilities. Thus, it treats the same set of problems as does probit regression using similar techniques; the first assumes a logistic function and the second a standard normal distribution function.

Logistic regression can be seen as a special case of generalized linear model and thus analogous to linear regression. The model of logistic regression, however, is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression. In particular the key differences of these two models can be seen in the following two features of logistic regression. First, the conditional distribution $p(y \mid x)$ is a Bernoulli distribution rather than a Gaussian distribution, because the dependent variable is binary. Second, the estimated probabilities are restricted to [0,1] through the logistic distribution function because logistic regression predicts the **probability** of the instance being positive.

Logistic regression is an alternative to Fisher's 1936 classification method, linear discriminant analysis.[4] If the assumptions of linear discriminant analysis hold, application of Bayes' rule to reverse the conditioning results in the logistic model, so if linear discriminant assumptions are true, logistic regression assumptions must hold. The converse is not true, so the logistic model has fewer assumptions than discriminant analysis and makes no assumption on the distribution of the independent variables.

36.1 Fields and example applications

Logistic regression is used widely in many fields, including the medical and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression.[5] Many other medical scales used to assess severity of a patient have been developed using logistic regression.[6][7][8][9] Logistic regression may be used to predict whether a patient has a given disease (e.g. diabetes; coronary heart disease), based on observed characteristics of the patient (age, sex, body mass index, results of various blood tests, etc.; age, blood cholesterol level, systolic blood pressure, relative weight, blood hemoglobin level, smoking (at 3 levels), and abnormal electrocardiogram.).[1][10] Another example might be to predict whether an American voter will vote Democratic or Republican, based on age, income, sex, race, state of residence, votes in previous elections, etc.[11] The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product.[12][13] It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc. In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a mortgage. Conditional random fields, an extension of logistic regression to sequential data, are used in natural language processing.

36.2 Basics

Logistic regression can be binomial or multinomial. Binomial or binary logistic regression deals with situations in which the observed outcome for a dependent variable can have only two possible types (for example, “dead” vs. “alive” or “win” vs. “loss”). Multinomial logistic regression deals with situations where the outcome can have three or more possible types (e.g., “disease A” vs. “disease B” vs. “disease C”). In binary logistic regression, the outcome is usually coded as “0” or “1”, as this leads to the most straightforward interpretation.[14] If a particular observed outcome for the dependent variable is the noteworthy possible outcome (referred to as a “success” or a “case”) it is usually coded as “1” and the contrary outcome (referred to as a “failure” or a “noncase”) as “0”. Logistic regression is used to predict the odds of being a case based on the values of the independent variables (predictors). The odds are defined as the probability that a particular outcome is a case divided by the probability that it is a noncase.

Like other forms of regression analysis, logistic regression makes use of one or more predictor variables that may be either continuous or categorical data. Unlike ordinary linear regression, however, logistic regression is used for predicting binary outcomes of the dependent variable (treating the dependent variable as the outcome of a Bernoulli trial) rather than a continuous outcome. Given this difference, it is necessary that logistic regression take the natural logarithm of the odds of the dependent variable being a case (referred to as the logit or log-odds) to create a continuous criterion as a transformed version of the dependent variable. Thus the logit transformation is referred to as the *link function* in logistic regression—although the dependent variable in logistic regression is binomial, the logit is the continuous criterion upon which linear regression is conducted.[14]

The logit of success is then fitted to the predictors using linear regression analysis. The predicted value of the logit is converted back into predicted odds via the inverse of the natural logarithm, namely the exponential function. Thus, although the observed dependent variable in logistic regression is a zero-or-one variable, the logistic regression estimates the odds, as a continuous variable, that the dependent variable is a success (a case). In some applications the odds are all that is needed. In others, a specific yes-or-no prediction is needed for whether the dependent variable is or is not a case; this categorical prediction can be based on the computed odds of a success, with predicted odds above some chosen cutoff value being translated into a prediction of a success.

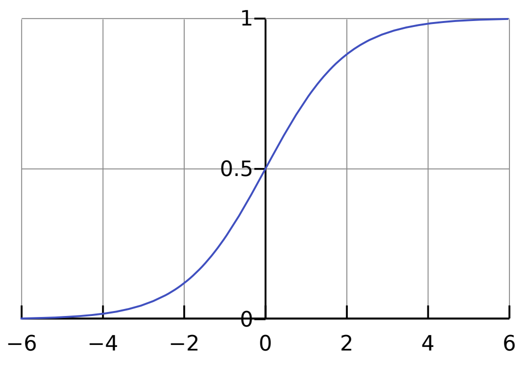


Figure 1. The logistic function $\sigma(t)$; note that $\sigma(t) \in [0,1]$ for all t .

36.3 Logistic function, odds, odds ratio, and logit

36.3.1 Definition of the logistic function

An explanation of logistic regression begins with an explanation of the logistic function. The logistic function is useful because it can take an input with any value from negative to positive infinity, whereas the output always takes values between zero and one[14] and hence is interpretable as a probability. The logistic function $\sigma(t)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}},$$

A graph of the logistic function is shown in Figure 1.

If t is viewed as a linear function of an explanatory variable x (or of a linear combination of explanatory variables), then we express t as follows:

$$t = \beta_0 + \beta_1 x$$

And the logistic function can now be written as:

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Note that $F(x)$ is interpreted as the probability of the dependent variable equaling a “success” or “case” rather than a failure or non-case. It’s clear that the response variables Y_i are not identically distributed: $P(Y_i = 1 \mid X)$ differs from one data point X_i to another, though they are independent given design matrix X and shared with parameters β .[1]

36.3.2 Definition of the inverse of the logistic function

We can now define the inverse of the logistic function, g , the logit (log odds):

$$g(F(x)) = \ln \frac{F(x)}{1 - F(x)} = \beta_0 + \beta_1 x,$$

and equivalently:

$$\frac{F(x)}{1 - F(x)} = e^{\beta_0 + \beta_1 x}.$$

36.3.3 Interpretation of these terms

In the above equations, the terms are as follows:

- $g(\cdot)$ refers to the logit function. The equation for $g(F(x))$ illustrates that the logit (i.e., log-odds or natural logarithm of the odds) is equivalent to the linear regression expression.
- ln denotes the natural logarithm.
- $F(x)$ is the probability that the dependent variable equals a case, given some linear combination x of the predictors. The formula for $F(x)$ illustrates that the probability of the dependent variable equaling a case is equal to the value of the logistic function of the linear regression expression. This is important in that it shows that the value of the linear regression expression can vary from negative to positive infinity and yet, after transformation, the resulting expression for the probability $F(x)$ ranges between 0 and 1.
- β_0 is the intercept from the linear regression equation (the value of the criterion when the predictor is equal to zero).
- $\beta_1 x$ is the regression coefficient multiplied by some value of the predictor.
- base e denotes the exponential function.

36.3.4 Definition of the odds

The odds of the dependent variable equaling a case (given some linear combination x of the predictors) is equivalent to the exponential function of the linear regression expression. This illustrates how the logit serves as a link function between the probability and the linear regression expression. Given that the logit ranges between negative and positive infinity, it provides an adequate criterion upon which to conduct linear regression and the logit is easily converted back into the odds.[14]

So we define odds of the dependent variable equaling a case (given some linear combination x of the predictors) as follows:

$$\text{odds} = e^{\beta_0 + \beta_1 x}.$$

36.3.5 Definition of the odds ratio

The odds ratio can be defined as:

$$OR = \text{odds}(x+1) / \text{odds}(x) = \frac{\frac{F(x+1)}{1-F(x+1)}}{\frac{F(x)}{1-F(x)}} = e^{\beta_0 + \beta_1 (x+1)} / e^{\beta_0 + \beta_1 x} = e^{\beta_1}$$

or for binary variable F(0) instead of F(x) and F(1) for F(x+1). This exponential relationship provides an interpretation for β_1 : The odds multiply by e^{β_1} for every 1-unit increase in x.[15]

36.3.6 Multiple explanatory variables

If there are multiple explanatory variables, the above expression $\beta_0 + \beta_1 x$ can be revised to $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$. Then when this is used in the equation relating the logged odds of a success to the values of the predictors, the linear regression will be a multiple regression with *m* explanators; the parameters β_j for all *j* = 0, 1, 2, ..., *m* are all estimated.

36.4 Model fitting

36.4.1 Estimation

Because the model can be expressed as a generalized linear model (see below), for 0<p<1, ordinary least squares can suffice, with R-squared as the measure of goodness of fit in the fitting space. When p=0 or 1, more complex methods are required.

Maximum likelihood estimation

The regression coefficients are usually estimated using maximum likelihood estimation.[16] Unlike linear regression with normally distributed residuals, it is not possible to find a closed-form expression for the coefficient values that maximize the likelihood function, so that an iterative process must be used instead; for example Newton's method. This process begins with a tentative solution, revises it slightly to see if it can be improved, and repeats this revision until improvement is minute, at which point the process is said to have converged.[17]

In some instances the model may not reach convergence. Nonconvergence of a model indicates that the coefficients are not meaningful because the iterative process was unable to find appropriate solutions. A failure to converge may occur for a number of reasons: having a large ratio of predictors to cases, multicollinearity, sparseness, or complete separation.

- Having a large ratio of variables to cases results in an overly conservative Wald statistic (discussed below) and can lead to nonconvergence.
- Multicollinearity refers to unacceptably high correlations between predictors. As multicollinearity increases, coefficients remain unbiased but standard errors increase and the likelihood of model convergence decreases.[16] To detect multicollinearity amongst the predictors, one can conduct a linear regression analysis with the predictors of interest for the sole purpose of examining the tolerance statistic [16] used to assess whether multicollinearity is unacceptably high.
- Sparseness in the data refers to having a large proportion of empty cells (cells with zero counts). Zero cell counts are particularly problematic with categorical predictors. With continuous predictors, the model can infer values for the zero cell counts, but this is not the case with categorical predictors. The model will not converge with zero cell counts for categorical predictors because the natural logarithm of zero is an undefined value, so that final solutions to the model cannot be reached. To remedy this problem, researchers may collapse categories in a theoretically meaningful way or add a constant to all cells.[16]
- Another numerical problem that may lead to a lack of convergence is complete separation, which refers to the instance in which the predictors perfectly predict the criterion – all cases are accurately classified. In such instances, one should reexamine the data, as there is likely some kind of error.[14]

As a general rule of thumb, logistic regression models require a minimum of about 10 events per explaining variable (where *event* denotes the cases belonging to the less frequent category in the dependent variable).[18]

Minimum chi-squared estimator for grouped data

While individual data will have a dependent variable with a value of zero or one for every observation, with grouped data one observation is on a group of people who all share the same characteristics (e.g., demographic characteristics); in this case the researcher observes the proportion of people in the group for whom the response variable falls into one category or the other. If this proportion is neither zero nor one for any group, the minimum chi-squared estimator involves using weighted least squares to estimate a linear model in which the dependent variable is the logit of the proportion: that is, the log of the ratio of the fraction in one group to the fraction in the other group.[19]:pp.686–9

36.4.2 Evaluating goodness of fit

Goodness of fit in linear regression models is generally measured using the R^2. Since this has no direct analog in logistic regression, various methods[19]:ch.21 including the following can be used instead.

Deviance and likelihood ratio tests

In linear regression analysis, one is concerned with partitioning variance via the sum of squares calculations – variance in the criterion is essentially divided into variance accounted for by the predictors and residual variance. In logistic regression analysis, deviance is used in lieu of sum of squares calculations.[20] Deviance is analogous to the sum of squares calculations in linear regression[14] and is a measure of the lack of fit to the data in a logistic regression model.[20] When a "saturated" model is available (a model with a theoretically perfect fit), deviance is calculated by comparing a given model with the saturated model.[14] This computation give the likelihood-ratio test:.[14]

$$D = -2 \ln \frac{\text{model fitted the of likelihood}}{\text{model saturated the of likelihood}}.$$

In the above equation D represents the deviance and ln represents the natural logarithm. The log of the likelihood ratio (the ratio of the fitted model to the saturated model) will produce a negative value, so the product is multiplied by negative two times its natural logarithm to produce a value with an approximate chi-squared distribution.[14] Smaller values indicate better fit as the fitted model deviates less from the saturated model. When assessed upon a chi-square distribution, nonsignificant chi-square values indicate very little unexplained variance and thus, good model fit. Conversely, a significant chi-square value indicates that a significant amount of the variance is unexplained.

When the saturated model is not available (a common case), deviance is calculated simply as (-2)x(log likelihood of the fitted model), and the reference to the saturated model's log likelihood can be removed from all that follows without harm.

Two measures of deviance are particularly important in logistic regression: null deviance and model deviance. The null deviance represents the difference between a model with only the intercept (which means "no predictors") and the saturated model. The model deviance represents the difference between a model with at least one predictor and the saturated model.[20] In this respect, the null model provides a baseline upon which to compare predictor models. Given that deviance is a measure of the difference between a given model and the saturated model, smaller values indicate better fit. Thus, to assess the contribution of a predictor or set of predictors, one can subtract the model deviance from the null deviance and assess the difference on a χ^2_{s-p}, chi-square distribution with degrees of freedom[14] equal to the difference in the number of parameters estimated.

Let

$$D_{\text{null}} = -2\ln\frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}}$$

$$D_{\text{fitted}} = -2\ln\frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}.$$

Then

$$\begin{aligned} D_{\text{null}} - D_{\text{fitted}} &= \left(-2\ln\frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}}\right) - \left(-2\ln\frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}\right) \\ &= -2\left(\ln\frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}} - \ln\frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}\right) \\ &= -\,2\ln\frac{\left(\frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}}\right)}{\left(\frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}\right)} \\ &= -\,2\ln\frac{\text{model null the of likelihood}}{\text{model fitted of likelihood}}. \end{aligned}$$

If the model deviance is significantly smaller than the null deviance then one can conclude that the predictor or set of predictors significantly improved model fit. This is analogous to the F-test used in linear regression analysis to assess the significance of prediction.[20]

Pseudo-R^{2}s

In linear regression the squared multiple correlation, R^2 is used to assess goodness of fit as it represents the proportion of variance in the criterion that is explained by the predictors.[20] In logistic regression analysis, there is no agreed upon analogous measure, but there are several competing measures each with limitations.[20] Three of the most commonly used indices are examined on this page beginning with the likelihood ratio R^2, R^2L:[20]

$$R^2_{\text{L}} = \frac{D_{\text{null}} - D_{\text{fitted}}}{D_{\text{null}}}.$$

This is the most analogous index to the squared multiple correlation in linear regression.[16] It represents the proportional reduction in the deviance wherein the deviance is treated as a measure of variation analogous but not identical to the variance in linear regression analysis.[16] One limitation of the likelihood ratio R^2 is that it is not monotonically related to the odds ratio,[20] meaning that it does not necessarily increase as the odds ratio increases and does not necessarily decrease as the odds ratio decreases.

The Cox and Snell R^2 is an alternative index of goodness of fit related to the R^2 value from linear regression. The Cox and Snell index is problematic as its maximum value is .75, when the variance is at its maximum (.25). The Nagelkerke R^2 provides a correction to the Cox and Snell R^2 so that the maximum value is equal to one. Nevertheless, the Cox and Snell and likelihood ratio R^2s show greater agreement with each other than either does with the Nagelkerke R^2.[20] Of course, this might not be the case for values exceeding .75 as the Cox and Snell index is capped at this value. The likelihood ratio R^2 is often preferred to the alternatives as it is most analogous to R^2 in linear regression, is independent of the base rate (both Cox and Snell and Nagelkerke R^2s increase as the proportion of cases increase from 0 to .5) and varies between 0 and 1.

A word of caution is in order when interpreting pseudo-R^2 statistics. The reason these indices of fit are referred to as *pseudo* R^2 is that they do not represent the proportionate reduction in error as the R^2 in linear regression does.[20] Linear regression assumes homoscedasticity, that the error variance is the same for all values of the criterion. Logistic regression will always be heteroscedastic – the error variances differ for each value of the predicted score. For each value of the predicted score there would be a different value of the proportionate reduction in error. Therefore, it is inappropriate to think of R^2 as a proportionate reduction in error in a universal sense in logistic regression.[20]

Hosmer–Lemeshow test

The Hosmer–Lemeshow test uses a test statistic that asymptotically follows a χ^2 distribution to assess whether or not the observed event rates match expected event rates in subgroups of the model population.

Evaluating binary classification performance

If the estimated probabilities are to be used to classify each observation of independent variable values as predicting the category that the dependent variable is found in, the various methods below for judging the model's suitability in out-of-sample forecasting can also be used on the data that were used for estimation—accuracy, precision (also called positive predictive value), recall (also called sensitivity), specificity and negative predictive value. In each of these evaluative methods, an aspect of the model's effectiveness in assigning instances to the correct categories is measured.

36.5 Coefficients

After fitting the model, it is likely that researchers will want to examine the contribution of individual predictors. To do so, they will want to examine the regression coefficients. In linear regression, the regression coefficients represent the change in the criterion for each unit change in the predictor.[20] In logistic regression, however, the regression coefficients represent the change in the logit for each unit change in the predictor. Given that the logit is not intuitive, researchers are likely to focus on a predictor's effect on the exponential function of the regression coefficient – the odds ratio (see definition). In linear regression, the significance of a regression coefficient is assessed by computing a *t* test. In logistic regression, there are several different tests designed to assess the significance of an individual predictor, most notably the likelihood ratio test and the Wald statistic.

36.5.1 Likelihood ratio test

The likelihood-ratio test discussed above to assess model fit is also the recommended procedure to assess the contribution of individual "predictors" to a given model.[14][16][20] In the case of a single predictor model, one simply compares the deviance of the predictor model with that of the null model on a chi-square distribution with a single degree of freedom. If the predictor model has a significantly smaller deviance (c.f chi-square using the difference in degrees of freedom of the two models), then one can conclude that there is a significant association between the "predictor" and the outcome. Although some common statistical packages (e.g. SPSS) do provide likelihood ratio test statistics, without this computationally intensive test it would be more difficult to assess the contribution of individual predictors in the multiple logistic regression case. To assess the contribution of individual predictors one can enter the predictors hierarchically, comparing each new model with the previous to determine the contribution of each predictor.[20] There is some debate among statisticians about the appropriateness of so-called "stepwise" procedures. The fear is that they may not preserve nominal statistical properties and may become misleading.

36.5.2 Wald statistic

Alternatively, when assessing the contribution of individual predictors in a given model, one may examine the significance of the Wald statistic. The Wald statistic, analogous to the *t*-test in linear regression, is used to assess the significance of coefficients. The Wald statistic is the ratio of the square of the regression coefficient to the square of the standard error of the coefficient and is asymptotically distributed as a chi-square distribution.[16]

$W_j = \frac{B_j^2}{SE_{B_j}^2}$

Although several statistical packages (e.g., SPSS, SAS) report the Wald statistic to assess the contribution of individual predictors, the Wald statistic has limitations. When the regression coefficient is large, the standard error of the regression coefficient also tends to be large increasing the probability of Type-II error. The Wald statistic also tends to be biased when data are sparse.[20]

36.5.3 Case-control sampling

Suppose cases are rare. Then we might wish to sample them more frequently than their prevalence in the population. For example, suppose there is a disease that affects 1 person in 10,000 and to collect our data we need to do a complete physical. It may be too expensive to do thousands of physicals of healthy people in order to obtain data for only a few diseased individuals. Thus, we may evaluate more diseased individuals. This is also called unbalanced data. As a rule of thumb, sampling controls at a rate of five times the number of cases will produce sufficient control data.[21]

If we form a logistic model from such data, if the model is correct, the β_j parameters are all correct except for β_0 . We can correct β_0 if we know the true prevalence as follows:[21]

$\hat{\beta}_0^* = \hat{\beta}_0 + \log \frac{\pi}{1-\pi} - \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$

where π is the true prevalence and $\tilde{\pi}$ is the prevalence in the sample.

36.6 Formal mathematical specification

There are various equivalent specifications of logistic regression, which fit into different types of more general models. These different specifications allow for different sorts of useful generalizations.

36.6.1 Setup

The basic setup of logistic regression is the same as for standard linear regression.

It is assumed that we have a series of N observed data points. Each data point *i* consists of a set of m explanatory variables x_1,i ... xm,i (also called independent variables, predictor variables, input variables, features, or attributes), and an associated binary-valued outcome variable Yi (also known as a dependent variable, response variable, output variable, outcome variable or class variable), i.e. it can assume only the two possible values 0 (often meaning "no" or "failure") or 1 (often meaning "yes" or "success"). The goal of logistic regression is to explain the relationship between the explanatory variables and the outcome, so that an outcome can be predicted for a new set of explanatory variables.

Some examples:

- The observed outcomes are the presence or absence of a given disease (e.g. diabetes) in a set of patients, and the explanatory variables might be characteristics of the patients thought to be pertinent (sex, race, age, blood pressure, body-mass index, etc.).
- The observed outcomes are the votes (e.g. Democratic or Republican) of a set of people in an election, and the explanatory variables are the demographic characteristics of each person (e.g. sex, race, age, income, etc.). In such a case, one of the two outcomes is arbitrarily coded as 1, and the other as 0.

As in linear regression, the outcome variables *Yi* are assumed to depend on the explanatory variables x_1,*i* ... *xm,i*.

Explanatory variables

As shown above in the above examples, the explanatory variables may be of any type: real-valued, binary, categorical, etc. The main distinction is between continuous variables (such as income, age and blood pressure) and discrete variables (such as sex or race). Discrete variables referring to more than two possible choices are typically coded using dummy variables (or indicator variables), that is, separate explanatory variables taking the value 0 or 1 are created for each possible value of the discrete variable, with a 1 meaning "variable does have the given value" and a 0 meaning "variable does not have that value". For example, a four-way discrete variable of blood type with the possible values "A, B, AB, O" can be converted to four separate two-way dummy variables, "is-A, is-B, is-AB, is-O", where only one of them has the value 1 and all the rest have the value 0. This allows for separate regression coefficients to be matched for each possible value of the discrete variable. (In a case like this, only three of the four dummy variables are independent of each other, in the sense that once the values of three of the variables are known, the fourth is automatically determined. Thus, it is necessary to encode only three of the four possibilities as dummy variables. This also means that when all four possibilities are encoded, the overall model is not identifiable in the absence of additional constraints such as a regularization constraint. Theoretically, this could cause problems, but in reality almost all logistic regression models are fitted with regularization constraints.)

Outcome variables

Formally, the outcomes *Yi* are described as being Bernoulli-distributed data, where each outcome is determined by an unobserved probability *pi* that is specific to the outcome at hand, but related to the explanatory variables. This can be expressed in any of the following equivalent forms:

$$\begin{aligned}
Y_i \mid x_{1,i}, \ldots, x_{m,i} &\sim \text{Bernoulli}(p_i) \\
\mathbb{E}[Y_i \mid x_{1,i}, \ldots, x_{m,i}] &= p_i \\
\Pr(Y_i = y_i \mid x_{1,i}, \ldots, x_{m,i}) &= \begin{cases} p_i & \text{if} y_i = 1 \\ 1 - p_i & \text{if} y_i = 0 \end{cases} \\
\Pr(Y_i = y_i \mid x_{1,i}, \ldots, x_{m,i}) &= p_i^{y_i} (1 - p_i)^{(1-y_i)}
\end{aligned}$$

The meanings of these four lines are:

1. The first line expresses the probability distribution of each *Yi*: Conditioned on the explanatory variables, it follows a Bernoulli distribution with parameters *pi*, the probability of the outcome of 1 for trial *i*. As noted above, each separate trial has its own probability of success, just as each trial has its own explanatory variables. The probability of success *pi* is not observed, only the outcome of an individual Bernoulli trial using that probability.
2. The second line expresses the fact that the expected value of each *Yi* is equal to the probability of success *pi*, which is a general property of the Bernoulli distribution. In other words, if we run a large number of Bernoulli trials using the same probability of success *pi*, then take the average of all the 1 and 0 outcomes, then the result would be close to *pi*. This is because doing an average this way simply computes the proportion of successes seen, which we expect to converge to the underlying probability of success.
3. The third line writes out the probability mass function of the Bernoulli distribution, specifying the probability of seeing each of the two possible outcomes.
4. The fourth line is another way of writing the probability mass function, which avoids having to write separate cases and is more convenient for certain types of calculations. This relies on the fact that *Yi* can take only the value 0 or 1. In each case, one of the exponents will be 1, "choosing" the value under it, while the other is 0, "canceling out" the value under it. Hence, the outcome is either *pi* or 1 − *pi*, as in the previous line.

Linear predictor function

The basic idea of logistic regression is to use the mechanism already developed for linear regression by modeling the probability *pi* using a linear predictor function, i.e. a linear combination of the explanatory variables and a set of regression coefficients that are specific to the model at hand but the same for all trials. The linear predictor function $f(i)$ for a particular data point *i* is written as:

$$f(i) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_m x_{m,i},$$

where $\beta_0, \ldots, \beta_m$ are regression coefficients indicating the relative effect of a particular explanatory variable on the outcome.

The model is usually put into a more compact form as follows:

- The regression coefficients β_0, β_1, ..., βm are grouped into a single vector $\boldsymbol{\beta}$ of size *m* + 1.
- For each data point *i*, an additional explanatory pseudo-variable x_0,*i* is added, with a fixed value of 1, corresponding to the intercept coefficient β_0.
- The resulting explanatory variables x_0,*i*, x_1,*i*, ..., *xm,i* are then grouped into a single vector ***Xi*** of size *m* + 1.

This makes it possible to write the linear predictor function as follows:

$$f(i) = \boldsymbol{\beta} \cdot \mathbf{X}_i,$$

using the notation for a dot product between two vectors.

36.6.2 As a generalized linear model

The particular model used by logistic regression, which distinguishes it from standard linear regression and from other types of regression analysis used for binary-valued outcomes, is the way the probability of a particular outcome is linked to the linear predictor function:

$$\operatorname{logit}(\mathbb{E}[Y_i \mid x_{1,i}, \ldots, x_{m,i}]) = \operatorname{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_m x_{m,i}$$

Written using the more compact notation described above, this is:

$$\operatorname{logit}(\mathbb{E}[Y_i \mid \mathbf{X}_i]) = \operatorname{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \boldsymbol{\beta} \cdot \mathbf{X}_i$$

This formulation expresses logistic regression as a type of generalized linear model, which predicts variables with various types of probability distributions by fitting a linear predictor function of the above form to some sort of arbitrary transformation of the expected value of the variable.

The intuition for transforming using the logit function (the natural log of the odds) was explained above. It also has the practical effect of converting the probability (which is bounded to be between 0 and 1) to a variable that ranges over $(-\infty, +\infty)$ — thereby matching the potential range of the linear prediction function on the right side of the equation.

Note that both the probabilities *pi* and the regression coefficients are unobserved, and the means of determining them is not part of the model itself. They are typically determined by some sort of optimization procedure, e.g. maximum likelihood estimation, that finds values that best fit the observed data (i.e. that give the most accurate predictions for the data already observed), usually subject to regularization conditions that seek to exclude unlikely values, e.g. extremely large values for any of the regression coefficients. The use of a regularization condition is equivalent to doing maximum a posteriori (MAP) estimation, an extension of maximum likelihood. (Regularization is most commonly done using a squared regularizing function, which is equivalent to placing a zero-mean Gaussian prior distribution on the coefficients, but other regularizers are also possible.) Whether or not regularization is used, it is usually not possible to find a closed-form solution; instead, an iterative numerical method must be used, such as iteratively reweighted least squares (IRLS) or, more commonly these days, a quasi-Newton method such as the L-BFGS method.

The interpretation of the βj parameter estimates is as the additive effect on the log of the odds for a unit change in the *j*th explanatory variable. In the case of a dichotomous explanatory variable, for instance gender, e^β is the estimate of the odds of having the outcome for, say, males compared with females.

An equivalent formula uses the inverse of the logit function, which is the logistic function, i.e.:

$$\mathbb{E}[Y_i \mid \mathbf{X}_i] = p_i = \operatorname{logit}^{-1}(\boldsymbol{\beta} \cdot \mathbf{X}_i) = \frac{1}{1+e^{-\boldsymbol{\beta} \cdot \mathbf{X}_i}}$$

The formula can also be written as a probability distribution (specifically, using a probability mass function):

$$\Pr(Y_i = y_i \mid \mathbf{X}_i) = {p_i}^{y_i}(1-p_i)^{1-y_i} = \left(\frac{e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}{1+e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}\right)^{y_i}\left(1-\frac{e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}{1+e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}\right)$$

36.6.3 As a latent-variable model

The above model has an equivalent formulation as a latent-variable model. This formulation is common in the theory of discrete choice models, and makes it easier to extend to certain more complicated models with multiple, correlated choices, as well as to compare logistic regression to the closely related probit model.

Imagine that, for each trial *i*, there is a continuous latent variable *Yi** (i.e. an unobserved random variable) that is distributed as follows:

$$Y_i^\ast = \boldsymbol\beta \cdot \mathbf{X}_i + \varepsilon$$

where

$$\varepsilon \sim \operatorname{Logistic}(0,1)$$

i.e. the latent variable can be written directly in terms of the linear predictor function and an additive random error variable that is distributed according to a standard logistic distribution.

Then *Yi* can be viewed as an indicator for whether this latent variable is positive:

$$Y_i = \begin{cases} 1 & \text{if} Y_i^\ast > 0 \ \text{ i.e. } -\varepsilon < \boldsymbol\beta \cdot \mathbf{X}_i, \\ 0 & \text{otherwise.} \end{cases}$$

The choice of modeling the error variable specifically with a standard logistic distribution, rather than a general logistic distribution with the location and scale set to arbitrary values, seems restrictive, but in fact it is not. It must be kept in mind that we can choose the regression coefficients ourselves, and very often can use them to offset changes in the parameters of the error variable's distribution. For example, a logistic error-variable distribution with a non-zero location parameter *μ* (which sets the mean) is equivalent to a distribution with a zero location parameter, where *μ* has been added to the intercept coefficient. Both situations produce the same value for *Yi** regardless of settings of explanatory variables. Similarly, an arbitrary scale parameter *s* is equivalent to setting the scale parameter to 1 and then dividing all regression coefficients by *s*. In the latter case, the resulting value of *Yi** will be smaller by a factor of *s* than in the former case, for all sets of explanatory variables — but critically, it will always remain on the same side of 0, and hence lead to the same *Yi* choice.

(Note that this predicts that the irrelevancy of the scale parameter may not carry over into more complex models where more than two choices are available.)

It turns out that this formulation is exactly equivalent to the preceding one, phrased in terms of the generalized linear model and without any latent variables. This can be shown as follows, using the fact that the cumulative distribution function (CDF) of the standard logistic distribution is the logistic function, which is the inverse of the logit function, i.e.

$$\Pr(\varepsilon < x) = \operatorname{logit}^{-1}(x)$$

Then:

$$\begin{aligned} \Pr(Y_i = 1 \mid \mathbf{X}_i) &= \Pr(Y_i^\ast > 0 \mid \mathbf{X}_i) \\ &= \Pr(\boldsymbol\beta \cdot \mathbf{X}_i + \varepsilon > 0) \\ &= \Pr(\varepsilon > -\boldsymbol\beta \cdot \mathbf{X}_i) \\ &= \Pr(\varepsilon < \boldsymbol\beta \cdot \mathbf{X}_i) && \text{symmetric) is distribution logisti} \\ &= \operatorname{logit}^{-1}(\boldsymbol\beta \cdot \mathbf{X}_i) \\ &= p_i && \text{above) (see} \end{aligned}$$

This formulation—which is standard in discrete choice models—makes clear the relationship between logistic regression (the "logit model") and the probit model, which uses an error variable distributed according to a standard normal distribution instead of a standard logistic distribution. Both the logistic and normal distributions are symmetric with a basic unimodal, "bell curve" shape. The only difference is that the logistic distribution has somewhat heavier tails, which means that it is less sensitive to outlying data (and hence somewhat more robust to model mis-specifications or erroneous data).

36.6.4 As a two-way latent-variable model

Yet another formulation uses two separate latent variables:

$$\begin{aligned} Y_i^{0\ast} &= \boldsymbol\beta_0 \cdot \mathbf{X}_i + \varepsilon_0 \\ Y_i^{1\ast} &= \boldsymbol\beta_1 \cdot \mathbf{X}_i + \varepsilon_1 \end{aligned}$$

where

$$\begin{aligned} \varepsilon_0 &\sim \operatorname{EV}_1(0,1) \\ \varepsilon_1 &\sim \operatorname{EV}_1(0,1) \end{aligned}$$

where $EV_1(0,1)$ is a standard type-1 extreme value distribution: i.e.

$$\Pr(\varepsilon_0 = x) = \Pr(\varepsilon_1 = x) = e^{-x} e^{-e^{-x}}$$

Then

$$Y_i = \begin{cases} 1 & \text{if} Y_i^{1\ast} > Y_i^{0\ast}, \\ 0 & \text{otherwise.} \end{cases}$$

This model has a separate latent variable and a separate set of regression coefficients for each possible outcome of the dependent variable. The reason for this separation is that it makes it easy to extend logistic regression to multi-outcome categorical variables, as in the multinomial logit model. In such a model, it is natural to model each possible outcome using a different set of regression coefficients. It is also possible to motivate each of the separate latent variables as the theoretical utility associated with making the associated choice, and thus motivate logistic regression in terms of utility theory. (In terms of utility theory, a rational actor always chooses the choice with the greatest associated utility.) This is the approach taken by economists when formulating discrete choice models, because it both provides a theoretically strong foundation and facilitates intuitions about the model, which in turn makes it easy to consider various sorts of extensions. (See the example below.)

The choice of the type-1 extreme value distribution seems fairly arbitrary, but it makes the mathematics work out, and it may be possible to justify its use through rational choice theory.

It turns out that this model is equivalent to the previous model, although this seems non-obvious, since there are now two sets of regression coefficients and error variables, and the error variables have a different distribution. In fact, this model reduces directly to the previous one with the following substitutions:

$$\boldsymbol\beta = \boldsymbol\beta_1 - \boldsymbol\beta_0$$

$$\varepsilon = \varepsilon_1 - \varepsilon_0$$

An intuition for this comes from the fact that, since we choose based on the maximum of two values, only their difference matters, not the exact values — and this effectively removes one degree of freedom. Another critical fact is that the difference of two type-1 extreme-value-distributed variables is a logistic distribution, i.e. if $\varepsilon = \varepsilon_1 - \varepsilon_0 \sim \text{Logistic}(0, 1)$.

We can demonstrate the equivalent as follows:

$$\begin{aligned}
&\Pr(Y_i = 1 \mid \mathbf{X}_i) \\
=\ &\Pr(Y_i^{1*} > Y_i^{0*} \mid \mathbf{X}_i) \\
=\ &\Pr(Y_i^{1*} - Y_i^{0*} > 0 \mid \mathbf{X}_i) \\
=\ &\Pr(\boldsymbol\beta_1 \cdot \mathbf{X}_i + \varepsilon_1 - (\boldsymbol\beta_0 \cdot \mathbf{X}_i + \varepsilon_0) > 0) \\
=\ &\Pr((\boldsymbol\beta_1 \cdot \mathbf{X}_i - \boldsymbol\beta_0 \cdot \mathbf{X}_i) + (\varepsilon_1 - \varepsilon_0) > 0) \\
=\ &\Pr((\boldsymbol\beta_1 - \boldsymbol\beta_0) \cdot \mathbf{X}_i + (\varepsilon_1 - \varepsilon_0) > 0) \\
=\ &\Pr((\boldsymbol\beta_1 - \boldsymbol\beta_0) \cdot \mathbf{X}_i + \varepsilon > 0) && \text{(substitute}\varepsilon\text{above) as} \\
=\ &\Pr(\boldsymbol\beta \cdot \mathbf{X}_i + \varepsilon > 0) && \text{(substitute}\boldsymbol\beta\text{above) as} \\
=\ &\Pr(\varepsilon > -\boldsymbol\beta \cdot \mathbf{X}_i) && \text{model) above as same (now} \\
=\ &\Pr(\varepsilon < \boldsymbol\beta \cdot \mathbf{X}_i) \\
=\ &\operatorname{logit}^{-1}(\boldsymbol\beta \cdot \mathbf{X}_i) \\
=\ &p_i
\end{aligned}$$

Example

As an example, consider a province-level election where the choice is between a right-of-center party, a left-of-center party, and a secessionist party (e.g. the Parti Québécois, which wants Quebec to secede from Canada). We would then use three latent variables, one for each choice. Then, in accordance with utility theory, we can then interpret the latent variables as expressing the utility that results from making each of the choices. We can also interpret the regression coefficients as indicating the strength that the associated factor (i.e. explanatory variable) has in contributing to the utility — or more correctly, the amount by which a unit change in an explanatory variable changes the utility of a given choice. A voter might expect that the right-of-center party would lower taxes, especially on rich people. This would give low-income people no benefit, i.e. no change in utility (since they usually don't pay taxes); would cause moderate benefit (i.e. somewhat more money, or moderate utility increase) for middle-incoming people; and would cause significant benefits for high-income people. On the other hand, the left-of-center party might be expected to raise taxes and offset it with increased welfare and other assistance for the lower and middle classes. This would cause significant positive benefit to low-income people, perhaps weak benefit to middle-income people, and significant negative benefit to high-income people. Finally, the secessionist party would take no direct actions on the economy, but simply secede. A low-income or middle-income voter might expect basically no clear utility gain or loss from this, but a high-income voter might expect negative utility, since he/she is likely to own companies, which will have a harder time doing business in such an environment and probably lose money.

These intuitions can be expressed as follows:

This clearly shows that

1. Separate sets of regression coefficients need to exist for each choice. When phrased in terms of utility, this can be seen very easily. Different choices have different effects on net utility; furthermore, the effects vary in complex ways that depend on the characteristics of each individual, so there need to be separate sets of coefficients for each characteristic, not simply a single extra per-choice characteristic.

2. Even though income is a continuous variable, its effect on utility is too complex for it to be treated as a single variable. Either it needs to be directly split up into ranges, or higher powers of income need to be added so that polynomial regression on income is effectively done.

36.6.5 As a "log-linear" model

Yet another formulation combines the two-way latent variable formulation above with the original formulation higher up without latent variables, and in the process provides a link to one of the standard formulations of the multinomial logit.

Here, instead of writing the logit of the probabilities *pi* as a linear predictor, we separate the linear predictor into two, one for each of the two outcomes:

$$\ln \Pr(Y_i = 0) = \boldsymbol\beta_0 \cdot \mathbf{X}_i - \ln Z$$
$$\ln \Pr(Y_i = 1) = \boldsymbol\beta_1 \cdot \mathbf{X}_i - \ln Z$$

Note that two separate sets of regression coefficients have been introduced, just as in the two-way latent variable model, and the two equations appear a form that writes the logarithm of the associated probability as a linear predictor, with an extra term $-lnZ$ at the end. This term, as it turns out, serves as the normalizing factor ensuring that the result is a distribution. This can be seen by exponentiating both sides:

$$\Pr(Y_i = 0) = \frac{1}{Z} e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i}$$
$$\Pr(Y_i = 1) = \frac{1}{Z} e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}$$

In this form it is clear that the purpose of Z is to ensure that the resulting distribution over Yi is in fact a probability distribution, i.e. it sums to 1. This means that Z is simply the sum of all un-normalized probabilities, and by dividing each probability by Z, the probabilities become "normalized". That is:

$$Z = e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}$$

and the resulting equations are

$$\Pr(Y_i = 0) = \frac{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}$$
$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}$$

Or generally:

$$\Pr(Y_i = c) = \frac{e^{\boldsymbol\beta_c \cdot \mathbf{X}_i}}{\sum_h e^{\boldsymbol\beta_h \cdot \mathbf{X}_i}}$$

This shows clearly how to generalize this formulation to more than two outcomes, as in multinomial logit. Note that this general formulation is exactly the Softmax function as in

$$\Pr(Y_i = c) = \operatorname{softmax}(c, \boldsymbol\beta_0 \cdot \mathbf{X}_i, \boldsymbol\beta_1 \cdot \mathbf{X}_i, \dots).$$

In order to prove that this is equivalent to the previous model, note that the above model is overspecified, in that $\Pr(Y_i = 0)$ and $\Pr(Y_i = 1)$ cannot be independently specified: rather $\Pr(Y_i = 0) + \Pr(Y_i = 1) = 1$ so knowing one automatically determines the other. As a result, the model is nonidentifiable, in that multiple combinations of $\boldsymbol\beta_0$ and $\boldsymbol\beta_1$ will produce the same probabilities for all possible explanatory variables. In fact, it can be seen that adding any constant vector to both of them will produce the same probabilities:

$$\begin{aligned}\Pr(Y_i = 1) &= \frac{e^{(\boldsymbol\beta_1 + \mathbf{C}) \cdot \mathbf{X}_i}}{e^{(\boldsymbol\beta_0 + \mathbf{C}) \cdot \mathbf{X}_i} + e^{(\boldsymbol\beta_1 + \mathbf{C}) \cdot \mathbf{X}_i}} \\ &= \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i} e^{\mathbf{C} \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} e^{\mathbf{C} \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i} e^{\mathbf{C} \cdot \mathbf{X}_i}} \\ &= \frac{e^{\mathbf{C} \cdot \mathbf{X}_i} e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{e^{\mathbf{C} \cdot \mathbf{X}_i} (e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i})} \\ &= \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}\end{aligned}$$

As a result, we can simplify matters, and restore identifiability, by picking an arbitrary value for one of the two vectors. We choose to set $\boldsymbol\beta_0 = \mathbf{0}$. Then,

$$e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} = e^{\mathbf{0} \cdot \mathbf{X}_i} = 1$$

and so

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{1 + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}} = \frac{1}{1 + e^{-\boldsymbol\beta_1 \cdot \mathbf{X}_i}} = p_i$$

which shows that this formulation is indeed equivalent to the previous formulation. (As in the two-way latent variable formulation, any settings where $\boldsymbol\beta = \boldsymbol\beta_1 - \boldsymbol\beta_0$ will produce equivalent results.)

Note that most treatments of the multinomial logit model start out either by extending the “log-linear” formulation presented here or the two-way latent variable formulation presented above, since both clearly show the way that the model could be extended to multi-way outcomes. In general, the presentation with latent variables is more common in econometrics and political science, where discrete choice models and utility theory reign, while the “log-linear” formulation here is more common in computer science, e.g. machine learning and natural language processing.

36.6.6 As a single-layer perceptron

The model has an equivalent formulation

$$p_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i})}}.$$

This functional form is commonly called a single-layer perceptron or single-layer artificial neural network. A single-layer neural network computes a continuous output instead of a step function. The derivative of *pi* with respect to $X = (x_1, ..., xk)$ is computed from the general form:

$$y = \frac{1}{1+e^{-f(X)}}$$

where $f(X)$ is an analytic function in X. With this choice, the single-layer neural network is identical to the logistic regression model. This function has a continuous derivative, which allows it to be used in backpropagation. This function is also preferred because its derivative is easily calculated:

$$\frac{\mathrm{d}y}{\mathrm{d}X} = y(1-y)\frac{\mathrm{d}f}{\mathrm{d}X}.$$

36.6.7 In terms of binomial data

A closely related model assumes that each i is associated not with a single Bernoulli trial but with ni independent identically distributed trials, where the observation Yi is the number of successes observed (the sum of the individual Bernoulli-distributed random variables), and hence follows a binomial distribution:

$$Y_i \ \sim \mathrm{Bin}(n_i, p_i), \text{ for } i = 1, \dots, n$$

An example of this distribution is the fraction of seeds (pi) that germinate after ni are planted.

In terms of expected values, this model is expressed as follows:

$$p_i = \mathbb{E}\left[\left.\frac{Y_i}{n_i}\,\right|\,\mathbf{X}_i\right],$$

so that

$$\operatorname{logit}\left(\mathbb{E}\left[\left.\frac{Y_i}{n_i}\,\right|\,\mathbf{X}_i\right]\right) = \operatorname{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \boldsymbol\beta\cdot\mathbf{X}_i,$$

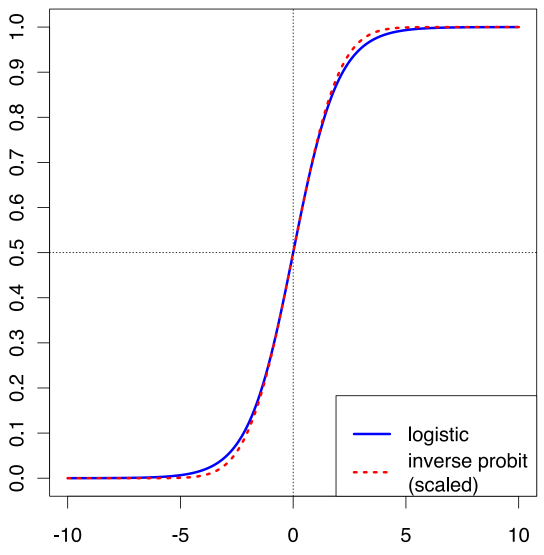
Or equivalently:

$$\Pr(Y_i = y_i \mid \mathbf{X}_i) = \binom{n_i}{y_i} p_i^{y_i}(1-p_i)^{n_i-y_i} = \binom{n_i}{y_i}\left(\frac{1}{1+e^{-\boldsymbol\beta\cdot\mathbf{X}_i}}\right)^{y_i}\left(1-\frac{1}{1+e^{-\boldsymbol\beta\cdot\mathbf{X}_i}}\right)^{n_i-y_i}$$

This model can be fit using the same sorts of methods as the above more basic model.

36.7 Bayesian logistic regression

In a Bayesian statistics context, prior distributions are normally placed on the regression coefficients, usually in



Comparison of logistic function with a scaled inverse probit function (i.e. the CDF of the normal distribution), comparing $\sigma(x)$ vs. $\Phi(\sqrt{\frac{\pi}{8}}x)$, which makes the slopes the same at the origin. This shows the heavier tails of the logistic distribution.

the form of Gaussian distributions. Unfortunately, the Gaussian distribution is not the conjugate prior of the likelihood function in logistic regression. As a result, the posterior distribution is difficult to calculate, even using standard simulation algorithms (e.g. Gibbs sampling).

There are various possibilities:

- Don't do a proper Bayesian analysis, but simply compute a maximum a posteriori point estimate of the parameters. This is common, for example, in “maximum entropy” classifiers in machine learning.
- Use a more general approximation method such as the Metropolis–Hastings algorithm.
- Draw a Markov chain Monte Carlo sample from the exact posterior by using the Independent Metropolis–Hastings algorithm with heavy-tailed multivariate candidate distribution found by matching the mode and curvature at the mode of the normal approximation to the posterior and then using the Student’s t shape with low degrees of freedom.[22] This is shown to have excellent convergence properties.
- Use a latent variable model and approximate the logistic distribution using a more tractable distribution, e.g. a Student’s t-distribution or a mixture of normal distributions.
- Do probit regression instead of logistic regression. This is actually a special case of the previous situation, using a normal distribution in place of a Student’s t, mixture of normals, etc. This will be less accurate but has the advantage that probit regression

is extremely common, and a ready-made Bayesian implementation may already be available.

- Use the Laplace approximation of the posterior distribution.[23] This approximates the posterior with a Gaussian distribution. This is not a terribly good approximation, but it suffices if all that is desired is an estimate of the posterior mean and variance. In such a case, an approximation scheme such as variational Bayes can be used.[24]

36.7.1 Gibbs sampling with an approximating distribution

As shown above, logistic regression is equivalent to a latent variable model with an error variable distributed according to a standard logistic distribution. The overall distribution of the latent variable Y_i* is also a logistic distribution, with the mean equal to $\boldsymbol{\beta} \cdot \mathbf{X}_i$ (i.e. the fixed quantity added to the error variable). This model considerably simplifies the application of techniques such as Gibbs sampling. However, sampling the regression coefficients is still difficult, because of the lack of conjugacy between the normal and logistic distributions. Changing the prior distribution over the regression coefficients is of no help, because the logistic distribution is not in the exponential family and thus has no conjugate prior.

One possibility is to use a more general Markov chain Monte Carlo technique, such as the Metropolis–Hastings algorithm, which can sample arbitrary distributions. Another possibility, however, is to replace the logistic distribution with a similar-shaped distribution that is easier to work with using Gibbs sampling. In fact, the logistic and normal distributions have a similar shape, and thus one possibility is simply to have normally distributed errors. Because the normal distribution is conjugate to itself, sampling the regression coefficients becomes easy. In fact, this model is exactly the model used in probit regression.

However, the normal and logistic distributions differ in that the logistic has heavier tails. As a result, it is more robust to inaccuracies in the underlying model (which are inevitable, in that the model is essentially always an approximation) or to errors in the data. Probit regression loses some of this robustness.

Another alternative is to use errors distributed as a Student's t-distribution. The Student's t-distribution has heavy tails, and is easy to sample from because it is the compound distribution of a normal distribution with variance distributed as an inverse gamma distribution. In other words, if a normal distribution is used for the error variable, and another latent variable, following an inverse gamma distribution, is added corresponding to the variance of this error variable, the marginal distribution of the error variable will follow a Student's t distribution. Because of the various conjugacy relationships, all variables in this model are easy to sample from.

The Student's t distribution that best approximates a standard logistic distribution can be determined by matching the moments of the two distributions. The Student's t distribution has three parameters, and since the skewness of both distributions is always 0, the first four moments can all be matched, using the following equations:

$$\begin{aligned}\mu &= 0\\ \frac{\nu}{\nu-2}s^2 &= \frac{\pi^2}{3}\\ \frac{6}{\nu-4} &= \frac{6}{5}\end{aligned}$$

This yields the following values:

$$\begin{aligned}&\mu = 0\\ &s = \sqrt{\frac{7}{9}\frac{\pi^2}{3}}\\ &\nu = 9\end{aligned}$$

The following graphs compare the standard logistic distribution with the Student's t distribution that matches the first four moments using the above-determined values, as well as the normal distribution that matches the first two moments. Note how much closer the Student's t distribution agrees, especially in the tails. Beyond about two standard deviations from the mean, the logistic and normal distributions diverge rapidly, but the logistic and Student's t distributions don't start diverging significantly until more than 5 standard deviations away.

(Another possibility, also amenable to Gibbs sampling, is to approximate the logistic distribution using a mixture density of normal distributions.)

36.8 Extensions

There are large numbers of extensions:

- Multinomial logistic regression (or **multinomial logit**) handles the case of a multi-way categorical dependent variable (with unordered values, also called "classification"). Note that the general case of having dependent variables with more than two values is termed *polytomous regression*.
- Ordered logistic regression (or **ordered logit**) handles ordinal dependent variables (ordered values).
- Mixed logit is an extension of multinomial logit that allows for correlations among the choices of the dependent variable.
- An extension of the logistic model to sets of interdependent variables is the conditional random field.

36.9 Model suitability

A way to measure a model's suitability is to assess the model against a set of data that was not used to create the model.[25] The class of techniques is called cross-validation. This holdout model assessment method is particularly valuable when data are collected in different settings (e.g., at different times or places) or when models are assumed to be generalizable.

To measure the suitability of a binary regression model, one can classify both the actual value and the predicted value of each observation as either 0 or 1.[26] The predicted value of an observation can be set equal to 1 if the estimated probability that the observation equals 1 is above $\frac{1}{2}$, and set equal to 0 if the estimated probability is below $\frac{1}{2}$. Here logistic regression is being used as a binary classification model. There are four possible combined classifications:

1. prediction of 0 when the holdout sample has a 0 (True Negatives, the number of which is TN)
2. prediction of 0 when the holdout sample has a 1 (False Negatives, the number of which is FN)
3. prediction of 1 when the holdout sample has a 0 (False Positives, the number of which is FP)
4. prediction of 1 when the holdout sample has a 1 (True Positives, the number of which is TP)

These classifications are used to calculate accuracy, precision (also called positive predictive value), recall (also called sensitivity), specificity and negative predictive value:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

$$\text{Precision} = \text{value predictive Positive} = \frac{TP}{TP+FP}$$

$$\text{value predictive Negative} = \frac{TN}{TN+FN}$$

$$\text{Recall} = \text{Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{Specificity} = \frac{TN}{TN+FP}$$

36.10 See also

- Logistic function
- Discrete choice
- Jarrow–Turnbull model
- Limited dependent variable
- Multinomial logit model
- Ordered logit
- Hosmer–Lemeshow test
- Brier score
- MLPACK - contains a C++ implementation of logistic regression
- Local case-control sampling

36.11 References

[1] David A. Freedman (2009). *Statistical Models: Theory and Practice*. Cambridge University Press. p. 128.

[2] Cox, DR (1958). "The regression analysis of binary sequences (with discussion)". *J Roy Stat Soc B* **20**: 215–242.

[3] Walker, SH; Duncan, DB (1967). "Estimation of the probability of an event as a function of several independent variables". *Biometrika* **54**: 167–178.

[4] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Springer. p. 6.

[5] Boyd, C. R.; Tolson, M. A.; Copes, W. S. (1987). "Evaluating trauma care: The TRISS method. Trauma Score and the Injury Severity Score". *The Journal of trauma* **27** (4): 370–378. doi:10.1097/00005373-198704000-00005. PMID 3106646.

[6] Kologlu M., Elker D., Altun H., Sayek I. Valdation of MPI and OIA II in two different groups of patients with secondary peritonitis // Hepato-Gastroenterology. – 2001. – Vol. 48, № 37. – P. 147-151.

[7] Biondo S., Ramos E., Deiros M. et al. Prognostic factors for mortality in left colonic peritonitis: a new scoring system // J. Am. Coll. Surg. – 2000. – Vol. 191, № 6. – P. 635-642.

[8] Marshall J.C., Cook D.J., Christou N.V. et al. Multiple Organ Dysfunction Score: A reliable descriptor of a complex clinical outcome // Crit. Care Med. – 1995. – Vol. 23. – P. 1638-1652.

[9] Le Gall J.-R., Lemeshow S., Saulnier F. A new Simplified Acute Physiology Score (SAPS II) based on a European/North American multicenter study // JAMA. – 1993. – Vol. 270. – P. 2957-2963.

[10] Truett, J; Cornfield, J; Kannel, W (1967). "A multivariate analysis of the risk of coronary heart disease in Framingham". *Journal of chronic diseases* **20** (7): 511–24. PMID 6028270.

[11] Harrell, Frank E. (2001). *Regression Modeling Strategies*. Springer-Verlag. ISBN 0-387-95232-2.

[12] M. Strano; B.M. Colosimo (2006). "Logistic regression analysis for experimental determination of forming limit diagrams". *International Journal of Machine Tools and Manufacture* **46** (6). doi:10.1016/j.ijmachtools.2005.07.005.

[13] Palei, S. K.; Das, S. K. (2009). "Logistic regression model for prediction of roof fall risks in bord and pillar workings in coal mines: An approach". *Safety Science* **47**: 88. doi:10.1016/j.ssci.2008.01.002.

[14] Hosmer, David W.; Lemeshow, Stanley (2000). *Applied Logistic Regression* (2nd ed.). Wiley. ISBN 0-471-35632-8.

[15] http://www.planta.cn/forum/files_planta/introduction_to_categorical_data_analysis_805.pdf

[16] Menard, Scott W. (2002). *Applied Logistic Regression* (2nd ed.). SAGE. ISBN 978-0-7619-2208-7.

[17] Menard ch 1.3

[18] Peduzzi, P; Concato, J; Kemper, E; Holford, TR; Feinstein, AR (December 1996). "A simulation study of the number of events per variable in logistic regression analysis.". *Journal of Clinical Epidemiology* **49** (12): 1373–9. doi:10.1016/s0895-4356(96)00236-3. PMID 8970487.

[19] Greene, William N. (2003). *Econometric Analysis* (Fifth ed.). Prentice-Hall. ISBN 0-13-066189-9.

[20] Cohen, Jacob; Cohen, Patricia; West, Steven G.; Aiken, Leona S. (2002). *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences* (3rd ed.). Routledge. ISBN 978-0-8058-2223-6.

[21] https://class.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/classification.pdf slide 16

[22] Bolstad, William M. (2010). *Understandeing Computational Bayesian Statistics*. Wiley. ISBN 978-0-470-04609-8.

[23] Bishop, Christopher M. "Chapter 4. Linear Models for Classification". *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC. pp. 217–218. ISBN 978-0387-31073-2.

[24] Bishop, Christopher M. "Chapter 10. Approximate Inference". *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC. pp. 498–505. ISBN 978-0387-31073-2.

[25] Jonathan Mark and Michael A. Goldberg (2001). Multiple Regression Analysis and Mass Assessment: A Review of the Issues. The Appraisal Journal, Jan. pp. 89–109

[26] Myers, J. H.; Forgy, E. W. (1963). "The Development of Numerical Credit Evaluation Systems". *J. Amer. Statist. Assoc.* **58** (303): 799–806. doi:10.1080/01621459.1963.10500889.

36.12 Further reading

- Agresti, Alan. (2002). *Categorical Data Analysis*. New York: Wiley-Interscience. ISBN 0-471-36093-7.
- Amemiya, T. (1985). *Advanced Econometrics*. Harvard University Press. ISBN 0-674-00560-0.
- Balakrishnan, N. (1991). *Handbook of the Logistic Distribution*. Marcel Dekker, Inc. ISBN 978-0-8247-8587-1.
- Greene, William H. (2003). *Econometric Analysis, fifth edition*. Prentice Hall. ISBN 0-13-066189-9.
- Hilbe, Joseph M. (2009). *Logistic Regression Models*. Chapman & Hall/CRC Press. ISBN 978-1-4200-7575-5.
- Howell, David C. (2010). *Statistical Methods for Psychology, 7th ed*. Belmont, CA; Thomson Wadsworth. ISBN 978-0-495-59786-5.
- Peduzzi, P.; J. Concato; E. Kemper; T.R. Holford; A.R. Feinstein (1996). "A simulation study of the number of events per variable in logistic regression analysis". *Journal of Clinical Epidemiology* **49** (12): 1373–1379. doi:10.1016/s0895-4356(96)00236-3. PMID 8970487.

36.13 External links

- Econometrics Lecture (topic: Logit model) on YouTube by Mark Thoma
- Logistic Regression Interpretation
- Logistic Regression tutorial
- Using open source software for building Logistic Regression models
- Logistic regression. Biomedical statistics

Chapter 37

Linear discriminant analysis

Not to be confused with latent Dirichlet allocation.

Linear discriminant analysis (**LDA**) is a generalization of **Fisher's linear discriminant**, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

LDA is closely related to analysis of variance (ANOVA) and regression analysis, which also attempt to express one dependent variable as a linear combination of other features or measurements.[1][2] However, ANOVA uses categorical independent variables and a continuous dependent variable, whereas discriminant analysis has continuous independent variables and a categorical dependent variable (*i.e.* the class label).[3] Logistic regression and probit regression are more similar to LDA than ANOVA is, as they also explain a categorical variable by the values of continuous independent variables. These other methods are preferable in applications where it is not reasonable to assume that the independent variables are normally distributed, which is a fundamental assumption of the LDA method.

LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data.[4] LDA explicitly attempts to model the difference between the classes of data. PCA on the other hand does not take into account any difference in class, and factor analysis builds the feature combinations based on differences rather than similarities. Discriminant analysis is also different from factor analysis in that it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made.

LDA works when the measurements made on independent variables for each observation are continuous quantities. When dealing with categorical independent variables, the equivalent technique is discriminant correspondence analysis.[5][6]

37.1 LDA for two classes

Consider a set of observations $\vec{x}$ (also called features, attributes, variables or measurements) for each sample of an object or event with known class y. This set of samples is called the training set. The classification problem is then to find a good predictor for the class y of any sample of the same distribution (not necessarily from the training set) given only an observation $\vec{x}$.[7]:338

LDA approaches the problem by assuming that the conditional probability density functions $p(\vec{x}|y=0)$ and $p(\vec{x}|y=1)$ are both normally distributed with mean and covariance parameters $(\vec{\mu}_0, \Sigma_0)$ and $(\vec{\mu}_1, \Sigma_1)$, respectively. Under this assumption, the Bayes optimal solution is to predict points as being from the second class if the log of the likelihood ratios is below some threshold T, so that;

$$(\vec{x}-\vec{\mu}_0)^T\Sigma_0^{-1}(\vec{x}-\vec{\mu}_0)+\ln|\Sigma_0|-(\vec{x}-\vec{\mu}_1)^T\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1)-\ln|\Sigma_1| < T$$

Without any further assumptions, the resulting classifier is referred to as QDA (quadratic discriminant analysis).

LDA instead makes the additional simplifying homoscedasticity assumption (*i.e.* that the class covariances are identical, so $\Sigma_0 = \Sigma_1 = \Sigma$) and that the covariances have full rank. In this case, several terms cancel:

$$\vec{x}^T\Sigma_0^{-1}\vec{x} = \vec{x}^T\Sigma_1^{-1}\vec{x}$$

$\vec{x}^T{\Sigma_i}^{-1}\vec{\mu}_i = {\vec{\mu}_i}^T{\Sigma_i}^{-1}\vec{x}$ because Σ_i is Hermitian

and the above decision criterion becomes a threshold on the dot product

$$\vec{w}\cdot\vec{x} > c$$

for some threshold constant c, where

$$\vec{w} = \Sigma^{-1}(\vec{\mu}_1 - \vec{\mu}_0)$$

$$c = \frac{1}{2}(T - \vec{\mu_0}^T \Sigma_0^{-1} \vec{\mu_0} + \vec{\mu_1}^T \Sigma_1^{-1} \vec{\mu_1})$$

This means that the criterion of an input $\vec{x}$ being in a class y is purely a function of this linear combination of the known observations.

It is often useful to see this conclusion in geometrical terms: the criterion of an input $\vec{x}$ being in a class y is purely a function of projection of multidimensional-space point $\vec{x}$ onto vector $\vec{w}$ (thus, we only consider its direction). In other words, the observation belongs to y if corresponding $\vec{x}$ is located on a certain side of a hyperplane perpendicular to $\vec{w}$. The location of the plane is defined by the threshold c.

37.2 Canonical discriminant analysis for *k* classes

Canonical discriminant analysis (CDA) finds axes (k - 1 canonical coordinates, k being the number of classes) that best separate the categories. These linear functions are uncorrelated and define, in effect, an optimal $k - 1$ space through the n-dimensional cloud of data that best separates (the projections in that space of) the k groups. See "Multiclass LDA" for details below.

37.3 Fisher's linear discriminant

The terms *Fisher's linear discriminant* and *LDA* are often used interchangeably, although Fisher's original article[1] actually describes a slightly different discriminant, which does not make some of the assumptions of LDA such as normally distributed classes or equal class covariances.

Suppose two classes of observations have means $\vec{\mu}_0, \vec{\mu}_1$ and covariances Σ_0, Σ_1 . Then the linear combination of features $\vec{w} \cdot \vec{x}$ will have means $\vec{w} \cdot \vec{\mu}_i$ and variances $\vec{w}^T \Sigma_i \vec{w}$ for $i = 0, 1$. Fisher defined the separation between these two distributions to be the ratio of the variance between the classes to the variance within the classes:

$$S = \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{within}}} = \frac{(\vec{w} \cdot \vec{\mu}_1 - \vec{w} \cdot \vec{\mu}_0)^2}{\vec{w}^T \Sigma_1 \vec{w} + \vec{w}^T \Sigma_0 \vec{w}} = \frac{(\vec{w} \cdot (\vec{\mu}_1 - \vec{\mu}_0))^2}{\vec{w}^T (\Sigma_0 + \Sigma_1) \vec{w}}$$

This measure is, in some sense, a measure of the signal-to-noise ratio for the class labelling. It can be shown that the maximum separation occurs when

$$\vec{w} \propto (\Sigma_0 + \Sigma_1)^{-1} (\vec{\mu}_1 - \vec{\mu}_0)$$

When the assumptions of LDA are satisfied, the above equation is equivalent to LDA.

Be sure to note that the vector $\vec{w}$ is the normal to the discriminant hyperplane. As an example, in a two dimensional problem, the line that best divides the two groups is perpendicular to $\vec{w}$.

Generally, the data points to be discriminated are projected onto $\vec{w}$; then the threshold that best separates the data is chosen from analysis of the one-dimensional distribution. There is no general rule for the threshold. However, if projections of points from both classes exhibit approximately the same distributions, a good choice would be the hyperplane between projections of the two means, $\vec{w} \cdot \vec{\mu}_0$ and $\vec{w} \cdot \vec{\mu}_1$. In this case the parameter c in threshold condition $\vec{w} \cdot \vec{x} > c$ can be found explicitly:

$$c = \vec{w} \cdot \frac{1}{2}(\vec{\mu}_0 + \vec{\mu}_1) = \frac{1}{2}\vec{\mu}_1^t \Sigma^{-1} \vec{\mu}_1 - \frac{1}{2}\vec{\mu}_0^t \Sigma^{-1} \vec{\mu}_0$$

Otsu's Method is related to Fisher's linear discriminant, and was created to binarize the histogram of pixels in a grayscale image by optimally picking the black/white threshold that minimizes intra-class variance and maximizes inter-class variance within/between grayscales assigned to black and white pixel classes.

37.4 Multiclass LDA

In the case where there are more than two classes, the analysis used in the derivation of the Fisher discriminant can be extended to find a subspace which appears to contain all of the class variability. This generalization is due to C.R. Rao.[8] Suppose that each of C classes has a mean μ_i and the same covariance Σ . Then the scatter between class variability may be defined by the sample covariance of the class means

$$\Sigma_b = \frac{1}{C} \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T$$

where μ is the mean of the class means. The class separation in a direction $\vec{w}$ in this case will be given by

$$S = \frac{\vec{w}^T \Sigma_b \vec{w}}{\vec{w}^T \Sigma \vec{w}}$$

This means that when $\vec{w}$ is an eigenvector of $\Sigma^{-1}\Sigma_b$ the separation will be equal to the corresponding eigenvalue.

If $\Sigma^{-1}\Sigma_b$ is diagonalizable, the variability between features will be contained in the subspace spanned by the eigenvectors corresponding to the $C - 1$ largest eigenvalues (since Σ_b is of rank $C - 1$ at most). These eigenvectors are primarily used in feature reduction, as in PCA. The eigenvectors corresponding to the smaller eigenvalues will tend to be very sensitive to the exact choice of training data, and it is often necessary to use regularisation as described in the next section.

If classification is required, instead of dimension reduction, there are a number of alternative techniques available. For instance, the classes may be partitioned, and a standard Fisher discriminant or LDA used to classify each partition. A common example of this is “one against the rest” where the points from one class are put in one group, and everything else in the other, and then LDA applied. This will result in C classifiers, whose results are combined. Another common method is pairwise classification, where a new classifier is created for each pair of classes (giving $C(C-1)/2$ classifiers in total), with the individual classifiers combined to produce a final classification.

37.5 Practical use

In practice, the class means and covariances are not known. They can, however, be estimated from the training set. Either the maximum likelihood estimate or the maximum a posteriori estimate may be used in place of the exact value in the above equations. Although the estimates of the covariance may be considered optimal in some sense, this does not mean that the resulting discriminant obtained by substituting these values is optimal in any sense, even if the assumption of normally distributed classes is correct.

Another complication in applying LDA and Fisher’s discriminant to real data occurs when the number of measurements of each sample exceeds the number of samples in each class.[4] In this case, the covariance estimates do not have full rank, and so cannot be inverted. There are a number of ways to deal with this. One is to use a pseudo inverse instead of the usual matrix inverse in the above formulae. However, better numeric stability may be achieved by first projecting the problem onto the subspace spanned by Σ_b .[9] Another strategy to deal with small sample size is to use a shrinkage estimator of the covariance matrix, which can be expressed mathematically as

$$\Sigma = (1-\lambda)\Sigma + \lambda I$$

where I is the identity matrix, and λ is the *shrinkage intensity* or *regularisation parameter*. This leads to the framework of regularized discriminant analysis[10] or shrinkage discriminant analysis.[11]

Also, in many practical cases linear discriminants are not suitable. LDA and Fisher’s discriminant can be extended for use in non-linear classification via the kernel trick. Here, the original observations are effectively mapped into a higher dimensional non-linear space. Linear classification in this non-linear space is then equivalent to non-linear classification in the original space. The most commonly used example of this is the kernel Fisher discriminant.

LDA can be generalized to multiple discriminant analysis, where *c* becomes a categorical variable with *N* possible states, instead of only two. Analogously, if the class-conditional densities $p(\vec{x}|c=i)$ are normal with shared covariances, the sufficient statistic for $P(c|\vec{x})$ are the values of N projections, which are the subspace spanned by the N means, affine projected by the inverse covariance matrix. These projections can be found by solving a generalized eigenvalue problem, where the numerator is the covariance matrix formed by treating the means as the samples, and the denominator is the shared covariance matrix.

37.6 Applications

In addition to the examples given below, LDA is applied in positioning and product management.

37.6.1 Bankruptcy prediction

In bankruptcy prediction based on accounting ratios and other financial variables, linear discriminant analysis was the first statistical method applied to systematically explain which firms entered bankruptcy vs. survived. Despite limitations including known nonconformance of accounting ratios to the normal distribution assumptions of LDA, Edward Altman's 1968 model is still a leading model in practical applications.

37.6.2 Face recognition

In computerised face recognition, each face is represented by a large number of pixel values. Linear discriminant analysis is primarily used here to reduce the number of features to a more manageable number before classification. Each of the new dimensions is a linear combination of pixel values, which form a template. The linear combinations obtained using Fisher’s linear discriminant are called *Fisher faces*, while those obtained using the related principal component analysis are called *eigenfaces*.

37.6.3 Marketing

In marketing, discriminant analysis was once often used to determine the factors which distinguish different types of customers and/or products on the basis of surveys or other forms of collected data. Logistic regression or other methods are now more commonly used. The use of discriminant analysis in marketing can be described by the following steps:

1. Formulate the problem and gather data — Identify the salient attributes consumers use to evaluate products in this category — Use quantitative marketing research techniques (such as surveys) to collect

data from a sample of potential customers concerning their ratings of all the product attributes. The data collection stage is usually done by marketing research professionals. Survey questions ask the respondent to rate a product from one to five (or 1 to 7, or 1 to 10) on a range of attributes chosen by the researcher. Anywhere from five to twenty attributes are chosen. They could include things like: ease of use, weight, accuracy, durability, colourfulness, price, or size. The attributes chosen will vary depending on the product being studied. The same question is asked about all the products in the study. The data for multiple products is codified and input into a statistical program such as R, SPSS or SAS. (This step is the same as in Factor analysis).

2. Estimate the Discriminant Function Coefficients and determine the statistical significance and validity — Choose the appropriate discriminant analysis method. The direct method involves estimating the discriminant function so that all the predictors are assessed simultaneously. The stepwise method enters the predictors sequentially. The two-group method should be used when the dependent variable has two categories or states. The multiple discriminant method is used when the dependent variable has three or more categorical states. Use Wilks's Lambda to test for significance in SPSS or F stat in SAS. The most common method used to test validity is to split the sample into an estimation or analysis sample, and a validation or holdout sample. The estimation sample is used in constructing the discriminant function. The validation sample is used to construct a classification matrix which contains the number of correctly classified and incorrectly classified cases. The percentage of correctly classified cases is called the hit ratio.

3. Plot the results on a two dimensional map, define the dimensions, and interpret the results. The statistical program (or a related module) will map the results. The map will plot each product (usually in two-dimensional space). The distance of products to each other indicate either how different they are. The dimensions must be labelled by the researcher. This requires subjective judgement and is often very challenging. See perceptual mapping.

37.6.4 Biomedical studies

The main application of discriminant analysis in medicine is the assessment of severity state of a patient and prognosis of disease outcome. For example, during retrospective analysis, patients are divided into groups according to severity of disease – mild, moderate and severe form. Then results of clinical and laboratory analyses are studied in order to reveal variables which are statistically different in studied groups. Using these variables, discriminant functions are built which help to objectively classify disease in a future patient into mild, moderate or severe form.

In biology, similar principles are used in order to classify and define groups of different biological objects, for example, to define phage types of Salmonella enteritidis based on Fourier transform infrared spectra,[12] to detect animal source of Escherichia coli studying its virulence factors[13] etc.

37.6.5 Earth Science

This method can be used to separate the alteration zones. For example, when different data from various zones are available, discriminate analysis can find the pattern within the data and classify the them effectively [14]

37.7 See also

- Data mining
- Decision tree learning
- Factor analysis
- Kernel Fisher discriminant analysis
- Logit (for logistic regression)
- Multidimensional scaling
- Multilinear subspace learning
- Pattern recognition
- Perceptron
- Preference regression
- Quadratic classifier

37.8 References

[1] Fisher, R. A. (1936). "The Use of Multiple Measurements in Taxonomic Problems". *Annals of Eugenics* **7** (2): 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x. hdl:2440/15227.

[2] McLachlan, G. J. (2004). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Interscience. ISBN 0-471-69115-1. MR 1190469.

[3] Analyzing Quantitative Data: An Introduction for Social Researchers, Debra Wetcher-Hendricks, p.288

[4] Martinez, A. M.; Kak, A. C. (2001). "PCA versus LDA" (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23** (=2): 228–233. doi:10.1109/34.908974.

[5] Abdi, H. (2007) "Discriminant correspondence analysis." In: N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistic*. Thousand Oaks (CA): Sage. pp. 270–275.

[6] Perriere, G.; & Thioulouse, J. (2003). "Use of Correspondence Discriminant Analysis to predict the subcellular location of bacterial proteins", *Computer Methods and Programs in Biomedicine*, 70, 99–105.

[7] Venables, W. N.; Ripley, B. D. (2002). *Modern Applied Statistics with S* (4th ed.). Springer Verlag. ISBN 0-387-95457-0.

[8] Rao, R. C. (1948). "The utilization of multiple measurements in problems of biological classification". *Journal of the Royal Statistical Society, Series B* **10** (2): 159–203.

[9] Yu, H.; Yang, J. (2001). "A direct LDA algorithm for high-dimensional data — with application to face recognition", *Pattern Recognition*, 34 (10), 2067–2069

[10] Friedman, J. H. (1989). "Regularized Discriminant Analysis" (PDF). *Journal of the American Statistical Association* (American Statistical Association) **84** (405): 165–175. doi:10.2307/2289860. JSTOR 2289860. MR 0999675.

[11] Ahdesmäki, M.; Strimmer K. (2010) "Feature selection in omics prediction problems using cat scores and false nondiscovery rate control", *Annals of Applied Statistics*, 4 (1), 503–519.

[12] Preisner O, Guiomar R, Machado J, Menezes JC, Lopes JA. Application of Fourier transform infrared spectroscopy and chemometrics for differentiation of Salmonella enterica serovar Enteritidis phage types. Appl Environ Microbiol. 2010;76(11):3538–3544.

[13] David DE, Lynne AM, Han J, Foley SL. Evaluation of virulence factor profiling in the characterization of veterinary Escherichia coli isolates. Appl Environ Microbiol. 2010;76(22):7509–7513.

[14] Tahmasebi, P., Hezarkhani, A., & Mortazavi, M. (2010). Application of discriminant analysis for alteration separation; sungun copper deposit, East Azerbaijan, Iran. Australian Journal of Basic and Applied Sciences, 6(4), 564-576.

37.9 Further reading

- Duda, R. O.; Hart, P. E.; Stork, D. H. (2000). *Pattern Classification* (2nd ed.). Wiley Interscience. ISBN 0-471-05669-3. MR 1802993.
- Hilbe, J. M. (2009). *Logistic Regression Models*. Chapman & Hall/CRC Press. ISBN 978-1-4200-7575-5.
- Mika, S. et al. (1999). "Fisher Discriminant Analysis with Kernels". *IEEE Conference on Neural Networks for Signal Processing IX*: 41–48. doi:10.1109/NNSP.1999.788121.

37.10 External links

- ALGLIB contains open-source LDA implementation in C# / C++ / Pascal / VBA.
- Psychometrica.de open-source LDA implementation in Java
- LDA tutorial using MS Excel
- Biomedical statistics. Discriminant analysis

Chapter 38

Naive Bayes classifier

In machine learning, **naive Bayes classifiers** are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s,[1]:488 and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate preprocessing, it is competitive in this domain with more advanced methods including support vector machines.[2] It also finds application in automatic medical diagnosis.[3]

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression,[1]:718 which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, Naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**.[4] All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method;[4] Russell and Norvig note that "[naive Bayes] is sometimes called a **Bayesian classifier**, a somewhat careless usage that has prompted true Bayesians to call it the **idiot Bayes** model."[1]:482

38.1 Introduction

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 3 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.[5] Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.[6]

An advantage of naive Bayes is that it only requires a small amount of training data to estimate the parameters necessary for classification.

38.2 Probabilistic model

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities

$$p(C_k|x_1, \dots, x_n)$$

for each of k possible outcomes or *classes*.[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(C_k)\ p(x_1, \dots, x_n|C_k) \\ &= p(C_k)\ p(x_1|C_k)\ p(x_2, \dots, x_n|C_k, x_1) \\ &= p(C_k)\ p(x_1|C_k)\ p(x_2|C_k, x_1)\ p(x_3, \dots, x_n|C_k, x_1, x_2) \\ &= p(C_k)\ p(x_1|C_k)\ p(x_2|C_k, x_1)\ \dots p(x_n|C_k, x_1, x_2, x_3, \dots, x_{n-1}) \end{aligned}$$

Now the “naive” conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$, given the category C . This means that

$$p(x_i|C_k, x_j) = p(x_i|C_k)$$

$$p(x_i|C_k, x_j, x_k) = p(x_i|C_k)$$

$$p(x_i|C_k, x_j, x_k, x_l) = p(x_i|C_k)$$

and so on, for $i \neq j, k, l$. Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k|x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k)\ p(x_1|C_k)\ p(x_2|C_k)\ p(x_3|C_k)\ \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k)\,. \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k|x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

where the evidence $Z = p(\mathbf{x})$ is a scaling factor dependent only on $x_1, \dots, x_n$, that is, a constant if the values of the feature variables are known.

38.2.1 Constructing a classifier from the probability model

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

38.3 Parameter estimation and event models

A class' prior may be calculated by assuming equiprobable classes (i.e., priors = 1 / (number of classes)), or by calculating an estimate for the class probability from the training set (i.e., (prior for a given class) = (number of samples in the class) / (total number of samples)). To estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set.[8]

The assumptions on distributions of features are called the *event model* of the Naive Bayes classifier. For discrete features like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular. These assumptions lead to two distinct models, which are often confused.[9][10]

38.3.1 Gaussian naive Bayes

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contain a continuous attribute, x . We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_c be the mean of the values in x associated with class c, and let σ_c^2 be the variance of the values in x associated with class c. Then, the probability *distribution* of some value given a class, $p(x = v|c)$, can be computed by plugging v into the equation for a Normal distribution parameterized by μ_c and σ_c^2 . That is,

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features; some literature in fact suggests that this is necessary to apply naive Bayes, but it is not, and the discretization may throw away discriminative information.[4]

38.3.2 Multinomial naive Bayes

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1, \dots, p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector $\mathbf{x} = (x_1, \dots, x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram $\mathbf{x}$ is given by

$$p(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:[2]

$$\begin{aligned} \log p(C_k|\mathbf{x}) &\propto \log \left(p(C_k) \prod_{i=1}^{n} {p_{ki}}^{x_i} \right) \\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki} \\ &= b + \mathbf{w}_k^\top \mathbf{x} \end{aligned}$$

where $b = \log p(C_k)$ and $w_{ki} = \log p_{ki}$.

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case.

Rennie *et al.* discuss problems with the multinomial assumption in the context of document classification and possible ways to alleviate those problems, including the use of tf–idf weights instead of raw term frequencies and document length normalization, to produce a naive Bayes classifier that is competitive with support vector machines.[2]

38.3.3 Bernoulli naive Bayes

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks,[9] where binary term occurrence features are used rather than term frequencies. If x_i is a boolean expressing the occurrence or absence of the i'th term from the vocabulary, then the likelihood of a document given a class C_k is given by[9]

$$p(\mathbf{x}|C_k) = \prod_{i=1}^{n} p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

where p_{ki} is the probability of class C_k generating the term w_i . This event model is especially popular for classifying short texts. It has the benefit of explicitly modelling the absence of terms. Note that a naive Bayes classifier with a Bernoulli event model is not the same as a multinomial NB classifier with frequency counts truncated to one.

38.3.4 Semi-supervised parameter estimation

Given a way to train a naive Bayes classifier from labeled data, it's possible to construct a semi-supervised training algorithm that can learn from a combination of labeled and unlabeled data by running the supervised learning algorithm in a loop:[11]

```
Given a collection $D = L \uplus U$ of labeled samples L and unlabeled samples U, start by training a naive Bayes classifier on L.
Until convergence, do:
    Predict class probabilities $P(C|x)$ for all examples x in $D$ .
    Re-train the model based on the *probabilities* (not the labels) predicted in the previous step.
```

Convergence is determined based on improvement to the model likelihood $P(D|\theta)$, where θ denotes the parameters of the naive Bayes model.

This training algorithm is an instance of the more general expectation–maximization algorithm (EM): the prediction step inside the loop is the *E*-step of EM, while the re-training of naive Bayes is the M-step. The algorithm is formally justified by the assumption that the data are generated by a mixture model, and the components of this mixture model are exactly the classes of the classification problem.[11]

38.4 Discussion

Despite the fact that the far-reaching independence assumptions are often inaccurate, the naive Bayes classifier has several properties that make it surprisingly useful in practice. In particular, the decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This helps alleviate problems stemming from the curse of dimensionality, such as the need for data sets that scale exponentially with the number of features. While naive Bayes often fails to produce a good estimate for the correct class probabilities,[12] this may not be a requirement for many applications. For example, the naive Bayes classifier will make the correct MAP decision rule classification so long as the correct class is more probable than any other class. This is true regardless of whether the probability estimate is slightly, or even grossly inaccurate. In this manner, the overall classifier can be robust enough to ignore serious deficiencies in its underlying naive probability model.[3] Other reasons for the observed success of the naive Bayes classifier are discussed in the literature cited below.

38.4.1 Relation to logistic regression

In the case of discrete inputs (indicator or frequency features for discrete events), naive Bayes classifiers form a *generative-discriminative* pair with (multinomial) logistic regression classifiers: each naive Bayes classifier can be considered a way of fitting a probability model that optimizes the joint likelihood $p(C, \mathbf{x})$, while logistic regression fits the same probability model to optimize the conditional $p(C|\mathbf{x})$.[13]

The link between the two can be seen by observing that the decision function for naive Bayes (in the binary case) can be rewritten as "predict class C_1 if the odds of $p(C_1|\mathbf{x})$ exceed those of $p(C_2|\mathbf{x})$ ". Expressing this in log-space gives:

$$\log\frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} = \log p(C_1|\mathbf{x}) - \log p(C_2|\mathbf{x}) > 0$$

The left-hand side of this equation is the log-odds, or *logit*, the quantity predicted by the linear model that underlies logistic regression. Since naive Bayes is also a linear model for the two "discrete" event models, it can be reparametrised as a linear function $b + \mathbf{w}^\top x > 0$. Obtaining the probabilities is then a matter of applying the logistic function to $b + \mathbf{w}^\top x$, or in the multiclass case, the softmax function.

Discriminative classifiers have lower asymptotic error than generative ones; however, research by Ng and Jordan has shown that in some practical cases naive Bayes can outperform logistic regression because it reaches its asymptotic error faster.[13]

38.5 Examples

38.5.1 Sex classification

Problem: classify whether a given person is a male or a female based on the measured features. The features include height, weight, and foot size.

Training

Example training set below.

The classifier created from the training set using a Gaussian distribution assumption would be (given variances are *unbiased* sample variances):

Let's say we have equiprobable classes so P(male)= P(female) = 0.5. This prior probability distribution might be based on our knowledge of frequencies in the larger population, or on frequency in the training set.

Testing

Below is a sample to be classified as a male or female.

We wish to determine which posterior is greater, male or female. For the classification as male the posterior is given by

$$posterior(male) = \frac{P(male)\, p(height|male)\, p(weight|male)\, p(foots}{evidence}$$

For the classification as female the posterior is given by

$$posterior(female) = \frac{P(female)\, p(height|female)\, p(weight|femal}{evidence}$$

The evidence (also termed normalizing constant) may be calculated:

$$evidence = P(male)\, p(height|male)\, p(weight|male)\, p(footsize|mal$$

$$+P(female)\, p(height|female)\, p(weight|female)\, p(footsize|femal$$

However, given the sample the evidence is a constant and thus scales both posteriors equally. It therefore does not affect classification and can be ignored. We now determine the probability distribution for the sex of the sample.

$$P(male) = 0.5$$

$$p(\text{height}|\text{male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6-\mu)^2}{2\sigma^2}\right) \approx 1.5789$$

where $\mu = 5.855$ and $\sigma^2 = 3.5033 \cdot 10^{-2}$ are the parameters of normal distribution which have been previously determined from the training set. Note that a value greater than 1 is OK here – it is a probability density rather than a probability, because height is a continuous variable.

$$p(\text{weight}|\text{male}) = 5.9881 \cdot 10^{-6}$$

$$p(\text{foot size}|\text{male}) = 1.3112 \cdot 10^{-3}$$

$$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$$

$$P(\text{female}) = 0.5$$

$$p(\text{height}|\text{female}) = 2.2346 \cdot 10^{-1}$$

$$p(\text{weight}|\text{female}) = 1.6789 \cdot 10^{-2}$$

$$p(\text{foot size}|\text{female}) = 2.8669 \cdot 10^{-1}$$

$$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$$

Since posterior numerator is greater in the female case, we predict the sample is female.

38.5.2 Document classification

Here is a worked example of naive Bayesian classification to the document classification problem. Consider the problem of classifying documents by their content, for example into spam and non-spam e-mails. Imagine that documents are drawn from a number of classes of documents which can be modelled as sets of words where the (independent) probability that the i-th word of a given document occurs in a document from class *C* can be written as

$$p(w_i|C)$$

(For this treatment, we simplify things further by assuming that words are randomly distributed in the document - that is, words are not dependent on the length of the document, position within the document with relation to other words, or other document-context.)

Then the probability that a given document *D* contains all of the words w_i , given a class *C*, is

$$p(D|C) = \prod_i p(w_i|C)$$

The question that we desire to answer is: “what is the probability that a given document *D* belongs to a given class *C*?" In other words, what is $p(C|D)$?

Now by definition

$$p(D|C) = \frac{p(D \cap C)}{p(C)}$$

and

$$p(C|D) = \frac{p(D \cap C)}{p(D)}$$

Bayes' theorem manipulates these into a statement of probability in terms of likelihood.

$$p(C|D) = \frac{p(C)}{p(D)} p(D|C)$$

Assume for the moment that there are only two mutually exclusive classes, *S* and ¬*S* (e.g. spam and not spam), such that every element (email) is in either one or the other;

$$p(D|S) = \prod_i p(w_i|S)$$

and

$$p(D|\neg S) = \prod_i p(w_i|\neg S)$$

Using the Bayesian result above, we can write:

$$p(S|D) = \frac{p(S)}{p(D)} \prod_i p(w_i|S)$$

$$p(\neg S|D) = \frac{p(\neg S)}{p(D)} \prod_i p(w_i|\neg S)$$

Dividing one by the other gives:

$$\frac{p(S|D)}{p(\neg S|D)} = \frac{p(S) \prod_i p(w_i|S)}{p(\neg S) \prod_i p(w_i|\neg S)}$$

Which can be re-factored as:

$$\frac{p(S|D)}{p(\neg S|D)} = \frac{p(S)}{p(\neg S)} \prod_i \frac{p(w_i|S)}{p(w_i|\neg S)}$$

Thus, the probability ratio p(*S* | *D*) / p(¬*S* | *D*) can be expressed in terms of a series of likelihood ratios. The actual probability p(*S* | *D*) can be easily computed from log (p(S | D) / p(¬S | D)) based on the observation that p(S | D) + p(¬S | D) = 1.

Taking the logarithm of all these ratios, we have:

$$\ln \frac{p(S|D)}{p(\neg S|D)} = \ln \frac{p(S)}{p(\neg S)} + \sum_i \ln \frac{p(w_i|S)}{p(w_i|\neg S)}$$

(This technique of "log-likelihood ratios" is a common technique in statistics. In the case of two mutually exclusive alternatives (such as this example), the conversion of a log-likelihood ratio to a probability takes the form of a sigmoid curve: see logit for details.)

Finally, the document can be classified as follows. It is spam if $p(S|D) > p(\neg S|D)$ (i.e., $\ln \frac{p(S|D)}{p(\neg S|D)} > 0$), otherwise it is not spam.

38.6 See also

- AODE
- Bayesian spam filtering
- Bayesian network
- Random naive Bayes
- Linear classifier
- Logistic regression
- Perceptron
- Take-the-best heuristic

38.7 References

[1] Russell, Stuart; Norvig, Peter (2003) [1995]. *Artificial Intelligence: A Modern Approach* (2nd ed.). Prentice Hall. ISBN 978-0137903955.

[2] Rennie, J.; Shih, L.; Teevan, J.; Karger, D. (2003). *Tackling the poor assumptions of Naive Bayes classifiers* (PDF). ICML.

[3] Rish, Irina (2001). *An empirical study of the naive Bayes classifier* (PDF). IJCAI Workshop on Empirical Methods in AI.

[4] Hand, D. J.; Yu, K. (2001). "Idiot's Bayes — not so stupid after all?". *International Statistical Review* **69** (3): 385–399. doi:10.2307/1403452. ISSN 0306-7734.

[5] Zhang, Harry. *The Optimality of Naive Bayes* (PDF). FLAIRS2004 conference.

[6] Caruana, R.; Niculescu-Mizil, A. (2006). *An empirical comparison of supervised learning algorithms*. Proc. 23rd International Conference on Machine Learning. CiteSeerX: 10.1.1.122.5901.

[7] Narasimha Murty, M.; Susheela Devi, V. (2011). *Pattern Recognition: An Algorithmic Approach*. ISBN 0857294946.

[8] John, George H.; Langley, Pat (1995). *Estimating Continuous Distributions in Bayesian Classifiers*. Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann. pp. 338–345.

[9] McCallum, Andrew; Nigam, Kamal (1998). *A comparison of event models for Naive Bayes text classification* (PDF). AAAI-98 workshop on learning for text categorization **752**.

[10] Metsis, Vangelis; Androutsopoulos, Ion; Paliouras, Georgios (2006). *Spam filtering with Naive Bayes—which Naive Bayes?*. Third conference on email and anti-spam (CEAS) **17**.

[11] Nigam, Kamal; McCallum, Andrew; Thrun, Sebastian; Mitchell, Tom (2000). "Learning to classify text from labeled and unlabeled documents using EM" (PDF). *Machine Learning*.

[12] Niculescu-Mizil, Alexandru; Caruana, Rich (2005). *Predicting good probabilities with supervised learning* (PDF). ICML. doi:10.1145/1102351.1102430.

[13] Ng, Andrew Y.; Jordan, Michael I. (2002). *On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes*. NIPS **14**.

38.7.1 Further reading

- Domingos, Pedro; Pazzani, Michael (1997). "On the optimality of the simple Bayesian classifier under zero-one loss". *Machine Learning* **29**: 103–137.
- Webb, G. I.; Boughton, J.; Wang, Z. (2005). "Not So Naive Bayes: Aggregating One-Dependence Estimators". *Machine Learning* (Springer) **58** (1): 5–24. doi:10.1007/s10994-005-4258-6.
- Mozina, M.; Demsar, J.; Kattan, M.; Zupan, B. (2004). *Nomograms for Visualization of Naive Bayesian Classifier* (PDF). Proc. PKDD-2004. pp. 337–348.
- Maron, M. E. (1961). "Automatic Indexing: An Experimental Inquiry". *JACM* **8** (3): 404–417. doi:10.1145/321075.321084.
- Minsky, M. (1961). *Steps toward Artificial Intelligence*. Proc. IRE **49** (1). pp. 8–30.

38.8 External links

- Book Chapter: Naive Bayes text classification, Introduction to Information Retrieval
- Naive Bayes for Text Classification with Unbalanced Classes

- Benchmark results of Naive Bayes implementations
- Hierarchical Naive Bayes Classifiers for uncertain data (an extension of the Naive Bayes classifier).

Software

- Naive Bayes classifiers are available in many general-purpose machine learning and NLP packages, including Apache Mahout, Mallet, NLTK, Orange, scikit-learn and Weka.
- IMSL Numerical Libraries Collections of math and statistical algorithms available in C/C++, Fortran, Java and C#/.NET. Data mining routines in the IMSL Libraries include a Naive Bayes classifier.
- Winnow content recommendation Open source Naive Bayes text classifier works with very small training and unbalanced training sets. High performance, C, any Unix.
- An interactive Microsoft Excel spreadsheet Naive Bayes implementation using VBA (requires enabled macros) with viewable source code.
- jBNC - Bayesian Network Classifier Toolbox
- Statistical Pattern Recognition Toolbox for Matlab.
- ifile - the first freely available (Naive) Bayesian mail/spam filter
- NClassifier - NClassifier is a .NET library that supports text classification and text summarization. It is a port of Classifier4J.
- Classifier4J - Classifier4J is a Java library designed to do text classification. It comes with an implementation of a Bayesian classifier.

Chapter 39

Cross-validation (statistics)

Cross-validation, sometimes called **rotation estimation**,[1][2][3] is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of *known data* on which training is run (*training dataset*), and a dataset of *unknown data* (or *first seen* data) against which the model is tested (*testing dataset*).[4] The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the *validation dataset*), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem), etc.

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the *training set*), and validating the analysis on the other subset (called the *validation set* or *testing set*). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

Cross-validation is important in guarding against testing hypotheses suggested by the data (called "Type III errors"[5]), especially where further samples are hazardous, costly or impossible to collect.

Furthermore, one of the main reasons for using cross-validation instead of using the conventional validation (e.g. partitioning the data set into two sets of 70% for training and 30% for test) is that the error (e.g. Root Mean Square Error) on the training set in the conventional validation is not a useful estimator of model performance and thus the error on the test data set does not properly represent the assessment of model performance. This may be due to the fact that there is not enough data available or there is not a good distribution and spread of data to partition it into separate training and test sets in the conventional validation method. In these cases, a fair way to properly estimate model prediction performance is to use cross-validation as a powerful general technique.[6]

In summary, cross-validation combines (averages) measures of fit (prediction error) to correct for the optimistic nature of training error and derive a more accurate estimate of model prediction performance.[6]

39.1 Purpose of cross-validation

Suppose we have a model with one or more unknown parameters, and a data set to which the model can be fit (the training data set). The fitting process optimizes the model parameters to make the model fit the training data as well as possible. If we then take an independent sample of validation data from the same population as the training data, it will generally turn out that the model does not fit the validation data as well as it fits the training data. This is called overfitting, and is particularly likely to happen when the size of the training data set is small, or when the number of parameters in the model is large. Cross-validation is a way to predict the fit of a model to a hypothetical validation set when an explicit validation set is not available.

Linear regression provides a simple illustration of overfitting. In linear regression we have real *response values* y_1, ..., yn, and n p-dimensional vector *covariates* $\boldsymbol{x}_1$, ..., $\boldsymbol{xn}$. The components of the vectors $\boldsymbol{x}_i$ are denoted x_{i1}, ..., x_{ip}. If we use least squares to fit a function in the form of a hyperplane $y = a + \boldsymbol{\beta}^T\boldsymbol{x}$ to the data $(\boldsymbol{x}_i, y_i)_{1 \le i \le n}$, we could then assess the fit using the mean squared error (MSE). The MSE for a given value of the parameters a and $\boldsymbol{\beta}$ on the training set $(\boldsymbol{x}_i, y_i)_{1 \le i \le n}$ is

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - a - \boldsymbol{\beta}^T\mathbf{x}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - a - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2$$

It can be shown under mild assumptions that the expected value of the MSE for the training set is $(n - p - 1)/(n + p + 1) < 1$ times the expected value of the MSE for the validation set (the expected value is taken over the distribution of training sets). Thus if we fit the model and compute the MSE on the training set, we will get an optimistically biased assessment of how well the model will fit an independent data set. This biased estimate is called the *in-sample* estimate of the fit, whereas the cross-validation estimate is an *out-of-sample* estimate.

Since in linear regression it is possible to directly compute the factor $(n - p - 1)/(n + p + 1)$ by which the training MSE underestimates the validation MSE, cross-validation is not practically useful in that setting (however, cross-validation remains useful in the context of linear regression in that it can be used to select an optimally regularized cost function). In most other regression procedures (e.g. logistic regression), there is no simple formula to make such an adjustment. Cross-validation is, thus, a generally applicable way to predict the performance of a model on a validation set using computation in place of mathematical analysis.

39.2 Common types of cross-validation

Two types of cross-validation can be distinguished, exhaustive and non-exhaustive cross-validation.

39.2.1 Exhaustive cross-validation

Exhaustive cross-validation methods are cross-validation methods which learn and test on all possible ways to divide the original sample into a training and a validation set.

Leave-p-out cross-validation

Leave-*p*-out cross-validation (**LpO CV**) involves using p observations as the validation set and the remaining observations as the training set. This is repeated on all ways to cut the original sample on a validation set of p observations and a training set.

LpO cross-validation requires to learn and validate C_p^n times (where n is the number of observations in the original sample). So as soon as n is quite big it becomes impossible to calculate. (See Binomial coefficient)

Leave-one-out cross-validation

Leave-*one*-out cross-validation (**LOOCV**) is a particular case of leave-*p*-out cross-validation with $p = 1$.

LOO cross-validation doesn't have the calculation problem of general LpO cross-validation because $C_1^n = n$.

39.2.2 Non-exhaustive cross-validation

Non-exhaustive cross validation methods do not compute all ways of splitting the original sample. Those methods are approximations of leave-*p*-out cross-validation.

k-fold cross-validation

In *k*-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data. The cross-validation process is then repeated k times (the *folds*), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random subsampling (see below) is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross-validation is commonly used,[7] but in general k remains an unfixed parameter.

When k=n (the number of observations), the k-fold cross-validation is exactly the leave-one-out cross-validation.

In *stratified* k-fold cross-validation, the folds are selected so that the mean response value is approximately equal in all the folds. In the case of a dichotomous classification, this means that each fold contains roughly the same proportions of the two types of class labels.

2-fold cross-validation

This is the simplest variation of k-fold cross-validation. Also called holdout method.[8] For each fold, we randomly assign data points to two sets d_0 and d_1, so that both sets are equal size (this is usually implemented by shuffling the data array and then splitting it in two). We then train on d_0 and test on d_1, followed by training on d_1 and testing on d_0.

This has the advantage that our training and test sets are both large, and each data point is used for both training and validation on each fold.

Repeated random sub-sampling validation

This method randomly splits the dataset into training and validation data. For each such split, the model is fit to the training data, and predictive accuracy is assessed using the validation data. The results are then averaged over the splits. The advantage of this method (over k-fold cross validation) is that the proportion of the training/validation split is not dependent on the number of iterations (folds). The disadvantage of this method is that some observations may never be selected in the validation subsample, whereas others may be selected more than once. In other words, validation subsets may overlap. This method also exhibits Monte Carlo variation, meaning that the results will vary if the analysis is repeated with different random splits.

When the number of random splits goes to infinity, the

Repeated random sub-sampling validation become arbitrary close to the leave-p-out cross-validation.

In a stratified variant of this approach, the random samples are generated in such a way that the mean response value (i.e. the dependent variable in the regression) is equal in the training and testing sets. This is particularly useful if the responses are dichotomous with an unbalanced representation of the two response values in the data.

39.3 Measures of fit

The goal of cross-validation is to estimate the expected level of fit of a model to a data set that is independent of the data that were used to train the model. It can be used to estimate any quantitative measure of fit that is appropriate for the data and model. For example, for binary classification problems, each case in the validation set is either predicted correctly or incorrectly. In this situation the misclassification error rate can be used to summarize the fit, although other measures like positive predictive value could also be used. When the value being predicted is continuously distributed, the mean squared error, root mean squared error or median absolute deviation could be used to summarize the errors.

39.4 Applications

Cross-validation can be used to compare the performances of different predictive modeling procedures. For example, suppose we are interested in optical character recognition, and we are considering using either support vector machines (SVM) or k nearest neighbors (KNN) to predict the true character from an image of a handwritten character. Using cross-validation, we could objectively compare these two methods in terms of their respective fractions of misclassified characters. If we simply compared the methods based on their in-sample error rates, the KNN method would likely appear to perform better, since it is more flexible and hence more prone to overfitting compared to the SVM method.

Cross-validation can also be used in *variable selection*.[9] Suppose we are using the expression levels of 20 proteins to predict whether a cancer patient will respond to a drug. A practical goal would be to determine which subset of the 20 features should be used to produce the best predictive model. For most modeling procedures, if we compare feature subsets using the in-sample error rates, the best performance will occur when all 20 features are used. However under cross-validation, the model with the best fit will generally include only a subset of the features that are deemed truly informative.

39.5 Statistical properties

Suppose we choose a measure of fit F, and use cross-validation to produce an estimate F^* of the expected fit EF of a model to an independent data set drawn from the same population as the training data. If we imagine sampling multiple independent training sets following the same distribution, the resulting values for F^* will vary. The statistical properties of F^* result from this variation.

The cross-validation estimator F^* is very nearly unbiased for EF. The reason that it is slightly biased is that the training set in cross-validation is slightly smaller than the actual data set (e.g. for LOOCV the training set size is $n-1$ when there are n observed cases). In nearly all situations, the effect of this bias will be conservative in that the estimated fit will be slightly biased in the direction suggesting a poorer fit. In practice, this bias is rarely a concern.

The variance of F^* can be large.[10][11] For this reason, if two statistical procedures are compared based on the results of cross-validation, it is important to note that the procedure with the better estimated performance may not actually be the better of the two procedures (i.e. it may not have the better value of EF). Some progress has been made on constructing confidence intervals around cross-validation estimates,[10] but this is considered a difficult problem.

39.6 Computational issues

Most forms of cross-validation are straightforward to implement as long as an implementation of the prediction method being studied is available. In particular, the prediction method need only be available as a "black box" – there is no need to have access to the internals of its implementation. If the prediction method is expensive to train, cross-validation can be very slow since the training must be carried out repeatedly. In some cases such as least squares and kernel regression, cross-validation can be sped up significantly by pre-computing certain values that are needed repeatedly in the training, or by using fast "updating rules" such as the Sherman–Morrison formula. However one must be careful to preserve the "total blinding" of the validation set from the training procedure, otherwise bias may result. An extreme example of accelerating cross-validation occurs in linear regression, where the results of cross-validation have a closed-form expression known as the *prediction residual error sum of squares* (PRESS).

39.7 Relationship to other forms of validation

In "true validation," or "holdout validation," a subset of observations is chosen randomly from the initial sample to form a validation or testing set, and the remaining observations are retained as the training data. Normally, less than a third of the initial sample is used for validation data.[12]

39.8 Limitations and misuse

Cross-validation only yields meaningful results if the validation set and training set are drawn from the same population and only if human biases are controlled.

In many applications of predictive modeling, the structure of the system being studied evolves over time. Both of these can introduce systematic differences between the training and validation sets. For example, if a model for predicting stock values is trained on data for a certain five-year period, it is unrealistic to treat the subsequent five-year period as a draw from the same population. As another example, suppose a model is developed to predict an individual's risk for being diagnosed with a particular disease within the next year. If the model is trained using data from a study involving only a specific population group (e.g. young people or males), but is then applied to the general population, the cross-validation results from the training set could differ greatly from the actual predictive performance.

In many applications, models also may be incorrectly specified and vary as a function of modeler biases and/or arbitrary choices. When this occurs, there may be an illusion that the system changes in external samples, whereas the reason is that the model has missed a critical predictor and/or included a confounded predictor. New evidence is that cross-validation by itself is not very predictive of external validity, whereas a form of experimental validation known as swap sampling that does control for human bias can be much more predictive of external validity.[13] As defined by this large MAQC-II study across 30,000 models, swap sampling incorporates cross-validation in the sense that predictions are tested across independent training and validation samples. Yet, models are also developed across these independent samples and by modelers who are blinded to one another. When there is a mismatch in these models developed across these swapped training and validation samples as happens quite frequently, MAQC-II shows that this will be much more predictive of poor external predictive validity than traditional cross-validation.

The reason for the success of the swapped sampling is a built-in control for human biases in model building. In addition to placing too much faith in predictions that may vary across modelers and lead to poor external validity due to these confounding modeler effects, these are some other ways that cross-validation can be misused:

- By performing an initial analysis to identify the most informative features using the entire data set – if feature selection or model tuning is required by the modeling procedure, this must be repeated on every training set. Otherwise, predictions will certainly be upwardly biased.[14] If cross-validation is used to decide which features to use, an *inner cross-validation* to carry out the feature selection on every training set must be performed.[15]
- By allowing some of the training data to also be included in the test set – this can happen due to "twinning" in the data set, whereby some exactly identical or nearly identical samples are present in the data set. Note that to some extent twinning always takes place even in perfectly independent training and validation samples. This is because some of the training sample observations will have nearly identical values of predictors as validation sample observations. And some of these will correlate with a target at better than chance levels in the same direction in both training and validation when they are actually driven by confounded predictors with poor external validity. If such a cross-validated model is selected from a k-fold set, human confirmation bias will be at work and determine that such a model has been validated. This is why traditional cross-validation needs to be supplemented with controls for human bias and confounded model specification like swap sampling and prospective studies.

It should be noted that some statisticians have questioned the usefulness of validation samples.[16]

39.9 See also

- Boosting (machine learning)
- Bootstrap aggregating (bagging)
- Bootstrapping (statistics)
- Resampling (statistics)

39.10 Notes and references

[1] Geisser, Seymour (1993). *Predictive Inference*. New York, NY: Chapman and Hall. ISBN 0-412-03471-9.

[2] Kohavi, Ron (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection". *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (San Mateo, CA: Morgan Kaufmann) **2** (12): 1137–1143. CiteSeerX: 10.1.1.48.529.

[3] Devijver, Pierre A.; Kittler, Josef (1982). *Pattern Recognition: A Statistical Approach*. London, GB: Prentice-Hall.

[4] “Newbie question: Confused about train, validation and test data!". Retrieved 2013-11-14.

[5] Mosteller, Frederick (1948). “A k-sample slippage test for an extreme population”. *Annals of Mathematical Statistics* **19** (1): 58–65. doi:10.1214/aoms/1177730290. JSTOR 2236056. MR 0024116.

[6] Grossman,, Robert; Seni, Giovanni; Elder, John; Agarwal, Nitin; Liu, Huan (2010). *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*. Morgan & Claypool. doi:10.2200/S00240ED1V01Y200912DMK002.

[7] McLachlan, Geoffrey J.; Do, Kim-Anh; Ambroise, Christophe (2004). *Analyzing microarray gene expression data*. Wiley.

[8] “Cross Validation”. Retrieved 11 November 2012.

[9] Picard, Richard; Cook, Dennis (1984). “Cross-Validation of Regression Models”. *Journal of the American Statistical Association* **79** (387): 575–583. doi:10.2307/2288403. JSTOR 2288403.

[10] Efron, Bradley; Tibshirani, Robert (1997). “Improvements on cross-validation: The .632 + Bootstrap Method”. *Journal of the American Statistical Association* **92** (438): 548–560. doi:10.2307/2965703. JSTOR 2965703. MR 1467848.

[11] Stone, Mervyn (1977). “Asymptotics for and against cross-validation”. *Biometrika* **64** (1): 29–35. doi:10.1093/biomet/64.1.29. JSTOR 2335766. MR 0474601.

[12] “Tutorial 12”. *Decision Trees Interactive Tutorial and Resources*. Archived from the original on 2006-06-23. Retrieved 2006-06-21.

[13] Consortium, MAQC (2010). “The Microarray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models”. *Nature Biotechnology* (London: Nature Publishing Group) **28**: 827–838.

[14] Bermingham, Mairead L.; Pong-Wong, Ricardo; Spiliopoulou, Athina; Hayward, Caroline; Rudan, Igor; Campbell, Harry; Wright, Alan F.; Wilson, James F.; Agakov, Felix; Navarro, Pau; Haley, Chris S. (2015). “Application of high-dimensional feature selection: evaluation for genomic prediction in man”. *Sci. Rep.* **5**.

[15] Varma, Sudhir; Simon, Richard (2006). “Bias in error estimation when using cross-validation for model selection”. *BMC Bioinformatics* **7**: 91. doi:10.1186/1471-2105-7-91. PMC 1397873. PMID 16504092.

[16] Hirsch, Robert (1991). “Validation Samples”. *Biometrics* **47** (3): 1193–1194.

Chapter 40

Unsupervised learning

In machine learning, the problem of **unsupervised learning** is that of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning.

Unsupervised learning is closely related to the problem of density estimation in statistics.[1] However unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data. Many methods employed in unsupervised learning are based on data mining methods used to preprocess data.

Approaches to unsupervised learning include:

- clustering (e.g., k-means, mixture models, hierarchical clustering),[2]
- Approaches for learning latent variable models such as
  - Expectation–maximization algorithm (EM)
  - Method of moments
  - Blind signal separation techniques, e.g.,
    - Principal component analysis,
    - Independent component analysis,
    - Non-negative matrix factorization,
    - Singular value decomposition.[3]

Among neural network models, the self-organizing map (SOM) and adaptive resonance theory (ART) are commonly used unsupervised learning algorithms. The SOM is a topographic organization in which nearby locations in the map represent inputs with similar properties. The ART model allows the number of clusters to vary with problem size and lets the user control the degree of similarity between members of the same clusters by means of a user-defined constant called the vigilance parameter. ART networks are also used for many pattern recognition tasks, such as automatic target recognition and seismic signal processing. The first version of ART was “ART1”, developed by Carpenter and Grossberg (1988).[4]

40.1 Method of moments

One of the approaches in unsupervised learning is the method of moments. In the method of moments, the unknown parameters (of interest) in the model are related to the moments of one or more random variables, and thus, these unknown parameters can be estimated given the moments. The moments are usually estimated from samples in an empirical way. The basic moments are first and second order moments. For a random vector, the first order moment is the mean vector, and the second order moment is the covariance matrix (when the mean is zero). Higher order moments are usually represented using tensors which are the generalization of matrices to higher orders as multi-dimensional arrays.

In particular, the method of moments is shown to be effective in learning the parameters of latent variable models.[5] Latent variable models are statistical models where in addition to the observed variables, a set of latent variables also exists which is not observed. A highly practical example of latent variable models in machine learning is the topic modeling which is a statistical model for generating the words (observed variables) in the document based on the topic (latent variable) of the document. In the topic modeling, the words in the document are generated according to different statistical parameters when the topic of the document is changed. It is shown that method of moments (tensor decomposition techniques) consistently recover the parameters of a large class of latent variable models under some assumptions.[5]

Expectation–maximization algorithm (EM) is also one of the most practical methods for learning latent variable models. But, it can be stuck in local optima, and the global convergence of the algorithm to the true unknown parameters of the model is not guaranteed. While, for the method of moments, the global convergence is guaranteed under some conditions.[5]

40.2 See also

- Cluster analysis
- Expectation–maximization algorithm

- Generative topographic map
- Multilinear subspace learning
- Multivariate analysis
- Radial basis function network

40.3 Notes

[1] Jordan, Michael I.; Bishop, Christopher M. (2004). "Neural Networks". In Allen B. Tucker. *Computer Science Handbook, Second Edition (Section VII: Intelligent Systems)*. Boca Raton, FL: Chapman & Hall/CRC Press LLC. ISBN 1-58488-360-X.

[2] Hastie,Trevor,Robert Tibshirani, Friedman,Jerome (2009). *The Elements of Statistical Learning: Data mining,Inference,and Prediction*. New York: Springer. pp. 485–586. ISBN 978-0-387-84857-0.

[3] Acharyya, Ranjan (2008); *A New Approach for Blind Source Separation of Convolutive Sources*, ISBN 978-3-639-07797-1 (this book focuses on unsupervised learning with Blind Source Separation)

[4] Carpenter, G.A. and Grossberg, S. (1988). "The ART of adaptive pattern recognition by a self-organizing neural network" (PDF). *Computer* **21**: 77–88. doi:10.1109/2.33.

[5] Anandkumar, Animashree; Ge, Rong; Hsu, Daniel; Kakade, Sham; Telgarsky, Matus (2014). "Tensor Decompositions for Learning Latent Variable Models" (PDF). *Journal of Machine Learning Research (JMLR)* **15**: 2773–2832.

40.4 Further reading

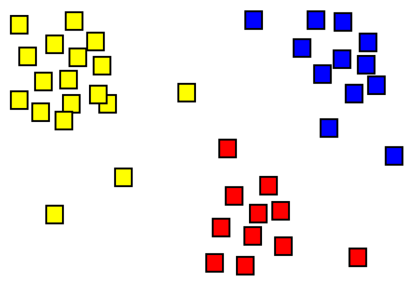
- Bousquet, O.; von Luxburg, U.; Raetsch, G., eds. (2004). *Advanced Lectures on Machine Learning*. Springer-Verlag. ISBN 978-3540231226.
- Duda, Richard O.; Hart, Peter E.; Stork, David G. (2001). "Unsupervised Learning and Clustering". *Pattern classification* (2nd ed.). Wiley. ISBN 0-471-05669-3.
- Hastie, Trevor; Tibshirani, Robert (2009). *The Elements of Statistical Learning: Data mining,Inference,and Prediction*. New York: Springer. pp. 485–586. doi:10.1007/978-0-387-84858-7_14. ISBN 978-0-387-84857-0.
- Hinton, Geoffrey; Sejnowski, Terrence J., eds. (1999). *Unsupervised Learning: Foundations of Neural Computation*. MIT Press. ISBN 0-262-58168-X. (This book focuses on unsupervised learning in neural networks)

Chapter 41

Cluster analysis

For the supervised learning approach, see Statistical classification.

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.


The result of a cluster analysis shown as the coloring of the squares into three clusters.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem. The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It will often be necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

Besides the term *clustering*, there are a number of terms with similar meanings, including *automatic classification*, *numerical taxonomy*, *botryology* (from Greek βότρυς "grape") and *typological analysis*. The subtle differences are often in the usage of the results: while in data mining, the resulting groups are the matter of interest, in automatic classification the resulting discriminative power is of interest. This often leads to misunderstandings between researchers coming from the fields of data mining and machine learning, since they use the same terms and often the same algorithms, but have different goals.

Cluster analysis was originated in anthropology by Driver and Kroeber in 1932 and introduced to psychology by Zubin in 1938 and Robert Tryon in 1939[1][2] and famously used by Cattell beginning in 1943[3] for trait theory classification in personality psychology.

41.1 Definition

According to Vladimir Estivill-Castro, the notion of a "cluster" cannot be precisely defined, which is one of the reasons why there are so many clustering algorithms.[4] There is a common denominator: a group of data objects. However, different researchers employ different cluster models, and for each of these cluster models again different algorithms can be given. The notion of a cluster, as found by different algorithms, varies significantly in its properties. Understanding these "cluster models" is key to understanding the differences between the various algorithms. Typical cluster models include:

- Connectivity models: for example hierarchical clustering builds models based on distance connectivity.
- Centroid models: for example the k-means algorithm represents each cluster by a single mean vector.
- Distribution models: clusters are modeled using statistical distributions, such as multivariate normal distributions used by the Expectation-maximization algorithm.

- Density models: for example DBSCAN and OPTICS defines clusters as connected dense regions in the data space.
- Subspace models: in Biclustering (also known as Co-clustering or two-mode-clustering), clusters are modeled with both cluster members and relevant attributes.
- Group models: some algorithms do not provide a refined model for their results and just provide the grouping information.
- Graph-based models: a clique, i.e., a subset of nodes in a graph such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster. Relaxations of the complete connectivity requirement (a fraction of the edges can be missing) are known as quasi-cliques, as in HCS clustering algorithm .

A “clustering” is essentially a set of such clusters, usually containing all objects in the data set. Additionally, it may specify the relationship of the clusters to each other, for example a hierarchy of clusters embedded in each other. Clusterings can be roughly distinguished as:

- hard clustering: each object belongs to a cluster or not
- soft clustering (also: fuzzy clustering): each object belongs to each cluster to a certain degree (e.g. a likelihood of belonging to the cluster)

There are also finer distinctions possible, for example:

- strict partitioning clustering: here each object belongs to exactly one cluster
- strict partitioning clustering with outliers: objects can also belong to no cluster, and are considered outliers.
- overlapping clustering (also: alternative clustering, multi-view clustering): while usually a hard clustering, objects may belong to more than one cluster.
- hierarchical clustering: objects that belong to a child cluster also belong to the parent cluster
- subspace clustering: while an overlapping clustering, within a uniquely defined subspace, clusters are not expected to overlap.

41.2 Algorithms

Main category: Data clustering algorithms

Clustering algorithms can be categorized based on their cluster model, as listed above. The following overview will only list the most prominent examples of clustering algorithms, as there are possibly over 100 published clustering algorithms. Not all provide models for their clusters and can thus not easily be categorized. An overview of algorithms explained in Wikipedia can be found in the list of statistics algorithms.

There is no objectively “correct” clustering algorithm, but as it was noted, “clustering is in the eye of the beholder.”[4] The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another. It should be noted that an algorithm that is designed for one kind of model has no chance on a data set that contains a radically different kind of model.[4] For example, k-means cannot find non-convex clusters.[4]

41.2.1 Connectivity based clustering (hierarchical clustering)

Main article: Hierarchical clustering

Connectivity based clustering, also known as *hierarchical clustering*, is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect “objects” to form “clusters” based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster. At different distances, different clusters will form, which can be represented using a dendrogram, which explains where the common name “hierarchical clustering” comes from: these algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances. In a dendrogram, the y-axis marks the distance at which the clusters merge, while the objects are placed along the x-axis such that the clusters don't mix.

Connectivity based clustering is a whole family of methods that differ by the way distances are computed. Apart from the usual choice of distance functions, the user also needs to decide on the linkage criterion (since a cluster consists of multiple objects, there are multiple candidates to compute the distance to) to use. Popular choices are known as single-linkage clustering (the minimum of object distances), complete linkage clustering (the maximum of object distances) or UPGMA (“Unweighted Pair Group Method with Arithmetic Mean”, also known as average linkage clustering). Furthermore, hierarchical clustering can be agglomerative (starting with single elements and aggregating them into clusters) or divisive (starting with the complete data set and dividing it into partitions).

These methods will not produce a unique partitioning of the data set, but a hierarchy from which the user still needs to choose appropriate clusters. They are not very robust towards outliers, which will either show up as additional clusters or even cause other clusters to merge (known as "chaining phenomenon", in particular with single-linkage clustering). In the general case, the complexity is $\mathcal{O}(n^3)$, which makes them too slow for large data sets. For some special cases, optimal efficient methods (of complexity $\mathcal{O}(n^2)$) are known: SLINK[5] for single-linkage and CLINK[6] for complete-linkage clustering. In the data mining community these methods are recognized as a theoretical foundation of cluster analysis, but often considered obsolete. They did however provide inspiration for many later methods such as density based clustering.

- Linkage clustering examples
- Single-linkage on Gaussian data. At 35 clusters, the biggest cluster starts fragmenting into smaller parts, while before it was still connected to the second largest due to the single-link effect.
- Single-linkage on density-based clusters. 20 clusters extracted, most of which contain single elements, since linkage clustering does not have a notion of "noise".

41.2.2 Centroid-based clustering

Main article: k-means clustering

In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

The optimization problem itself is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximative method is Lloyd's algorithm,[7] often actually referred to as "*k-means algorithm*". It does however only find a local optimum, and is commonly run multiple times with different random initializations. Variations of k-means often include such optimizations as choosing the best of multiple runs, but also restricting the centroids to members of the data set (k-medoids), choosing medians (k-medians clustering), choosing the initial centers less randomly (K-means++) or allowing a fuzzy cluster assignment (Fuzzy c-means).

Most k-means-type algorithms require the number of clusters - k - to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. This often leads to incorrectly cut borders in between of clusters (which is not surprising, as the algorithm optimized cluster centers, not cluster borders).

K-means has a number of interesting theoretical properties. On the one hand, it partitions the data space into a structure known as a Voronoi diagram. On the other hand, it is conceptually close to nearest neighbor classification, and as such is popular in machine learning. Third, it can be seen as a variation of model based classification, and Lloyd's algorithm as a variation of the Expectation-maximization algorithm for this model discussed below.

- k-Means clustering examples
- K-means separates data into Voronoi-cells, which assumes equal-sized clusters (not adequate here)
- K-means cannot represent density-based clusters

41.2.3 Distribution-based clustering

The clustering model most closely related to statistics is based on distribution models. Clusters can then easily be defined as objects belonging most likely to the same distribution. A convenient property of this approach is that this closely resembles the way artificial data sets are generated: by sampling random objects from a distribution.

While the theoretical foundation of these methods is excellent, they suffer from one key problem known as overfitting, unless constraints are put on the model complexity. A more complex model will usually be able to explain the data better, which makes choosing the appropriate model complexity inherently difficult.

One prominent method is known as Gaussian mixture models (using the expectation-maximization algorithm). Here, the data set is usually modelled with a fixed (to avoid overfitting) number of Gaussian distributions that are initialized randomly and whose parameters are iteratively optimized to fit better to the data set. This will converge to a local optimum, so multiple runs may produce different results. In order to obtain a hard clustering, objects are often then assigned to the Gaussian distribution they most likely belong to; for soft clusterings, this is not necessary.

Distribution-based clustering produces complex models for clusters that can capture correlation and dependence between attributes. However, these algorithms put an extra burden on the user: for many real data sets, there may be no concisely defined mathematical model (e.g. assuming Gaussian distributions is a rather strong assumption on the data).

- Expectation-Maximization (EM) clustering examples

- On Gaussian-distributed data, EM works well, since it uses Gaussians for modelling clusters
- Density-based clusters cannot be modeled using Gaussian distributions

41.2.4 Density-based clustering

In density-based clustering,[8] clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points.

The most popular[9] density based clustering method is DBSCAN.[10] In contrast to many newer methods, it features a well-defined cluster model called "density-reachability". Similar to linkage based clustering, it is based on connecting points within certain distance thresholds. However, it only connects points that satisfy a density criterion, in the original variant defined as a minimum number of other objects within this radius. A cluster consists of all density-connected objects (which can form a cluster of an arbitrary shape, in contrast to many other methods) plus all objects that are within these objects' range. Another interesting property of DBSCAN is that its complexity is fairly low - it requires a linear number of range queries on the database - and that it will discover essentially the same results (it is deterministic for core and noise points, but not for border points) in each run, therefore there is no need to run it multiple times. OPTICS[11] is a generalization of DBSCAN that removes the need to choose an appropriate value for the range parameter ε , and produces a hierarchical result related to that of linkage clustering. DeLi-Clu,[12] Density-Link-Clustering combines ideas from single-linkage clustering and OPTICS, eliminating the ε parameter entirely and offering performance improvements over OPTICS by using an R-tree index.

The key drawback of DBSCAN and OPTICS is that they expect some kind of density drop to detect cluster borders. Moreover, they cannot detect intrinsic cluster structures which are prevalent in the majority of real life data. A variation of DBSCAN, EnDBSCAN,[13] efficiently detects such kinds of structures. On data sets with, for example, overlapping Gaussian distributions - a common use case in artificial data - the cluster borders produced by these algorithms will often look arbitrary, because the cluster density decreases continuously. On a data set consisting of mixtures of Gaussians, these algorithms are nearly always outperformed by methods such as EM clustering that are able to precisely model this kind of data.

Mean-shift is a clustering approach where each object is moved to the densest area in its vicinity, based on kernel density estimation. Eventually, objects converge to local maxima of density. Similar to k-means clustering, these "density attractors" can serve as representatives for the data set, but mean-shift can detect arbitrary-shaped clusters similar to DBSCAN. Due to the expensive iterative procedure and density estimation, mean-shift is usually slower than DBSCAN or k-Means.

- Density-based clustering examples
- Density-based clustering with DBSCAN.
- DBSCAN assumes clusters of similar density, and may have problems separating nearby clusters
- OPTICS is a DBSCAN variant that handles different densities much better

41.2.5 Recent developments

In recent years considerable effort has been put into improving the performance of existing algorithms.[14][15] Among them are *CLARANS* (Ng and Han, 1994),[16] and *BIRCH* (Zhang et al., 1996).[17] With the recent need to process larger and larger data sets (also known as big data), the willingness to trade semantic meaning of the generated clusters for performance has been increasing. This led to the development of pre-clustering methods such as canopy clustering, which can process huge data sets efficiently, but the resulting "clusters" are merely a rough pre-partitioning of the data set to then analyze the partitions with existing slower methods such as k-means clustering. Various other approaches to clustering have been tried such as seed based clustering.[18]

For high-dimensional data, many of the existing methods fail due to the curse of dimensionality, which renders particular distance functions problematic in high-dimensional spaces. This led to new clustering algorithms for high-dimensional data that focus on subspace clustering (where only some attributes are used, and cluster models include the relevant attributes for the cluster) and correlation clustering that also looks for arbitrary rotated ("correlated") subspace clusters that can be modeled by giving a correlation of their attributes. Examples for such clustering algorithms are CLIQUE[19] and SUBCLU.[20]

Ideas from density-based clustering methods (in particular the DBSCAN/OPTICS family of algorithms) have been adopted to subspace clustering (HiSC,[21] hierarchical subspace clustering and DiSH[22]) and correlation clustering (HiCO,[23] hierarchical correlation clustering, 4C[24] using "correlation connectivity" and ERiC[25] exploring hierarchical density-based correlation clusters).

Several different clustering systems based on mutual information have been proposed. One is Marina Meilă's *variation of information* metric;[26] another provides hierarchical clustering.[27] Using genetic algorithms, a wide range of different fit-functions can be optimized, including mutual information.[28] Also message passing algorithms, a recent development in Computer Science and Statistical Physics, has led to the creation of new types of clustering algorithms.[29]

41.2.6 Other methods

- Basic sequential algorithmic scheme (BSAS)

41.3 Evaluation and assessment

Evaluation of clustering results sometimes is referred to as cluster validation.

There have been several suggestions for a measure of similarity between two clusterings. Such a measure can be used to compare how well different data clustering algorithms perform on a set of data. These measures are usually tied to the type of criterion being considered in assessing the quality of a clustering method.

41.3.1 Internal evaluation

When a clustering result is evaluated based on the data that was clustered itself, this is called internal evaluation. These methods usually assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters. One drawback of using internal criteria in cluster evaluation is that high scores on an internal measure do not necessarily result in effective information retrieval applications.[30] Additionally, this evaluation is biased towards algorithms that use the same cluster model. For example k-Means clustering naturally optimizes object distances, and a distance-based internal criterion will likely overrate the resulting clustering.

Therefore, the internal evaluation measures are best suited to get some insight into situations where one algorithm performs better than another, but this shall not imply that one algorithm produces more valid results than another.[4] Validity as measured by such an index depends on the claim that this kind of structure exists in the data set. An algorithm designed for some kind of models has no chance if the data set contains a radically different set of models, or if the evaluation measures a radically different criterion.[4] For example, k-means clustering can only find convex clusters, and many evaluation indexes assume convex clusters. On a data set with non-convex clusters neither the use of k-means, nor of an evaluation criterion that assumes convexity, is sound.

The following methods can be used to assess the quality of clustering algorithms based on internal criterion:

- **Davies–Bouldin index**

  The Davies–Bouldin index can be calculated by the following formula:

  $DB = \frac{1}{n}\sum_{i=1}^{n}\max_{j\neq i}\left(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\right)$

  where n is the number of clusters, c_x is the centroid of cluster x , σ_x is the average distance of all elements in cluster x to centroid c_x , and $d(c_i, c_j)$ is the distance between centroids c_i and c_j . Since algorithms that produce clusters with low intra-cluster distances (high intra-cluster similarity) and high inter-cluster distances (low inter-cluster similarity) will have a low Davies–Bouldin index, the clustering algorithm that produces a collection of clusters with the smallest Davies–Bouldin index is considered the best algorithm based on this criterion.

- **Dunn index**

  The Dunn index aims to identify dense and well-separated clusters. It is defined as the ratio between the minimal inter-cluster distance to maximal intra-cluster distance. For each cluster partition, the Dunn index can be calculated by the following formula:[31]

  $D = \frac{\min_{1\leq i<j\leq n} d(i,j)}{\max_{1\leq k\leq n} d'(k)}$,

  where *d*(*i*,*j*) represents the distance between clusters *i* and *j*, and *d* '(*k*) measures the intra-cluster distance of cluster *k*. The inter-cluster distance *d*(*i*,*j*) between two clusters may be any number of distance measures, such as the distance between the centroids of the clusters. Similarly, the intra-cluster distance *d* '(*k*) may be measured in a variety ways, such as the maximal distance between any pair of elements in cluster *k*. Since internal criterion seek clusters with high intra-cluster similarity and low inter-cluster similarity, algorithms that produce clusters with high Dunn index are more desirable.

- Silhouette coefficient

  The silhouette coefficient contrasts the average distance to elements in the same cluster with the average distance to elements in other clusters. Objects with a high silhouette value are considered well clustered, objects with a low value may be outliers. This index works well with k-means clustering, and is also used to determine the optimal number of clusters.

41.3.2 External evaluation

In external evaluation, clustering results are evaluated based on data that was not used for clustering, such as known class labels and external benchmarks. Such benchmarks consist of a set of pre-classified items, and these sets are often created by human (experts). Thus, the benchmark sets can be thought of as a gold standard for evaluation. These types of evaluation methods measure how close the clustering is to the predetermined benchmark classes. However, it has recently been discussed whether this is adequate for real data, or only on synthetic data sets with a factual ground truth, since classes can contain internal structure, the attributes present may not allow separation of clusters or the classes may contain anomalies.[32] Additionally, from a knowledge discovery point of view, the reproduction of known knowledge may not necessarily be the intended result.[32]

A number of measures are adapted from variants used to evaluate classification tasks. In place of counting the number of times a class was correctly assigned to a single data point (known as true positives), such *pair counting* metrics assess whether each pair of data points that is truly in the same cluster is predicted to be in the same cluster.

Some of the measures of quality of a cluster algorithm using external criterion include:

- **Rand measure** (William M. Rand 1971)[33]

  The Rand index computes how similar the clusters (returned by the clustering algorithm) are to the benchmark classifications. One can also view the Rand index as a measure of the percentage of correct decisions made by the algorithm. It can be computed using the following formula:

  $RI = \frac{TP+TN}{TP+FP+FN+TN}$

  where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. One issue with the Rand index is that false positives and false negatives are equally weighted. This may be an undesirable characteristic for some clustering applications. The F-measure addresses this concern, as does the chance-corrected adjusted Rand index.

- **F-measure**

  The F-measure can be used to balance the contribution of false negatives by weighting recall through a parameter $\beta \geq 0$. Let precision and recall be defined as follows:

  $P = \frac{TP}{TP+FP}$

  $R = \frac{TP}{TP+FN}$

  where P is the precision rate and R is the recall rate. We can calculate the F-measure by using the following formula:[30]

  $F_\beta = \frac{(\beta^2+1)\cdot P\cdot R}{\beta^2\cdot P+R}$

  Notice that when $\beta = 0$, $F_0 = P$. In other words, recall has no impact on the F-measure when $\beta = 0$, and increasing β allocates an increasing amount of weight to recall in the final F-measure.

- **Jaccard index**

  The Jaccard index is used to quantify the similarity between two datasets. The Jaccard index takes on a value between 0 and 1. An index of 1 means that the two dataset are identical, and an index of 0 indicates that the datasets have no common elements. The Jaccard index is defined by the following formula:

  $J(A,B) = \frac{|A\cap B|}{|A\cup B|} = \frac{TP}{TP+FP+FN}$

  This is simply the number of unique elements common to both sets divided by the total number of unique elements in both sets.

- **Fowlkes–Mallows index** (E. B. Fowlkes & C. L. Mallows 1983)[34]

  The Fowlkes-Mallows index computes the similarity between the clusters returned by the clustering algorithm and the benchmark classifications. The higher the value of the Fowlkes-Mallows index the more similar the clusters and the benchmark classifications are. It can be computed using the following formula:

  $FM = \sqrt{\frac{TP}{TP+FP}\cdot\frac{TP}{TP+FN}}$

  where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. The FM index is the geometric mean of the precision and recall P and R , while the F-measure is their harmonic mean.[35] Moreover, precision and recall are also known as Wallace's indices B^I and B^{II} .[36]

- The **Mutual Information** is an information theoretic measure of how much information is shared between a clustering and a ground-truth classification that can detect a non-linear similarity between two clusterings. Adjusted mutual information is the corrected-for-chance variant of this that has a reduced bias for varying cluster numbers.

- **Confusion matrix**

  A confusion matrix can be used to quickly visualize the results of a classification (or clustering) algorithm. It shows how different a cluster is from the gold standard cluster.

41.4 Applications

41.5 See also

41.5.1 Specialized types of cluster analysis

Others

Social science

Computer science

World wide web

Business and marketing

Medicine

Biology, computational biology and bioinformatics

Plant and animal ecologycluster analysis is used to describe and to make spatial and temporal comparisons of communities (assemblages) of organisms in heterogeneous environments; it is also used in plant systematics to generate artificial phylogenies or clusters of organisms (individuals) at the species, genus or higher level that share a number of attributes

Transcriptomicsclustering is used to build groups of genes with related expression patterns (also known as coexpressed genes) as in HCS clustering algorithm . Often such groups contain functionally related proteins, such as enzymes for a specific pathway, or genes that are co-regulated. High throughput experiments using expressed sequence tags (ESTs) or DNA microarrays can be a powerful tool for genome annotation, a general aspect of genomics.

Sequence analysisclustering is used to group homologous sequences into gene families. This is a very important concept in bioinformatics, and evolutionary biology in general. See evolution by gene duplication.

High-throughput genotyping platformsclustering algorithms are used to automatically assign genotypes.

Human genetic clustering The similarity of genetic data is used in clustering to infer population structures.

Medical imaging

On PET scans, cluster analysis can be used to differentiate between different types of tissue and blood in a three-dimensional image. In this application, actual position does not matter, but the voxel intensity is considered as a vector, with a dimension for each image that was taken over time. This technique allows, for example, accurate measurement of the rate a radioactive tracer is delivered to the area of interest, without a separate sampling of arterial blood, an intrusive technique that is most common today.

Analysis of antimicrobial activity Cluster analysis can be used to analyse patterns of antibiotic resistance, to classify antimicrobial compounds according to their mechanism of action, to classify antibiotics according to their antibacterial activity.

IMRT segmentation Clustering can be used to divide a fluence map into distinct regions for conversion into deliverable fields in MLC-based Radiation Therapy.

Market research

Social network analysis

Software evolution

Crime analysis

Clustering high-dimensional data

- Conceptual clustering
- Consensus clustering
- Constrained clustering
- Data stream clustering
- Sequence clustering
- Spectral clustering
- HCS clustering

41.5.2 Techniques used in cluster analysis

- Artificial neural network (ANN)
- Nearest neighbor search
- Neighbourhood components analysis
- Latent class analysis

41.5.3 Data projection and preprocessing

- Dimension reduction
- Principal component analysis
- Multidimensional scaling

41.5.4 Other

- Cluster-weighted modeling
- Curse of dimensionality
- Determining the number of clusters in a data set
- Parallel coordinates
- Structured data analysis

41.6 References

[1] Bailey, Ken (1994). "Numerical Taxonomy and Cluster Analysis". *Typologies and Taxonomies*. p. 34. ISBN 9780803952591.

[2] Tryon, Robert C. (1939). *Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality*. Edwards Brothers.

[3] Cattell, R. B. (1943). "The description of personality: Basic traits resolved into clusters". *Journal of Abnormal and Social Psychology* **38**: 476–506. doi:10.1037/h0054116.

[4] Estivill-Castro, Vladimir (20 June 2002). "Why so many clustering algorithms — A Position Paper". *ACM SIGKDD Explorations Newsletter* **4** (1): 65–75. doi:10.1145/568574.568575.

[5] Sibson, R. (1973). "SLINK: an optimally efficient algorithm for the single-link cluster method" (PDF). *The Computer Journal* (British Computer Society) **16** (1): 30–34. doi:10.1093/comjnl/16.1.30.

[6] Defays, D. (1977). "An efficient algorithm for a complete link method". *The Computer Journal* (British Computer Society) **20** (4): 364–366. doi:10.1093/comjnl/20.4.364.

[7] Lloyd, S. (1982). "Least squares quantization in PCM". *IEEE Transactions on Information Theory* **28** (2): 129–137. doi:10.1109/TIT.1982.1056489.

[8] Kriegel, Hans-Peter; Kröger, Peer; Sander, Jörg; Zimek, Arthur (2011). "Density-based Clustering". *WIREs Data Mining and Knowledge Discovery* **1** (3): 231–240. doi:10.1002/widm.30.

[9] Microsoft academic search: most cited data mining articles: DBSCAN is on rank 24, when accessed on: 4/18/2010

[10] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press. pp. 226–231. ISBN 1-57735-004-9. CiteSeerX: 10.1.1.71.1980.

[11] Ankerst, Mihael; Breunig, Markus M.; Kriegel, Hans-Peter; Sander, Jörg (1999). "OPTICS: Ordering Points To Identify the Clustering Structure". *ACM SIGMOD international conference on Management of data*. ACM Press. pp. 49–60. CiteSeerX: 10.1.1.129.6542.

[12] Achtert, E.; Böhm, C.; Kröger, P. (2006). "DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking". *LNCS: Advances in Knowledge Discovery and Data Mining*. Lecture Notes in Computer Science **3918**: 119–128. doi:10.1007/11731139_16. ISBN 978-3-540-33206-0.

[13] Roy, S.; Bhattacharyya, D. K. (2005). "An Approach to find Embedded Clusters Using Density Based Techniques". *LNCS Vol.3816*. Springer Verlag. pp. 523–535.

[14] Sculley, D. (2010). *Web-scale k-means clustering*. Proc. 19th WWW.

[15] Huang, Z. (1998). "Extensions to the *k*-means algorithm for clustering large data sets with categorical values". *Data Mining and Knowledge Discovery* **2**: 283–304.

[16] R. Ng and J. Han. "Efficient and effective clustering method for spatial data mining". In: Proceedings of the 20th VLDB Conference, pages 144-155, Santiago, Chile, 1994.

[17] Tian Zhang, Raghu Ramakrishnan, Miron Livny. "An Efficient Data Clustering Method for Very Large Databases." In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 103–114.

[18] Can, F.; Ozkarahan, E. A. (1990). "Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases". *ACM Transactions on Database Systems* **15** (4): 483. doi:10.1145/99935.99938.

[19] Agrawal, R.; Gehrke, J.; Gunopulos, D.; Raghavan, P. (2005). "Automatic Subspace Clustering of High Dimensional Data". *Data Mining and Knowledge Discovery* **11**: 5. doi:10.1007/s10618-005-1396-1.

[20] Karin Kailing, Hans-Peter Kriegel and Peer Kröger. *Density-Connected Subspace Clustering for High-Dimensional Data*. In: *Proc. SIAM Int. Conf. on Data Mining (SDM'04)*, pp. 246-257, 2004.

[21] Achtert, E.; Böhm, C.; Kriegel, H. P.; Kröger, P.; Müller-Gorman, I.; Zimek, A. (2006). "Finding Hierarchies of Subspace Clusters". *LNCS: Knowledge Discovery in Databases: PKDD 2006*. Lecture Notes in Computer Science **4213**: 446–453. doi:10.1007/11871637_42. ISBN 978-3-540-45374-1.

[22] Achtert, E.; Böhm, C.; Kriegel, H. P.; Kröger, P.; Müller-Gorman, I.; Zimek, A. (2007). "Detection and Visualization of Subspace Cluster Hierarchies". *LNCS: Advances in Databases: Concepts, Systems and Applications*. Lecture Notes in Computer Science **4443**: 152–163. doi:10.1007/978-3-540-71703-4_15. ISBN 978-3-540-71702-7.

[23] Achtert, E.; Böhm, C.; Kröger, P.; Zimek, A. (2006). "Mining Hierarchies of Correlation Clusters". *Proc. 18th International Conference on Scientific and Statistical Database Management (SSDBM)*: 119–128. doi:10.1109/SSDBM.2006.35. ISBN 0-7695-2590-3.

[24] Böhm, C.; Kailing, K.; Kröger, P.; Zimek, A. (2004). "Computing Clusters of Correlation Connected objects". *Proceedings of the 2004 ACM SIGMOD international conference on Management of data - SIGMOD '04*. p. 455. doi:10.1145/1007568.1007620. ISBN 1581138598.

[25] Achtert, E.; Bohm, C.; Kriegel, H. P.; Kröger, P.; Zimek, A. (2007). "On Exploring Complex Relationships of Correlation Clusters". *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*. p. 7. doi:10.1109/SSDBM.2007.21. ISBN 0-7695-2868-6.

[26] Meilă, Marina (2003). "Comparing Clusterings by the Variation of Information". *Learning Theory and Kernel Machines*. Lecture Notes in Computer Science **2777**: 173–187. doi:10.1007/978-3-540-45167-9_14. ISBN 978-3-540-40720-1.

[27] Kraskov, Alexander; Stögbauer, Harald; Andrzejak, Ralph G.; Grassberger, Peter (1 December 2003) [28 November 2003]. "Hierarchical Clustering Based on Mutual Information". arXiv:q-bio/0311039.

[28] Auffarth, B. (July 18–23, 2010). "Clustering by a Genetic Algorithm with Biased Mutation Operator". *WCCI CEC* (IEEE). CiteSeerX: 10.1.1.170.869.

[29] Frey, B. J.; Dueck, D. (2007). "Clustering by Passing Messages Between Data Points". *Science* **315** (5814): 972–976. doi:10.1126/science.1136800. PMID 17218491.

[30] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. *Introduction to Information Retrieval*. Cambridge University Press. ISBN 978-0-521-86571-5.

[31] Dunn, J. (1974). "Well separated clusters and optimal fuzzy partitions". *Journal of Cybernetics* **4**: 95–104. doi:10.1080/01969727408546059.

[32] Färber, Ines; Günnemann, Stephan; Kriegel, Hans-Peter; Kröger, Peer; Müller, Emmanuel; Schubert, Erich; Seidl, Thomas; Zimek, Arthur (2010). "On Using Class-Labels in Evaluation of Clusterings" (PDF). In Fern, Xiaoli Z.; Davidson, Ian; Dy, Jennifer. *MultiClust: Discovering, Summarizing, and Using Multiple Clusterings*. ACM SIGKDD.

[33] Rand, W. M. (1971). "Objective criteria for the evaluation of clustering methods". *Journal of the American Statistical Association* (American Statistical Association) **66** (336): 846–850. doi:10.2307/2284239. JSTOR 2284239.

[34] E. B. Fowlkes & C. L. Mallows (1983), "A Method for Comparing Two Hierarchical Clusterings", Journal of the American Statistical Association 78, 553–569.

[35] L. Hubert et P. Arabie. Comparing partitions. J. of Classification, 2(1), 1985.

[36] D. L. Wallace. Comment. Journal of the American Statistical Association, 78 :569– 579, 1983.

[37] Bewley, A. et al. "Real-time volume estimation of a dragline payload". *IEEE International Conference on Robotics and Automation* **2011**: 1571–1576.

[38] Basak, S.C.; Magnuson, V.R.; Niemi, C.J.; Regal, R.R. "Determining Structural Similarity of Chemicals Using Graph Theoretic Indices". *Discr. Appl. Math., 19* **1988**: 17–44.

[39] Huth, R. et al. (2008). "Classifications of Atmospheric Circulation Patterns: Recent Advances and Applications". *Ann. N.Y. Acad. Sci.* **1146**: 105–152.

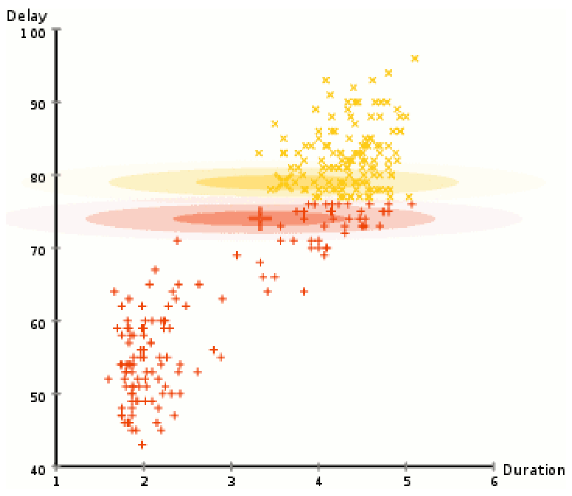
41.7 External links

- Data Mining at DMOZ

Chapter 42

Expectation–maximization algorithm

In statistics, an **expectation–maximization (EM) algorithm** is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the *E* step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.



EM clustering of Old Faithful eruption data. The random initial model (which, due to the different scales of the axes, appears to be two very flat and wide spheres) is fit to the observed data. In the first iterations, the model changes substantially, but then converges to the two modes of the geyser. Visualized using ELKI.

42.1 History

The EM algorithm was explained and given its name in a classic 1977 paper by Arthur Dempster, Nan Laird, and Donald Rubin.[1] They pointed out that the method had been "proposed many times in special circumstances" by earlier authors. In particular, a very detailed treatment of the EM method for exponential families was published by Rolf Sundberg in his thesis and several papers[2][3][4] following his collaboration with Per Martin-Löf and Anders Martin-Löf.[5][6][7][8][9][10][11] The Dempster-Laird-Rubin paper in 1977 generalized the method and sketched a convergence analysis for a wider class of problems. Regardless of earlier inventions, the innovative Dempster-Laird-Rubin paper in the *Journal of the Royal Statistical Society* received an enthusiastic discussion at the Royal Statistical Society meeting with Sundberg calling the paper "brilliant". The Dempster-Laird-Rubin paper established the EM method as an important tool of statistical analysis.

The convergence analysis of the Dempster-Laird-Rubin paper was flawed and a correct convergence analysis was published by C.F. Jeff Wu in 1983.[12] Wu's proof established the EM method's convergence outside of the exponential family, as claimed by Dempster-Laird-Rubin.[13]

42.2 Introduction

The EM algorithm is used to find (locally) maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent variables in addition to unknown parameters and known data observations. That is, either there are missing values among the data, or the model can be formulated more simply by assuming the existence of additional unobserved data points. For example, a mixture model can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component that each data point belongs to.

Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values — viz. the parameters and the latent variables — and simultaneously solving the resulting equations. In statistical models with latent variables, this usually is not possible. Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice versa, but substituting one set of equations into the other produces an unsolvable equation.

The EM algorithm proceeds from the observation that the following is a way to solve these two sets of equations numerically. One can simply pick arbitrary values for one of the two sets of unknowns, use them to estimate the second set, then use these new values to find a better estimate of the first set, and then keep alternating between the two until the resulting values both converge to fixed points. It's not obvious that this will work at all, but in fact it can be proven that in this particular context it does, and that the derivative of the likelihood is (arbitrarily close to) zero at that point, which in turn means that the point is either a maximum or a saddle point. In general there may be multiple maxima, and there is no guarantee that the global maximum will be found. Some likelihoods also have singularities in them, i.e. nonsensical maxima. For example, one of the "solutions" that may be found by EM in a mixture model involves setting one of the components to have zero variance and the mean parameter for the same component to be equal to one of the data points.

42.3 Description

Given a statistical model which generates a set $\mathbf{X}$ of observed data, a set of unobserved latent data or missing values $\mathbf{Z}$, and a vector of unknown parameters $\boldsymbol{\theta}$, along with a likelihood function $L(\boldsymbol{\theta};\mathbf{X},\mathbf{Z}) = p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data

$$L(\boldsymbol{\theta};\mathbf{X}) = p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$$

However, this quantity is often intractable (e.g. if $\mathbf{Z}$ is a sequence of events, so that the number of values grows exponentially with the sequence length, making the exact calculation of the sum extremely difficult).

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps:

Expectation step (E step): Calculate the expected value of the log likelihood function, with respect to the conditional distribution of $\mathbf{Z}$ given $\mathbf{X}$ under the current estimate of the parameters $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \operatorname{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}}\left[\log L(\boldsymbol{\theta};\mathbf{X},\mathbf{Z})\right]$$

Maximization step (M step): Find the parameter that maximizes this quantity:

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{arg\,max}}\ Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$$

Note that in typical models to which EM is applied:

1. The observed data points $\mathbf{X}$ may be discrete (taking values in a finite or countably infinite set) or continuous (taking values in an uncountably infinite set). There may in fact be a vector of observations associated with each data point.

2. The missing values (aka latent variables) $\mathbf{Z}$ are discrete, drawn from a fixed number of values, and there is one latent variable per observed data point.

3. The parameters are continuous, and are of two kinds: Parameters that are associated with all data points, and parameters associated with a particular value of a latent variable (i.e. associated with all data points whose corresponding latent variable has a particular value).

However, it is possible to apply EM to other sorts of models.

The motivation is as follows. If we know the value of the parameters $\boldsymbol{\theta}$, we can usually find the value of the latent variables $\mathbf{Z}$ by maximizing the log-likelihood over all possible values of $\mathbf{Z}$, either simply by iterating over $\mathbf{Z}$ or through an algorithm such as the Viterbi algorithm for hidden Markov models. Conversely, if we know the value of the latent variables $\mathbf{Z}$, we can find an estimate of the parameters $\boldsymbol{\theta}$ fairly easily, typically by simply grouping the observed data points according to the value of the associated latent variable and averaging the values, or some function of the values, of the points in each group. This suggests an iterative algorithm, in the case where both $\boldsymbol{\theta}$ and $\mathbf{Z}$ are unknown:

1. First, initialize the parameters $\boldsymbol{\theta}$ to some random values.

2. Compute the best value for $\mathbf{Z}$ given these parameter values.

3. Then, use the just-computed values of $\mathbf{Z}$ to compute a better estimate for the parameters $\boldsymbol{\theta}$. Parameters associated with a particular value of $\mathbf{Z}$ will use only those data points whose associated latent variable has that value.

4. Iterate steps 2 and 3 until convergence.

The algorithm as just described monotonically approaches a local minimum of the cost function, and is commonly called *hard EM*. The k-means algorithm is an example of this class of algorithms.

However, one can do somewhat better: Rather than making a hard choice for $\mathbf{Z}$ given the current parameter values and averaging only over the set of data points associated with a particular value of $\mathbf{Z}$, one can instead determine the probability of each possible value of $\mathbf{Z}$ for each data point, and then use the probabilities associated with a particular value of **Z** to compute a weighted average over the entire set of data points. The resulting algorithm is commonly called *soft EM*, and is the type of algorithm normally associated with EM. The counts used to compute these weighted averages are called *soft counts* (as opposed to the *hard counts* used in a hard-EM-type algorithm such as *k*-means). The probabilities computed for **Z** are posterior probabilities and are what is computed in the E step. The soft counts used to compute new parameter values are what is computed in the M step.

42.4 Properties

Speaking of an expectation (E) step is a bit of a misnomer. What is calculated in the first step are the fixed, data-dependent parameters of the function *Q*. Once the parameters of *Q* are known, it is fully determined and is maximized in the second (M) step of an EM algorithm.

Although an EM iteration does increase the observed data (i.e. marginal) likelihood function there is no guarantee that the sequence converges to a maximum likelihood estimator. For multimodal distributions, this means that an EM algorithm may converge to a local maximum of the observed data likelihood function, depending on starting values. There are a variety of heuristic or metaheuristic approaches for escaping a local maximum such as random restart (starting with several different random initial estimates $\theta^{(t)}$), or applying simulated annealing methods.

EM is particularly useful when the likelihood is an exponential family: the E step becomes the sum of expectations of sufficient statistics, and the M step involves maximizing a linear function. In such a case, it is usually possible to derive closed form updates for each step, using the Sundberg formula (published by Rolf Sundberg using unpublished results of Per Martin-Löf and Anders Martin-Löf).[3][4][7][8][9][10][11]

The EM method was modified to compute maximum a posteriori (MAP) estimates for Bayesian inference in the original paper by Dempster, Laird, and Rubin.

There are other methods for finding maximum likelihood estimates, such as gradient descent, conjugate gradient or variations of the Gauss–Newton method. Unlike EM, such methods typically require the evaluation of first and/or second derivatives of the likelihood function.

42.5 Proof of correctness

Expectation-maximization works to improve $Q(\boldsymbol\theta|\boldsymbol\theta^{(t)})$ rather than directly improving $\log p(\mathbf{X}|\boldsymbol\theta)$. Here we show that improvements to the former imply improvements to the latter.[14]

For any **Z** with non-zero probability $p(\mathbf{Z}|\mathbf{X},\boldsymbol\theta)$, we can write

$$\log p(\mathbf{X}|\boldsymbol\theta) = \log p(\mathbf{X},\mathbf{Z}|\boldsymbol\theta) - \log p(\mathbf{Z}|\mathbf{X},\boldsymbol\theta)\,.$$

We take the expectation over values of **Z** by multiplying both sides by $p(\mathbf{Z}|\mathbf{X},\boldsymbol\theta^{(t)})$ and summing (or integrating) over **Z** . The left-hand side is the expectation of a constant, so we get:

$$\begin{aligned}\log p(\mathbf{X}|\boldsymbol\theta) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\boldsymbol\theta^{(t)})\log p(\mathbf{X},\mathbf{Z}|\boldsymbol\theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\boldsymbol\theta^{(t)})\log p(\mathbf{Z}|\mathbf{X}\\ &= Q(\boldsymbol\theta|\boldsymbol\theta^{(t)}) + H(\boldsymbol\theta|\boldsymbol\theta^{(t)})\,,\end{aligned}$$

where $H(\boldsymbol\theta|\boldsymbol\theta^{(t)})$ is defined by the negated sum it is replacing. This last equation holds for any value of $\boldsymbol\theta$ including $\boldsymbol\theta = \boldsymbol\theta^{(t)}$,

$$\log p(\mathbf{X}|\boldsymbol\theta^{(t)}) = Q(\boldsymbol\theta^{(t)}|\boldsymbol\theta^{(t)}) + H(\boldsymbol\theta^{(t)}|\boldsymbol\theta^{(t)})\,,$$

and subtracting this last equation from the previous equation gives

$$\log p(\mathbf{X}|\boldsymbol\theta) - \log p(\mathbf{X}|\boldsymbol\theta^{(t)}) = Q(\boldsymbol\theta|\boldsymbol\theta^{(t)}) - Q(\boldsymbol\theta^{(t)}|\boldsymbol\theta^{(t)}) + H(\boldsymbol\theta|\boldsymbol\theta^{(t)}) - H(\boldsymbol\theta^{($$

However, Gibbs' inequality tells us that $H(\boldsymbol\theta|\boldsymbol\theta^{(t)}) \geq H(\boldsymbol\theta^{(t)}|\boldsymbol\theta^{(t)})$, so we can conclude that

$$\log p(\mathbf{X}|\boldsymbol\theta) - \log p(\mathbf{X}|\boldsymbol\theta^{(t)}) \geq Q(\boldsymbol\theta|\boldsymbol\theta^{(t)}) - Q(\boldsymbol\theta^{(t)}|\boldsymbol\theta^{(t)})\,.$$

In words, choosing $\boldsymbol\theta$ to improve $Q(\boldsymbol\theta|\boldsymbol\theta^{(t)})$ beyond $Q(\boldsymbol\theta^{(t)}|\boldsymbol\theta^{(t)})$ will improve $\log p(\mathbf{X}|\boldsymbol\theta)$ beyond $\log p(\mathbf{X}|\boldsymbol\theta^{(t)})$ at least as much.

42.6 Alternative description

Under some circumstances, it is convenient to view the EM algorithm as two alternating maximization steps.[15][16] Consider the function:

$$F(q,\theta) = \operatorname{E}_q[\log L(\theta;x,Z)] + H(q) = -D_{\text{KL}}\big(q\big\|p_{Z|X}(\cdot|x;\theta)\big) + \log L(\theta;$$

where *q* is an arbitrary probability distribution over the unobserved data *z*, *pZ*|*X*(· |*x*;*θ*) is the conditional distribution of the unobserved data given the observed data *x*, *H* is the entropy and *D*KL is the Kullback–Leibler divergence.

Then the steps in the EM algorithm may be viewed as:

Expectation step: Choose q to maximize F:

$q^{(t)} = * \, arg\, max_q \; F(q, \theta^{(t)})$

Maximization step: Choose θ to maximize F:

$\theta^{(t+1)} = * \, arg\, max_\theta \; F(q^{(t)}, \theta)$

42.7 Applications

EM is frequently used for data clustering in machine learning and computer vision. In natural language processing, two prominent instances of the algorithm are the Baum-Welch algorithm and the inside-outside algorithm for unsupervised induction of probabilistic context-free grammars.

In psychometrics, EM is almost indispensable for estimating item parameters and latent abilities of item response theory models.

With the ability to deal with missing data and observe unidentified variables, EM is becoming a useful tool to price and manage risk of a portfolio.[ref?]

The EM algorithm (and its faster variant Ordered subset expectation maximization) is also widely used in medical image reconstruction, especially in positron emission tomography and single photon emission computed tomography. See below for other faster variants of EM.

42.8 Filtering and smoothing EM algorithms

A Kalman filter is typically used for on-line state estimation and a minimum-variance smoother may be employed for off-line or batch state estimation. However, these minimum-variance solutions require estimates of the state-space model parameters. EM algorithms can be used for solving joint state and parameter estimation problems.

Filtering and smoothing EM algorithms arise by repeating the following two-step procedure:

E-step Operate a Kalman filter or a minimum-variance smoother designed with current parameter estimates to obtain updated state estimates.

M-step Use the filtered or smoothed state estimates within maximum-likelihood calculations to obtain updated parameter estimates.

Suppose that a Kalman filter or minimum-variance smoother operates on noisy measurements of a single-input-single-output system. An updated measurement noise variance estimate can be obtained from the maximum likelihood calculation

$$\hat{\sigma}_v^2 = \frac{1}{N} \sum_{k=1}^{N} (z_k - \hat{x}_k)^2$$

where $\hat{x}_k$ are scalar output estimates calculated by a filter or a smoother from N scalar measurements z_k . Similarly, for a first-order auto-regressive process, an updated process noise variance estimate can be calculated by

$$\hat{\sigma}_w^2 = \frac{1}{N} \sum_{k=1}^{N} (\hat{x}_{k+1} - \hat{F}\hat{x}_k)^2$$

where $\hat{x}_k$ and $\hat{x}_{k+1}$ are scalar state estimates calculated by a filter or a smoother. The updated model coefficient estimate is obtained via

$$\hat{F} = \frac{\sum_{k=1}^{N} (\hat{x}_{k+1} - \hat{F}\hat{x}_k)}{\sum_{k=1}^{N} \hat{x}_k^2}$$

The convergence of parameter estimates such as those above are well studied.[17][18][19]

42.9 Variants

A number of methods have been proposed to accelerate the sometimes slow convergence of the EM algorithm, such as those using conjugate gradient and modified Newton–Raphson techniques.[20] Additionally EM can be used with constrained estimation techniques.

Expectation conditional maximization (ECM) replaces each M step with a sequence of conditional maximization (CM) steps in which each parameter θi is maximized individually, conditionally on the other parameters remaining fixed.[21]

This idea is further extended in **generalized expectation maximization (GEM)** algorithm, in which one only seeks an increase in the objective function F for both the E step and M step under the alternative description.[15] GEM is further developed in a distributed environment and shows promising results.[22]

It is also possible to consider the EM algorithm as a subclass of the **MM** (Majorize/Minimize or Minorize/Maximize, depending on context) algorithm,[23] and therefore use any machinery developed in the more general case.

42.9.1 α-EM algorithm

The Q-function used in the EM algorithm is based on the log likelihood. Therefore, it is regarded as the log-EM algorithm. The use of the log likelihood can be generalized to that of the α-log likelihood ratio. Then, the α-log likelihood ratio of the observed data can be exactly expressed as equality by using the Q-function of the α-log likelihood ratio and the α-divergence. Obtaining this Q-function is a generalized E step. Its maximization is a generalized M step. This pair is called the α-EM algorithm [24] which contains the log-EM algorithm as its subclass. Thus, the α-EM algorithm by Yasuo Matsuyama is an exact generalization of the log-EM algorithm. No computation of gradient or Hessian matrix is needed. The α-EM shows faster convergence than the log-EM algorithm by choosing an appropriate α. The α-EM algorithm leads to a faster version of the Hidden Markov model estimation algorithm α-HMM. [25]

42.10 Relation to variational Bayes methods

EM is a partially non-Bayesian, maximum likelihood method. Its final result gives a probability distribution over the latent variables (in the Bayesian style) together with a point estimate for θ (either a maximum likelihood estimate or a posterior mode). We may want a fully Bayesian version of this, giving a probability distribution over θ as well as the latent variables. In fact the Bayesian approach to inference is simply to treat θ as another latent variable. In this paradigm, the distinction between the E and M steps disappears. If we use the factorized Q approximation as described above (variational Bayes), we may iterate over each latent variable (now including θ) and optimize them one at a time. There are now k steps per iteration, where k is the number of latent variables. For graphical models this is easy to do as each variable's new Q depends only on its Markov blanket, so local message passing can be used for efficient inference.

42.11 Geometric interpretation

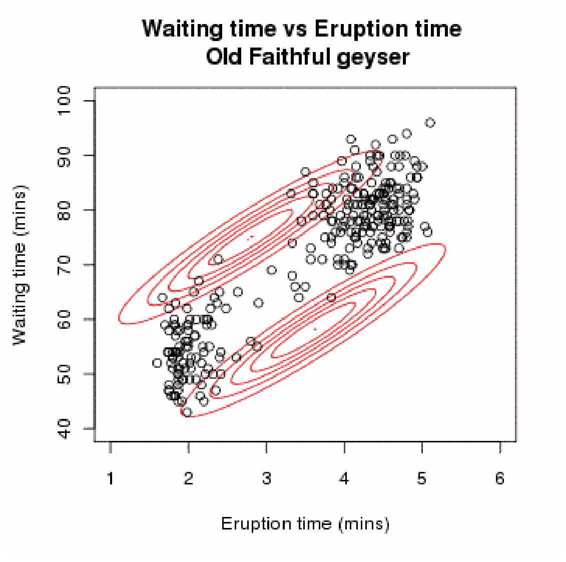
For more details on this topic, see Information geometry.

In information geometry, the E step and the M step are interpreted as projections under dual affine connections, called the e-connection and the m-connection; the Kullback–Leibler divergence can also be understood in these terms.

42.12 Examples

42.12.1 Gaussian mixture

Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ be a sample of n independent observations from a mixture of two multivariate normal distributions of dimension d, and let $\mathbf{z} = (z_1, z_2, \ldots, z_n)$



An animation demonstrating the EM algorithm fitting a two component Gaussian mixture model to the Old Faithful dataset. The algorithm steps through from a random initialization to convergence.

be the latent variables that determine the component from which the observation originates.[16]

$$X_i|(Z_i = 1) \sim \mathcal{N}_d(\boldsymbol{\mu}_1, \Sigma_1) \text{ and } X_i|(Z_i = 2) \sim \mathcal{N}_d(\boldsymbol{\mu}_2, \Sigma_2)$$

where

$$\operatorname{P}(Z_i = 1) = \tau_1 \text{ and } \operatorname{P}(Z_i = 2) = \tau_2 = 1 - \tau_1$$

The aim is to estimate the unknown parameters representing the "mixing" value between the Gaussians and the means and covariances of each:

$$\theta = \big(\boldsymbol{\tau}, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2\big)$$

where the incomplete-data likelihood function is

$$L(\theta; \mathbf{x}) = \prod_{i=1}^{n} \sum_{j=1}^{2} \tau_j \ f(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)$$

and the complete-data likelihood function is

$$L(\theta; \mathbf{x}, \mathbf{z}) = P(\mathbf{x}, \mathbf{z}|\theta) = \prod_{i=1}^{n} \sum_{j=1}^{2} \mathbb{I}(z_i = j) \ f(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j) \tau_j$$

or

$$L(\theta; \mathbf{x}, \mathbf{z}) = \exp\left\{ \sum_{i=1}^{n} \sum_{j=1}^{2} \mathbb{I}(z_i = j) \big[\log \tau_j - \tfrac{1}{2} \log |\Sigma_j| - \tfrac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^\top \Sigma \right.$$

where $\mathbb{I}$ is an indicator function and f is the probability density function of a multivariate normal.

To see the last equality, note that for each *i* all indicators $\mathbb{I}(z_i = j)$ are equal to zero, except for one which is equal to one. The inner sum thus reduces to a single term.

E step

Given our current estimate of the parameters $\theta^{(t)}$, the conditional distribution of the *Zi* is determined by Bayes theorem to be the proportional height of the normal density weighted by τ:

$$T_{j,i}^{(t)} := \operatorname{P}(Z_i = j \mid X_i = \mathbf{x}_i ;\theta^{(t)}) = \frac{\tau_j^{(t)}\ f(\mathbf{x}_i;\boldsymbol{\mu}_j^{(t)},\Sigma_j^{(t)})}{\tau_1^{(t)}\ f(\mathbf{x}_i;\boldsymbol{\mu}_1^{(t)},\Sigma_1^{(t)}) + \tau_2^{(t)}\ f(\mathbf{x}_i;\boldsymbol{\mu}_2^{(t)},\Sigma_2^{(t)})}$$

These are called the "membership probabilities" which are normally considered the output of the E step (although this is not the Q function of below).

Note that this E step corresponds with the following function for Q:

$$\begin{aligned}
Q(\theta|\theta^{(t)}) &= \operatorname{E}[\log L(\theta;\mathbf{x},\mathbf{Z})] \\
&= \operatorname{E}[\log \prod_{i=1}^{n}L(\theta;\mathbf{x}_i,\mathbf{z}_i)] \\
&= \operatorname{E}[\sum_{i=1}^n \log L(\theta;\mathbf{x}_i,\mathbf{z}_i)] \\
&= \sum_{i=1}^n\operatorname{E}[\log L(\theta;\mathbf{x}_i,\mathbf{z}_i)] \\
&= \sum_{i=1}^n \sum_{j=1}^2 T_{j,i}^{(t)} \big[\log \tau_j -\tfrac{1}{2} \log |\Sigma_j| -\tfrac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^\top\Sigma_j^{-1} (\mathbf{x}_i-\boldsymbol{\mu}_j) -\tfrac{d}{2} \log(2\pi) \big]
\end{aligned}$$

This does not need to be calculated, because in the M step we only require the terms depending on τ when we maximize for τ, or only the terms depending on $\boldsymbol{\mu}$ if we maximize for $\boldsymbol{\mu}$.

M step

The fact that $Q(\theta|\theta^{(t)})$ is quadratic in form means that determining the maximizing values of θ is relatively straightforward. Note that τ, ($\boldsymbol{\mu}_1$,Σ_1) and ($\boldsymbol{\mu}_2$,Σ_2) may all be maximized independently since they all appear in separate linear terms.

To begin, consider τ, which has the constraint $\tau_1 + \tau_2$=1:

$$\begin{aligned}
\boldsymbol{\tau}^{(t+1)} &= \underset{\boldsymbol{\tau}}{\operatorname{arg\,max}}\ Q(\theta | \theta^{(t)}) \\
&= \underset{\boldsymbol{\tau}}{\operatorname{arg\,max}} \ \left\{ \left[\sum_{i=1}^n T_{1,i}^{(t)} \right] \log \tau_1 + \left[\sum_{i=1}^n T_{2,i}^{(t)} \right] \log \tau_2 \right\}
\end{aligned}$$

This has the same form as the MLE for the binomial distribution, so

$$\tau^{(t+1)}_j = \frac{\sum_{i=1}^n T_{j,i}^{(t)}}{\sum_{i=1}^n (T_{1,i}^{(t)} + T_{2,i}^{(t)}) } = \frac{1}{n} \sum_{i=1}^n T_{j,i}^{(t)}$$

For the next estimates of ($\boldsymbol{\mu}_1$,σ_1):

$$\begin{aligned}
(\boldsymbol{\mu}_1^{(t+1)},\Sigma_1^{(t+1)}) &= \underset{\boldsymbol{\mu}_1,\Sigma_1}{\operatorname{arg\,max}}\ Q(\theta | \theta^{(t)}) \\
&= \underset{\boldsymbol{\mu}_1,\Sigma_1}{\operatorname{arg\,max}}\ \sum_{i=1}^n T_{1,i}^{(t)} \left\{ -\tfrac{1}{2} \log |\Sigma_1| -\tfrac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_1)^\top\Sigma_1^{-1} (\mathbf{x}
\end{aligned}$$

This has the same form as a weighted MLE for a normal distribution, so

$$\boldsymbol{\mu}_1^{(t+1)} = \frac{\sum_{i=1}^n T_{1,i}^{(t)} \mathbf{x}_i}{\sum_{i=1}^n T_{1,i}^{(t)}} \text{ and } \Sigma_1^{(t+1)} = \frac{\sum_{i=1}^n T_{1,i}^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_1^{(t+1)}) (\mathbf{x}_i - \boldsymbol{\mu}_1^{(t+1)})^\top }{\sum_{i=1}^n T_{1,i}^{(t)}}$$

and, by symmetry

$$\boldsymbol{\mu}_2^{(t+1)} = \frac{\sum_{i=1}^n T_{2,i}^{(t)} \mathbf{x}_i}{\sum_{i=1}^n T_{2,i}^{(t)}} \text{ and } \Sigma_2^{(t+1)} = \frac{\sum_{i=1}^n T_{2,i}^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_2^{(t+1)}) (\mathbf{x}_i - \boldsymbol{\mu}_2^{(t+1)})^\top }{\sum_{i=1}^n T_{2,i}^{(t)}} .$$

Termination

Conclude the iterative process if $\log L(\theta^t;\mathbf{x},\mathbf{Z}) \leq \log L(\theta^{(t-1)};\mathbf{x},\mathbf{Z}) + \epsilon$ for ϵ below some preset threshold.

Generalization

The algorithm illustrated above can be generalized for mixtures of more than two multivariate normal distributions.

42.12.2 Truncated and censored regression

The EM algorithm has been implemented in the case where there is an underlying linear regression model explaining the variation of some quantity, but where the values actually observed are censored or truncated versions of those represented in the model.[26] Special cases of this model include censored or truncated observations from a single normal distribution.[26]

42.13 Alternatives to EM

EM typically converges to a local optimum--not necessarily the global optimum--and there is no bound on the convergence rate in general. It is possible that it can be arbitrarily poor in high dimensions and there can be an exponential number of local optima. Hence, there is a need for alternative techniques for guaranteed learning, especially in the high-dimensional setting. There are alternatives to EM with better guarantees in terms of consistency which are known as moment-based approaches or the so-called "spectral techniques". Moment-based approaches to learning the parameters of a probabilistic model are of increasing interest recently since they enjoy guarantees such as global convergence under certain conditions unlike EM which is often plagued by the issue of getting stuck in local optima. Algorithms with guarantees for learning can be derived for a number of important models such as mixture models, HMMs etc. For these spectral methods, there are no spurious local optima and the true parameters can be consistently estimated under some regularity conditions.

42.14 See also

- Density estimation
- Total absorption spectroscopy
- The EM algorithm can be viewed as a special case of the majorize-minimization (MM) algorithm.[27]

42.15 Further reading

- Robert Hogg, Joseph McKean and Allen Craig. *Introduction to Mathematical Statistics*. pp. 359–364. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.
- The on-line textbook: Information Theory, Inference, and Learning Algorithms, by David J.C. MacKay includes simple examples of the EM algorithm such as clustering using the soft k-means algorithm, and emphasizes the variational view of the EM algorithm, as described in Chapter 33.7 of version 7.2 (fourth edition).
- Dellaert, Frank. "The Expectation Maximization Algorithm". CiteSeerX: 10.1.1.9.9735, gives an easier explanation of EM algorithm in terms of lowerbound maximization.
- Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer. ISBN 0-387-31073-8.
- M. R. Gupta and Y. Chen (2010). *Theory and Use of the EM Algorithm*. doi:10.1561/2000000034. A well-written short book on EM, including detailed derivation of EM for GMMs, HMMs, and Dirichlet.
- Bilmes, Jeff. "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models". CiteSeerX: 10.1.1.28.613, includes a simplified derivation of the EM equations for Gaussian Mixtures and Gaussian Mixture Hidden Markov Models.
- Variational Algorithms for Approximate Bayesian Inference, by M. J. Beal includes comparisons of EM to Variational Bayesian EM and derivations of several models including Variational Bayesian HMMs (chapters).
- The Expectation Maximization Algorithm: A short tutorial, A self-contained derivation of the EM Algorithm by Sean Borman.
- The EM Algorithm, by Xiaojin Zhu.
- EM algorithm and variants: an informal tutorial by Alexis Roche. A concise and very clear description of EM and many interesting variants.
- Einicke, G.A. (2012). *Smoothing, Filtering and Prediction: Estimating the Past, Present and Future*. Rijeka, Croatia: Intech. ISBN 978-953-307-752-9.

42.16 References

[1] Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B* **39** (1): 1–38. JSTOR 2984875. MR 0501537.

[2] Sundberg, Rolf (1974). "Maximum likelihood theory for incomplete data from an exponential family". *Scandinavian Journal of Statistics* **1** (2): 49–58. JSTOR 4615553. MR 381110.

[3] Rolf Sundberg. 1971. *Maximum likelihood theory and applications for distributions generated when observing a function of an exponential family variable*. Dissertation, Institute for Mathematical Statistics, Stockholm University.

[4] Sundberg, Rolf (1976). "An iterative method for solution of the likelihood equations for incomplete data from exponential families". *Communications in Statistics – Simulation and Computation* **5** (1): 55–64. doi:10.1080/03610917608812007. MR 443190.

[5] See the acknowledgement by Dempster, Laird and Rubin on pages 3, 5 and 11.

[6] G. Kulldorff. 1961. *Contributions to the theory of estimation from grouped and partially grouped samples*. Almqvist & Wiksell.

[7] Anders Martin-Löf. 1963. "Utvärdering av livslängder i subnanosekundsområdet" ("Evaluation of sub-nanosecond lifetimes"). ("Sundberg formula")

[8] Per Martin-Löf. 1966. *Statistics from the point of view of statistical mechanics*. Lecture notes, Mathematical Institute, Aarhus University. ("Sundberg formula" credited to Anders Martin-Löf).

[9] Per Martin-Löf. 1970. *Statistika Modeller (Statistical Models): Anteckningar från seminarier läsåret 1969–1970 (Notes from seminars in the academic year 1969-1970), with the assistance of Rolf Sundberg.* Stockholm University. ("Sundberg formula")

[10] Martin-Löf, P. The notion of redundancy and its use as a quantitative measure of the deviation between a statistical hypothesis and a set of observational data. With a discussion by F. Abildgård, A. P. Dempster, D. Basu, D. R. Cox, A. W. F. Edwards, D. A. Sprott, G. A. Barnard, O. Barndorff-Nielsen, J. D. Kalbfleisch and G. Rasch and a reply by the author. *Proceedings of Conference on Foundational Questions in Statistical Inference* (Aarhus, 1973), pp. 1–42. Memoirs, No. 1, Dept. Theoret. Statist., Inst. Math., Univ. Aarhus, Aarhus, 1974.

[11] Martin-Löf, Per The notion of redundancy and its use as a quantitative measure of the discrepancy between a statistical hypothesis and a set of observational data. *Scand. J. Statist.* 1 (1974), no. 1, 3–18.

[12] Wu, C. F. Jeff (1983). "On the Convergence Properties of the EM Algorithm". *The Annals of Statistics* (Institute of Mathematical Statistics) **11** (1): 95–103. doi:10.1214/aos/1176346060. Retrieved 11 December 2014.

[13] Wu, C. F. Jeff (Mar 1983). "On the Convergence Properties of the EM Algorithm". *Annals of Statistics* **11** (1): 95–103. doi:10.1214/aos/1176346060. JSTOR 2240463. MR 684867.

[14] Little, Roderick J.A.; Rubin, Donald B. (1987). *Statistical Analysis with Missing Data*. Wiley Series in Probability and Mathematical Statistics. New York: John Wiley & Sons. pp. 134–136. ISBN 0-471-80254-9.

[15] Neal, Radford; Hinton, Geoffrey (1999). Michael I. Jordan, ed. "A view of the EM algorithm that justifies incremental, sparse, and other variants" (PDF). *Learning in Graphical Models* (Cambridge, MA: MIT Press): 355–368. ISBN 0-262-60032-3. Retrieved 2009-03-22.

[16] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2001). "8.5 The EM algorithm". *The Elements of Statistical Learning*. New York: Springer. pp. 236–243. ISBN 0-387-95284-5.

[17] Einicke, G.A.; Malos, J.T.; Reid, D.C.; Hainsworth, D.W. (January 2009). "Riccati Equation and EM Algorithm Convergence for Inertial Navigation Alignment". *IEEE Trans. Signal Processing* **57** (1): 370–375. doi:10.1109/TSP.2008.2007090.

[18] Einicke, G.A.; Falco, G.; Malos, J.T. (May 2010). "EM Algorithm State Matrix Estimation for Navigation". *IEEE Signal Processing Letters* **17** (5): 437–440. Bibcode:2010ISPL...17..437E. doi:10.1109/LSP.2010.2043151.

[19] Einicke, G.A.; Falco, G.; Dunn, M.T.; Reid, D.C. (May 2012). "Iterative Smoother-Based Variance Estimation". *IEEE Signal Processing Letters* **19** (5): 275–278. Bibcode:2012ISPL...19..275E. doi:10.1109/LSP.2012.2190278.

[20] Jamshidian, Mortaza; Jennrich, Robert I. (1997). "Acceleration of the EM Algorithm by using Quasi-Newton Methods". *Journal of the Royal Statistical Society, Series B* **59** (2): 569–587. doi:10.1111/1467-9868.00083. MR 1452026.

[21] Meng, Xiao-Li; Rubin, Donald B. (1993). "Maximum likelihood estimation via the ECM algorithm: A general framework". *Biometrika* **80** (2): 267–278. doi:10.1093/biomet/80.2.267. MR 1243503.

[22] Jiangtao Yin, Yanfeng Zhang, and Lixin Gao (2012). "Accelerating Expectation-Maximization Algorithms with Frequent Updates" (PDF). *Proceedings of the IEEE International Conference on Cluster Computing*.

[23] Hunter DR and Lange K (2004), A Tutorial on MM Algorithms, The American Statistician, 58: 30-37

[24] Matsuyama, Yasuo (2003). "The α-EM algorithm: Surrogate likelihood maximization using α-logarithmic information measures". *IEEE Transactions on Information Theory* **49** (3): 692–706. doi:10.1109/TIT.2002.808105.

[25] Matsuyama, Yasuo (2011). "Hidden Markov model estimation based on alpha-EM algorithm: Discrete and continuous alpha-HMMs". *International Joint Conference on Neural Networks*: 808–816.

[26] Wolynetz, M.S. (1979). "Maximum likelihood estimation in a linear model from confined and censored normal data". *Journal of the Royal Statistical Society, Series C* **28** (2): 195–206. doi:10.2307/2346749.

[27] Lange, Kenneth. "The MM Algorithm" (PDF).

42.17 External links

- Various 1D, 2D and 3D demonstrations of EM together with Mixture Modeling are provided as part of the paired SOCR activities and applets. These applets and activities show empirically the properties of the EM algorithm for parameter estimation in diverse settings.
- k-MLE: A fast algorithm for learning statistical mixture models
- Class hierarchy in C++ (GPL) including Gaussian Mixtures
- Fast and clean C implementation of the Expectation Maximization (EM) algorithm for estimating Gaussian Mixture Models (GMMs).

Chapter 43

k-means clustering

***k*-means clustering** is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. *k*-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data; however, *k*-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has nothing to do with and should not be confused with *k*-nearest neighbor, another popular machine learning technique.

43.1 Description

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* (≤ *n*) sets $\mathbf{S} = \{S_1, S_2, \ldots, Sk\}$ so as to minimize the within-cluster sum of squares (WCSS). In other words, its objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \left\|\mathbf{x} - \boldsymbol{\mu}_i\right\|^2$$

where $\boldsymbol{\mu}i$ is the mean of points in *Si*.

43.2 History

The term "*k*-means" was first used by James MacQueen in 1967,[1] though the idea goes back to Hugo Steinhaus in 1957.[2] The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published outside of Bell Labs until 1982.[3] In 1965, E.W.Forgy published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy.[4] A more efficient version was proposed and published in Fortran by Hartigan and Wong in 1975/1979.[5][6]

43.3 Algorithms

43.3.1 Standard algorithm

The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the ***k*-means algorithm**; it is also referred to as **Lloyd's algorithm**, particularly in the computer science community.

Given an initial set of *k* means $m_1{}^{(1)}$,…,$mk^{(1)}$ (see below), the algorithm proceeds by alternating between two steps:[7]

Assignment step: Assign each observation to the cluster whose mean yields the least within-cluster sum of squares (WCSS). Since the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean.[8] (Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means).

$$S_i^{(t)} = \left\{x_p : \left\|x_p - m_i^{(t)}\right\|^2 \le \left\|x_p - m_j^{(t)}\right\|^2 \; \forall j, 1 \le j \le k\right\},$$

where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

Update step: Calculate the new means to be the centroids of the observations in the new clusters.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Since the arithmetic mean is a least-squares estimator, this also minimizes the within-cluster sum of squares (WCSS) objective.

The algorithm has converged when the assignments no longer change. Since both steps optimize the WCSS objective, and there only exists a finite number of such partitionings, the algorithm must converge to a (local) optimum. There is no guarantee that the global optimum is found using this algorithm.

The algorithm is often presented as assigning objects to the nearest cluster by distance. The standard algorithm aims at minimizing the WCSS objective, and thus assigns by "least sum of squares", which is exactly equivalent to assigning by the smallest Euclidean distance. Using a different distance function other than (squared) Euclidean distance may stop the algorithm from converging. Various modifications of k-means such as spherical k-means and k-medoids have been proposed to allow using other distance measures.

Initialization methods

Commonly used initialization methods are Forgy and Random Partition.[9] The Forgy method randomly chooses *k* observations from the data set and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points. The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to Hamerly et al.,[9] the Random Partition method is generally preferable for algorithms such as the *k*-harmonic means and fuzzy *k*-means. For expectation maximization and standard *k*-means algorithms, the Forgy method of initialization is preferable.

- Demonstration of the standard algorithm
- 1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).
- 2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.
- 3. The centroid of each of the *k* clusters becomes the new mean.
- 4. Steps 2 and 3 are repeated until convergence has been reached.

As it is a heuristic algorithm, there is no guarantee that it will converge to the global optimum, and the result may depend on the initial clusters. As the algorithm is usually very fast, it is common to run it multiple times with different starting conditions. However, in the worst case, *k*-means can be very slow to converge: in particular it has been shown that there exist certain point sets, even in 2 dimensions, on which *k*-means takes exponential time, that is $2^{\Omega(n)}$, to converge.[10] These point sets do not seem to arise in practice: this is corroborated by the fact that the smoothed running time of *k*-means is polynomial.[11]

The "assignment" step is also referred to as **expectation step**, the "update step" as **maximization step**, making this algorithm a variant of the *generalized* expectation-maximization algorithm.

43.3.2 Complexity

Regarding computational complexity, finding the optimal solution to the *k*-means clustering problem for observations in *d* dimensions is:

- NP-hard in general Euclidean space *d* even for 2 clusters[12][13]
- NP-hard for a general number of clusters *k* even in the plane[14]
- If *k* and *d* (the dimension) are fixed, the problem can be exactly solved in time $O(n^{dk+1}\log n)$, where *n* is the number of entities to be clustered[15]

Thus, a variety of heuristic algorithms such as Lloyds algorithm given above are generally used.

The running time of Lloyds algorithm is often given as $O(nkdi)$, where *n* is the number of *d*-dimensional vectors, *k* the number of clusters and *i* the number of iterations needed until convergence. On data that does have a clustering structure, the number of iterations until convergence is often small, and results only improve slightly after the first dozen iterations. Lloyds algorithm is therefore often considered to be of "linear" complexity in practice.

Following are some recent insights into this algorithm complexity behaviour.

- Lloyd's *k*-means algorithm has polynomial smoothed running time. It is shown that[11] for arbitrary set of *n* points in $[0,1]^d$, if each point is independently perturbed by a normal distribution with mean 0 and variance σ^2 , then the expected running time of k -means algorithm is bounded by $O(n^{34}k^{34}d^8log^4(n)/\sigma^6)$, which is a polynomial in n , k , d and $1/\sigma$.

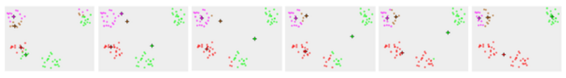
- Better bounds are proved for simple cases. For example,[16] showed that the running time of *k*-means algorithm is bounded by $O(dn^4M^2)$ for n points in an integer lattice $\{1,\dots,M\}^d$.

43.3.3 Variations

- Jenks natural breaks optimization: *k*-means applied to univariate data

- k-medians clustering uses the median in each dimension instead of the mean, and this way minimizes L_1 norm (Taxicab geometry).
- k-medoids (also: Partitioning Around Medoids, PAM) uses the medoid instead of the mean, and this way minimizes the sum of distances for *arbitrary* distance functions.
- Fuzzy C-Means Clustering is a soft version of K-means, where each data point has a fuzzy degree of belonging to each cluster.
- Gaussian mixture models trained with expectation-maximization algorithm (EM algorithm) maintains probabilistic assignments to clusters, instead of deterministic assignments, and multivariate Gaussian distributions instead of means.
- k-means++ chooses initial centers in a way that gives a provable upper bound on the WCCS objective.
- The filtering algorithm uses kd-trees to speed up each k-means step.[17]
- Some methods attempt to speed up each k-means step using coresets[18] or the triangle inequality.[19]
- Escape local optima by swapping points between clusters.[6]
- The Spherical k-means clustering algorithm is suitable for directional data.[20]
- The Minkowski metric weighted k-means deals with irrelevant features by assigning cluster specific weights to each feature[21]

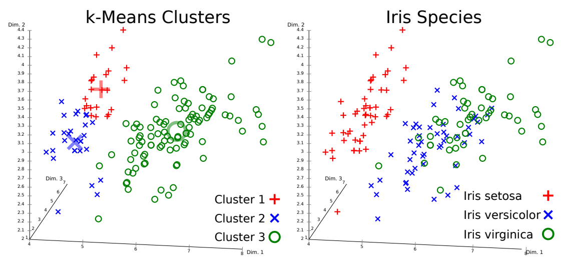
43.4 Discussion



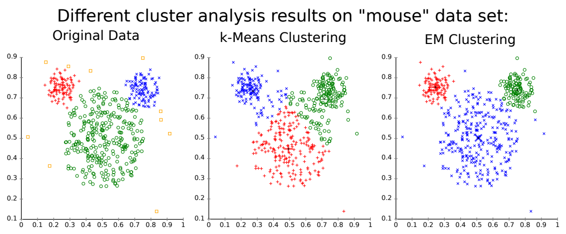
A typical example of the k-means convergence to a local minimum. In this example, the result of k-means clustering (the right figure) contradicts the obvious cluster structure of the data set. The small circles are the data points, the four ray stars are the centroids (means). The initial configuration is on the left figure. The algorithm converges after five iterations presented on the figures, from the left to the right. The illustration was prepared with the Mirkes Java applet.[22]

The two key features of k-means which make it efficient are often regarded as its biggest drawbacks:

- Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.



k-*means clustering result for the Iris flower data set and actual species visualized using ELKI. Cluster means are marked using larger, semi-transparent symbols.*



k-*means clustering and EM clustering on an artificial dataset ("mouse"). The tendency of* k-*means to produce equi-sized clusters leads to bad results, while EM benefits from the Gaussian distribution present in the data set*

- The number of clusters k is an input parameter: an inappropriate choice of k may yield poor results. That is why, when performing k-means, it is important to run diagnostic checks for determining the number of clusters in the data set.
- Convergence to a local minimum may produce counterintuitive ("wrong") results (see example in Fig.).

A key limitation of k-means is its cluster model. The concept is based on spherical clusters that are separable in a way so that the mean value converges towards the cluster center. The clusters are expected to be of similar size, so that the assignment to the nearest cluster center is the correct assignment. When for example applying k-means with a value of $k = 3$ onto the well-known Iris flower data set, the result often fails to separate the three Iris species contained in the data set. With $k = 2$, the two visible clusters (one containing two species) will be discovered, whereas with $k = 3$ one of the two clusters will be split into two even parts. In fact, $k = 2$ is more appropriate for this data set, despite the data set containing 3 *classes*. As with any other clustering algorithm, the k-means result relies on the data set to satisfy the assumptions made by the clustering algorithms. It works well on some data sets, while failing on others.

The result of k-means can also be seen as the Voronoi cells of the cluster means. Since data is split halfway between cluster means, this can lead to suboptimal splits as can be seen in the "mouse" example. The Gaussian models used by the Expectation-maximization algorithm (which can be seen as a generalization of *k*-means) are more flexible here by having both variances and covariances. The EM result is thus able to accommodate clusters of variable size much better than *k*-means as well as correlated clusters (not in this example).

43.5 Applications

k-means clustering in particular when using heuristics such as Lloyd's algorithm is rather easy to implement and apply even on large data sets. As such, it has been successfully used in various topics, including market segmentation, computer vision, geostatistics,[23] astronomy and agriculture. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.

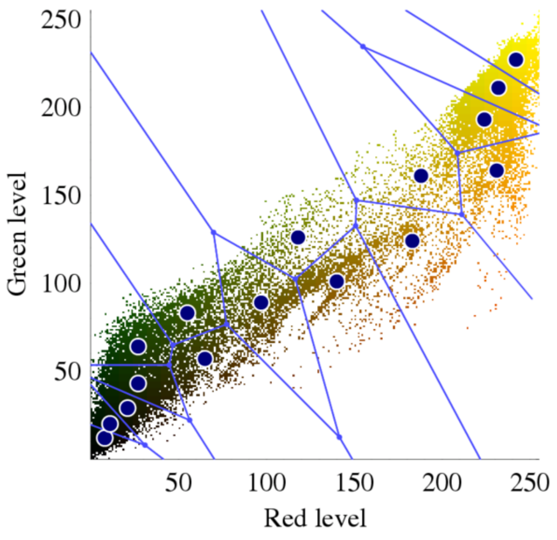
43.5.1 Vector quantization

Main article: Vector quantization

k-means originates from signal processing, and still finds

Two-channel (for illustration purposes -- red and green only) color image.

use in this domain. For example in computer graphics, color quantization is the task of reducing the color palette of an image to a fixed number of colors *k*. The *k*-means algorithm can easily be used for this task and produces competitive results. Other uses of vector quantization include non-random sampling, as *k*-means can easily be used to choose *k* different but prototypical objects from a large data set for further analysis.



Vector quantization of colors present in the image above into Voronoi cells using k*-means.*

43.5.2 Cluster analysis

Main article: Cluster analysis

In cluster analysis, the *k*-means algorithm can be used to partition the input data set into *k* partitions (clusters).

However, the pure *k*-means algorithm is not very flexible, and as such of limited use (except for when vector quantization as above is actually the desired use case!). In particular, the parameter *k* is known to be hard to choose (as discussed above) when not given by external constraints. Another limitation of the algorithm is that it cannot be used with arbitrary distance functions or on non-numerical data. For these use cases, many other algorithms have been developed since.

43.5.3 Feature learning

k-means clustering has been used as a feature learning (or dictionary learning) step, in either (semi-)supervised learning or unsupervised learning.[24] The basic approach is first to train a *k*-means clustering representation, using the input training data (which need not be labelled). Then, to project any input datum into the new feature space, we have a choice of "encoding" functions, but we can use for example the thresholded matrix-product of the datum with the centroid locations, the distance from the datum to each centroid, or simply an indicator function for the nearest centroid,[24][25] or some smooth transformation of the distance.[26] Alternatively, by transforming the sample-cluster distance through a Gaussian RBF, one effectively obtains the hidden layer of a radial basis function network.[27]

This use of *k*-means has been successfully combined with simple, linear classifiers for semi-supervised learning in NLP (specifically for named entity recognition)[28] and in computer vision. On an object recognition task, it was found to exhibit comparable performance with more sophisticated feature learning approaches such as autoencoders and restricted Boltzmann machines.[26] However, it generally requires more data than the sophisticated methods, for equivalent performance, because each data point only contributes to one "feature" rather than multiple.[24]

43.6 Relation to other statistical machine learning algorithms

k-means clustering, and its associated expectation-maximization algorithm, is a special case of a Gaussian mixture model, specifically, the limit of taking all covariances as diagonal, equal, and small. It is often easy to generalize a *k*-means problem into a Gaussian mixture model.[29] Another generalization of the k-means algorithm is the K-SVD algorithm, which estimates data points as a sparse linear combination of "codebook vectors". K-means corresponds to the special case of using a single codebook vector, with a weight of 1.[30]

43.6.1 Mean shift clustering

Basic mean shift clustering algorithms maintain a set of data points the same size as the input data set. Initially, this set is copied from the input set. Then this set is iteratively replaced by the mean of those points in the set that are within a given distance of that point. By contrast, *k*-means restricts this updated set to *k* points usually much less than the number of points in the input data set, and replaces each point in this set by the mean of all points in the *input set* that are closer to that point than any other (e.g. within the Voronoi partition of each updating point). A mean shift algorithm that is similar then to *k*-means, called *likelihood mean shift*, replaces the set of points undergoing replacement by the mean of all points in the input set that are within a given distance of the changing set.[31] One of the advantages of mean shift over *k*-means is that there is no need to choose the number of clusters, because mean shift is likely to find only a few clusters if indeed only a small number exist. However, mean shift can be much slower than *k*-means, and still requires selection of a bandwidth parameter. Mean shift has soft variants much as *k*-means does.

43.6.2 Principal component analysis (PCA)

It was asserted in[32][33] that the relaxed solution of k-means clustering, specified by the cluster indicators, is given by the PCA (principal component analysis) principal components, and the PCA subspace spanned by the principal directions is identical to the cluster centroid subspace. However, that PCA is a useful relaxation of k-means clustering was not a new result (see, for example,[34]), and it is straightforward to uncover counterexamples to the statement that the cluster centroid subspace is spanned by the principal directions.[35]

43.6.3 Independent component analysis (ICA)

It has been shown in [36] that under sparsity assumptions and when input data is pre-processed with the whitening transformation *k*-means produces the solution to the linear Independent component analysis task. This aids in explaining the successful application of *k*-means to feature learning.

43.6.4 Bilateral filtering

k-means implicitly assumes that the ordering of the input data set does not matter. The bilateral filter is similar to K-means and mean shift in that it maintains a set of data points that are iteratively replaced by means. However, the bilateral filter restricts the calculation of the (kernel weighted) mean to include only points that are close in the ordering of the input data.[31] This makes it applicable to problems such as image denoising, where the spatial arrangement of pixels in an image is of critical importance.

43.7 Similar problems

The set of squared error minimizing cluster functions also includes the k-medoids algorithm, an approach which forces the center point of each cluster to be one of the actual points, i.e., it uses medoids in place of centroids.

43.8 Software Implementations

43.8.1 Free

- CrimeStat implements two spatial *k*-means algorithms, one of which allows the user to define the starting locations.
- ELKI contains *k*-means (with Lloyd and MacQueen iteration, along with different initializations such as *k*-means++ initialization) and various more advanced clustering algorithms.
- Julia contains a *k*-means implementation in the Clustering package.[37]
- Mahout contains a MapReduce based *k*-means.

- MLPACK contains a C++ implementation of *k*-means.
- Octave contains *k*-means.
- OpenCV contains a *k*-means implementation.
- R contains three *k*-means variations.[1][3][6]
- SciPy and scikit-learn contain multiple *k*-means implementations.
- Spark MLlib implements a distributed *k*-means algorithm.
- Torch contains an *unsup* package that provides *k*-means clustering.
- Weka contains *k*-means and *x*-means.

43.8.2 Commercial

- Grapheme
- MATLAB
- Mathematica
- SAS
- Stata

43.9 See also

- Canopy clustering algorithm
- Centroidal Voronoi tessellation
- k q-flats
- Linde–Buzo–Gray algorithm
- Nearest centroid classifier
- Self-organizing map
- Silhouette clustering
- Head/tail Breaks

43.10 References

[1] MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations*. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability **1**. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.

[2] Steinhaus, H. (1957). “Sur la division des corps matériels en parties”. *Bull. Acad. Polon. Sci.* (in French) **4** (12): 801–804. MR 0090073. Zbl 0079.16403.

[3] Lloyd, S. P. (1957). “Least square quantization in PCM”. *Bell Telephone Laboratories Paper*. Published in journal much later: Lloyd., S. P. (1982). “Least squares quantization in PCM” (PDF). *IEEE Transactions on Information Theory* **28** (2): 129–137. doi:10.1109/TIT.1982.1056489. Retrieved 2009-04-15.

[4] E.W. Forgy (1965). “Cluster analysis of multivariate data: efficiency versus interpretability of classifications”. *Biometrics* **21**: 768–769.

[5] J.A. Hartigan (1975). *Clustering algorithms*. John Wiley & Sons, Inc.

[6] Hartigan, J. A.; Wong, M. A. (1979). “Algorithm AS 136: A K-Means Clustering Algorithm”. *Journal of the Royal Statistical Society, Series C* **28** (1): 100–108. JSTOR 2346830.

[7] MacKay, David (2003). “Chapter 20. An Example Inference Task: Clustering” (PDF). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. pp. 284–292. ISBN 0-521-64298-1. MR 2012999.

[8] Since the square root is a monotone function, this also is the minimum Euclidean distance assignment.

[9] Hamerly, G. and Elkan, C. (2002). “Alternatives to the k-means algorithm that find better clusterings” (PDF). *Proceedings of the eleventh international conference on Information and knowledge management (CIKM)*.

[10] Vattani., A. (2011). “k-means requires exponentially many iterations even in the plane” (PDF). *Discrete and Computational Geometry* **45** (4): 596–616. doi:10.1007/s00454-011-9340-1.

[11] Arthur, D.; Manthey, B.; Roeglin, H. (2009). “k-means has polynomial smoothed complexity”. *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*.

[12] Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. (2009). “NP-hardness of Euclidean sum-of-squares clustering”. *Machine Learning* **75**: 245–249. doi:10.1007/s10994-009-5103-0.

[13] Dasgupta, S. and Freund, Y. (July 2009). “Random Projection Trees for Vector Quantization”. *Information Theory, IEEE Transactions on* **55**: 3229–3242. arXiv:0805.1390. doi:10.1109/TIT.2009.2021326.

[14] Mahajan, M.; Nimbhorkar, P.; Varadarajan, K. (2009). “The Planar k-Means Problem is NP-Hard”. *Lecture Notes in Computer Science* **5431**: 274–285. doi:10.1007/978-3-642-00202-1_24.

[15] Inaba, M.; Katoh, N.; Imai, H. (1994). *Applications of weighted Voronoi diagrams and randomization to variance-based* k*-clustering*. Proceedings of 10th ACM Symposium on Computational Geometry. pp. 332–339. doi:10.1145/177424.178042.

[16] Arthur; Abhishek Bhowmick (2009). *A theoretical analysis of Lloyd’s algorithm for k-means clustering* (Thesis).

[17] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; Piatko, C. D.; Silverman, R.; Wu, A. Y. (2002). "An efficient k-means clustering algorithm: Analysis and implementation" (PDF). *IEEE Trans. Pattern Analysis and Machine Intelligence* **24**: 881–892. doi:10.1109/TPAMI.2002.1017616. Retrieved 2009-04-24.

[18] Frahling, G.; Sohler, C. (2006). *A fast k-means implementation using coresets* (PDF). Proceedings of the twenty-second annual symposium on Computational geometry (SoCG).

[19] Elkan, C. (2003). "Using the triangle inequality to accelerate k-means" (PDF). *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*.

[20] Dhillon, I. S.; Modha, D. M. (2001). "Concept decompositions for large sparse text data using clustering". *Machine Learning* **42** (1): 143–175. doi:10.1023/a:1007612920971.

[21] Amorim, R. C.; Mirkin, B (2012). "Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering". *Pattern Recognition* **45** (3): 1061–1075. doi:10.1016/j.patcog.2011.08.012.

[22] Mirkes, E.M. "K-means and K-medoids applet.". *http://www.math.le.ac.uk". Retrieved 1 May 2015.*

[23] Honarkhah, M; Caers, J (2010). "Stochastic Simulation of Patterns Using Distance-Based Pattern Modeling". *Mathematical Geosciences* **42**: 487–517. doi:10.1007/s11004-010-9276-7.

[24] Coates, Adam; Ng, Andrew Y. (2012). "Learning feature representations with k-means" (PDF). In G. Montavon, G. B. Orr, K.-R. Müller. *Neural Networks: Tricks of the Trade*. Springer.

[25] Csurka, Gabriella; Dance, Christopher C.; Fan, Lixin; Willamowski, Jutta; Bray, Cédric (2004). *Visual categorization with bags of keypoints* (PDF). ECCV Workshop on Statistical Learning in Computer Vision.

[26] Coates, Adam; Lee, Honglak; Ng, Andrew Y. (2011). *An analysis of single-layer networks in unsupervised feature learning* (PDF). International Conference on Artificial Intelligence and Statistics (AISTATS).

[27] Schwenker, Friedhelm; Kestler, Hans A.; Palm, Günther (2001). "Three learning phases for radial-basis-function networks". *Neural Networks* **14**: 439–458. doi:10.1016/s0893-6080(01)00027-2. CiteSeerX: 10.1.1.109.312.

[28] Lin, Dekang; Wu, Xiaoyun (2009). *Phrase clustering for discriminative learning* (PDF). Annual Meeting of the ACL and IJCNLP. pp. 1030–1038.

[29] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 16.1. Gaussian Mixture Models and k-Means Clustering". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

[30] Aharon, Michal; Elad, Michael; Bruckstein, Alfred (2006). "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation" (PDF).

[31] Little, M.A.; Jones, N.S. (2011). "Generalized Methods and Solvers for Piecewise Constant Signals: Part I" (PDF). *Proceedings of the Royal Society A* **467**: 3088–3114. doi:10.1098/rspa.2010.0671.

[32] H. Zha, C. Ding, M. Gu, X. He and H.D. Simon (Dec 2001). "Spectral Relaxation for K-means Clustering" (PDF). *Neural Information Processing Systems vol.14 (NIPS 2001)* (Vancouver, Canada): 1057–1064.

[33] Chris Ding and Xiaofeng He (July 2004). "K-means Clustering via Principal Component Analysis" (PDF). *Proc. of Int'l Conf. Machine Learning (ICML 2004)*: 225–232.

[34] Drineas, P.; A. Frieze; R. Kannan; S. Vempala; V. Vinay (2004). "Clustering large graphs via the singular value decomposition" (PDF). *Machine learning* **56**: 9–33. doi:10.1023/b:mach.0000033113.59016.96. Retrieved 2012-08-02.

[35] Cohen, M.; S. Elder; C. Musco; C. Musco; M. Persu (2014). "Dimensionality reduction for k-means clustering and low rank approximation (Appendix B)". *ArXiv*. Retrieved 2014-11-29.

[36] Alon Vinnikov and Shai Shalev-Shwartz (2014). "K-means Recovers ICA Filters when Independent Components are Sparse" (PDF). *Proc. of Int'l Conf. Machine Learning (ICML 2014)*.

[37] Clustering.jl www.github.com

Chapter 44

Hierarchical clustering

In data mining and statistics, **hierarchical clustering** (also called **hierarchical cluster analysis** or **HCA**) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types: [1]

- **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram.

In the general case, the complexity of agglomerative clustering is $O(n^3)$, which makes them too slow for large data sets. Divisive clustering with an exhaustive search is $O(2^n)$, which is even worse. However, for some special cases, optimal efficient agglomerative methods (of complexity $O(n^2)$) are known: SLINK[2] for single-linkage and CLINK[3] for complete-linkage clustering.

44.1 Cluster dissimilarity

In order to decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, this is achieved by use of an appropriate metric (a measure of distance between pairs of observations), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.

44.1.1 Metric

Further information: metric (mathematics)

The choice of an appropriate metric will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. For example, in a 2-dimensional space, the distance between the point (1,0) and the origin (0,0) is always 1 according to the usual norms, but the distance between the point (1,1) and the origin (0,0) can be 2 under Manhattan distance, $\sqrt{2}$ under Euclidean distance, or 1 under maximum distance.

Some commonly used metrics for hierarchical clustering are:[4]

For text or other non-numeric data, metrics such as the Hamming distance or Levenshtein distance are often used.

A review of cluster analysis in health psychology research found that the most common distance measure in published studies in that research area is the Euclidean distance or the squared Euclidean distance.

44.1.2 Linkage criteria

The linkage criterion determines the distance between sets of observations as a function of the pairwise distances between observations.

Some commonly used linkage criteria between two sets of observations *A* and *B* are:[5][6]

where *d* is the chosen metric. Other linkage criteria include:

- The sum of all intra-cluster variance.
- The decrease in variance for the cluster being merged (Ward's criterion).[7]
- The probability that candidate clusters spawn from the same distribution function (V-linkage).
- The product of in-degree and out-degree on a k-nearest-neighbor graph (graph degree linkage).[8]
- The increment of some cluster descriptor (i.e., a quantity defined for measuring the quality of a cluster) after merging two clusters.[9][10][11]

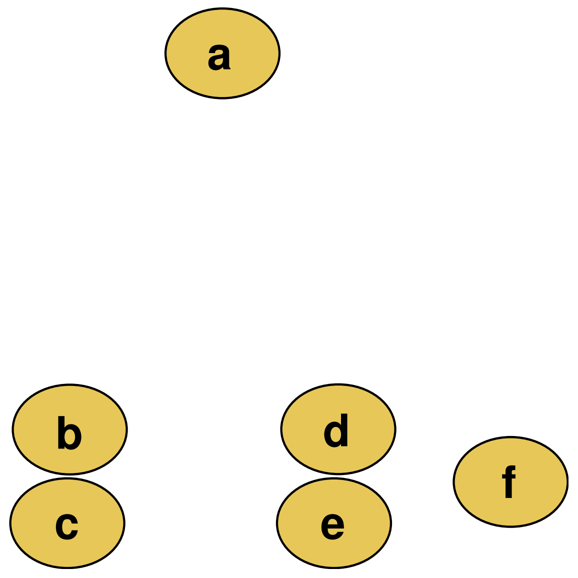
44.2 Discussion

Hierarchical clustering has the distinct advantage that any valid measure of distance can be used. In fact, the observations themselves are not required: all that is used is a matrix of distances.

44.3 Example for Agglomerative Clustering

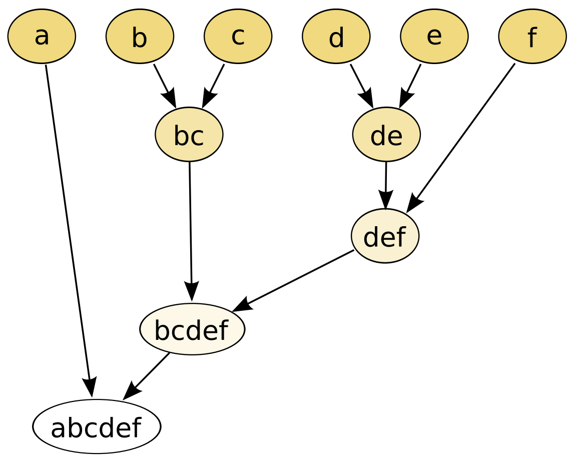
For example, suppose this data is to be clustered, and the Euclidean distance is the distance metric.

Cutting the tree at a given height will give a partitioning clustering at a selected precision. In this example, cutting after the second row of the dendrogram will yield clusters {a} {b c} {d e} {f}. Cutting after the third row will yield clusters {a} {b c} {d e f}, which is a coarser clustering, with a smaller number but larger clusters.



Raw data

The hierarchical clustering dendrogram would be as such:



Traditional representation

This method builds the hierarchy from the individual elements by progressively merging clusters. In our example, we have six elements {a} {b} {c} {d} {e} and {f}. The first step is to determine which elements to merge in a cluster. Usually, we want to take the two closest elements, according to the chosen distance.

Optionally, one can also construct a distance matrix at this stage, where the number in the *i*-th row *j*-th column is the distance between the *i*-th and *j*-th elements. Then, as clustering progresses, rows and columns are merged as the clusters are merged and the distances updated. This is a common way to implement this type of clustering, and has the benefit of caching distances between clusters. A simple agglomerative clustering algorithm is described in the single-linkage clustering page; it can easily be adapted to different types of linkage (see below).

Suppose we have merged the two closest elements b and c, we now have the following clusters $\{a\}$, $\{b, c\}$, $\{d\}$, $\{e\}$ and $\{f\}$, and want to merge them further. To do that, we need to take the distance between {a} and {b c}, and therefore define the distance between two clusters. Usually the distance between two clusters $\mathcal{A}$ and $\mathcal{B}$ is one of the following:

- The maximum distance between elements of each cluster (also called complete-linkage clustering):

$$\max\{\, d(x,y) : x \in \mathcal{A},\, y \in \mathcal{B}\,\}.$$

- The minimum distance between elements of each cluster (also called single-linkage clustering):

$$\min\{\, d(x,y) : x \in \mathcal{A},\, y \in \mathcal{B}\,\}.$$

- The mean distance between elements of each cluster (also called average linkage clustering, used e.g. in UPGMA):

$$\frac{1}{|\mathcal{A}|\cdot|\mathcal{B}|}\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} d(x,y).$$

- The sum of all intra-cluster variance.
- The increase in variance for the cluster being merged (Ward's method<ref name="[7])
- The probability that candidate clusters spawn from the same distribution function (V-linkage).

Each agglomeration occurs at a greater distance between clusters than the previous agglomeration, and one can decide to stop clustering either when the clusters are too far apart to be merged (distance criterion) or when there is a sufficiently small number of clusters (number criterion).

44.4 Software

44.4.1 Open Source Frameworks

- R has several functions for hierarchical clustering: see CRAN Task View: Cluster Analysis & Finite Mixture Models for more information.
- Cluster 3.0 provides a nice Graphical User Interface to access to different clustering routines and is available for Windows, Mac OS X, Linux, Unix.
- ELKI includes multiple hierarchical clustering algorithms, various linkage strategies and also includes the efficient SLINK[2] algorithm, flexible cluster extraction from dendrograms and various other cluster analysis algorithms.
- Octave, the GNU analog to MATLAB implements hierarchical clustering in linkage function
- Orange, a free data mining software suite, module orngClustering for scripting in Python, or cluster analysis through visual programming.
- scikit-learn implements a hierarchical clustering.
- Weka includes hierarchical cluster analysis.
- fastCluster efficiently implements the seven most widely used clustering schemes.
- SCaViS computing environment in Java that implements this algorithm.

44.4.2 Standalone implementations

- CrimeStat implements two hierarchical clustering routines, a nearest neighbor (Nnh) and a risk-adjusted(Rnnh).
- figue is a JavaScript package that implements some agglomerative clustering functions (single-linkage, complete-linkage, average-linkage) and functions to visualize clustering output (e.g. dendrograms).
- hcluster is a Python implementation, based on NumPy, which supports hierarchical clustering and plotting.
- Hierarchical Agglomerative Clustering implemented as C# visual studio project that includes real text files processing, building of document-term matrix with stop words filtering and stemming.
- MultiDendrograms An open source Java application for variable-group agglomerative hierarchical clustering, with graphical user interface.
- Graph Agglomerative Clustering (GAC) toolbox implemented several graph-based agglomerative clustering algorithms.
- Hierarchical Clustering Explorer provides tools for interactive exploration of multidimensional data.

44.4.3 Commercial

- MATLAB includes hierarchical cluster analysis.
- SAS includes hierarchical cluster analysis.
- Mathematica includes a Hierarchical Clustering Package.
- NCSS (statistical software) includes hierarchical cluster analysis.
- SPSS includes hierarchical cluster analysis.
- Qlucore Omics Explorer includes hierarchical cluster analysis.
- Stata includes hierarchical cluster analysis.

44.5 See also

- Statistical distance
- Brown clustering
- Cluster analysis
- CURE data clustering algorithm
- Dendrogram
- Determining the number of clusters in a data set
- Hierarchical clustering of networks
- Nearest-neighbor chain algorithm
- Numerical taxonomy
- OPTICS algorithm
- Nearest neighbor search
- Locality-sensitive hashing

44.6 Notes

[1] Rokach, Lior, and Oded Maimon. "Clustering methods." Data mining and knowledge discovery handbook. Springer US, 2005. 321-352.

[2] R. Sibson (1973). "SLINK: an optimally efficient algorithm for the single-link cluster method" (PDF). *The Computer Journal* (British Computer Society) **16** (1): 30–34. doi:10.1093/comjnl/16.1.30.

[3] D. Defays (1977). "An efficient algorithm for a complete link method". *The Computer Journal* (British Computer Society) **20** (4): 364–366. doi:10.1093/comjnl/20.4.364.

[4] "The DISTANCE Procedure: Proximity Measures". *SAS/STAT 9.2 Users Guide*. SAS Institute. Retrieved 2009-04-26.

[5] "The CLUSTER Procedure: Clustering Methods". *SAS/STAT 9.2 Users Guide*. SAS Institute. Retrieved 2009-04-26.

[6] Székely, G. J. and Rizzo, M. L. (2005) Hierarchical clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method, Journal of Classification 22, 151-183.

[7] Ward, Joe H. (1963). "Hierarchical Grouping to Optimize an Objective Function". *Journal of the American Statistical Association* **58** (301): 236–244. doi:10.2307/2282967. JSTOR 2282967. MR 0148188.

[8] Zhang, et al. "Graph degree linkage: Agglomerative clustering on a directed graph." 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012. http://arxiv.org/abs/1208.5092

[9] Zhang, et al. "Agglomerative clustering via maximum incremental path integral." Pattern Recognition (2013).

[10] Zhao, and Tang. "Cyclizing clusters via zeta function of a graph."Advances in Neural Information Processing Systems. 2008.

[11] Ma, et al. "Segmentation of multivariate mixed data via lossy data coding and compression." IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(9) (2007): 1546-1562.

44.7 References and further reading

- Kaufman, L.; Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis* (1 ed.). New York: John Wiley. ISBN 0-471-87876-6.
- Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009). "14.3.12 Hierarchical clustering". *The Elements of Statistical Learning* (PDF) (2nd ed.). New York: Springer. pp. 520–528. ISBN 0-387-84857-6. Retrieved 2009-10-20.
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 16.4. Hierarchical Clustering by Phylogenetic Trees". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.
- Hierarchical Cluster Analysis
- Free statistical software. An overview of statistical software and methods used in published microbiological studies

Chapter 45

Instance-based learning

In machine learning, **instance-based learning** (sometimes called **memory-based learning**[1]) is a family of learning algorithms that, instead of performing explicit generalization, compares new problem instances with instances seen in training, which have been stored in memory. Instance-based learning is a kind of lazy learning.

It is called instance-based because it constructs hypotheses directly from the training instances themselves.[2] This means that the hypothesis complexity can grow with the data:[2] in the worst case, a hypothesis is a list of n training items and the computational complexity of classifying a single new instance is $O(n)$. One advantage that instance-based learning has over other methods of machine learning is its ability to adapt its model to previously unseen data: instance-based learners may simply store a new instance or throw an old instance away.

Examples of instance-based learning algorithm are the k-nearest neighbor algorithm, kernel machines and RBF networks.[3]:ch. 8 These store (a subset of) their training set; when predicting a value/class for a new instance, they compute distances or similarities between this instance and the training instances to make a decision.

To battle the memory complexity of storing all training instances, as well as the risk of overfitting to noise in the training set, *instance reduction* algorithms have been proposed.[4]

Gagliardi[5] applies this family of classifiers in medical field as second-opinion diagnostic tools and as tools for the knowledge extraction phase in the process of knowledge discovery in databases. One of these classifiers (called *Prototype exemplar learning classifier* (PEL-C) is able to extract a mixture of abstracted prototypical cases (that are syndromes) and selected atypical clinical cases.

45.1 See also

- Analogical modeling

45.2 References

[1] Walter Daelemans; Antal van den Bosch (2005). *Memory-Based Language Processing*. Cambridge University Press.

[2] Stuart Russell and Peter Norvig (2003). *Artificial Intelligence: A Modern Approach*, second edition, p. 733. Prentice Hall. ISBN 0-13-080302-2

[3] Tom Mitchell (1997). *Machine Learning*. McGraw-Hill.

[4] D. Randall Wilson; Tony R. Martinez (2000). "Reduction techniques for instance-based learning algorithms". *Machine Learning* (Kluwer).

[5] Gagliardi, F (2011). "Instance-based classifiers applied to medical databases: Diagnosis and knowledge extraction". *Artificial Intelligence in Medicine* **52** (3): 123–139. doi:10.1016/j.artmed.2011.04.002.

Chapter 46

k-nearest neighbors algorithm

In pattern recognition, the ***k*-Nearest Neighbors algorithm** (or ***k*-NN** for short) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

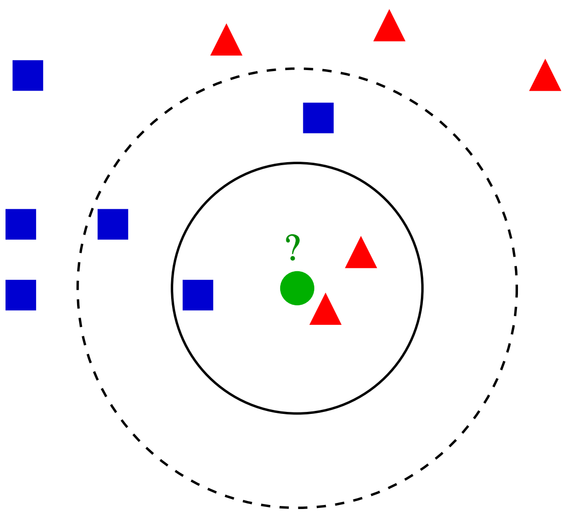
k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A shortcoming of the *k*-NN algorithm is that it is sensitive to the local structure of the data. The algorithm has nothing to do with and is not to be confused with *k*-means, another popular machine learning technique.

46.1 Algorithm



Example of k*-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If* k = 3 *(solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If* k = 5 *(dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).*

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or Hamming distance). In the context of gene expression microarray data, for example, *k*-NN has also been employed with correlation coefficients such as Pearson and Spearman.[3] Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number.[4] One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. K-NN can then be applied to the SOM.

46.2 Parameter selection

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification,[5] but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques (see hyperparameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when $k = 1$) is called the nearest neighbor algorithm.

The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling.[6] Another popular approach is to scale features by the mutual information of the training data with the training classes.

In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal k in this setting is via bootstrap method.[7]

46.3 Properties

k-NN is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel.[8] [9]

The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets. Using an appropriate nearest neighbor search algorithm makes k-NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed.

k-NN has some strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).[10] k-NN is guaranteed to approach the Bayes error rate for some value of k (where k increases as a function of the number of data points). Various improvements to k-NN are possible by using proximity graphs.[11]

46.4 Metric Learning

The K-nearest neighbor classification performance can often be significantly improved through (supervised) metric learning. Popular algorithms are Neighbourhood components analysis and Large margin nearest neighbor. Supervised metric learning algorithms use the label information to learn a new metric or pseudo-metric.

46.5 Feature extraction

When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying k-NN algorithm on the transformed data in feature space.

An example of a typical computer vision computation pipeline for face recognition using k-NN including feature extraction and dimension reduction pre-processing steps (usually implemented with OpenCV):

1. Haar face detection
2. Mean-shift tracking analysis
3. PCA or Fisher LDA projection into feature space, followed by k-NN classification

46.6 Dimension reduction

For high-dimensional data (e.g., with number of dimensions more than 10) dimension reduction is usually performed prior to applying the k-NN algorithm in order to avoid the effects of the curse of dimensionality. [12]

The curse of dimensionality in the k-NN context basically means that Euclidean distance is unhelpful in high dimensions because all vectors are almost equidistant to the search query vector (imagine multiple points lying more or less on a circle with the query point at the center; the distance from the query to all data points in the search space is almost the same).

Feature extraction and dimension reduction can be combined in one step using principal component analysis (PCA), linear discriminant analysis (LDA), or canonical correlation analysis (CCA) techniques as a pre-processing step, followed by clustering by k-NN on feature vectors in reduced-dimension space. In machine learning this process is also called low-dimensional embedding.[13]

For very-high-dimensional datasets (e.g. when performing a similarity search on live video streams, DNA data or high-dimensional time series) running a fast **approximate** k-NN search using locality sensitive hashing, "random projections",[14] "sketches" [15] or other high-dimensional similarity search techniques from VLDB toolbox might be the only feasible option.

46.7 Decision boundary

Nearest neighbor rules in effect implicitly compute the decision boundary. It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity.[16]

46.8 Data reduction

Data reduction is one of the most important problems for work with huge data sets. Usually, only some of the data points are needed for accurate classification. Those data are called the *prototypes* and can be found as follows:

1. Select the *class-outliers*, that is, training data that are classified incorrectly by k-NN (for a given k)
2. Separate the rest of the data into two sets: (i) the prototypes that are used for the classification decisions and (ii) the *absorbed points* that can be correctly classified by k-NN using prototypes. The absorbed points can then be removed from the training set.

46.8.1 Selection of class-outliers

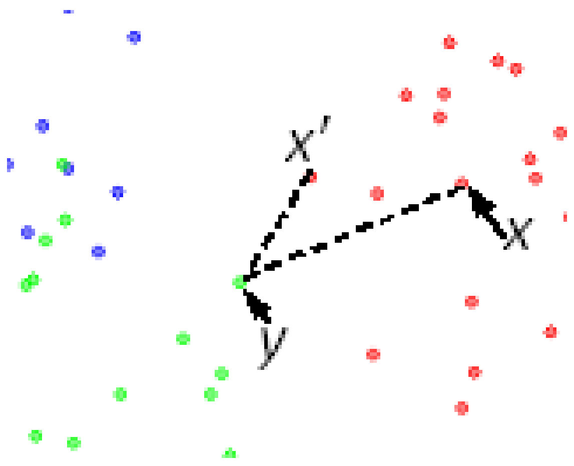
A training example surrounded by examples of other classes is called a class outlier. Causes of class outliers include:

- random error
- insufficient training examples of this class (an isolated example appears instead of a cluster)
- missing important features (the classes are separated in other dimensions which we do not know)
- too many training examples of other classes (unbalanced classes) that create a "hostile" background for the given small class

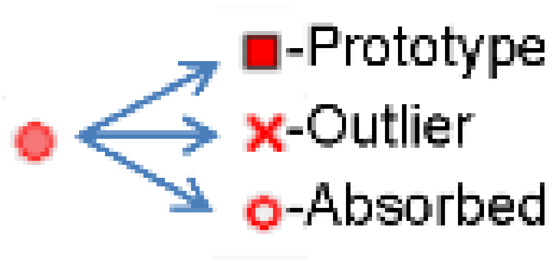
Class outliers with k-NN produce noise. They can be detected and separated for future analysis. Given two natural numbers, $k>r>0$, a training example is called a (k,r)NN class-outlier if its k nearest neighbors include more than r examples of other classes.

46.8.2 CNN for data reduction

Condensed nearest neighbor (CNN, the *Hart algorithm*) is an algorithm designed to reduce the data set for k-NN classification.[17] It selects the set of prototypes U from the training data, such that 1NN with U can classify the examples almost as accurately as 1NN does with the whole data set.



Calculation of the border ratio.



Three types of points: prototypes, class-outliers, and absorbed points.

Given a training set X, CNN works iteratively:

1. Scan all elements of X, looking for an element x whose nearest prototype from U has a different label than x.
2. Remove x from X and add it to U
3. Repeat the scan until no more prototypes are added to U.

Use U instead of X for classification. The examples that are not prototypes are called “absorbed” points.

It is efficient to scan the training examples in order of decreasing border ratio.[18] The border ratio of a training example x is defined as

$$a(x) = \|x'-y\| / \|x-y\|$$

where $\|x-y\|$ is the distance to the closest example y having a different color than x, and $\|x'-y\|$ is the distance from y to its closest example x' with the same label as x.

The border ratio is in the interval [0,1] because $\|x'-y\|$ never exceeds $\|x-y\|$. This ordering gives preference to the borders of the classes for inclusion in the set of prototypes$U$. A point of a different label than x is called external to x. The calculation of the border ratio is illustrated by the figure on the right. The data points are labeled by colors: the initial point is x and its label is red. External points are blue and green. The closest to x external point is y. The closest to y red point is x' . The border ratio $a(x)=\|x'-y\|/\|x-y\|$ is the attribute of the initial point x.

Below is an illustration of CNN in a series of figures. There are three classes (red, green and blue). Fig. 1: initially there are 60 points in each class. Fig. 2 shows the 1NN classification map: each pixel is classified by 1NN using all the data. Fig. 3 shows the 5NN classification map. White areas correspond to the unclassified regions, where 5NN voting is tied (for example, if there are two green, two red and one blue points among 5 nearest neighbors). Fig. 4 shows the reduced data set. The crosses are the class-outliers selected by the (3,2)NN rule (all the three nearest neighbors of these instances belong to other classes); the squares are the prototypes, and the empty circles are the absorbed points. The left bottom corner shows the numbers of the class-outliers, prototypes and absorbed points for all three classes. The number of prototypes varies from 15% to 20% for different classes in this example. Fig. 5 shows that the 1NN classification map with the prototypes is very similar to that with the initial data set. The figures were produced using the Mirkes applet.[18]

- CNN model reduction for k-NN classifiers
- Fig. 1. The dataset.
- Fig. 2. The 1NN classification map.
- Fig. 3. The 5NN classification map.
- Fig. 4. The CNN reduced dataset.
- Fig. 5. The 1NN classification map based on the CNN extracted prototypes.

46.9 *k*-NN regression

In k-NN regression, the k-NN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the k nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

1. Compute the Euclidean or Mahalanobis distance from the query example to the labeled examples.
2. Order the labeled examples by increasing distance.
3. Find a heuristically optimal number k of nearest neighbors, based on RMSE. This is done using cross validation.
4. Calculate an inverse distance weighted average with the k-nearest multivariate neighbors.

46.10 Validation of results

A confusion matrix or “matching matrix” is often used as a tool to validate the accuracy of k-NN classification. More robust statistical methods such as likelihood-ratio test can also be applied.

46.11 See also

- Instance-based learning
- Nearest neighbor search
- Statistical classification
- Cluster analysis
- Data mining
- Nearest centroid classifier
- Pattern recognition
- Curse of dimensionality
- Dimension reduction
- Principal Component Analysis
- Locality Sensitive Hashing
- MinHash
- Cluster hypothesis
- Closest pair of points problem

46.12 References

[1] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". *The American Statistician* **46** (3): 175–185. doi:10.1080/00031305.1992.10475879.

[2] This scheme is a generalization of linear interpolation.

[3] Jaskowiak, P. A.; Campello, R. J. G. B. "Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data". *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.208.993". Brazilian Symposium on Bioinformatics (BSB 2011). pp. 1–8. Retrieved 16 October 2014.*

[4] D. Coomans; D.L. Massart (1982). "Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-Nearest neighbour classification by using alternative voting rules". *Analytica Chimica Acta* **136**: 15–27. doi:10.1016/S0003-2670(01)95359-0.

[5] Everitt, B. S., Landau, S., Leese, M. and Stahl, D. (2011) Miscellaneous Clustering Methods, in Cluster Analysis, 5th Edition, John Wiley & Sons, Ltd, Chichester, UK.

[6] Nigsch F, Bender A, van Buuren B, Tissen J, Nigsch E, Mitchell JB (2006). "Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization". *Journal of Chemical Information and Modeling* **46** (6): 2412–2422. doi:10.1021/ci060149f. PMID 17125183.

[7] Hall P, Park BU, Samworth RJ (2008). "Choice of neighbor order in nearest-neighbor classification". *Annals of Statistics* **36** (5): 2135–2152. doi:10.1214/07-AOS537.

[8] D. G. Terrell; D. W. Scott (1992). "Variable kernel density estimation". *Annals of Statistics* **20** (3): 1236–1265. doi:10.1214/aos/1176348768.

[9] Mills, Peter. "Efficient statistical classification of satellite measurements". *International Journal of Remote Sensing*.

[10] Cover TM, Hart PE (1967). "Nearest neighbor pattern classification". *IEEE Transactions on Information Theory* **13** (1): 21–27. doi:10.1109/TIT.1967.1053964.

[11] Toussaint GT (April 2005). "Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining". *International Journal of Computational Geometry and Applications* **15** (2): 101–150. doi:10.1142/S0218195905001622.

[12] Beyer, Kevin, et al.. 'When is "nearest neighbor" meaningful?. Database Theory—ICDT'99, 217-235|year 1999

[13] Shaw, Blake, and Tony Jebara. 'Structure preserving embedding. Proceedings of the 26th Annual International Conference on Machine Learning. ACM,2009

[14] Bingham, Ella, and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM | year 2001

[15] Shasha, D High Performance Discovery in Time Series.Berlin: Springer, 2004, ISBN 0-387-00857-8

[16] Bremner D, Demaine E, Erickson J, Iacono J, Langerman S, Morin P, Toussaint G (2005). "Output-sensitive algorithms for computing nearest-neighbor decision boundaries". *Discrete and Computational Geometry* **33** (4): 593–604. doi:10.1007/s00454-004-1152-0.

[17] P. E. Hart, The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory 18 (1968) 515–516. doi: 10.1109/TIT.1968.1054155

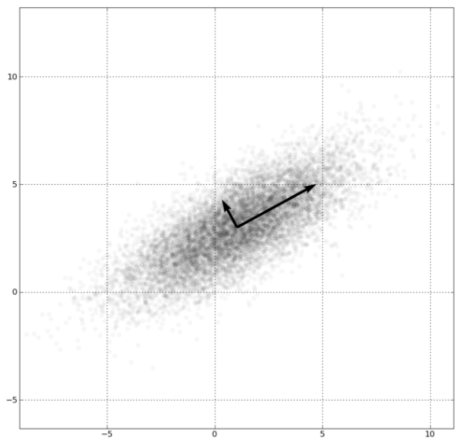
[18] E. M. Mirkes, KNN and Potential Energy: applet. University of Leicester, 2011.

46.13 Further reading

- When Is "Nearest Neighbor" Meaningful?
- Belur V. Dasarathy, ed. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. ISBN 0-8186-8930-7.
- Shakhnarovish, Darrell, and Indyk, ed. (2005). *Nearest-Neighbor Methods in Learning and Vision*. MIT Press. ISBN 0-262-19547-X.
- Mäkelä H Pekkarinen A (2004-07-26). "Estimation of forest stand volumes by Landsat TM imagery and stand-level field-inventory data". *Forest Ecology and Management* **196** (2–3): 245–255. doi:10.1016/j.foreco.2004.02.049.
- Fast k nearest neighbor search using GPU. In Proceedings of the CVPR Workshop on Computer Vision on GPU, Anchorage, Alaska, USA, June 2008. V. Garcia and E. Debreuve and M. Barlaud.
- Scholarpedia article on *k*-NN
- google-all-pairs-similarity-search

Chapter 47

Principal component analysis



PCA of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3 in roughly the (0.878, 0.478) direction and of 1 in the orthogonal direction. The vectors shown are the eigenvectors of the covariance matrix scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean.

Principal component analysis (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.

Depending on the field of application, it is also named the discrete Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, proper orthogonal decomposition (POD) in mechanical engineering, singular value decomposition (SVD) of $\mathbf{X}$ (Golub and Van Loan, 1983), eigenvalue decomposition (EVD) of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ in linear algebra, factor analysis (for a discussion of the differences between PCA and factor analysis see Ch. 7 of[1]), Eckart–Young theorem (Harman, 1960), or Schmidt–Mirsky theorem in psychometrics, empirical orthogonal functions (EOF) in meteorological science, empirical eigenfunction decomposition (Sirovich, 1987), empirical component analysis (Lorenz, 1956), quasiharmonic modes (Brooks et al., 1988), spectral decomposition in noise and vibration, and empirical modal analysis in structural dynamics.

PCA was invented in 1901 by Karl Pearson,[2] as an analogue of the principal axis theorem in mechanics; it was later independently developed (and named) by Harold Hotelling in the 1930s.[3] The method is mostly used as a tool in exploratory data analysis and for making predictive models. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing or using Z-scores) the data matrix for each attribute.[4] The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).[5]

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualised as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection or “shadow” of this object when viewed from its (in some sense; see below) most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

PCA is closely related to factor analysis. Factor analysis typically incorporates more domain specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix.

PCA is also related to canonical correlation analysis (CCA). CCA defines coordinate systems that optimally describe the cross-covariance between two datasets while PCA defines a new orthogonal coordinate system that optimally describes variance in a single dataset.[6][7]

47.1 Intuition

PCA can be thought of as fitting an n-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipse is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.

To find the axes of the ellipse, we must first subtract the mean of each variable from the dataset to center the data around the origin. Then, we compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix. Then, we must orthogonalize the set of eigenvectors, and normalize each to become unit vectors. Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data. The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

It is important to note that this procedure is sensitive to the scaling of the data, and that there is no consensus as to how to best scale the data to obtain optimal results.

47.2 Details

PCA is mathematically defined[1] as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Consider a data matrix, **X**, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the n rows represents a different repetition of the experiment, and each of the p columns gives a particular kind of datum (say, the results from a particular sensor).

Mathematically, the transformation is defined by a set of p-dimensional vectors of weights or *loadings* $\mathbf{w}_{(k)} = (w_1, \dots, w_p)_{(k)}$ that map each row vector $\mathbf{x}_{(i)}$ of **X** to a new vector of principal component *scores* $\mathbf{t}_{(i)} = (t_1, \dots, t_p)_{(i)}$, given by

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}$$

in such a way that the individual variables of **t** considered over the data set successively inherit the maximum possible variance from **x**, with each loading vector **w** constrained to be a unit vector.

47.2.1 First component

The first loading vector $\mathbf{w}_{(1)}$ thus has to satisfy

$$\mathbf{w}_{(1)} = \underset{\Vert\mathbf{w}\Vert=1}{\operatorname{arg\,max}} \left\{ \sum_i (t_1)^2_{(i)} \right\} = \underset{\Vert\mathbf{w}\Vert=1}{\operatorname{arg\,max}} \left\{ \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w}\right)^2 \right\}$$

Equivalently, writing this in matrix form gives

$$\mathbf{w}_{(1)} = \underset{\Vert\mathbf{w}\Vert=1}{\operatorname{arg\,max}} \left\{ \Vert\mathbf{Xw}\Vert^2 \right\} = \underset{\Vert\mathbf{w}\Vert=1}{\operatorname{arg\,max}} \left\{ \mathbf{w}^T\mathbf{X}^T\mathbf{Xw} \right\}$$

Since $\mathbf{w}_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_{(1)} = \operatorname{arg\,max} \left\{ \frac{\mathbf{w}^T\mathbf{X}^T\mathbf{Xw}}{\mathbf{w}^T\mathbf{w}} \right\}$$

The quantity to be maximised can be recognised as a Rayleigh quotient. A standard result for a symmetric matrix such as $\mathbf{X}^\mathrm{T}\mathbf{X}$ is that the quotient's maximum possible value is the largest eigenvalue of the matrix, which occurs when w is the corresponding eigenvector.

With $\mathbf{w}_{(1)}$ found, the first component of a data vector $\mathbf{x}_{(i)}$ can then be given as a score $t_{1(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}$ in the transformed co-ordinates, or as the corresponding vector in the original variables, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}\}\ \mathbf{w}_{(1)}$.

47.2.2 Further components

The kth component can be found by subtracting the first $k - 1$ principal components from **X**:

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X}\mathbf{w}_{(s)}\mathbf{w}_{(s)}^\mathrm{T}$$

and then finding the loading vector which extracts the maximum variance from this new data matrix

$$\mathbf{w}_{(k)} = \underset{\Vert\mathbf{w}\Vert=1}{\operatorname{arg\,max}} \left\{ \Vert\hat{\mathbf{X}}_k\mathbf{w}\Vert^2 \right\} = \operatorname{arg\,max} \left\{ \tfrac{\mathbf{w}^T\hat{\mathbf{X}}_k^T\hat{\mathbf{X}}_k\mathbf{w}}{\mathbf{w}^T\mathbf{w}} \right\}$$

It turns out that this gives the remaining eigenvectors of $\mathbf{X}^\mathrm{T}\mathbf{X}$, with the maximum values for the quantity in brackets given by their corresponding eigenvalues.

The kth principal component of a data vector $\mathbf{x}_{(i)}$ can therefore be given as a score $t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}$ in the transformed co-ordinates, or as the corresponding vector in the space of the original variables, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}\}\ \mathbf{w}_{(k)}$, where $\mathbf{w}_{(k)}$ is the kth eigenvector of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

The full principal components decomposition of $\mathbf{X}$ can therefore be given as

$$\mathbf{T} = \mathbf{X}\mathbf{W}$$

where $\mathbf{W}$ is a p-by-p matrix whose columns are the eigenvectors of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$

47.2.3 Covariances

$\mathbf{X}^{\mathrm{T}}\mathbf{X}$ itself can be recognised as proportional to the empirical sample covariance matrix of the dataset $\mathbf{X}$.

The sample covariance Q between two of the different principal components over the dataset is given by:

$$\begin{aligned} Q(\mathrm{PC}_{(j)}, \mathrm{PC}_{(k)}) &\propto (\mathbf{X}\mathbf{w}_{(j)})^T \cdot (\mathbf{X}\mathbf{w}_{(k)}) \\ &= \mathbf{w}_{(j)}^T \mathbf{X}^T \mathbf{X}\mathbf{w}_{(k)} \\ &= \mathbf{w}_{(j)}^T \lambda_{(k)} \mathbf{w}_{(k)} \\ &= \lambda_{(k)} \mathbf{w}_{(j)}^T \mathbf{w}_{(k)} \end{aligned}$$

where the eigenvalue property of $\mathbf{w}_{(k)}$ has been used to move from line 2 to line 3. However eigenvectors $\mathbf{w}_{(j)}$ and $\mathbf{w}_{(k)}$ corresponding to eigenvalues of a symmetric matrix are orthogonal (if the eigenvalues are different), or can be orthogonalised (if the vectors happen to share an equal repeated value). The product in the final line is therefore zero; there is no sample covariance between different principal components over the dataset.

Another way to characterise the principal components transformation is therefore as the transformation to coordinates which diagonalise the empirical sample covariance matrix.

In matrix form, the empirical covariance matrix for the original variables can be written

$$\mathbf{Q} \propto \mathbf{X}^T\mathbf{X} = \mathbf{W}⍰\mathbf{W}^T$$

The empirical covariance matrix between the principal components becomes

$$\mathbf{W}^T\mathbf{Q}\mathbf{W} \propto \mathbf{W}^T\mathbf{W}⍰\mathbf{W}^T\mathbf{W} = ⍰$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues $\lambda_{(k)}$ of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$

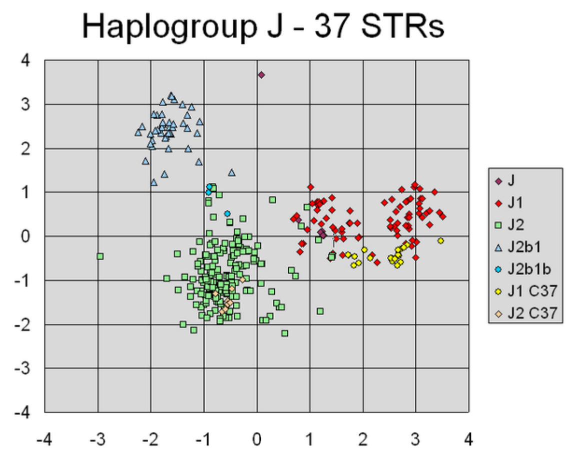
($\lambda_{(k)}$ being equal to the sum of the squares over the dataset associated with each component k: $\lambda_{(k)} = \Sigma i\ t k^2{}_{(i)} = \Sigma i\ (\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)})^2$)

47.2.4 Dimensionality reduction

The faithful transformation $\mathbf{T} = \mathbf{X}\ \mathbf{W}$ maps a data vector $\mathbf{x}_{(i)}$ from an original space of p variables to a new space of p variables which are uncorrelated over the dataset. However, not all the principal components need to be kept. Keeping only the first L principal components, produced by using only the first L loading vectors, gives the truncated transformation

$$\mathbf{T}_L = \mathbf{X}\mathbf{W}_L$$

where the matrix **TL** now has n rows but only L columns. In other words, PCA learns a linear transformation $t = W^T x, x \in R^p, t \in R^L$, where the columns of $p \times L$ matrix W form an orthogonal basis for the L features (the components of representation t) that are decorrelated.[8] By construction, of all the transformed data matrices with only L columns, this score matrix maximises the variance in the original data that has been preserved, while minimising the total squared reconstruction error $\|\mathbf{T}\mathbf{W}^T - \mathbf{T}_L\mathbf{W}_L^T\|_2^2$ or $\|\mathbf{X} - \mathbf{X}_L\|_2^2$.



A principal components analysis scatterplot of Y-STR haplotypes calculated from repeat-count values for 37 Y-chromosomal STR markers from 354 individuals.
PCA has successfully found linear combinations of the different markers, that separate out different clusters corresponding to different lines of individuals' Y-chromosomal genetic descent.

Such dimensionality reduction can be a very useful step for visualising and processing high-dimensional datasets, while still retaining as much of the variance in the dataset as possible. For example, selecting $L = 2$ and keeping only the first two principal components finds the two-dimensional plane through the high-dimensional dataset in which the data is most spread out, so if the data contains clusters these too may be most spread out, and therefore most visible to be plotted out in a two-dimensional diagram; whereas if two directions through the data (or two of the original variables) are chosen at random, the clusters may be much less spread apart from each other, and may in fact be much more likely to substantially overlay each other, making them indistinguishable.

Similarly, in regression analysis, the larger the number of explanatory variables allowed, the greater is the chance of overfitting the model, producing conclusions that fail to generalise to other datasets. One approach, especially when there are strong correlations between different possible explanatory variables, is to reduce them to a few principal components and then run the regression against them, a method called principal component regression.

Dimensionality reduction may also be appropriate when the variables in a dataset are noisy. If each column of the dataset contains independent identically distributed Gaussian noise, then the columns of **T** will also contain similarly identically distributed Gaussian noise (such a distribution is invariant under the effects of the matrix **W**, which can be thought of as a high-dimensional rotation of the co-ordinate axes). However, with more of the total variance concentrated in the first few principal components compared to the same noise variance, the proportionate effect of the noise is less—the first few components achieve a higher signal-to-noise ratio. PCA thus can have the effect of concentrating much of the signal into the first few principal components, which can usefully be captured by dimensionality reduction; while the later principal components may be dominated by noise, and so disposed of without great loss.

47.2.5 Singular value decomposition

The principal components transformation can also be associated with another matrix factorisation, the singular value decomposition (SVD) of **X**,

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{W}^T$$

Here **Σ** is a n-by-p rectangular diagonal matrix of positive numbers $\sigma_{(k)}$, called the singular values of **X**; **U** is an n-by-n matrix, the columns of which are orthogonal unit vectors of length n called the left singular vectors of **X**; and **W** is a p-by-p whose columns are orthogonal unit vectors of length p and called the right singular vectors of **X**.

In terms of this factorisation, the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ can be written

$$\begin{aligned}\mathbf{X}^T\mathbf{X} &= \mathbf{W}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{W}^T \\ &= \mathbf{W}\mathbf{\Sigma}^2\mathbf{W}^T\end{aligned}$$

Comparison with the eigenvector factorisation of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ establishes that the right singular vectors **W** of **X** are equivalent to the eigenvectors of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, while the singular values $\sigma_{(k)}$ of **X** are equal to the square roots of the eigenvalues $\lambda_{(k)}$ of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

Using the singular value decomposition the score matrix **T** can be written

$$\begin{aligned}\mathbf{T} &= \mathbf{X}\mathbf{W} \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{W}^T\mathbf{W} \\ &= \mathbf{U}\mathbf{\Sigma}\end{aligned}$$

so each column of **T** is given by one of the left singular vectors of **X** multiplied by the corresponding singular value. This form is also the polar decomposition of **T**.

Efficient algorithms exist to calculate the SVD of **X** without having to form the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, so computing the SVD is now the standard way to calculate a principal components analysis from a data matrix, unless only a handful of components are required.

As with the eigen-decomposition, a truncated $n \times L$ score matrix **TL** can be obtained by considering only the first L largest singular values and their singular vectors:

$$\mathbf{T}_L = \mathbf{U}_L\mathbf{\Sigma}_L = \mathbf{X}\mathbf{W}_L$$

The truncation of a matrix **M** or **T** using a truncated singular value decomposition in this way produces a truncated matrix that is the nearest possible matrix of rank L to the original matrix, in the sense of the difference between the two having the smallest possible Frobenius norm, a result known as the Eckart–Young theorem [1936].

47.3 Further considerations

Given a set of points in Euclidean space, the first principal component corresponds to a line that passes through the multidimensional mean and minimizes the sum of squares of the distances of the points from the line. The second principal component corresponds to the same concept after all correlation with the first principal component has been subtracted from the points. The singular values (in **Σ**) are the square roots of the eigenvalues of the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Each eigenvalue is proportional to the portion of the "variance" (more correctly of the sum of the squared distances of the points from their multidimensional mean) that is correlated with each eigenvector. The sum of all the eigenvalues is equal to the sum of the squared distances of the points from their multidimensional mean. PCA essentially rotates the set of points around their mean in order to align with the principal components. This moves as much of the variance as possible (using an orthogonal transformation) into the first few dimensions. The values in the remaining dimensions, therefore, tend to be small and may be dropped with minimal loss of information (see below). PCA is often used in this manner for dimensionality reduction. PCA has the distinction of being the optimal orthogonal transformation for keeping the subspace that has largest "variance" (as defined above). This advantage, however, comes at the price of greater computational requirements if compared, for example and when applicable, to the discrete cosine transform, and in particular to the DCT-II which is simply known as the "DCT". Nonlinear dimensionality reduction techniques tend to be more computationally demanding than PCA.

PCA is sensitive to the scaling of the variables. If we have just two variables and they have the same sample variance and are positively correlated, then the PCA will entail a rotation by 45° and the "loadings" for the two variables with respect to the principal component will be equal. But if we multiply all values of the first variable by 100, then the first principal component will be almost the same as that variable, with a small contribution from the other variable, whereas the second component will be almost aligned with the second original variable. This means that whenever the different variables have different units (like temperature and mass), PCA is a somewhat arbitrary method of analysis. (Different results would be obtained if one used Fahrenheit rather than Celsius for example.) Note that Pearson's original paper was entitled "On Lines and Planes of Closest Fit to Systems of Points in Space" – "in space" implies physical Euclidean space where such concerns do not arise. One way of making the PCA less arbitrary is to use variables scaled so as to have unit variance, by standardizing the data and hence use the autocorrelation matrix instead of the autocovariance matrix as a basis for PCA. However, this compresses (or expands) the fluctuations in all dimensions of the signal space to unit variance.

Mean subtraction (a.k.a. "mean centering") is necessary for performing PCA to ensure that the first principal component describes the direction of maximum variance. If mean subtraction is not performed, the first principal component might instead correspond more or less to the mean of the data. A mean of zero is needed for finding a basis that minimizes the mean square error of the approximation of the data.[9]

PCA is equivalent to empirical orthogonal functions (EOF), a name which is used in meteorology.

An autoencoder neural network with a linear hidden layer is similar to PCA. Upon convergence, the weight vectors of the *K* neurons in the hidden layer will form a basis for the space spanned by the first *K* principal components. Unlike PCA, this technique will not necessarily produce orthogonal vectors.

PCA is a popular primary technique in pattern recognition. It is not, however, optimized for class separability.[10] An alternative is the linear discriminant analysis, which does take this into account.

47.4 Table of symbols and abbreviations

47.5 Properties and limitations of PCA

47.5.1 Properties[11]

***Property 1*:** For any integer *q, 1 ≤ q ≤ p,* consider the orthogonal linear transformation

$y = \mathbf{B}'x$

where y is a *q-element* vector and $\mathbf{B}'$ is a *(q × p)* matrix, and let $⍰_y = \mathbf{B}'⍰\mathbf{B}$ be the variance-covariance matrix for y . Then the trace of $⍰_y$, denoted $\text{tr}(⍰_y)$, is maximized by taking $\mathbf{B} = \mathbf{A}_q$, where $\mathbf{A}_q$ consists of the first q columns of $\mathbf{A}$ ($\mathbf{B}'$ is the transposition of $\mathbf{B}$) .

***Property 2*:** Consider again the orthonormal transformation

$y = \mathbf{B}'x$

with $x, \mathbf{B}, \mathbf{A}$ and $⍰_y$ defined as before. Then $\text{tr}(⍰_y)$ is minimized by taking $\mathbf{B} = \mathbf{A}^*_q$, where $\mathbf{A}^*_q$ consists of the last q columns of $\mathbf{A}$.

The statistical implication of this property is that the last few PCs are not simply unstructured left-overs after removing the important PCs. Because these last PCs have variances as small as possible they are useful in their own right. They can help to detect unsuspected near-constant linear relationships between the elements of x, and they may also be useful in regression, in selecting a subset of variables from x, and in outlier detection.

***Property 3*:** (Spectral Decomposition of $\mathbf{\Sigma}$)

$⍰ = \lambda_1\alpha_1\alpha_1' + \cdots + \lambda_p\alpha_p\alpha_p'$

Before we look at its usage, we first look at diagonal elements,

$$\text{Var}(x_j) = \sum_{k=1}^{P} \lambda_k\alpha_{kj}^2$$

Then, perhaps the main statistical implication of the result is that not only can we decompose the combined variances of all the elements of x into decreasing contributions due to each PC, but we can also decompose the whole covariance matrix into contributions $\lambda_k\alpha_k\alpha_k'$ from each PC. Although not strictly decreasing, the elements of $\lambda_k\alpha_k\alpha_k'$ will tend to become smaller as k increases, as $\lambda_k\alpha_k\alpha_k'$ decreases for increasing k , whereas the elements of α_k tend to stay 'about the same size'because of the normalization constraints: $\alpha_k'\alpha_k = 1, k = 1, \cdots , p$

47.5.2 Limitations

As noted above, the results of PCA depend on the scaling of the variables. A scale-invariant form of PCA has been developed.[12]

The applicability of PCA is limited by certain assumptions[13] made in its derivation.

47.5.3 PCA and information theory

The claim that the PCA used for dimensionality reduction preserves most of the information of the data is misleading. Indeed, without any assumption on the signal model, PCA cannot help to reduce the amount of information lost during dimensionality reduction, where information was measured using Shannon entropy.[14]

Under the assumption that

$$\mathbf{x} = \mathbf{s} + \mathbf{n}$$

i.e., that the data vector $\mathbf{x}$ is the sum of the desired information-bearing signal $\mathbf{s}$ and a noise signal $\mathbf{n}$ one can show that PCA can be optimal for dimensionality reduction also from an information-theoretic point-of-view.

In particular, Linsker showed that if $\mathbf{s}$ is Gaussian and $\mathbf{n}$ is Gaussian noise with a covariance matrix proportional to the identity matrix, the PCA maximizes the mutual information $I(\mathbf{y};\mathbf{s})$ between the desired information $\mathbf{s}$ and the dimensionality-reduced output $\mathbf{y} = \mathbf{W}_L^T\mathbf{x}$.[15]

If the noise is still Gaussian and has a covariance matrix proportional to the identity matrix (i.e., the components of the vector $\mathbf{n}$ are iid), but the information-bearing signal $\mathbf{s}$ is non-Gaussian (which is a common scenario), PCA at least minimizes an upper bound on the *information loss*, which is defined as[16][17]

$$I(\mathbf{x};\mathbf{s}) - I(\mathbf{y};\mathbf{s}).$$

The optimality of PCA is also preserved if the noise $\mathbf{n}$ is iid and at least more Gaussian (in terms of the Kullback–Leibler divergence) than the information-bearing signal $\mathbf{s}$.[18] In general, even if the above signal model holds, PCA loses its information-theoretic optimality as soon as the noise $\mathbf{n}$ becomes dependent.

47.6 Computing PCA using the covariance method

The following is a detailed description of PCA using the covariance method (see also here) as opposed to the correlation method.[19] But note that it is better to use the singular value decomposition (using standard software).

The goal is to transform a given data set $\mathbf{X}$ of dimension p to an alternative data set $\mathbf{Y}$ of smaller dimension L. Equivalently, we are seeking to find the matrix $\mathbf{Y}$, where $\mathbf{Y}$ is the Karhunen–Loève transform (KLT) of matrix $\mathbf{X}$:

$$\mathbf{Y} = \mathbb{KLT}\{\mathbf{X}\}$$

47.6.1 Organize the data set

Suppose you have data comprising a set of observations of p variables, and you want to reduce the data so that each observation can be described with only L variables, $L < p$. Suppose further, that the data are arranged as a set of n data vectors $\mathbf{x}_1 \ldots \mathbf{x}_n$ with each $\mathbf{x}_i$ representing a single grouped observation of the p variables.

- Write $\mathbf{x}_1 \ldots \mathbf{x}_n$ as row vectors, each of which has p columns.
- Place the row vectors into a single matrix $\mathbf{X}$ of dimensions $n \times p$.

47.6.2 Calculate the empirical mean

- Find the empirical mean along each dimension $j = 1, ..., p$.
- Place the calculated mean values into an empirical mean vector $\mathbf{u}$ of dimensions $p \times 1$.

$$u[j] = \frac{1}{n}\sum_{i=1}^{n} X[i,j]$$

47.6.3 Calculate the deviations from the mean

Mean subtraction is an integral part of the solution towards finding a principal component basis that minimizes the mean square error of approximating the data.[20] Hence we proceed by centering the data as follows:

- Subtract the empirical mean vector $\mathbf{u}$ from each row of the data matrix $\mathbf{X}$.
- Store mean-subtracted data in the $n \times p$ matrix $\mathbf{B}$.

$$\mathbf{B} = \mathbf{X} - \mathbf{h}\mathbf{u}^T$$

where $\mathbf{h}$ is an $n \times 1$ column vector of all 1s:

$$h[i] = 1 \qquad \text{for} i = 1, \ldots, n$$

47.6.4 Find the covariance matrix

- Find the $p \times p$ empirical covariance matrix **C** from the outer product of matrix **B** with itself:

  $$\mathbf{C} = \frac{1}{n-1}\mathbf{B}^* \cdot \mathbf{B}$$

  where $*$ is the conjugate transpose operator. Note that if **B** consists entirely of real numbers, which is the case in many applications, the "conjugate transpose" is the same as the regular transpose.

- Please note that outer products apply to vectors. For tensor cases we should apply tensor products, but the covariance matrix in PCA is a sum of outer products between its sample vectors; indeed, it could be represented as B*.B. See the covariance matrix sections on the discussion page for more information.

- The reasoning behind using $N - 1$ instead of N to calculate the covariance is Bessel's correction

47.6.5 Find the eigenvectors and eigenvalues of the covariance matrix

- Compute the matrix **V** of eigenvectors which diagonalizes the covariance matrix **C**:

  $$\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$$

  where **D** is the diagonal matrix of eigenvalues of **C**. This step will typically involve the use of a computer-based algorithm for computing eigenvectors and eigenvalues. These algorithms are readily available as sub-components of most matrix algebra systems, such as R, MATLAB,[21][22] Mathematica,[23] SciPy, IDL (Interactive Data Language), or GNU Octave as well as OpenCV.

- Matrix **D** will take the form of an $p \times p$ diagonal matrix, where

  $$D[k,l] = \lambda_k \qquad \text{for} k = l$$

  is the jth eigenvalue of the covariance matrix **C**, and

  $$D[k,l] = 0 \qquad \text{for} k \neq l.$$

- Matrix **V**, also of dimension $p \times p$, contains p column vectors, each of length p, which represent the p eigenvectors of the covariance matrix **C**.

- The eigenvalues and eigenvectors are ordered and paired. The jth eigenvalue corresponds to the jth eigenvector.

47.6.6 Rearrange the eigenvectors and eigenvalues

- Sort the columns of the eigenvector matrix **V** and eigenvalue matrix **D** in order of *decreasing* eigenvalue.

- Make sure to maintain the correct pairings between the columns in each matrix.

47.6.7 Compute the cumulative energy content for each eigenvector

- The eigenvalues represent the distribution of the source data's energy among each of the eigenvectors, where the eigenvectors form a basis for the data. The cumulative energy content g for the jth eigenvector is the sum of the energy content across all of the eigenvalues from 1 through j:

  $$g[j] = \sum_{k=1}^{j} D[k,k] \qquad \text{for} \qquad j = 1, \ldots, p$$

47.6.8 Select a subset of the eigenvectors as basis vectors

- Save the first L columns of **V** as the $p \times L$ matrix **W**:

  $$W[k,l] = V[k,l] \qquad \text{for} \qquad k = 1, \ldots, p \qquad l = 1, \ldots, L$$

  where

  $$1 \leq L \leq p.$$

- Use the vector **g** as a guide in choosing an appropriate value for L. The goal is to choose a value of L as small as possible while achieving a reasonably high value of g on a percentage basis. For example, you may want to choose L so that the cumulative energy g is above a certain threshold, like 90 percent. In this case, choose the smallest value of L such that

  $$\frac{g[L]}{g[p]} \geq 0.9$$

47.6.9 Convert the source data to z-scores (optional)

- Create an $p \times 1$ empirical standard deviation vector **s** from the square root of each element along the main diagonal of the diagonalized covariance matrix **C**. (Note, that scaling operations do not commute with the KLT thus we must scale by the variances of the already-decorrelated vector, which is the diagonal of **C**) :

$$\mathbf{s} = \{s[j]\} = \{\sqrt{C[j,j]}\} \qquad \text{for} j = 1, \ldots, p$$

- Calculate the $n \times p$ z-score matrix:

$$\mathbf{Z} = \frac{\mathbf{B}}{\mathbf{h} \cdot \mathbf{s}^T}$$

- Note: While this step is useful for various applications as it normalizes the data set with respect to its variance, it is not integral part of PCA/KLT

47.6.10 Project the z-scores of the data onto the new basis

- The projected vectors are the columns of the matrix

$$\mathbf{T} = \mathbf{Z} \cdot \mathbf{W} = \mathbb{KLT}\{\mathbf{X}\}.$$

- The rows of matrix **T** represent the Karhunen–Loeve transforms (KLT) of the data vectors in the rows of matrix **X**.

47.7 Derivation of PCA using the covariance method

Let **X** be a d-dimensional random vector expressed as column vector. Without loss of generality, assume **X** has zero mean.

We want to find $(*)$ a $d \times d$ orthonormal transformation matrix **P** so that **PX** has a diagonal covariant matrix (*i.e.* **PX** is a random vector with all its distinct components pairwise uncorrelated).

A quick computation assuming P were unitary yields:

$$\begin{aligned}\operatorname{var}(PX) &= \mathbb{E}[PX\ (PX)^\dagger] \\ &= \mathbb{E}[PX\ X^\dagger P^\dagger] \\ &= P\ \mathbb{E}[XX^\dagger]P^\dagger \\ &= P\ \operatorname{var}(X)P^{-1}\end{aligned}$$

Hence $(*)$ holds if and only if $\operatorname{var}(X)$ were diagonalisable by P .

This is very constructive, as var(**X**) is guaranteed to be a non-negative definite matrix and thus is guaranteed to be diagonalisable by some unitary matrix.

47.7.1 Iterative computation

In practical implementations especially with high dimensional data (large p), the covariance method is rarely used because it is not efficient. One way to compute the first principal component efficiently[24] is shown in the following pseudo-code, for a data matrix **X** with zero mean, without ever computing its covariance matrix.

$\mathbf{r}$ = a random vector of length p do c times: $\mathbf{s} = 0$ (a vector of length p) for each row $\mathbf{x} \in \mathbf{X}$ $\mathbf{s} = \mathbf{s} + (\mathbf{x} \cdot \mathbf{r})\mathbf{x}$ $\mathbf{r} = \frac{\mathbf{s}}{|\mathbf{s}|}$ return $\mathbf{r}$

This algorithm is simply an efficient way of calculating $\mathbf{X^TX}\ \mathbf{r}$, normalizing, and placing the result back in **r** (power iteration). It avoids the np^2 operations of calculating the covariance matrix. **r** will typically get close to the first principal component of **X** within a small number of iterations, c. (The magnitude of **s** will be larger after each iteration. Convergence can be detected when it increases by an amount too small for the precision of the machine.)

Subsequent principal components can be computed by subtracting component **r** from **X** (see Gram–Schmidt) and then repeating this algorithm to find the next principal component. However this simple approach is not numerically stable if more than a small number of principal components are required, because imprecisions in the calculations will additively affect the estimates of subsequent principal components. More advanced methods build on this basic idea, as with the closely related Lanczos algorithm.

One way to compute the eigenvalue that corresponds with each principal component is to measure the difference in mean-squared-distance between the rows and the centroid, before and after subtracting out the principal component. The eigenvalue that corresponds with the component that was removed is equal to this difference.

47.7.2 The NIPALS method

Main article: Non-linear iterative partial least squares

For very-high-dimensional datasets, such as those generated in the *omics sciences (e.g., genomics, metabolomics) it is usually only necessary to compute the first few PCs. The non-linear iterative partial least squares (NIPALS) algorithm calculates $\mathbf{t}_1$ and $\mathbf{w}_1{}^T$ from **X**. The outer product, $\mathbf{t}_1\mathbf{w}_1{}^T$ can then be subtracted from **X** leaving the residual matrix $\mathbf{E}_1$. This can be then used to calculate subsequent PCs.[25] This results in a dramatic reduction in computational time since calculation of the covariance matrix is avoided.

However, for large data matrices, or matrices that have a high degree of column collinearity, NIPALS suffers from loss of orthogonality due to machine precision limitations accumulated in each iteration step.[26] A Gram–Schmidt (GS) re-orthogonalization algorithm is applied to both the scores and the loadings at each iteration step to eliminate this loss of orthogonality.[27]

47.7.3 Online/sequential estimation

In an "online" or "streaming" situation with data arriving piece by piece rather than being stored in a single batch, it is useful to make an estimate of the PCA projection that can be updated sequentially. This can be done efficiently, but requires different algorithms.[28]

47.8 PCA and qualitative variables

In PCA, it is common that we want to introduce qualitative variables as supplementary elements. For example, many quantitative variables have been measured on plants. For these plants, some qualitative variables are available as, for example, the species to which the plant belongs. These data were subjected to PCA for quantitative variables. When analyzing the results, it is natural to connect the principal components to the qualitative variable *species*. For this, the following results are produced.

- Identification, on the factorial planes, of the different species e.g. using different colors.
- Representation, on the factorial planes, of the centers of gravity of plants belonging to the same species.
- For each center of gravity and each axis, p-value to judge the significance of the difference between the center of gravity and origin.

These results are what is called *introducing a qualitative variable as supplementary element*. This procedure is detailed in and Husson, Lê & Pagès 2009 and Pagès 2013. Few software offer this option in an "automatic" way. This is the case of SPAD that historically, following the work of Ludovic Lebart, was the first to propose this option, and the R package FactoMineR.

47.9 Applications

47.9.1 Neuroscience

A variant of principal components analysis is used in neuroscience to identify the specific properties of a stimulus that increase a neuron's probability of generating an action potential.[29] This technique is known as spike-triggered covariance analysis. In a typical application an experimenter presents a white noise process as a stimulus (usually either as a sensory input to a test subject, or as a current injected directly into the neuron) and records a train of action potentials, or spikes, produced by the neuron as a result. Presumably, certain features of the stimulus make the neuron more likely to spike. In order to extract these features, the experimenter calculates the covariance matrix of the *spike-triggered ensemble*, the set of all stimuli (defined and discretized over a finite time window, typically on the order of 100 ms) that immediately preceded a spike. The eigenvectors of the difference between the spike-triggered covariance matrix and the covariance matrix of the *prior stimulus ensemble* (the set of all stimuli, defined over the same length time window) then indicate the directions in the space of stimuli along which the variance of the spike-triggered ensemble differed the most from that of the prior stimulus ensemble. Specifically, the eigenvectors with the largest positive eigenvalues correspond to the directions along which the variance of the spike-triggered ensemble showed the largest positive change compared to the variance of the prior. Since these were the directions in which varying the stimulus led to a spike, they are often good approximations of the sought after relevant stimulus features.

In neuroscience, PCA is also used to discern the identity of a neuron from the shape of its action potential. Spike sorting is an important procedure because extracellular recording techniques often pick up signals from more than one neuron. In spike sorting, one first uses PCA to reduce the dimensionality of the space of action potential waveforms, and then performs clustering analysis to associate specific action potentials with individual neurons.

47.10 Relation between PCA and *K*-means clustering

It was asserted in [30][31] that the relaxed solution of k-means clustering, specified by the cluster indicators, is given by the PCA (principal component analysis) principal components, and the PCA subspace spanned by the principal directions is identical to the cluster centroid subspace. However, that PCA is a useful relaxation of k-means clustering was not a new result (see, for example,[32]), and it is straightforward to uncover counterexamples to the statement that the cluster centroid subspace is spanned by the principal directions.[33]

47.11 Relation between PCA and factor analysis[34]

Principal component analysis creates variables that are linear combinations of the original variables. The new variables have the property that the variables are all orthogonal. The principal components can be used to find clusters in a set of data. PCA is a variance-focused approach seeking to reproduce the total variable variance, in which components reflect both common and unique variance of the variable. PCA is generally preferred for purposes of data reduction (i.e., translating variable space into optimal factor space) but not when the goal is to detect the latent construct or factors.

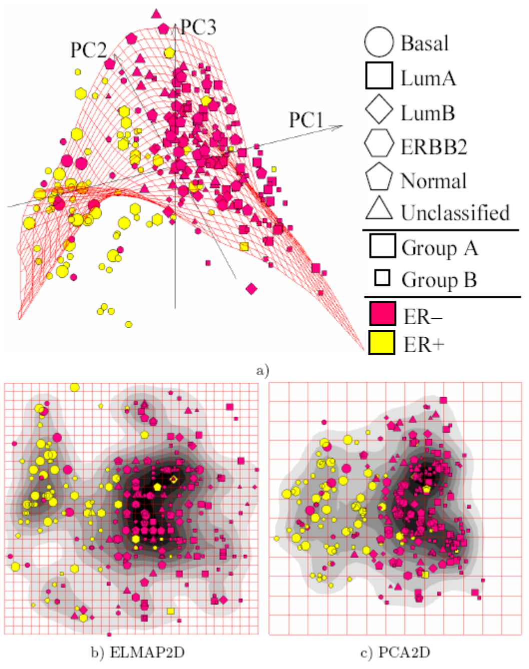
Factor analysis is similar to principal component analysis, in that factor analysis also involves linear combinations of variables. Different from PCA, factor analysis is a correlation-focused approach seeking to reproduce the inter-correlations among variables, in which the factors "represent the common variance of variables, excluding unique variance[35]" . Factor analysis is generally used when the research purpose is detecting data structure (i.e., latent constructs or factors) or causal modeling.

47.12 Correspondence analysis

Correspondence analysis (CA) was developed by Jean-Paul Benzécri[36] and is conceptually similar to PCA, but scales the data (which should be non-negative) so that rows and columns are treated equivalently. It is traditionally applied to contingency tables. CA decomposes the chi-squared statistic associated to this table into orthogonal factors.[37] Because CA is a descriptive technique, it can be applied to tables for which the chi-squared statistic is appropriate or not. Several variants of CA are available including detrended correspondence analysis and canonical correspondence analysis. One special extension is multiple correspondence analysis, which may be seen as the counterpart of principal component analysis for categorical data.[38]

47.13 Generalizations

47.13.1 Nonlinear generalizations



Linear PCA versus nonlinear Principal Manifolds[39] for visualization of breast cancer microarray data: a) Configuration of nodes and 2D Principal Surface in the 3D PCA linear manifold. The dataset is curved and cannot be mapped adequately on a 2D principal plane; b) The distribution in the internal 2D non-linear principal surface coordinates (ELMap2D) together with an estimation of the density of points; c) The same as b), but for the linear 2D PCA manifold (PCA2D). The "basal" breast cancer subtype is visualized more adequately with ELMap2D and some features of the distribution become better resolved in comparison to PCA2D. Principal manifolds are produced by the elastic maps algorithm. Data are available for public competition.[40] Software is available for free non-commercial use.[41]

Most of the modern methods for nonlinear dimensionality reduction find their theoretical and algorithmic roots in PCA or K-means. Pearson's original idea was to take a straight line (or plane) which will be "the best fit" to a set of data points. **Principal curves and manifolds**[42] give the natural geometric framework for PCA generalization and extend the geometric interpretation of PCA by explicitly constructing an embedded manifold for data approximation, and by encoding using standard geometric projection onto the manifold, as it is illustrated by Fig. See also the elastic map algorithm and principal geodesic analysis. Another popular generalization is kernel PCA, which corresponds to PCA performed in a reproducing kernel Hilbert space associated with a positive definite kernel.

47.13.2 Multilinear generalizations

In multilinear subspace learning,[43] PCA is generalized to multilinear PCA (MPCA) that extracts features directly from tensor representations. MPCA is solved by performing PCA in each mode of the tensor iteratively. MPCA has been applied to face recognition, gait recognition, etc. MPCA is further extended to uncorrelated MPCA, non-negative MPCA and robust MPCA.

47.13.3 Higher order

N-way principal component analysis may be performed with models such as Tucker decomposition, PARAFAC, multiple factor analysis, co-inertia analysis, STATIS, and DISTATIS.

47.13.4 Robustness – weighted PCA

While PCA finds the mathematically optimal method (as in minimizing the squared error), it is sensitive to outliers in the data that produce large errors PCA tries to avoid. It therefore is common practice to remove outliers before computing PCA. However, in some contexts, outliers can be difficult to identify. For example in data mining algorithms like correlation clustering, the assignment of points to clusters and outliers is not known beforehand. A recently proposed generalization of PCA[44] based on a **weighted PCA** increases robustness by assigning different weights to data objects based on their estimated relevancy.

47.13.5 Robust PCA via Decomposition in Low Rank and Sparse Matrices

Robust principal component analysis (RPCA) is a modification of the widely used statistical procedure Principal component analysis (PCA) which works well with respect to grossly corrupted observations.

47.13.6 Sparse PCA

A particular disadvantage of PCA is that the principal components are usually linear combinations of all input variables. Sparse PCA overcomes this disadvantage by finding linear combinations that contain just a few input variables.

47.14 Software/source code

- An Open Source Code and Tutorial in MATLAB and C++.
- FactoMineR – Probably the more complete library of functions for exploratory data analysis.
- XLSTAT - Principal Compent Analysis is a part of XLSTAT core module[45]
- Mathematica – Implements principal component analysis with the PrincipalComponents command[46] using both covariance and correlation methods.
- DataMelt - A Java free program that implements several classes to build PCA analysis and to calculate eccentricity of random distributions.
- NAG Library – Principal components analysis is implemented via the g03aa routine (available in both the Fortran[47] and the C[48] versions of the Library).
- SIMCA – Commercial software package available to perform PCA analysis.[49]
- CORICO - Commercial software, offers principal components analysis coupled with Iconographie des correlations.
- MATLAB Statistics Toolbox – The functions princomp and pca (R2012b) give the principal components, while the function pcares gives the residuals and reconstructed matrix for a low-rank PCA approximation. An example MATLAB implementation of PCA is available.[50]
- Oracle Database 12c – Implemented via DBMS_DATA_MINING.SVDS_SCORING_MODE by specifying setting value SVDS_SCORING_PCA [51]
- GNU Octave – Free software computational environment mostly compatible with MATLAB, the function princomp[52] gives the principal component.
- R – Free statistical package, the functions princomp[53] and prcomp[54] can be used for principal component analysis; prcomp uses singular value decomposition which generally gives better numerical accuracy. Some packages that implement PCA in R, include, but are not limited to: ade4, vegan, ExPosition, and FactoMineR[55]
- SAS, PROC FACTOR – Offers principal components analysis.[56]
- MLPACK – Provides an implementation of principal component analysis in C++.
- XLMiner – The principal components tab can be used for principal component analysis.
- Stata – The pca command provides principal components analysis.[57]
- Cornell Spectrum Imager – Open-source toolset built on ImageJ, enables PCA analysis for 3D datacubes.[58]
- imDEV – Free Excel addon to calculate principal components using R package[59][60]
- ViSta: The Visual Statistics System – Free software that provides principal components analysis, simple and multiple correspondence analysis.[61]
- Spectramap – Software to create a biplot using principal components analysis, correspondence analysis or spectral map analysis.[62]
- FinMath – .NET numerical library containing an implementation of PCA.[63]

- Unscrambler X – Multivariate analysis software enabling Principal Component Analysis (PCA) with PCA Projection.{[64]}
- OpenCV[65]
- NMath – Proprietary numerical library containing PCA for the .NET Framework.
- IDL – The principal components can be calculated using the function pcomp.[66]
- Weka – Computes principal components.[67]
- Qlucore – Commercial software for analyzing multivariate data with instant response using PCA[68]
- Orange (software) – Supports PCA through its Linear Projection widget.
- EIGENSOFT – Provides a version of PCA adapted for population genetics analysis.[69]
- Partek Genomics Suite – Statistical software able to perform PCA.[70]
- libpca C++ library – Offers PCA and corresponding transformations.
- Origin – Contains PCA in its Pro version.
- Scikit-learn – Python library for machine learning which contains PCA, Probabilistic PCA, Kernel PCA, Sparse PCA and other techniques in the decomposition module.[71]
- Knime[72]- A java based nodal arrenging software for Analysis, in this the nodes called PCA, PCA compute, PCA Apply, PCA inverse make it easily.
- Julia – Supports PCA with the pca function in the MultivariateStats package [73]
- Netflix Surus – Provides a Java implementation of robust PCA with wrappers for Pig.
- Insightomics - Run principal component analysis directly on your browser.

47.15 See also

- Correspondence analysis (for contingency tables)
- Multiple correspondence analysis (for qualitative variables)
- Factor analysis of mixed data (for quantitative **and** qualitative variables)
- Canonical correlation
- CUR matrix approximation (can replace of low-rank SVD approximation)
- Detrended correspondence analysis
- Dynamic mode decomposition
- Eigenface
- Exploratory factor analysis (Wikiversity)
- Factorial code
- Functional principal component analysis
- Geometric data analysis
- Independent component analysis
- Kernel PCA
- Low-rank approximation
- Matrix decomposition
- Non-negative matrix factorization
- Nonlinear dimensionality reduction
- Oja's rule
- Point distribution model (PCA applied to morphometry and computer vision)
- Principal component analysis (Wikibooks)
- Principal component regression
- Singular spectrum analysis
- Singular value decomposition
- Sparse PCA
- Transform coding
- Weighted least squares

47.16 Notes

[1] Jolliffe I.T. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4

[2] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (PDF). *Philosophical Magazine* **2** (11): 559–572. doi:10.1080/14786440109462720.

[3] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, **24**, 417–441, and 498–520.
Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, **27**, 321–77

[4] Abdi. H., & Williams, L.J. (2010). "Principal component analysis.". *Wiley Interdisciplinary Reviews: Computational Statistics*, **2**: 433–459. doi:10.1002/wics.101.

[5] Shaw P.J.A. (2003) *Multivariate statistics for the Environmental Sciences*, Hodder-Arnold. ISBN 0-340-80763-6.

[6] Barnett, T. P., and R. Preisendorfer. (1987). "Origins and levels of monthly and seasonal forecast skill for United States surface air temperatures determined by canonical correlation analysis.". *Monthly Weather Review 115*.

[7] Hsu, Daniel, Sham M. Kakade, and Tong Zhang (2008). "A spectral algorithm for learning hidden markov models.". *arXiv preprint arXiv:0811.4413*.

[8] Bengio, Y. et al. (2013). "Representation Learning: A Review and New Perspectives" (PDF). *Pattern Analysis and Machine Intelligence* **35** (8). doi:10.1109/TPAMI.2013.50.

[9] A. A. Miranda, Y. A. Le Borgne, and G. Bontempi. New Routes from Minimal Approximation Error to Principal Components, Volume 27, Number 3 / June, 2008, Neural Processing Letters, Springer

[10] Fukunaga, Keinosuke (1990). *Introduction to Statistical Pattern Recognition*. Elsevier. ISBN 0-12-269851-7.

[11] Jolliffe, I. T. (2002). *Principal Component Analysis,* second edition Springer-Verlag. ISBN 978-0-387-95442-4.

[12] Leznik, M; Tofallis, C. 2005 [uhra.herts.ac.uk/bitstream/handle/2299/715/S56.pdf Estimating Invariant Principal Components Using Diagonal Regression.]

[13] Jonathon Shlens, A Tutorial on Principal Component Analysis.

[14] Geiger, Bernhard; Kubin, Gernot (Sep 2012). "Relative Information Loss in the PCA". *Proc. IEEE Information Theory Workshop*: 562–566.

[15] Linsker, Ralph (March 1988). "Self-organization in a perceptual network". *IEEE Computer* **21** (3): 105–117. doi:10.1109/2.36.

[16] Deco & Obradovic (1996). *An Information-Theoretic Approach to Neural Computing*. New York, NY: Springer.

[17] Plumbley, Mark (1991). "Information theory and unsupervised neural networks".Tech Note

[18] Geiger, Bernhard; Kubin, Gernot (January 2013). "Signal Enhancement as Minimization of Relevant Information Loss". *Proc. ITG Conf. on Systems, Communication and Coding*.

[19] "Engineering Statistics Handbook Section 6.5.5.2". Retrieved 19 January 2015.

[20] A.A. Miranda, Y.-A. Le Borgne, and G. Bontempi. New Routes from Minimal Approximation Error to Principal Components, Volume 27, Number 3 / June, 2008, Neural Processing Letters, Springer

[21] eig function Matlab documentation

[22] MATLAB PCA-based Face recognition software

[23] Eigenvalues function Mathematica documentation

[24] Roweis, Sam. "EM Algorithms for PCA and SPCA." Advances in Neural Information Processing Systems. Ed. Michael I. Jordan, Michael J. Kearns, and Sara A. Solla The MIT Press, 1998.

[25] Geladi, Paul; Kowalski, Bruce (1986). "Partial Least Squares Regression:A Tutorial". *Analytica Chimica Acta* **185**: 1–17. doi:10.1016/0003-2670(86)80028-9.

[26] Kramer, R. (1998). *Chemometric Techniques for Quantitative Analysis*. New York: CRC Press.

[27] Andrecut, M. (2009). "Parallel GPU Implementation of Iterative PCA Algorithms". *Journal of Computational Biology* **16** (11): 1593–1599. doi:10.1089/cmb.2008.0221.

[28] Warmuth, M. K.; Kuzmin, D. (2008). "Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension". *Journal of Machine Learning Research* **9**: 2287–2320.

[29] Brenner, N., Bialek, W., & de Ruyter van Steveninck, R.R. (2000).

[30] H. Zha, C. Ding, M. Gu, X. He and H.D. Simon (Dec 2001). "Spectral Relaxation for K-means Clustering" (PDF). *Neural Information Processing Systems vol.14 (NIPS 2001)* (Vancouver, Canada): 1057–1064.

[31] Chris Ding and Xiaofeng He (July 2004). "K-means Clustering via Principal Component Analysis" (PDF). *Proc. of Int'l Conf. Machine Learning (ICML 2004)*: 225–232.

[32] Drineas, P.; A. Frieze; R. Kannan; S. Vempala; V. Vinay (2004). "Clustering large graphs via the singular value decomposition" (PDF). *Machine learning* **56**: 9–33. doi:10.1023/b:mach.0000033113.59016.96. Retrieved 2012-08-02.

[33] Cohen, M.; S. Elder; C. Musco; C. Musco; M. Persu (2014). "Dimensionality reduction for k-means clustering and low rank approximation (Appendix B)". *ArXiv*. Retrieved 2014-11-29.

[34] http://www.linkedin.com/groups/What-is-difference-between-factor-107833.S.162765950

[35] Timothy A. Brown. Confirmatory Factor Analysis for Applied Research Methodology in the social sciences. Guilford Press, 2006

[36] Benzécri, J.-P. (1973). *L'Analyse des Données. Volume II. L'Analyse des Correspondances*. Paris, France: Dunod.

[37] Greenacre, Michael (1983). *Theory and Applications of Correspondence Analysis*. London: Academic Press. ISBN 0-12-299050-1.

[38] Le Roux, Brigitte and Henry Rouanet (2004). *Geometric Data Analysis, From Correspondence Analysis to Structured Data Analysis*. Dordrecht: Kluwer.

[39] A. N. Gorban, A. Y. Zinovyev, Principal Graphs and Manifolds, In: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques, Olivas E.S. et al Eds. Information Science Reference, IGI Global: Hershey, PA, USA, 2009. 28–59.

[40] Wang, Y., Klijn, J.G., Zhang, Y., Sieuwerts, A.M., Look, M.P., Yang, F., Talantov, D., Timmermans, M., Meijer-van Gelder, M.E., Yu, J. et al.: Gene expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer" *Lancet* 365, 671–679 (2005); Data online

[41] A. Zinovyev, ViDaExpert – Multidimensional Data Visualization Tool (free for non-commercial use). Institut Curie, Paris.

[42] A.N. Gorban, B. Kegl, D.C. Wunsch, A. Zinovyev (Eds.), Principal Manifolds for Data Visualisation and Dimension Reduction, LNCSE 58, Springer, Berlin – Heidelberg – New York, 2007. ISBN 978-3-540-73749-0

[43] Lu, Haiping; Plataniotis, K.N.; Venetsanopoulos, A.N. (2011). "A Survey of Multilinear Subspace Learning for Tensor Data" (PDF). *Pattern Recognition* **44** (7): 1540–1551. doi:10.1016/j.patcog.2011.01.004.

[44] Kriegel, H. P.; Kröger, P.; Schubert, E.; Zimek, A. (2008). "A General Framework for Increasing the Robustness of PCA-Based Correlation Clustering Algorithms". *Scientific and Statistical Database Management*. Lecture Notes in Computer Science **5069**: 418. doi:10.1007/978-3-540-69497-7_27. ISBN 978-3-540-69476-2.

[45] http://www.kovcomp.co.uk/support/XL-Tut/

[46] PrincipalComponents Mathematica Documentation

[47] The Numerical Algorithms Group. "NAG Library Routine Document: nagf_mv_prin_comp (g03aaf)" (PDF). *NAG Library Manual, Mark 23*. Retrieved 2012-02-16.

[48] The Numerical Algorithms Group. "NAG Library Routine Document: nag_mv_prin_comp (g03aac)" (PDF). *NAG Library Manual, Mark 9*. Retrieved 2012-02-16.

[49] PcaPress http://www.umetrics.com/products/simca

[50] PcaPress www.utdallas.edu

[51] Oracle documentation http://docs.oracle.com

[52] princomp octave.sourceforge.net

[53] princomp

[54] prcomp

[55] Multivariate cran.r-project.org

[56] http://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_factor_sect028.htm

[57] "pca — Principal component analysis" (PDF). *Stata Manual*. Retrieved April 3, 2015.

[58] Cornell Spectrum Imager https://code.google.com/p/cornell-spectrum-imager/wiki/Home

[59] imDEV sourceforge.net

[60] pcaMethods www.bioconductor.org

[61] "ViSta: The Visual Statistics System" www.mdp.edu.ar

[62] "Spectramap" www.coloritto.com

[63] FinMath rtmath.net

[64] http://www.camo.com

[65] Computer Vision Library sourceforge.net

[66] PCOMP (IDL Reference) | Exelis VIS Docs Center IDL online documentation

[67] javadoc weka.sourceforge.net

[68] Software for analyzing multivariate data with instant response using PCA www.qlucore.com

[69] EIGENSOFT genepath.med.harvard.edu

[70] Partek Genomics Suite www.partek.com

[71] http://scikit-learn.org

[72]

[73] MultivariateStats.jl www.github.com

47.17 References

- Jackson, J.E. (1991). *A User's Guide to Principal Components* (Wiley).
- Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag. p. 487. doi:10.1007/b98835. ISBN 978-0-387-95442-4.
- Jolliffe, I.T. (2002). *Principal Component Analysis,* second edition (Springer).
- Husson François, Lê Sébastien & Pagès Jérôme (2009). *Exploratory Multivariate Analysis by Example Using R*. Chapman & Hall/CRC The R Series, London. 224p. isbn=978-2-7535-0938-2
- Pagès Jérôme (2014). *Multiple Factor Analysis by Example Using R*. Chapman & Hall/CRC The R Series London 272 p

47.18 External links

- University of Copenhagen video by Rasmus Bro on YouTube
- Stanford University video by Andrew Ng on YouTube
- A Tutorial on Principal Component Analysis
- A layman's introduction to principal component analysis on YouTube (a video of less than 100 seconds.)
- See also the list of Software implementations

Chapter 48

Dimensionality reduction

For dimensional reduction in physics, see Dimensional reduction.

In machine learning and statistics, **dimensionality reduction** or **dimension reduction** is the process of reducing the number of random variables under consideration,[1] and can be divided into feature selection and feature extraction.[2]

48.1 Feature selection

Main article: Feature selection

Feature selection approaches try to find a subset of the original variables (also called features or attributes). Two strategies are *filter* (e.g. information gain) and *wrapper* (e.g. search guided by the accuracy) approaches. See also combinatorial optimization problems.

In some cases, data analysis such as regression or classification can be done in the reduced space more accurately than in the original space.

48.2 Feature extraction

Main article: Feature extraction

Feature extraction transforms the data in the high-dimensional space to a space of fewer dimensions. The data transformation may be linear, as in principal component analysis (PCA), but many nonlinear dimensionality reduction techniques also exist.[3][4] For multidimensional data, tensor representation can be used in dimensionality reduction through multilinear subspace learning.[5]

The main linear technique for dimensionality reduction, principal component analysis, performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. In practice, the correlation matrix of the data is constructed and the eigenvectors on this matrix are computed. The eigenvectors that correspond to the largest eigenvalues (the principal components) can now be used to reconstruct a large fraction of the variance of the original data. Moreover, the first few eigenvectors can often be interpreted in terms of the large-scale physical behavior of the system. The original space (with dimension of the number of points) has been reduced (with data loss, but hopefully retaining the most important variance) to the space spanned by a few eigenvectors.

Principal component analysis can be employed in a nonlinear way by means of the kernel trick. The resulting technique is capable of constructing nonlinear mappings that maximize the variance in the data. The resulting technique is entitled kernel PCA. Other prominent nonlinear techniques include manifold learning techniques such as Isomap, locally linear embedding (LLE), Hessian LLE, Laplacian eigenmaps, and LTSA. These techniques construct a low-dimensional data representation using a cost function that retains local properties of the data, and can be viewed as defining a graph-based kernel for Kernel PCA. More recently, techniques have been proposed that, instead of defining a fixed kernel, try to learn the kernel using semidefinite programming. The most prominent example of such a technique is maximum variance unfolding (MVU). The central idea of MVU is to exactly preserve all pairwise distances between nearest neighbors (in the inner product space), while maximizing the distances between points that are not nearest neighbors. A dimensionality reduction technique that is sometimes used in neuroscience is maximally informative dimensions, which finds a lower-dimensional representation of a dataset such that as much information as possible about the original data is preserved.

An alternative approach to neighborhood preservation is through the minimization of a cost function that measures differences between distances in the input and output spaces. Important examples of such techniques include classical multidimensional scaling (which is identical to PCA), Isomap (which uses geodesic distances in the data space), diffusion maps (which uses diffusion distances in the data space), t-SNE (which minimizes the divergence between distributions over pairs of points), and curvilinear component analysis.

A different approach to nonlinear dimensionality reduction is through the use of autoencoders, a special kind of feed-forward neural networks with a bottle-neck hidden layer.[6] The training of deep encoders is typically performed using a greedy layer-wise pre-training (e.g., using a stack of restricted Boltzmann machines) that is followed by a finetuning stage based on backpropagation.

48.3 Dimension reduction

For high-dimensional datasets (i.e. with number of dimensions more than 10), dimension reduction is usually performed prior to applying a K-nearest neighbors algorithm(k-NN) in order to avoid the effects of the curse of dimensionality. [7]

Feature extraction and dimension reduction can be combined in one step using principal component analysis (PCA), linear discriminant analysis (LDA), or canonical correlation analysis (CCA) techniques as a pre-processing step followed by clustering by K-NN on feature vectors in reduced-dimension space. In machine learning this process is also called low-dimensional embedding.[8]

For very-high-dimensional datasets (e.g. when performing similarity search on live video streams, DNA data or high-dimensional Time series) running a fast **approximate** K-NN search using locality sensitive hashing, "random projections",[9] "sketches" [10] or other high-dimensional similarity search techniques from the VLDB toolbox might be the only feasible option.

48.4 See also

- Nearest neighbor search
- MinHash
- Information gain in decision trees
- Semidefinite embedding
- Multifactor dimensionality reduction
- Multilinear subspace learning
- Multilinear PCA
- Singular value decomposition
- Latent semantic analysis
- Semantic mapping
- Topological data analysis
- Locality sensitive hashing
- Sufficient dimension reduction
- Data transformation (statistics)
- Weighted correlation network analysis

48.5 Notes

[1] Roweis, S. T.; Saul, L. K. (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". *Science* **290** (5500): 2323–2326. doi:10.1126/science.290.5500.2323. PMID 11125150.

[2] Pudil, P.; Novovičová, J. (1998). "Novel Methods for Feature Subset Selection with Respect to Problem Knowledge". In Liu, Huan; Motoda, Hiroshi. *Feature Extraction, Construction and Selection*. p. 101. doi:10.1007/978-1-4615-5725-8_7. ISBN 978-1-4613-7622-4.

[3] Samet, H. (2006) *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann. ISBN 0-12-369446-9

[4] C. Ding, , X. He , H. Zha , H.D. Simon, Adaptive Dimension Reduction for Clustering High Dimensional Data,Proceedings of International Conference on Data Mining, 2002

[5] Lu, Haiping; Plataniotis, K.N.; Venetsanopoulos, A.N. (2011). "A Survey of Multilinear Subspace Learning for Tensor Data" (PDF). *Pattern Recognition* **44** (7): 1540–1551. doi:10.1016/j.patcog.2011.01.004.

[6] Hongbing Hu, Stephen A. Zahorian, (2010) "Dimensionality Reduction Methods for HMM Phonetic Recognition," ICASSP 2010, Dallas, TX

[7] Kevin Beyer , Jonathan Goldstein , Raghu Ramakrishnan , Uri Shaft (1999) "When is "nearest neighbor" meaningful?". *Database Theory—ICDT99*, 217-235

[8] Shaw, B.; Jebara, T. (2009). "Structure preserving embedding". *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09* (PDF). p. 1. doi:10.1145/1553374.1553494. ISBN 9781605585161.

[9] Bingham, E.; Mannila, H. (2001). "Random projection in dimensionality reduction". *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*. p. 245. doi:10.1145/502512.502546. ISBN 158113391X.

[10] Shasha, D High (2004) *Performance Discovery in Time Series* Berlin: Springer. ISBN 0-387-00857-8

48.6 References

- Fodor,I. (2002) "A survey of dimension reduction techniques". Center for Applied Scientific Computing, Lawrence Livermore National, Technical Report UCRL-ID-148494
- Cunningham, P. (2007) "Dimension Reduction" University College Dublin, Technical Report UCD-CSI-2007-7
- Zahorian, Stephen A.; Hu, Hongbing (2011). "Nonlinear Dimensionality Reduction Methods for Use

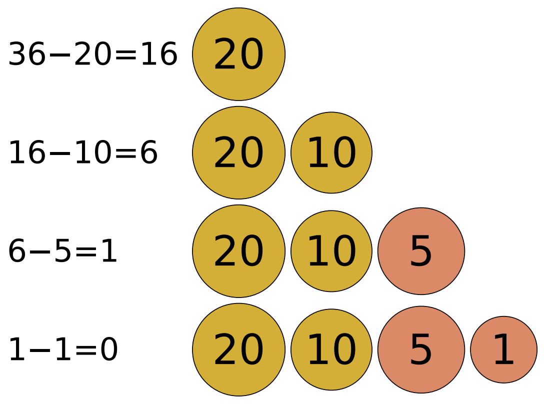
with Automatic Speech Recognition”. *Speech Technologies*. doi:10.5772/16863. ISBN 978-953-307-996-7.

48.7 External links

- JMLR Special Issue on Variable and Feature Selection
- ELastic MAPs
- Locally Linear Embedding
- A Global Geometric Framework for Nonlinear Dimensionality Reduction

Chapter 49

Greedy algorithm



Greedy algorithms determine minimum number of coins to give while making change. These are the steps a human would take to emulate a greedy algorithm to represent 36 cents using only coins with values {1, 5, 10, 20}. The coin of the highest value, less than the remaining change owed, is the local optimum. (Note that in general the change-making problem requires dynamic programming or integer programming to find an optimal solution; however, most currency systems, including the Euro and US Dollar, are special cases where the greedy strategy does find an optimal solution.)

A **greedy algorithm** is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage[1] with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

For example, a greedy strategy for the traveling salesman problem (which is of a high computational complexity) is the following heuristic: “At each stage visit an unvisited city nearest to the current city”. This heuristic need not find a best solution, but terminates in a reasonable number of steps; finding an optimal solution typically requires unreasonably many steps. In mathematical optimization, greedy algorithms solve combinatorial problems having the properties of matroids.

49.1 Specifics

In general, greedy algorithms have five components:

1. A candidate set, from which a solution is created
2. A selection function, which chooses the best candidate to be added to the solution
3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution
4. An objective function, which assigns a value to a solution, or a partial solution, and
5. A solution function, which will indicate when we have discovered a complete solution

Greedy algorithms produce good solutions on some mathematical problems, but not on others. Most problems for which they work will have two properties:

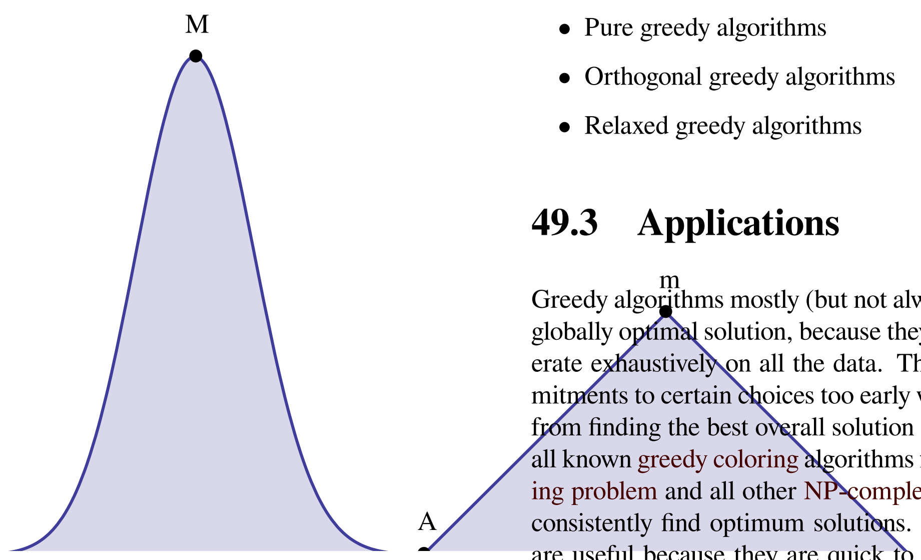
Greedy choice property We can make whatever choice seems best at the moment and then solve the subproblems that arise later. The choice made by a greedy algorithm may depend on choices made so far, but not on future choices or all the solutions to the subproblem. It iteratively makes one greedy choice after another, reducing each given problem into a smaller one. In other words, a greedy algorithm never reconsiders its choices. This is the main difference from dynamic programming, which is exhaustive and is guaranteed to find the solution.

After every stage, dynamic programming makes decisions based on all the decisions made in the previous stage, and may reconsider the previous stage’s algorithmic path to solution.

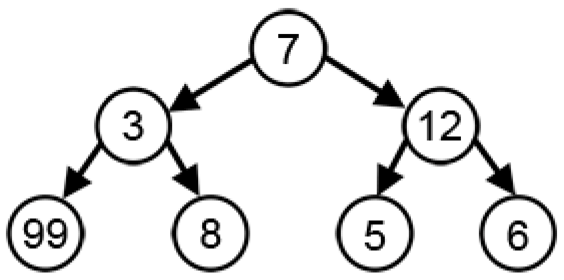
Optimal substructure “A problem exhibits optimal substructure if an optimal solution to the problem contains optimal solutions to the sub-problems.”[2]

49.1.1 Cases of failure

Examples on how a greedy algorithm may fail to achieve the optimal solution.



Starting at A, a greedy algorithm will find the local maximum at “m”, oblivious of the global maximum at “M”.



With a goal of reaching the largest-sum, at each step, the greedy algorithm will choose what appears to be the optimal immediate choice, so it will choose 12 instead of 3 at the second step, and will not reach the best solution, which contains 99.

For many other problems, greedy algorithms fail to produce the optimal solution, and may even produce the *unique worst possible* solution. One example is the traveling salesman problem mentioned above: for each number of cities, there is an assignment of distances between the cities for which the nearest neighbor heuristic produces the unique worst possible tour.[3]

49.2 Types

Greedy algorithms can be characterized as being 'short sighted', and also as 'non-recoverable'. They are ideal only for problems which have 'optimal substructure'. Despite this, for many simple problems (e.g. giving change), the best suited algorithms are greedy algorithms. It is important, however, to note that the greedy algorithm can be used as a selection algorithm to prioritize options within a search, or branch and bound algorithm. There are a few variations to the greedy algorithm:

- Pure greedy algorithms
- Orthogonal greedy algorithms
- Relaxed greedy algorithms

49.3 Applications

Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. They can make commitments to certain choices too early which prevent them from finding the best overall solution later. For example, all known greedy coloring algorithms for the graph coloring problem and all other NP-complete problems do not consistently find optimum solutions. Nevertheless, they are useful because they are quick to think up and often give good approximations to the optimum.

If a greedy algorithm can be proven to yield the global optimum for a given problem class, it typically becomes the method of choice because it is faster than other optimization methods like dynamic programming. Examples of such greedy algorithms are Kruskal’s algorithm and Prim’s algorithm for finding minimum spanning trees, and the algorithm for finding optimum Huffman trees.

The theory of matroids, and the more general theory of greedoids, provide whole classes of such algorithms.

Greedy algorithms appear in network routing as well. Using greedy routing, a message is forwarded to the neighboring node which is “closest” to the destination. The notion of a node’s location (and hence “closeness”) may be determined by its physical location, as in geographic routing used by ad hoc networks. Location may also be an entirely artificial construct as in small world routing and distributed hash table.

49.4 Examples

- The activity selection problem is characteristic to this class of problems, where the goal is to pick the maximum number of activities that do not clash with each other.
- In the Macintosh computer game *Crystal Quest* the objective is to collect crystals, in a fashion similar to the travelling salesman problem. The game has a demo mode, where the game uses a greedy algorithm to go to every crystal. The artificial intelligence does not account for obstacles, so the demo mode often ends quickly.

- The matching pursuit is an example of greedy algorithm applied on signal approximation.
- A greedy algorithm finds the optimal solution to Malfatti's problem of finding three disjoint circles within a given triangle that maximize the total area of the circles; it is conjectured that the same greedy algorithm is optimal for any number of circles.
- A greedy algorithm is used to construct a Huffman tree during Huffman coding where it finds an optimal solution.
- In decision tree learning, greedy algorithms are commonly used, however they are not guaranteed to find the optimal solution.

49.5 See also

- Epsilon-greedy strategy
- Greedy algorithm for Egyptian fractions
- Greedy source
- Matroid

49.6 Notes

[1] Black, Paul E. (2 February 2005). "greedy algorithm". *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology (NIST). Retrieved 17 August 2012.

[2] Introduction to Algorithms (Cormen, Leiserson, Rivest, and Stein) 2001, Chapter 16 "Greedy Algorithms".

[3] (G. Gutin, A. Yeo and A. Zverovich, 2002)

49.7 References

- *Introduction to Algorithms* (Cormen, Leiserson, and Rivest) 1990, Chapter 17 "Greedy Algorithms" p. 329.
- *Introduction to Algorithms* (Cormen, Leiserson, Rivest, and Stein) 2001, Chapter 16 "Greedy Algorithms".
- G. Gutin, A. Yeo and A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discrete Applied Mathematics 117 (2002), 81–86.
- J. Bang-Jensen, G. Gutin and A. Yeo, When the greedy algorithm fails. Discrete Optimization 1 (2004), 121–127.
- G. Bendall and F. Margot, Greedy Type Resistance of Combinatorial Problems, Discrete Optimization 3 (2006), 288–298.

49.8 External links

- Hazewinkel, Michiel, ed. (2001), "Greedy algorithm", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Greedy algorithm visualization A visualization of a greedy solution to the N-Queens puzzle by Yuval Baror.
- Python greedy coin example by Noah Gift.
- Java implementation used in a Checkers Game

Chapter 50

Reinforcement learning

For reinforcement learning in psychology, see Reinforcement.

Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take *actions* in an *environment* so as to maximize some notion of cumulative *reward*. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics, and genetic algorithms. In the operations research and control literature, the field where reinforcement learning methods are studied is called *approximate dynamic programming*. The problem has been studied in the theory of optimal control, though most studies are concerned with the existence of optimal solutions and their characterization, and not with the learning or approximation aspects. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality.

In machine learning, the environment is typically formulated as a Markov decision process (MDP) as many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical techniques and reinforcement learning algorithms is that the latter do not need knowledge about the MDP and they target large MDPs where exact methods become infeasible.

Reinforcement learning differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The exploration vs. exploitation trade-off in reinforcement learning has been most thoroughly studied through the multi-armed bandit problem and in finite MDPs.

50.1 Introduction

The basic reinforcement learning model consists of:

1. a set of environment states S ;
2. a set of actions A ;
3. rules of transitioning between states;
4. rules that determine the *scalar immediate reward* of a transition; and
5. rules that describe what the agent observes.

The rules are often stochastic. The observation typically involves the scalar immediate reward associated with the last transition. In many works, the agent is also assumed to observe the current environmental state, in which case we talk about *full observability*, whereas in the opposing case we talk about *partial observability*. Sometimes the set of actions available to the agent is restricted (e.g., you cannot spend more money than what you possess).

A reinforcement learning agent interacts with its environment in discrete time steps. At each time t , the agent receives an observation o_t , which typically includes the reward r_t . It then chooses an action a_t from the set of actions available, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the *transition* (s_t, a_t, s_{t+1}) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible. The agent can choose any action as a function of the history and it can even randomize its action selection.

When the agent's performance is compared to that of an agent which acts optimally from the beginning, the difference in performance gives rise to the notion of *regret*. Note that in order to act near optimally, the agent must reason about the long term consequences of its actions: In order to maximize my future income I had better go to school now, although the immediate monetary reward associated with this might be negative.

Thus, reinforcement learning is particularly well suited to problems which include a long-term versus short-term reward trade-off. It has been applied successfully to various problems, including robot control, elevator scheduling, telecommunications, backgammon and checkers (Sutton and Barto 1998, Chapter 11).

Two components make reinforcement learning powerful: The use of samples to optimize performance and the use of function approximation to deal with large environments. Thanks to these two key components, reinforcement learning can be used in large environments in any of the following situations:

- A model of the environment is known, but an analytic solution is not available;
- Only a simulation model of the environment is given (the subject of simulation-based optimization);[1]
- The only way to collect information about the environment is by interacting with it.

The first two of these problems could be considered planning problems (since some form of the model is available), while the last one could be considered as a genuine learning problem. However, under a reinforcement learning methodology both planning problems would be converted to machine learning problems.

50.2 Exploration

The reinforcement learning problem as described requires clever exploration mechanisms. Randomly selecting actions, without reference to an estimated probability distribution, is known to give rise to very poor performance. The case of (small) finite MDPs is relatively well understood by now. However, due to the lack of algorithms that would provably scale well with the number of states (or scale to problems with infinite state spaces), in practice people resort to simple exploration methods. One such method is ϵ -greedy, when the agent chooses the action that it believes has the best long-term effect with probability $1-\epsilon$, and it chooses an action uniformly at random, otherwise. Here, $0<\epsilon<1$ is a tuning parameter, which is sometimes changed, either according to a fixed schedule (making the agent explore less as time goes by), or adaptively based on some heuristics (Tokic & Palm, 2011).

50.3 Algorithms for control learning

Even if the issue of exploration is disregarded and even if the state was observable (which we assume from now on), the problem remains to find out which actions are good based on past experience.

50.3.1 Criterion of optimality

For simplicity, assume for a moment that the problem studied is *episodic*, an episode ending when some *terminal state* is reached. Assume further that no matter what course of actions the agent takes, termination is inevitable. Under some additional mild regularity conditions the expectation of the total reward is then well-defined, for *any* policy and any initial distribution over the states. Here, a policy refers to a mapping that assigns some probability distribution over the actions to all possible histories.

Given a fixed initial distribution μ , we can thus assign the expected return ρ^π to policy π :

$$\rho^\pi = E[R|\pi],$$

where the random variable R denotes the *return* and is defined by

$$R=\sum_{t=0}^{N-1} r_{t+1},$$

where r_{t+1} is the reward received after the t -th transition, the initial state is sampled at random from μ and actions are selected by policy π . Here, N denotes the (random) time when a terminal state is reached, i.e., the time when the episode terminates.

In the case of non-episodic problems the return is often *discounted*,

$$R=\sum_{t=0}^{\infty} \gamma^t r_{t+1},$$

giving rise to the total expected discounted reward criterion. Here $0 \le \gamma \le 1$ is the so-called *discount-factor*. Since the undiscounted return is a special case of the discounted return, from now on we will assume discounting. Although this looks innocent enough, discounting is in fact problematic if one cares about online performance. This is because discounting makes the initial time steps more important. Since a learning agent is likely to make mistakes during the first few steps after its “life” starts, no uninformed learning algorithm can achieve near-optimal performance under discounting even if the class of environments is restricted to that of finite MDPs. (This does not mean though that, given enough time, a learning agent cannot figure how to act near-optimally, if time was restarted.)

The problem then is to specify an algorithm that can be used to find a policy with maximum expected return. From the theory of MDPs it is known that, without loss of generality, the search can be restricted to the set of the so-called *stationary* policies. A policy is called stationary if the action-distribution returned by it depends only on the last state visited (which is part of the observation history of the agent, by our simplifying assumption). In fact, the search can be further restricted to *deterministic* stationary policies. A deterministic stationary policy is one which deterministically selects actions based on the current state. Since any such policy can be identified with a mapping from the set of states to the set of actions, these policies can be identified with such mappings with no loss of generality.

50.3.2 Brute force

The brute force approach entails the following two steps:

1. For each possible policy, sample returns while following it
2. Choose the policy with the largest expected return

One problem with this is that the number of policies can be extremely large, or even infinite. Another is that variance of the returns might be large, in which case a large number of samples will be required to accurately estimate the return of each policy.

These problems can be ameliorated if we assume some structure and perhaps allow samples generated from one policy to influence the estimates made for another. The two main approaches for achieving this are value function estimation and direct policy search.

50.3.3 Value function approaches

Value function approaches attempt to find a policy that maximizes the return by maintaining a set of estimates of expected returns for some policy (usually either the "current" or the optimal one).

These methods rely on the theory of MDPs, where optimality is defined in a sense which is stronger than the above one: A policy is called optimal if it achieves the best expected return from *any* initial state (i.e., initial distributions play no role in this definition). Again, one can always find an optimal policy amongst stationary policies.

To define optimality in a formal manner, define the value of a policy π by

$$V^\pi(s) = E[R|s,\pi],$$

where R stands for the random return associated with following π from the initial state s . Define $V^*(s)$ as the maximum possible value of $V^\pi(s)$, where π is allowed to change:

$$V^*(s) = \sup_\pi V^\pi(s).$$

A policy which achieves these *optimal values* in *each* state is called *optimal*. Clearly, a policy optimal in this strong sense is also optimal in the sense that it maximizes the expected return ρ^π , since $\rho^\pi = E[V^\pi(S)]$, where S is a state randomly sampled from the distribution μ .

Although state-values suffice to define optimality, it will prove to be useful to define action-values. Given a state s , an action a and a policy π , the action-value of the pair (s,a) under π is defined by

$$Q^\pi(s,a) = E[R|s,a,\pi],$$

where, now, R stands for the random return associated with first taking action a in state s and following π , thereafter.

It is well-known from the theory of MDPs that if someone gives us Q for an optimal policy, we can always choose optimal actions (and thus act optimally) by simply choosing the action with the highest value at each state. The *action-value function* of such an optimal policy is called the *optimal action-value function* and is denoted by Q^* . In summary, the knowledge of the optimal action-value function *alone* suffices to know how to act optimally.

Assuming full knowledge of the MDP, there are two basic approaches to compute the optimal action-value function, value iteration and policy iteration. Both algorithms compute a sequence of functions Q_k ($k = 0,1,2,\ldots,$) which converge to Q^* . Computing these functions involves computing expectations over the whole state-space, which is impractical for all, but the smallest (finite) MDPs, never mind the case when the MDP is unknown. In reinforcement learning methods the expectations are approximated by averaging over samples and one uses function approximation techniques to cope with the need to represent value functions over large state-action spaces.

Monte Carlo methods

The simplest Monte Carlo methods can be used in an algorithm that mimics policy iteration. Policy iteration consists of two steps: *policy evaluation* and *policy improvement*. The Monte Carlo methods are used in the policy evaluation step. In this step, given a stationary, deterministic policy π , the goal is to compute the function values $Q^\pi(s,a)$ (or a good approximation to them) for all state-action pairs (s,a) . Assume (for simplicity) that the MDP is finite and in fact a table representing the action-values fits into the memory. Further, assume that the problem is episodic and after each episode a new one starts from some random initial state. Then, the estimate of the value of a given state-action pair (s,a) can be computed by simply averaging the sampled returns which originated from (s, a) over time. Given enough time, this procedure can thus construct a precise estimate Q of the action-value function Q^π . This finishes the description of the policy evaluation step. In the policy improvement step, as it is done in the standard policy iteration algorithm, the next policy is obtained by computing a *greedy* policy with respect to Q : Given a state s , this new policy returns an action that maximizes $Q(s, \cdot)$. In practice one often avoids computing and storing the new policy, but uses lazy evaluation to defer the computation of the maximizing actions to when they are actually needed.

A few problems with this procedure are as follows:

- The procedure may waste too much time on evaluating a suboptimal policy;
- It uses samples inefficiently in that a long trajectory is used to improve the estimate only of the *single* state-action pair that started the trajectory;
- When the returns along the trajectories have *high variance*, convergence will be slow;
- It works in *episodic problems only*;
- It works in *small, finite MDPs only*.

Temporal difference methods

The first issue is easily corrected by allowing the procedure to change the policy (at all, or at some states) before the values settle. However good this sounds, this may be dangerous as this might prevent convergence. Still, most current algorithms implement this idea, giving rise to the class of *generalized policy iteration* algorithm. We note in passing that actor critic methods belong to this category.

The second issue can be corrected within the algorithm by allowing trajectories to contribute to any state-action pair in them. This may also help to some extent with the third problem, although a better solution when returns have high variance is to use Sutton's temporal difference (TD) methods which are based on the recursive Bellman equation. Note that the computation in TD methods can be incremental (when after each transition the memory is changed and the transition is thrown away), or batch (when the transitions are collected and then the estimates are computed once based on a large number of transitions). Batch methods, a prime example of which is the least-squares temporal difference method due to Bradtke and Barto (1996), may use the information in the samples better, whereas incremental methods are the only choice when batch methods become infeasible due to their high computational or memory complexity. In addition, there exist methods that try to unify the advantages of the two approaches. Methods based on temporal differences also overcome the second but last issue.

In order to address the last issue mentioned in the previous section, *function approximation methods* are used. In *linear function approximation* one starts with a mapping ϕ that assigns a finite-dimensional vector to each state-action pair. Then, the action values of a state-action pair (s, a) are obtained by linearly combining the components of $\phi(s, a)$ with some *weights* θ :

$$Q(s, a) = \sum_{i=1}^{d} \theta_i \phi_i(s, a)$$

The algorithms then adjust the weights, instead of adjusting the values associated with the individual state-action pairs. However, linear function approximation is not the only choice. More recently, methods based on ideas from nonparametric statistics (which can be seen to construct their own features) have been explored.

So far, the discussion was restricted to how policy iteration can be used as a basis of the designing reinforcement learning algorithms. Equally importantly, value iteration can also be used as a starting point, giving rise to the Q-Learning algorithm (Watkins 1989) and its many variants.

The problem with methods that use action-values is that they may need highly precise estimates of the competing action values, which can be hard to obtain when the returns are noisy. Though this problem is mitigated to some extent by temporal difference methods and if one uses the so-called compatible function approximation method, more work remains to be done to increase generality and efficiency. Another problem specific to temporal difference methods comes from their reliance on the recursive Bellman equation. Most temporal difference methods have a so-called λ parameter $(0 \leq \lambda \leq 1)$ that allows one to continuously interpolate between Monte-Carlo methods (which do not rely on the Bellman equations) and the basic temporal difference methods (which rely entirely on the Bellman equations), which can thus be effective in palliating this issue.

50.3.4 Direct policy search

An alternative method to find a good policy is to search directly in (some subset of) the policy space, in which case the problem becomes an instance of stochastic optimization. The two approaches available are gradient-based and gradient-free methods.

Gradient-based methods (giving rise to the so-called *policy gradient methods*) start with a mapping from a finite-dimensional (parameter) space to the space of policies: given the parameter vector θ , let π_θ denote the policy associated to θ . Define the performance function by

$$\rho(\theta) = \rho^{\pi_\theta}.$$

Under mild conditions this function will be differentiable as a function of the parameter vector θ . If the gradient of ρ was known, one could use gradient ascent. Since an analytic expression for the gradient is not available, one must rely on a noisy estimate. Such an estimate can be constructed in many ways, giving rise to algorithms like Williams' REINFORCE method (which is also known as the likelihood ratio method in the simulation-based optimization literature). Policy gradient methods have received a lot of attention in the last couple of years (e.g., Peters et al. (2003)), but they remain an active field. An overview of policy search methods in the context of robotics has been given by Deisenroth, Neumann and Peters.[2] The issue with many of these methods is that they may get stuck in local optima (as they are based on local search).

A large class of methods avoids relying on gradient information. These include simulated annealing, cross-entropy search or methods of evolutionary computation. Many gradient-free methods can achieve (in theory and in the limit) a global optimum. In a number of cases they have indeed demonstrated remarkable performance.

The issue with policy search methods is that they may converge slowly if the information based on which they act is noisy. For example, this happens when in episodic problems the trajectories are long and the variance of the returns is large. As argued beforehand, value-function based methods that rely on temporal differences might help in this case. In recent years, several actor-critic algorithms have been proposed following this idea and were demonstrated to perform well in various problems.

50.4 Theory

The theory for small, finite MDPs is quite mature. Both the asymptotic and finite-sample behavior of most algorithms is well-understood. As mentioned beforehand, algorithms with provably good online performance (addressing the exploration issue) are known. The theory of large MDPs needs more work. Efficient exploration is largely untouched (except for the case of bandit problems). Although finite-time performance bounds appeared for many algorithms in the recent years, these bounds are expected to be rather loose and thus more work is needed to better understand the relative advantages, as well as the limitations of these algorithms. For incremental algorithm asymptotic convergence issues have been settled. Recently, new incremental, temporal-difference-based algorithms have appeared which converge under a much wider set of conditions than was previously possible (for example, when used with arbitrary, smooth function approximation).

50.5 Current research

Current research topics include: adaptive methods which work with fewer (or no) parameters under a large number of conditions, addressing the exploration problem in large MDPs, large-scale empirical evaluations, learning and acting under partial information (e.g., using Predictive State Representation), modular and hierarchical reinforcement learning, improving existing value-function and policy search methods, algorithms that work well with large (or continuous) action spaces, transfer learning, lifelong learning, efficient sample-based planning (e.g., based on Monte-Carlo tree search). Multiagent or Distributed Reinforcement Learning is also a topic of interest in current research. There is also a growing interest in real life applications of reinforcement learning. Successes of reinforcement learning are collected on here and here.

Reinforcement learning algorithms such as TD learning are also being investigated as a model for Dopamine-based learning in the brain. In this model, the dopaminergic projections from the substantia nigra to the basal ganglia function as the prediction error. Reinforcement learning has also been used as a part of the model for human skill learning, especially in relation to the interaction between implicit and explicit learning in skill acquisition (the first publication on this application was in 1995-1996, and there have been many follow-up studies). See http://webdocs.cs.ualberta.ca/~{ }sutton/RL-FAQ.html#behaviorism for further details of these research areas above.

50.6 Literature

50.6.1 Conferences, journals

Most reinforcement learning papers are published at the major machine learning and AI conferences (ICML, NIPS, AAAI, IJCAI, UAI, AI and Statistics) and journals (JAIR, JMLR, Machine learning journal, IEEE T-CIAIG). Some theory papers are published at COLT and ALT. However, many papers appear in robotics conferences (IROS, ICRA) and the "agent" conference AAMAS. Operations researchers publish their papers at the INFORMS conference and, for example, in the Operation Research, and the Mathematics of Operations Research journals. Control researchers publish their papers at the CDC and ACC conferences, or, e.g., in the journals IEEE Transactions on Automatic Control, or Automatica, although applied works tend to be published in more specialized journals. The Winter Simulation Conference also publishes many relevant papers. Other than this, papers also published in the major conferences of the neural networks, fuzzy, and evolutionary computation communities. The annual IEEE symposium titled Approximate Dynamic Programming and Reinforcement Learning (ADPRL) and the biannual European Workshop on Reinforcement Learning (EWRL) are two regularly held meetings where RL researchers meet.

50.7 See also

- Temporal difference learning
- Q-learning
- SARSA
- Fictitious play
- Learning classifier system
- Optimal control
- Dynamic treatment regimes
- Error-driven learning
- Multi-agent system
- Distributed artificial intelligence

50.8 Implementations

- RL-Glue provides a standard interface that allows you to connect agents, environments, and experiment programs together, even if they are written in different languages.
- Maja Machine Learning Framework The Maja Machine Learning Framework (MMLF) is a general framework for problems in the domain of Reinforcement Learning (RL) written in python.
- Software Tools for Reinforcement Learning (Matlab and Python)
- PyBrain(Python)
- TeachingBox is a Java reinforcement learning framework supporting many features like RBF networks, gradient descent learning methods, ...
- C++ and Python implementations for some well known reinforcement learning algorithms with source.
- Orange, a free data mining software suite, module orngReinforcement
- Policy Gradient Toolbox provides a package for learning about policy gradient approaches.
- BURLAP is an open source Java library that provides a wide range of single and multi-agent learning and planning methods.

50.9 References

- Sutton, Richard S. (1984). *Temporal Credit Assignment in Reinforcement Learning* (PhD thesis). University of Massachusetts, Amherst, MA.
- Williams, Ronald J. (1987). "A class of gradient-estimating algorithms for reinforcement learning in neural networks". *Proceedings of the IEEE First International Conference on Neural Networks.*
- Sutton, Richard S. (1988). "Learning to predict by the method of temporal differences". *Machine Learning* (Springer) **3**: 9–44. doi:10.1007/BF00115009.
- Watkins, Christopher J.C.H. (1989). *Learning from Delayed Rewards* (PDF) (PhD thesis). King's College, Cambridge, UK.
- Bradtke, Steven J.; Andrew G. Barto (1996). "Learning to predict by the method of temporal differences". *Machine Learning* (Springer) **22**: 33–57. doi:10.1023/A:1018056104778.
- Bertsekas, Dimitri P.; John Tsitsiklis (1996). *Neuro-Dynamic Programming*. Nashua, NH: Athena Scientific. ISBN 1-886529-10-8.
- Kaelbling, Leslie P.; Michael L. Littman; Andrew W. Moore (1996). "Reinforcement Learning: A Survey". *Journal of Artificial Intelligence Research* **4**: 237–285.
- Sutton, Richard S.; Barto, Andrew G. (1998). *Reinforcement Learning: An Introduction*. MIT Press. ISBN 0-262-19398-1.
- Peters, Jan; Sethu Vijayakumar; Stefan Schaal (2003). "Reinforcement Learning for Humanoid Robotics" (PDF). *IEEE-RAS International Conference on Humanoid Robots.*
- Powell, Warren (2007). *Approximate dynamic programming: solving the curses of dimensionality*. Wiley-Interscience. ISBN 0-470-17155-3.
- Auer, Peter; Thomas Jaksch; Ronald Ortner (2010). "Near-optimal regret bounds for reinforcement learning". *Journal of Machine Learning Research* **11**: 1563–1600.
- Szita, Istvan; Csaba Szepesvari (2010). "Model-based Reinforcement Learning with Nearly Tight Exploration Complexity Bounds" (PDF). *ICML 2010*. Omnipress. pp. 1031–1038.

- Bertsekas, Dimitri P. (August 2010). "Chapter 6 (online): Approximate Dynamic Programming". *Dynamic Programming and Optimal Control* (PDF) **II** (3 ed.).

- Busoniu, Lucian; Robert Babuska ; Bart De Schutter ; Damien Ernst (2010). *Reinforcement Learning and Dynamic Programming using Function Approximators*. Taylor & Francis CRC Press. ISBN 978-1-4398-2108-4.

- Tokic, Michel; Günther Palm ; (2011). "Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax". *KI 2011: Advances in Artificial Intelligence* (PDF). Lecture Notes in Computer Science **7006**. Springer Berlin / Heidelberg. pp. 335–346.

- Röttger, Michael C.; Andreas W. Liehr (2009). "Control task for Reinforcement Learning with known optimal solution for discrete and continuous actions". *Journal of Intelligent Learning Systems and Applications* **1**: 26–39. doi:10.4236/jilsa.2009.11002.

- Deisenroth, Marc Peter; Gerhard Neumann; Jan Peters (2013). *A Survey on Policy Search for Robotics*. Foundations and Trends in Robotics **2**. NOW Publishers. pp. 1–142.

[1] Gosavi, Abhijit (2003). *Simulation-based Optimization: Parametric Optimization Techniques and Reinforcement*. Springer. ISBN 1-4020-7454-9.

[2] Deisenroth, Marc Peter; Neumann, Gerhard; Peters, Jan (2013). *A Survey on Policy Search for Robotics*. NOW Publishers. pp. 1–142. ISBN 978-1-60198-702-0.

50.10 External links

- Website for *Reinforcement Learning: An Introduction* (1998), by Rich Sutton and Andrew Barto, MIT Press, including a link to an html version of the book.

- Reinforcement Learning Repository

- Reinforcement Learning and Artificial Intelligence (RLAI, Rich Sutton's lab at the University of Alberta)

- Autonomous Learning Laboratory (ALL, Andrew Barto's lab at the University of Massachusetts Amherst)

- RL-Glue

- Software Tools for Reinforcement Learning (Matlab and Python)

- The Reinforcement Learning Toolbox from the (Graz University of Technology)

- Hybrid reinforcement learning

- Piqle: a Generic Java Platform for Reinforcement Learning

- A Short Introduction To Some Reinforcement Learning Algorithms

- Reinforcement Learning applied to Tic-Tac-Toe Game

- Scholarpedia Reinforcement Learning

- Scholarpedia Temporal Difference Learning

- Stanford Reinforcement Learning Course

- Real-world reinforcement learning experiments at Delft University of Technology

- Reinforcement Learning Tools for Matlab

- Stanford University Andrew Ng Lecture on Reinforcement Learning

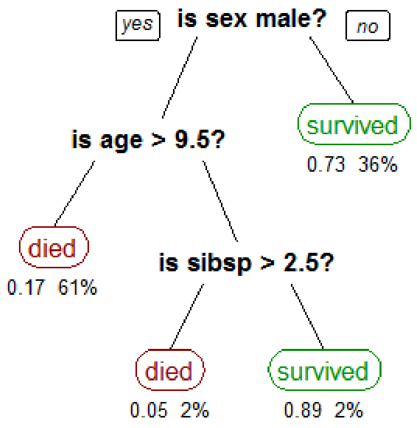
Chapter 51

Decision tree learning

This article is about decision trees in machine learning. For the use of the term in decision analysis, see Decision tree.

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called **classification trees**. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called **regression trees**.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. This page deals with decision trees in data mining.



A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

51.1 General

Decision tree learning is a method commonly used in data mining.[1] The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown on the right. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

A decision tree is a simple representation for classifying examples. Decision tree learning is one of the most successful techniques for supervised classification learning. For this section, assume that all of the features have finite discrete domains, and there is a single target feature called the classification. Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of *top-down induction of decision trees* (TDIDT) [2] is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.

In data mining, decision trees can be described also as the combination of mathematical and computational techniques to aid the description, categorisation and generalisation of a given set of data.

Data comes in records of the form:

$(\mathbf{x}, Y) = (x_1, x_2, x_3, ..., x_k, Y)$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector **x** is composed of the input variables, x_1, x_2, x_3 etc., that are used for that task.

51.2 Types

Decision trees used in data mining are of two main types:

- **Classification tree** analysis is when the predicted outcome is the class to which the data belongs.
- **Regression tree** analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

The term **Classification And Regression Tree (CART)** analysis is an umbrella term used to refer to both of the above procedures, first introduced by Breiman et al.[3] Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split.[3]

Some techniques, often called *ensemble* methods, construct more than one decision tree:

- **Bagging** decision trees, an early ensemble method, builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction.[4]
- A **Random Forest** classifier uses a number of decision trees, in order to improve the classification rate.
- **Boosted Trees** can be used for regression-type and classification-type problems.[5][6]
- **Rotation forest** - in which every decision tree is trained by first applying principal component analysis (PCA) on a random subset of the input features.[7]

Decision tree learning is the construction of a decision tree from class-labeled training tuples. A decision tree is a flow-chart-like structure, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf (or terminal) node holds a class label. The topmost node in a tree is the root node.

There are many specific decision-tree algorithms. Notable ones include:

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.[8]
- MARS: extends decision trees to handle numerical data better.
- Conditional Inference Trees. Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning.[9][10]

ID3 and CART were invented independently at around the same time (between 1970 and 1980), yet follow a similar approach for learning decision tree from training tuples.

51.3 Metrics

Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items.[11] Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets. Some examples are given below. These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

51.3.1 Gini impurity

Not to be confused with Gini coefficient.

Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. Gini impurity can be computed by summing the probability of each item being chosen times the probability of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

To compute Gini impurity for a set of items, suppose $i \in \{1, 2, ..., m\}$, and let f_i be the fraction of items labeled with value i in the set.

$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m}(f_i - {f_i}^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} {f_i}^2 = 1 - \sum_{i=1}^{m} {f_i}^2$

51.3.2 Information gain

Main article: Information gain in decision trees

Used by the ID3, C4.5 and C5.0 tree-generation algorithms. Information gain is based on the concept of entropy from information theory.

$I_E(f) = -\sum_{i=1}^{m} f_i \log_2 f_i$

51.3.3 Variance reduction

Introduced in CART,[3] variance reduction is often employed in cases where the target variable is continuous (regression tree), meaning that use of many other metrics would first require discretization before being applied. The variance reduction of a node N is defined as the total reduction of the variance of the target variable x due to the split at this node:

$$I_V(N) = \frac{1}{|S|^2}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|^2}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|^2}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i - x_j)^2\right)$$

where S , S_t , and S_f are the set of presplit sample indices, set of sample indices for which the split test is true, and set of sample indices for which the split test is false, respectively. Each of the above summands are indeed variance estimates, though, written in a form without directly referring to the mean.

51.4 Decision tree advantages

Amongst other data mining methods, decision trees have various advantages:

- **Simple to understand and interpret.** People are able to understand decision tree models after a brief explanation.
- **Requires little data preparation.** Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed.
- **Able to handle both numerical and categorical data.** Other techniques are usually specialised in analysing datasets that have only one type of variable. (For example, relation rules can be used only with nominal variables while neural networks can be used only with numerical variables.)
- **Uses a white box model.** If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. (An example of a black box model is an artificial neural network since the explanation for the results is difficult to understand.)
- **Possible to validate a model using statistical tests.** That makes it possible to account for the reliability of the model.
- **Robust.** Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.
- **Performs well with large datasets.** Large amounts of data can be analysed using standard computing resources in reasonable time.

51.5 Limitations

- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts.[12][13] Consequently, practical decision-tree learning algorithms are based on heuristics such as the greedy algorithm where locally-optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally-optimal decision tree. To reduce the greedy effect of local-optimality some methods such as the dual information distance (DID) tree were proposed.[14]
- Decision-tree learners can create over-complex trees that do not generalise well from the training data. (This is known as overfitting.[15]) Mechanisms such as pruning are necessary to avoid this problem (with the exception of some algorithms such as the Conditional Inference approach, that does not require pruning [9][10]).
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems. In such cases, the decision tree becomes prohibitively large. Approaches to solve the problem involve either changing the representation of the problem domain (known as propositionalisation)[16] or using learning algorithms based on more expressive representations (such as statistical relational learning or inductive logic programming).
- For data including categorical variables with different numbers of levels, information gain in decision trees is biased in favor of those attributes with more levels.[17] However, the issue of biased predictor selection is avoided by the Conditional Inference approach.[9]

51.6 Extensions

51.6.1 Decision graphs

In a decision tree, all paths from the root node to the leaf node proceed by way of conjunction, or *AND*. In a decision graph, it is possible to use disjunctions (ORs) to join two more paths together using Minimum message length (MML).[18] Decision graphs have been further extended to allow for previously unstated new attributes to be learnt dynamically and used at different places within the graph.[19] The more general coding scheme results in better predictive accuracy and log-loss probabilistic scoring. In general, decision graphs infer models with fewer leaves than decision trees.

51.6.2 Alternative search methods

Evolutionary algorithms have been used to avoid local optimal decisions and search the decision tree space with little *a priori* bias.[20][21]

It is also possible for a tree to be sampled using MCMC.[22]

The tree can be searched for in a bottom-up fashion.[23]

51.7 See also

- Decision tree pruning
- Binary decision diagram
- CHAID
- CART
- ID3 algorithm
- C4.5 algorithm
- Decision stump
- Incremental decision tree
- Alternating decision tree
- Structured data analysis (statistics)

51.8 Implementations

Many data mining software packages provide implementations of one or more decision tree algorithms. Several examples include Salford Systems CART (which licensed the proprietary code of the original CART authors[3]), IBM SPSS Modeler, RapidMiner, SAS Enterprise Miner, Matlab, R (an open source software environment for statistical computing which includes several CART implementations such as rpart, party and randomForest packages), Weka (a free and open-source data mining suite, contains many decision tree algorithms), Orange (a free data mining software suite, which includes the tree module orngTree), KNIME, Microsoft SQL Server , and scikit-learn (a free and open-source machine learning library for the Python programming language).

51.9 References

[1] Rokach, Lior; Maimon, O. (2008). *Data mining with decision trees: theory and applications*. World Scientific Pub Co Inc. ISBN 978-9812771711.

[2] Quinlan, J. R., (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers

[3] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. ISBN 978-0-412-04841-8.

[4] Breiman, L. (1996). Bagging Predictors. "Machine Learning, 24": pp. 123-140.

[5] Friedman, J. H. (1999). *Stochastic gradient boosting*. Stanford University.

[6] Hastie, T., Tibshirani, R., Friedman, J. H. (2001). *The elements of statistical learning : Data mining, inference, and prediction*. New York: Springer Verlag.

[7] Rodriguez, J.J. and Kuncheva, L.I. and Alonso, C.J. (2006), Rotation forest: A new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(10):1619-1630.

[8] Kass, G. V. (1980). "An exploratory technique for investigating large quantities of categorical data". *Applied Statistics* **29** (2): 119–127. doi:10.2307/2986296. JSTOR 2986296.

[9] Hothorn, T.; Hornik, K.; Zeileis, A. (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework". *Journal of Computational and Graphical Statistics* **15** (3): 651–674. doi:10.1198/106186006X133933. JSTOR 27594202.

[10] Strobl, C.; Malley, J.; Tutz, G. (2009). "An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classification and Regression Trees, Bagging and Random Forests". *Psychological Methods* **14** (4): 323–348. doi:10.1037/a0016973.

[11] Rokach, L.; Maimon, O. (2005). "Top-down induction of decision trees classifiers-a survey". *IEEE Transactions on Systems, Man, and Cybernetics, Part C* **35** (4): 476–487. doi:10.1109/TSMCC.2004.843247.

[12] Hyafil, Laurent; Rivest, RL (1976). "Constructing Optimal Binary Decision Trees is NP-complete". *Information Processing Letters* **5** (1): 15–17. doi:10.1016/0020-0190(76)90095-8.

[13] Murthy S. (1998). Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*

[14] Ben-Gal I. Dana A., Shkolnik N. and Singer (20). "Efficient Construction of Decision Trees by the Dual Information Distance Method" (PDF). Quality Technology & Quantitative Management (QTQM), 11(1), 133-147. Check date values in: |date= (help)

[15] "Principles of Data Mining". 2007. doi:10.1007/978-1-84628-766-4. ISBN 978-1-84628-765-7.

[16] Horváth, Tamás; Yamamoto, Akihiro, eds. (2003). "Inductive Logic Programming". Lecture Notes in Computer Science **2835**. doi:10.1007/b13700. ISBN 978-3-540-20144-1.

[17] Deng,H.; Runger, G.; Tuv, E. (2011). *Bias of importance measures for multi-valued attributes and solutions*. Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN). pp. 293–300.

[18] http://citeseer.ist.psu.edu/oliver93decision.html

[19] Tan & Dowe (2003)

[20] Papagelis A., Kalles D.(2001). Breeding Decision Trees Using Evolutionary Techniques, Proceedings of the Eighteenth International Conference on Machine Learning, p.393-400, June 28-July 01, 2001

[21] Barros, Rodrigo C., Basgalupp, M. P., Carvalho, A. C. P. L. F., Freitas, Alex A. (2011). A Survey of Evolutionary Algorithms for Decision-Tree Induction. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, vol. 42, n. 3, p. 291-312, May 2012.

[22] Chipman, Hugh A., Edward I. George, and Robert E. McCulloch. "Bayesian CART model search." Journal of the American Statistical Association 93.443 (1998): 935-948.

[23] Barros R. C., Cerri R., Jaskowiak P. A., Carvalho, A. C. P. L. F., A bottom-up oblique decision tree induction algorithm. Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011).

51.10 External links

- Building Decision Trees in Python From O'Reilly.
- An Addendum to "Building Decision Trees in Python" From O'Reilly.
- Decision Trees Tutorial using Microsoft Excel.
- Decision Trees page at aitopics.org, a page with commented links.
- Decision tree implementation in Ruby (AI4R)
- Evolutionary Learning of Decision Trees in C++
- Java implementation of Decision Trees based on Information Gain
- A very explicit explanation of information gain as splitting criterion

Chapter 52

Information gain in decision trees

For other uses, see Gain (disambiguation).

In information theory and machine learning, **information gain** is a synonym for *Kullback–Leibler divergence*. However, in the context of decision trees, the term is sometimes used synonymously with mutual information, which is the expectation value of the Kullback–Leibler divergence of a conditional probability distribution.

In particular, the information gain about a random variable *X* obtained from an observation that a random variable *A* takes the value *A*=*a* is the Kullback-Leibler divergence *D*KL(*p*(*x* | *a*) || *p*(*x* | I)) of the prior distribution *p*(*x* | I) for x from the posterior distribution *p*(*x* | *a*) for *x* given *a*.

The expected value of the information gain is the mutual information *I(X; A)* of *X* and *A* – i.e. the reduction in the entropy of *X* achieved by learning the state of the random variable *A*.

In machine learning, this concept can be used to define a preferred sequence of attributes to investigate to most rapidly narrow down the state of *X*. Such a sequence (which depends on the outcome of the investigation of previous attributes at each stage) is called a decision tree. Usually an attribute with high mutual information should be preferred to other attributes.

52.1 General definition

In general terms, the expected information gain is the change in information entropy H from a prior state to a state that takes some information as given:

$IG(T,a) = H(T) - H(T|a)$

52.2 Formal definition

Let T denote a set of training examples, each of the form $(\mathbf{x}, y) = (x_1, x_2, x_3, ..., x_k, y)$ where $x_a \in vals(a)$ is the value of the a th attribute of example **x** and y is the corresponding class label. The information gain for an attribute a is defined in terms of entropy $H()$ as follows:

$IG(T,a) = H(T) - \sum_{v \in vals(a)} \frac{|\{\mathbf{x} \in T | x_a = v\}|}{|T|} \cdot H(\{\mathbf{x} \in T | x_a = v\})$

The mutual information is equal to the total entropy for an attribute if for each of the attribute values a unique classification can be made for the result attribute. In this case, the relative entropies subtracted from the total entropy are 0.

52.3 Drawbacks

Although information gain is usually a good measure for deciding the relevance of an attribute, it is not perfect. A notable problem occurs when information gain is applied to attributes that can take on a large number of distinct values. For example, suppose that one is building a decision tree for some data describing the customers of a business. Information gain is often used to decide which of the attributes are the most relevant, so they can be tested near the root of the tree. One of the input attributes might be the customer's credit card number. This attribute has a high mutual information, because it uniquely identifies each customer, but we do *not* want to include it in the decision tree: deciding how to treat a customer based on their credit card number is unlikely to generalize to customers we haven't seen before (overfitting).

Information gain ratio is sometimes used instead. This biases the decision tree against considering attributes with a large number of distinct values. However, attributes with very low information values then appeared to receive an unfair advantage.

52.4 References

- Mitchell, Tom M. (1997). *Machine Learning*. The Mc-Graw-Hill Companies, Inc. ISBN 0070428077.

Chapter 53

Ensemble learning

For an alternative meaning, see variational Bayesian methods.

In statistics and machine learning, **ensemble methods** use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms.[1][2][3] Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble refers only to a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives.

53.1 Overview

Supervised learning algorithms are commonly described as performing the task of searching through a hypothesis space to find a suitable hypothesis that will make good predictions with a particular problem. Even if the hypothesis space contains hypotheses that are very well-suited for a particular problem, it may be very difficult to find a good one. Ensembles combine multiple hypotheses to form a (hopefully) better hypothesis. The term *ensemble* is usually reserved for methods that generate multiple hypotheses using the same base learner. The broader term of *multiple classifier systems* also covers hybridization of hypotheses that are not induced by the same base learner.

Evaluating the prediction of an ensemble typically requires more computation than evaluating the prediction of a single model, so ensembles may be thought of as a way to compensate for poor learning algorithms by performing a lot of extra computation. Fast algorithms such as decision trees are commonly used with ensembles (for example *Random Forest*), although slower algorithms can benefit from ensemble techniques as well.

53.2 Ensemble theory

An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. This flexibility can, in theory, enable them to over-fit the training data more than a single model would, but in practice, some ensemble techniques (especially bagging) tend to reduce problems related to over-fitting of the training data.

Empirically, ensembles tend to yield better results when there is a significant diversity among the models.[4][5] Many ensemble methods, therefore, seek to promote diversity among the models they combine.[6][7] Although perhaps non-intuitive, more random algorithms (like random decision trees) can be used to produce a stronger ensemble than very deliberate algorithms (like entropy-reducing decision trees).[8] Using a variety of strong learning algorithms, however, has been shown to be more effective than using techniques that attempt to *dumb-down* the models in order to promote diversity.[9]

53.3 Common types of ensembles

53.3.1 Bayes optimal classifier

The Bayes Optimal Classifier is a classification technique. It is an ensemble of all the hypotheses in the hypothesis space. On average, no other ensemble can outperform it.[10] Each hypothesis is given a vote proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis were true. To facilitate training data of finite size, the vote of each hypothesis is also multiplied by the prior probability of that hypothesis. The Bayes Optimal Classifier can be expressed with the following equation:

$$y = \mathrm{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(T|h_i)P(h_i)$$

where y is the predicted class, C is the set of all possible classes, H is the hypothesis space, P refers to a *probability*, and T is the training data. As an ensemble, the Bayes

Optimal Classifier represents a hypothesis that is not necessarily in H . The hypothesis represented by the Bayes Optimal Classifier, however, is the optimal hypothesis in *ensemble space* (the space of all possible ensembles consisting only of hypotheses in H).

Unfortunately, Bayes Optimal Classifier cannot be practically implemented for any but the most simple of problems. There are several reasons why the Bayes Optimal Classifier cannot be practically implemented:

1. Most interesting hypothesis spaces are too large to iterate over, as required by the argmax .
2. Many hypotheses yield only a predicted class, rather than a probability for each class as required by the term $P(c_j|h_i)$.
3. Computing an unbiased estimate of the probability of the training set given a hypothesis ($P(T|h_i)$) is non-trivial.
4. Estimating the prior probability for each hypothesis ($P(h_i)$) is rarely feasible.

53.3.2 Bootstrap aggregating (bagging)

Main article: Bootstrap aggregating

Bootstrap aggregating, often abbreviated as *bagging*, involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set. As an example, the random forest algorithm combines random decision trees with bagging to achieve very high classification accuracy.[11] An interesting application of bagging in unsupervised learning is provided here.[12][13]

53.3.3 Boosting

Main article: Boosting (meta-algorithm)

Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified. In some cases, boosting has been shown to yield better accuracy than bagging, but it also tends to be more likely to over-fit the training data. By far, the most common implementation of Boosting is Adaboost, although some newer algorithms are reported to achieve better results .

53.3.4 Bayesian model averaging

Bayesian model averaging (BMA) is an ensemble technique that seeks to approximate the Bayes Optimal Classifier by sampling hypotheses from the hypothesis space, and combining them using Bayes' law.[14] Unlike the Bayes optimal classifier, Bayesian model averaging can be practically implemented. Hypotheses are typically sampled using a Monte Carlo sampling technique such as MCMC. For example, Gibbs sampling may be used to draw hypotheses that are representative of the distribution $P(T|H)$. It has been shown that under certain circumstances, when hypotheses are drawn in this manner and averaged according to Bayes' law, this technique has an expected error that is bounded to be at most twice the expected error of the Bayes optimal classifier.[15] Despite the theoretical correctness of this technique, it has been found to promote over-fitting and to perform worse, empirically, compared to simpler ensemble techniques such as bagging;[16] however, these conclusions appear to be based on a misunderstanding of the purpose of Bayesian model averaging vs. model combination.[17]

53.3.5 Bayesian model combination

Bayesian model combination (BMC) is an algorithmic correction to BMA. Instead of sampling each model in the ensemble individually, it samples from the space of possible ensembles (with model weightings drawn randomly from a Dirichlet distribution having uniform parameters). This modification overcomes the tendency of BMA to converge toward giving all of the weight to a single model. Although BMC is somewhat more computationally expensive than BMA, it tends to yield dramatically better results. The results from BMC have been shown to be better on average (with statistical significance) than BMA, and bagging.[18]

The use of Bayes' law to compute model weights necessitates computing the probability of the data given each model. Typically, none of the models in the ensemble are exactly the distribution from which the training data were generated, so all of them correctly receive a value close to zero for this term. This would work well if the ensemble were big enough to sample the entire model-space, but such is rarely possible. Consequently, each pattern in the training data will cause the ensemble weight to shift toward the model in the ensemble that is closest to the distribution of the training data. It essentially reduces to an unnecessarily complex method for doing model selection.

The possible weightings for an ensemble can be visualized as lying on a simplex. At each vertex of the simplex, all of the weight is given to a single model in the ensemble. BMA converges toward the vertex that is closest to the distribution of the training data. By contrast, BMC converges toward the point where this distribution projects onto the simplex. In other words, instead of selecting the one model that is closest to the generating distribution, it seeks the combination of models that is closest to the generating distribution.

The results from BMA can often be approximated by using cross-validation to select the best model from a bucket of models. Likewise, the results from BMC may be approximated by using cross-validation to select the best ensemble combination from a random sampling of possible weightings.

53.3.6 Bucket of models

A "bucket of models" is an ensemble in which a model selection algorithm is used to choose the best model for each problem. When tested with only one problem, a bucket of models can produce no better results than the best model in the set, but when evaluated across many problems, it will typically produce much better results, on average, than any model in the set.

The most common approach used for model-selection is cross-validation selection (sometimes called a "bake-off contest"). It is described with the following pseudo-code:

For each model m in the bucket: Do c times: (where 'c' is some constant) Randomly divide the training dataset into two datasets: A, and B. Train m with A Test m with B Select the model that obtains the highest average score

Cross-Validation Selection can be summed up as: "try them all with the training set, and pick the one that works best".[19]

Gating is a generalization of Cross-Validation Selection. It involves training another learning model to decide which of the models in the bucket is best-suited to solve the problem. Often, a perceptron is used for the gating model. It can be used to pick the "best" model, or it can be used to give a linear weight to the predictions from each model in the bucket.

When a bucket of models is used with a large set of problems, it may be desirable to avoid training some of the models that take a long time to train. Landmark learning is a meta-learning approach that seeks to solve this problem. It involves training only the fast (but imprecise) algorithms in the bucket, and then using the performance of these algorithms to help determine which slow (but accurate) algorithm is most likely to do best.[20]

53.3.7 Stacking

Stacking (sometimes called *stacked generalization*) involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. If an arbitrary combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described in this article, although in practice, a single-layer logistic regression model is often used as the combiner.

Stacking typically yields performance better than any single one of the trained models.[21] It has been successfully used on both supervised learning tasks (regression,[22] classification and distance learning [23]) and unsupervised learning (density estimation).[24] It has also been used to estimate bagging's error rate.[3][25] It has been reported to out-perform Bayesian model-averaging.[26] The two top-performers in the Netflix competition utilized *blending*, which may be considered to be a form of stacking.[27]

53.4 References

[1] Opitz, D.; Maclin, R. (1999). "Popular ensemble methods: An empirical study". *Journal of Artificial Intelligence Research* **11**: 169–198. doi:10.1613/jair.614.

[2] Polikar, R. (2006). "Ensemble based systems in decision making". *IEEE Circuits and Systems Magazine* **6** (3): 21–45. doi:10.1109/MCAS.2006.1688199.

[3] Rokach, L. (2010). "Ensemble-based classifiers". *Artificial Intelligence Review* **33** (1-2): 1–39. doi:10.1007/s10462-009-9124-7.

[4] Kuncheva, L. and Whitaker, C., Measures of diversity in classifier ensembles, *Machine Learning*, 51, pp. 181-207, 2003

[5] Sollich, P. and Krogh, A., *Learning with ensembles: How overfitting can be useful*, Advances in Neural Information Processing Systems, volume 8, pp. 190-196, 1996.

[6] Brown, G. and Wyatt, J. and Harris, R. and Yao, X., Diversity creation methods: a survey and categorisation., *Information Fusion*, 6(1), pp.5-20, 2005.

[7] *Accuracy and Diversity in Ensembles of Text Categorisers*. J. J. García Adeva, Ulises Cerviño, and R. Calvo, CLEI Journal, Vol. 8, No. 2, pp. 1 - 12, December 2005.

[8] Ho, T., Random Decision Forests, *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 278-282, 1995.

[9] Gashler, M. and Giraud-Carrier, C. and Martinez, T., *Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous*, The Seventh International Conference on Machine Learning and Applications, 2008, pp. 900-905., DOI 10.1109/ICMLA.2008.154

[10] Tom M. Mitchell, *Machine Learning*, 1997, pp. 175

[11] Breiman, L., Bagging Predictors, *Machine Learning*, 24(2), pp.123-140, 1996.

[12] Sahu, A., Runger, G., Apley, D., Image denoising with a multi-phase kernel principal component approach and an ensemble version, IEEE Applied Imagery Pattern Recognition Workshop, pp.1-7, 2011.

[13] Shinde, Amit, Anshuman Sahu, Daniel Apley, and George Runger. "Preimages for Variation Patterns from Kernel PCA and Bagging." IIE Transactions, Vol. 46, Iss. 5, 2014.

[14] Hoeting, J. A.; Madigan, D.; Raftery, A. E.; Volinsky, C. T. (1999). “Bayesian Model Averaging: A Tutorial”. *Statistical Science* **14** (4): 382–401. doi:10.2307/2676803. JSTOR 2676803.

[15] David Haussler, Michael Kearns, and Robert E. Schapire. *Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension*. Machine Learning, 14:83–113, 1994

[16] Domingos, Pedro (2000). *Bayesian averaging of classifiers and the overfitting problem* (PDF). Proceedings of the 17th International Conference on Machine Learning (ICML). pp. 223—230.

[17] Minka, Thomas (2002), *Bayesian model averaging is not model combination* (PDF)

[18] Monteith, Kristine; Carroll, James; Seppi, Kevin; Martinez, Tony. (2011). *Turning Bayesian Model Averaging into Bayesian Model Combination* (PDF). Proceedings of the International Joint Conference on Neural Networks IJCNN'11. pp. 2657–2663.

[19] Bernard Zenko, *Is Combining Classifiers Better than Selecting the Best One*, Machine Learning, 2004, pp. 255-—273

[20] Bensusan, Hilan and Giraud-Carrier, Christophe G., Discovering Task Neighbourhoods Through Landmark Learning Performances, PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, Springer-Verlag, 2000, pages 325-—330

[21] Wolpert, D., *Stacked Generalization.*, Neural Networks, 5(2), pp. 241-259., 1992

[22] Breiman, L., *Stacked Regression*, Machine Learning, 24, 1996

[23] M. Ozay and F. T. Yarman Vural, *A New Fuzzy Stacked Generalization Technique and Analysis of its Performance*, 2012, arXiv:1204.0171

[24] Smyth, P. and Wolpert, D. H., *Linearly Combining Density Estimators via Stacking*, Machine Learning Journal, 36, 59-83, 1999

[25] Wolpert, D.H., and Macready, W.G., *An Efficient Method to Estimate Bagging's Generalization Error*, Machine Learning Journal, 35, 41-55, 1999

[26] Clarke, B., *Bayes model averaging and stacking when model approximation error cannot be ignored*, Journal of Machine Learning Research, pp 683-712, 2003

[27] Sill, J. and Takacs, G. and Mackey L. and Lin D., *Feature-Weighted Linear Stacking*, 2009, arXiv:0911.0460

53.5 Further reading

- Zhou Zhihua (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC. ISBN 978-1-439-83003-1.
- Robert Schapire; Yoav Freund (2012). *Boosting: Foundations and Algorithms*. MIT. ISBN 978-0-262-01718-3.

53.6 External links

- Ensemble learning at Scholarpedia, curated by Robi Polikar.
- The Waffles (machine learning) toolkit contains implementations of Bagging, Boosting, Bayesian Model Averaging, Bayesian Model Combination, Bucket-of-models, and other ensemble techniques

Chapter 54

Random forest

This article is about the machine learning technique. For other kinds of random tree, see Random tree (disambiguation).

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set.

The algorithm for inducing a random forest was developed by Leo Breiman[1] and Adele Cutler,[2] and "Random Forests" is their trademark. The method combines Breiman's "bagging" idea and the random selection of features, introduced independently by Ho[3][4] and Amit and Geman[5] in order to construct a collection of decision trees with controlled variance.

The selection of a random subset of features is an example of the random subspace method, which, in Ho's formulation, is a way to implement classification proposed by Eugene Kleinberg.[6]

54.1 History

The early development of random forests was influenced by the work of Amit and Geman[5] who introduced the idea of searching over a random subset of the available decisions when splitting a node, in the context of growing a single tree. The idea of random subspace selection from Ho[4] was also influential in the design of random forests. In this method a forest of trees is grown, and variation among the trees is introduced by projecting the training data into a randomly chosen subspace before fitting each tree. Finally, the idea of randomized node optimization, where the decision at each node is selected by a randomized procedure, rather than a deterministic optimization was first introduced by Dietterich.[7]

The introduction of random forests proper was first made in a paper by Leo Breiman.[1] This paper describes a method of building a forest of uncorrelated trees using a CART like procedure, combined with randomized node optimization and bagging. In addition, this paper combines several ingredients, some previously known and some novel, which form the basis of the modern practice of random forests, in particular:

1. Using out-of-bag error as an estimate of the generalization error.

2. Measuring variable importance through permutation.

The report also offers the first theoretical result for random forests in the form of a bound on the generalization error which depends on the strength of the trees in the forest and their correlation.

54.2 Algorithm

54.2.1 Preliminaries: decision tree learning

Main article: Decision tree learning

Decision trees are a popular method for various machine learning tasks. Tree learning "come[s] closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say Hastie *et al.*, because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspectable models. However, they are seldom accurate.[8]:352

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, because they have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance.[8]:587–588 This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.

54.2.2 Tree bagging

Main article: Bootstrap aggregating

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set X = x_1, …, x_n with responses Y = y_1, …, y_n, bagging repeatedly selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, …, B:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a decision or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x')$$

or by taking the majority vote in the case of decision trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the *out-of-bag error*: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.[9] The training and test error tend to level off after some number of trees have been fit.

54.2.3 From bagging to random forests

Main article: Random subspace method

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called “feature bagging”. The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.

Typically, for a dataset with p features, √p features are used in each split.

54.2.4 Extensions

Adding one further step of randomization yields *extremely randomized trees*, or ExtraTrees. These are trained using bagging and the random subspace method, like in an ordinary random forest, but additionally the top-down splitting in the tree learner is randomized. Instead of computing the locally *optimal* feature/split combination (based on, e.g., information gain or the Gini impurity), for each feature under consideration a random value is selected in the feature’s empirical range (in the tree’s training set, i.e., the bootstrap sample). The best of these is then chosen as the split.[10]

54.3 Properties

54.3.1 Variable importance

Random forests can be used to rank the importance of variables in a regression or classification problem in a natural way. The following technique was described in Breiman’s original paper[1] and is implemented in the R package randomForest.[2]

The first step in measuring the variable importance in a data set $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$ is to fit a random forest to the data. During the fitting process the out-of-bag error for each data point is recorded and averaged over the forest (errors on an independent test set can be substituted if bagging is not used during training).

To measure the importance of the j -th feature after training, the values of the j -th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set. The importance score for the j -th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences.

Features which produce large values for this score are ranked as more important than features which produce small values.

This method of determining variable importance has some drawbacks. For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Methods such as partial permutations[11][12] and growing unbiased trees[13] can be used to solve the problem. If the data contain groups of correlated features of similar relevance for the output, then smaller groups are favored over larger groups.[14]

54.3.2 Relationship to nearest neighbors

A relationship between random forests and the k-nearest neighbor algorithm (k-NN) was pointed out by Lin and Jeon in 2002.[15] It turns out that both can be viewed as so-called *weighted neighborhoods schemes*. These are models built from a training set $\{(x_i, y_i)\}_{i=1}^n$ that make predictions $\hat{y}$ for new points x' by looking at the “neighborhood” of the point, formalized by a weight function W:

$$\hat{y} = \sum_{i=1}^{n} W(x_i, x')\, y_i.$$

Here, $W(x_i, x')$ is the non-negative weight of the i'th training point relative to the new point x'. For any particular x', the weights must sum to one. Weight functions are given as follows:

- In k-NN, the weights are $W(x_i, x') = \frac{1}{k}$ if x_i is one of the k points closest to x', and zero otherwise.
- In a tree, $W(x_i, x')$ is the fraction of the training data that falls into the same leaf as x'.

Since a forest averages the predictions of a set of m trees with individual weight functions W_j , its predictions are

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right) y_i.$$

This shows that the whole forest is again a weighted neighborhood scheme, with weights that average those of the individual trees. The neighbors of x' in this interpretation are the points x_i which fall in the same leaf as x' in at least one tree of the forest. In this way, the neighborhood of x' depends in a complex way on the structure of the trees, and thus on the structure of the training set. Lin and Jeon show that the shape of the neighborhood used by a random forest adapts to the local importance of each feature.[15]

54.4 Unsupervised learning with random forests

As part of their construction, RF predictors naturally lead to a dissimilarity measure between the observations. One can also define an RF dissimilarity measure between unlabeled data: the idea is to construct an RF predictor that distinguishes the “observed” data from suitably generated synthetic data.[1][16] The observed data are the original unlabeled data and the synthetic data are drawn from a reference distribution. An RF dissimilarity can be attractive because it handles mixed variable types well, is invariant to monotonic transformations of the input variables, and is robust to outlying observations. The RF dissimilarity easily deals with a large number of semi-continuous variables due to its intrinsic variable selection; for example, the “Addcl 1” RF dissimilarity weighs the contribution of each variable according to how dependent it is on other variables. The RF dissimilarity has been used in a variety of applications, e.g. to find clusters of patients based on tissue marker data.[17]

54.5 Variants

Instead of decision trees, linear models have been proposed and evaluated as base estimators in random forests, in particular multinomial logistic regression and naive Bayes classifiers.[18][19]

54.6 See also

- Decision tree learning
- Gradient boosting
- Randomized algorithm
- Bootstrap aggregating (bagging)
- Ensemble learning
- Boosting
- Non-parametric statistics
- Kernel random forest

54.7 References

[1] Breiman, Leo (2001). “Random Forests”. *Machine Learning* **45** (1): 5–32. doi:10.1023/A:1010933404324.

[2] Liaw, Andy (16 October 2012). “Documentation for R package randomForest” (PDF). Retrieved 15 March 2013.

[3] Ho, Tin Kam (1995). *Random Decision Forest* (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[4] Ho, Tin Kam (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20** (8): 832–844. doi:10.1109/34.709601.

[5] Amit, Yali; Geman, Donald (1997). "Shape quantization and recognition with randomized trees" (PDF). *Neural Computation* **9** (7): 1545–1588. doi:10.1162/neco.1997.9.7.1545.

[6] Kleinberg, Eugene (1996). "An Overtraining-Resistant Stochastic Modeling Method for Pattern Recognition" (PDF). *Annals of Statistics* **24** (6): 2319–2349. doi:10.1214/aos/1032181157. MR 1425956.

[7] Dietterich, Thomas (2000). "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization". *Machine Learning*: 139–157.

[8] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning* (2nd ed.). Springer. ISBN 0-387-95284-5.

[9] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Springer. pp. 316–321.

[10] Geurts, P.; Ernst, D.; Wehenkel, L. (2006). "Extremely randomized trees" (PDF). *Machine Learning* **63**: 3. doi:10.1007/s10994-006-6226-1.

[11] Deng,H.; Runger, G.; Tuv, E. (2011). *Bias of importance measures for multi-valued attributes and solutions*. Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN). pp. 293–300.

[12] Altmann A, Tolosi L, Sander O, Lengauer T (2010). "Permutation importance:a corrected feature importance measure". *Bioinformatics*. doi:10.1093/bioinformatics/btq134.

[13] Strobl,C.; Boulesteix,A.; Augustin,T. (2007). "Unbiased split selection for classification trees based on the Gini index". *Computational Statistics & Data Analysis*: 483–501.

[14] Tolosi L, Lengauer T (2011). "Classification with correlated features: unreliability of feature ranking and solutions.". *Bioinformatics*. doi:10.1093/bioinformatics/btr300.

[15] Lin, Yi; Jeon, Yongho (2002). *Random forests and adaptive nearest neighbors* (Technical report). Technical Report No. 1055. University of Wisconsin.

[16] Shi, T., Horvath, S. (2006). "Unsupervised Learning with Random Forest Predictors". *Journal of Computational and Graphical Statistics* **15** (1): 118–138. doi:10.1198/106186006X94072.

[17] Shi, T., Seligson D., Belldegrun AS., Palotie A, Horvath, S. (2005). "Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma". *Modern Pathology* **18** (4): 547–557. doi:10.1038/modpathol.3800322. PMID 15529185.

[18] Prinzie, A., Van den Poel, D. (2008). "Random Forests for multiclass classification: Random MultiNomial Logit". *Expert Systems with Applications* **34** (3): 1721–1732. doi:10.1016/j.eswa.2007.01.029.

[19] Prinzie, A., Van den Poel, D. (2007). Random Multiclass Classification: Generalizing Random Forests to Random MNL and Random NB, Dexa 2007, Lecture Notes in Computer Science, 4653, 349–358.

54.8 External links

- Random Forests classifier description (Site of Leo Breiman)
- Liaw, Andy & Wiener, Matthew "Classification and Regression by randomForest" R News (2002) Vol. 2/3 p. 18 (Discussion of the use of the random forest package for R)
- Ho, Tin Kam (2002). "A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors". Pattern Analysis and Applications 5, p. 102-112 (Comparison of bagging and random subspace method)
- Prinzie, Anita; Poel, Dirk (2007). "Database and Expert Systems Applications". Lecture Notes in Computer Science **4653**. p. 349. doi:10.1007/978-3-540-74469-6_35. ISBN 978-3-540-74467-2. |chapter= ignored (help)
- C# implementation of random forest algorithm for categorization of text documents supporting reading of documents, making dictionaries, filtering stop words, stemming, counting words, making document-term matrix and its usage for building random forest and further categorization.
- A python implementation of the random forest algorithm working in regression, classification with multi-output support.

Chapter 55

Boosting (machine learning)

Boosting is a machine learning ensemble meta-algorithm for reducing bias primarily and also variance[1] in supervised learning, and a family of machine learning algorithms which convert weak learners to strong ones.[2] Boosting is based on the question posed by Kearns and Valiant (1988, 1989):[3][4] Can a set of **weak learners** create a single **strong learner**? A weak learner is defined to be a classifier which is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

Robert Schapire's affirmative answer in a 1990 paper[5] to the question of Kearns and Valiant has had significant ramifications in machine learning and statistics, most notably leading to the development of boosting.[6]

When first introduced, the *hypothesis boosting problem* simply referred to the process of turning a weak learner into a strong learner. "Informally, [the hypothesis boosting] problem asks whether an efficient learning algorithm […] that outputs a hypothesis whose performance is only slightly better than random guessing [i.e. a weak learner] implies the existence of an efficient algorithm that outputs a hypothesis of arbitrary accuracy [i.e. a strong learner]."[3] Algorithms that achieve hypothesis boosting quickly became simply known as "boosting". Freund and Schapire's arcing (Adapt[at]ive Resampling and Combining),[7] as a general technique, is more or less synonymous with boosting.[8]

55.1 Boosting algorithms

While boosting is not algorithmically constrained, most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. When they are added, they are typically weighted in some way that is usually related to the weak learners' accuracy. After a weak learner is added, the data is reweighted: examples that are misclassified gain weight and examples that are classified correctly lose weight (some boosting algorithms actually decrease the weight of repeatedly misclassified examples, e.g., boost by majority and BrownBoost). Thus, future weak learners focus more on the examples that previous weak learners misclassified.

There are many boosting algorithms. The original ones, proposed by Robert Schapire (a recursive majority gate formulation[5]) and Yoav Freund (boost by majority[9]), were not adaptive and could not take full advantage of the weak learners. However, Schapire and Freund then developed AdaBoost, an adaptive boosting algorithm that won the prestigious Gödel Prize. Only algorithms that are provable boosting algorithms in the probably approximately correct learning formulation are called boosting algorithms. Other algorithms that are similar in spirit to boosting algorithms are sometimes called "leveraging algorithms", although they are also sometimes incorrectly called boosting algorithms.[9]

55.2 Examples of boosting algorithms

The main variation between many boosting algorithms is their method of weighting training data points and hypotheses. AdaBoost is very popular and perhaps the most significant historically as it was the first algorithm that could adapt to the weak learners. However, there are many more recent algorithms such as LPBoost, TotalBoost, BrownBoost, MadaBoost, LogitBoost, and others. Many boosting algorithms fit into the AnyBoost framework,[9] which shows that boosting performs gradient descent in function space using a convex cost function.

Boosting algorithms are used in Computer Vision, where individual classifiers detecting contrast changes can be combined to identify Facial Features.[10]

55.3 Criticism

In 2008 Phillip Long (at Google) and Rocco A. Servedio (Columbia University) published a paper[11] at the 25th International Conference for Machine Learning suggesting that many of these algorithms are probably flawed.

They conclude that “convex potential boosters cannot withstand random classification noise,” thus making the applicability of such algorithms for real world, noisy data sets questionable. The paper shows that if any non-zero fraction of the training data is mis-labeled, the boosting algorithm tries extremely hard to correctly classify these training examples, and fails to produce a model with accuracy better than 1/2. This result does not apply to branching program based boosters but does apply to AdaBoost, LogitBoost, and others.[12][11]

55.4 See also

55.5 Implementations

- Orange, a free data mining software suite, module Orange.ensemble
- Weka is a machine learning set of tools that offers variate implementations of boosting algorithms like AdaBoost and LogitBoost
- R package GBM (Generalized Boosted Regression Models) implements extensions to Freund and Schapire’s AdaBoost algorithm and Friedman’s gradient boosting machine.
- jboost; AdaBoost, LogitBoost, RobustBoost, Boostexter and alternating decision trees

55.6 References

55.6.1 Footnotes

[1] Leo Breiman (1996). “BIAS, VARIANCE, AND ARCING CLASSIFIERS” (PDF). TECHNICAL REPORT. Retrieved 19 January 2015. Arcing [Boosting] is more successful than bagging in variance reduction

[2] Zhou Zhi-Hua (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC. p. 23. ISBN 978-1439830031. The term boosting refers to a family of algorithms that are able to convert weak learners to strong learners

[3] Michael Kearns(1988); *Thoughts on Hypothesis Boosting*, Unpublished manuscript (Machine Learning class project, December 1988)

[4] Michael Kearns; Leslie Valiant (1989). “Crytographic limitations on learning Boolean formulae and finite automata”. *Symposium on Theory of computing* (ACM) **21**: 433–444. doi:10.1145/73007.73049. Retrieved 18 January 2015.

[5] Schapire, Robert E. (1990). “The Strength of Weak Learnability” (PDF). *Machine Learning* (Boston, MA: Kluwer Academic Publishers) **5** (2): 197–227. doi:10.1007/bf00116037. CiteSeerX: 10.1.1.20.723.

[6] Leo Breiman (1998). “Arcing classifier (with discussion and a rejoinder by the author)". *Ann. Statist.* **26** (3): 801–849. Retrieved 18 January 2015. Schapire (1990) proved that boosting is possible. (Page 823)

[7] Yoav Freund and Robert E. Schapire (1997); *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, Journal of Computer and System Sciences, 55(1):119-139

[8] Leo Breiman (1998); *Arcing Classifier (with Discussion and a Rejoinder by the Author)*, Annals of Statistics, vol. 26, no. 3, pp. 801-849: “The concept of weak learning was introduced by Kearns and Valiant (1988, 1989), who left open the question of whether weak and strong learnability are equivalent. The question was termed the *boosting problem* since [a solution must] boost the low accuracy of a weak learner to the high accuracy of a strong learner. Schapire (1990) proved that boosting is possible. A *boosting algorithm* is a method that takes a weak learner and converts it into a strong learner. Freund and Schapire (1997) proved that an algorithm similar to arc-fs is boosting.

[9] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean (2000); *Boosting Algorithms as Gradient Descent*, in S. A. Solla, T. K. Leen, and K.-R. Muller, editors, *Advances in Neural Information Processing Systems* 12, pp. 512-518, MIT Press

[10] OpenCV CascadeClassifier http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html

[11] Random Classification Noise Defeats All Convex Potential Boosters

[12] Long version published as Phillip M. Long and Rocco A. Servedio (2010); *Random Classification Noise Defeats All Convex Potential Boosters*, Machine Learning 78(3), pp. 287-304

55.6.2 Notations

- Yoav Freund and Robert E. Schapire (1997); *A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting*, Journal of Computer and System Sciences, 55(1):119-139
- Robert E. Schapire and Yoram Singer (1999); *Improved Boosting Algorithms Using Confidence-Rated Predictors*, Machine Learning, 37(3):297-336

55.7 External links

- Robert E. Schapire (2003); *The Boosting Approach to Machine Learning: An Overview*, MSRI (Mathematical Sciences Research Institute) Workshop on Nonlinear Estimation and Classification
- Zhou Zhi-Hua (2014) *Boosting 25 years*, CCL 2014 Keynote.

Chapter 56

Bootstrap aggregating

Bootstrap aggregating, also called **bagging**, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

56.1 Description of the technique

Given a standard training set D of size n, bagging generates m new training sets D_i , each of size n', by sampling from D uniformly and with replacement. By sampling with replacement, some observations may be repeated in each D_i . If n'=n, then for large n the set D_i is expected to have the fraction (1 - 1/e) (≈63.2%) of the unique examples of D, the rest being duplicates.[1] This kind of sample is known as a bootstrap sample. The m models are fitted using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

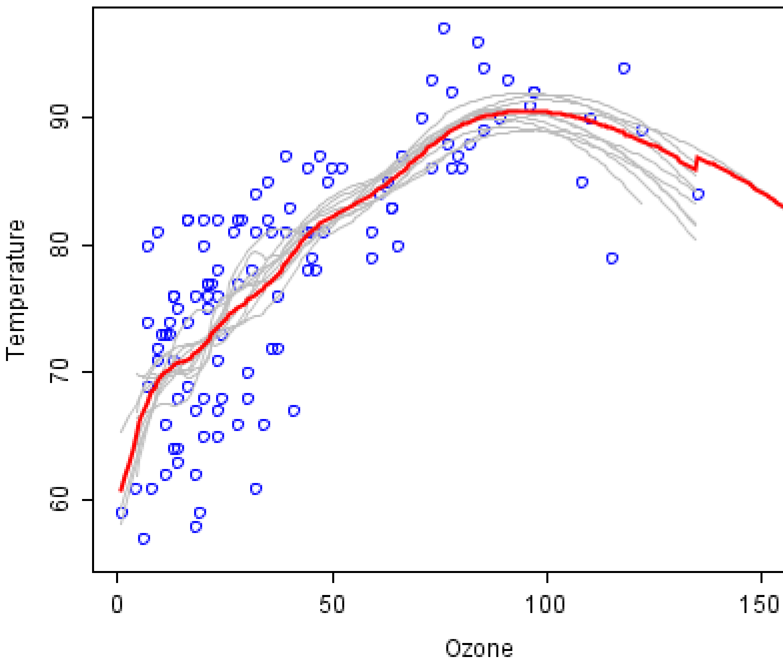
Bagging leads to "improvements for unstable procedures" (Breiman, 1996), which include, for example, artificial neural networks, classification and regression trees, and subset selection in linear regression (Breiman, 1994). An interesting application of bagging showing improvement in preimage learning is provided here.[2][3] On the other hand, it can mildly degrade the performance of stable methods such as K-nearest neighbors (Breiman, 1996).

56.2 Example: Ozone data

To illustrate the basic principles of bagging, below is an analysis on the relationship between ozone and temperature (data from Rousseeuw and Leroy (1986), available at classic data sets, analysis done in R).

The relationship between temperature and ozone in this data set is apparently non-linear, based on the scatter plot. To mathematically describe this relationship, LOESS smoothers (with span 0.5) are used. Instead of building a single smoother from the complete data set, 100 bootstrap samples of the data were drawn. Each sample is different from the original data set, yet resembles it in distribution and variability. For each bootstrap sample, a LOESS smoother was fit. Predictions from these 100 smoothers were then made across the range of the data. The first 10 predicted smooth fits appear as grey lines in the figure below. The lines are clearly very *wiggly* and they overfit the data - a result of the span being too low.

By taking the average of 100 smoothers, each fitted to a subset of the original data set, we arrive at one bagged predictor (red line). Clearly, the mean is more stable and there is less overfit.



56.3 Bagging for nearest neighbour classifiers

The risk of a 1 nearest neighbour (1NN) classifier is at most twice the risk of the Bayes classifier,[4] but there are no guarantees that this classifier will be consistent. By careful choice of the size of the resamples, bagging can lead to substantial improvements of the performance of the 1NN classifier. By taking a large number of resamples of the data of size n' , the bagged nearest neighbour classifier will be consistent provided $n' \to \infty$ diverges but $n'/n \to 0$ as the sample size $n \to \infty$.

Under infinite simulation, the bagged nearest neighbour classifier can be viewed as a weighted nearest neighbour classifier. Suppose that the feature space is d dimensional and denote by $C^{bnn}_{n,n'}$ the bagged nearest neighbour classifier based on a training set of size n , with resamples of size n' . In the infinite sampling case, under certain regularity conditions on the class distributions, the excess risk has the following asymptotic expansion[5]

$$\mathcal{R}_{\mathcal{R}}(C^{bnn}_{n,n'})-\mathcal{R}_{\mathcal{R}}(C^{Bayes}) = \left(B_1\frac{n'}{n} + B_2\frac{1}{(n')^{4/d}}\right)\{1+o(1)\},$$

for some constants B_1 and B_2 . The optimal choice of n' , that balances the two terms in the asymptotic expansion, is given by $n' = Bn^{d/(d+4)}$ for some constant B .

56.4 History

Bagging (**B**ootstrap **agg**regat**ing**) was proposed by Leo Breiman in 1994 to improve the classification by combining classifications of randomly generated training sets. See Breiman, 1994. Technical Report No. 421.

56.5 See also

- Boosting (meta-algorithm)
- Bootstrapping (statistics)
- Cross-validation (statistics)
- Random forest
- Random subspace method (attribute bagging)

56.6 References

[1] Aslam, Javed A.; Popa, Raluca A.; and Rivest, Ronald L. (2007); *On Estimating the Size and Confidence of a Statistical Audit*, Proceedings of the Electronic Voting Technology Workshop (EVT '07), Boston, MA, August 6, 2007. More generally, when drawing with replacement n' values out of a set of n (different and equally likely), the expected number of unique draws is $n(1 - e^{-n'/n})$.

[2] Sahu, A., Runger, G., Apley, D., Image denoising with a multi-phase kernel principal component approach and an ensemble version, IEEE Applied Imagery Pattern Recognition Workshop, pp.1-7, 2011.

[3] Shinde, Amit, Anshuman Sahu, Daniel Apley, and George Runger. "Preimages for Variation Patterns from Kernel PCA and Bagging." IIE Transactions, Vol.46, Iss.5, 2014

[4] Castelli, Vittorio. "Nearest Neighbor Classifiers, p.5" (PDF). *columbia.edu*. Columbia University. Retrieved 25 April 2015.

[5] Samworth R. J. (2012). "Optimal weighted nearest neighbour classifiers". *Annals of Statistics* **40** (5): 2733–2763. doi:10.1214/12-AOS1049.

- Breiman, Leo (1996). "Bagging predictors". *Machine Learning* **24** (2): 123–140. doi:10.1007/BF00058655. CiteSeerX: 10.1.1.121.7654.
- Alfaro, E., Gámez, M. and García, N. (2012). "adabag: An R package for classification with AdaBoost.M1, AdaBoost-SAMME and Bagging".

Chapter 57

Gradient boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The idea of gradient boosting originated in the observation by Leo Breiman [1] that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman[2][3] simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean .[4][5] The latter two papers introduced the abstract view of boosting algorithms as iterative *functional gradient descent* algorithms. That is, algorithms that optimize a cost *functional* over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

57.1 Informal introduction

(This section follows the exposition of gradient boosting by Li.[6])

Like other boosting methods, gradient boosting combines weak learners into a single strong learner, in an iterative fashion. It is easiest to explain in the least-squares regression setting, where the goal is to learn a model F that predicts values $\hat{y} = F(x)$, minimizing the mean squared error $(\hat{y} - y)^2$ to the true values y (averaged over some training set).

At each stage $1 \le m \le M$ of gradient boosting, it may be assumed that there is some imperfect model F_m (at the outset, a very weak model that just predicts the mean y in the training set could be used). The gradient boosting algorithm does not change F_m in any way; instead, it improves on it by constructing a new model that adds an estimator h to provide a better model $F_{m+1}(x) = F_m(x) + h(x)$. The question is now, how to find h ? The gradient boosting solution starts with the observation that a perfect h would imply

$$F_{m+1} = F_m(x) + h(x) = y$$

or, equivalently,

$$h(x) = y - F_m(x)$$

Therefore, gradient boosting will fit h to the *residual* $y - F_m(x)$. Like in other boosting variants, each F_{m+1} learns to correct its predecessor F_m . A generalization of this idea to other loss functions than squared error (and to classification and ranking problems) follows from the observation that residuals $y - F(x)$ are the negative gradients of the squared error loss function $\frac{1}{2}(y - F(x))^2$. So, gradient boosting is a gradient descent algorithm; and generalizing it entails "plugging in" a different loss and its gradient.

57.2 Algorithm

In many supervised learning problems one has an output variable y and a vector of input variables x connected together via a joint probability distribution $P(x, y)$. Using a training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$ of known values of x and corresponding values of y, the goal is to find an approximation $\hat{F}(x)$ to a function $F^*(x)$ that minimizes the expected value of some specified loss function $L(y, F(x))$:

$$F^* = \arg\min_F \mathbb{E}_{x,y}[L(y, F(x))]$$

Gradient boosting method assumes a real-valued y and seeks an approximation $\hat{F}(x)$ in the form of a weighted sum of functions $hi(x)$ from some class $\mathscr{H}$, called base (or weak) learners:

$$F(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + \text{const}$$

In accordance with the empirical risk minimization principle, the method tries to find an approximation $\hat{F}(x)$ that minimizes the average value of the loss function on the training set. It does so by starting with a model, consisting of a constant function $F_0(x)$, and incrementally expanding it in a greedy fashion:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

$$F_m(x) = F_{m-1}(x) + \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + f(x_i))$$

where f is restricted to be a function from the class $\mathscr{H}$ of base learner functions.

However, the problem of choosing at each step the best f for an arbitrary loss function L is a hard optimization problem in general, and so we'll "cheat" by solving a much easier problem instead.

The idea is to apply a steepest descent step to this minimization problem. If we only cared about predictions at the points of the training set, and f were unrestricted, we'd update the model per the following equation, where we view $L(y, f)$ not as a functional of f, but as a function of a vector of values $f(x_1), \ldots, f(x_n)$:

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_f L(y_i, F_{m-1}(x_i)),$$

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial f(x_i)}\right).$$

But as f must come from a restricted class of functions (that's what allows us to generalize), we'll just choose the one that most closely approximates the gradient of L. Having chosen f, the multiplier γ is then selected using line search just as shown in the second equation above.

In pseudocode, the generic gradient boosting method is:[2][7]

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:

   (a) Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   (b) Fit a base learner $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.

   (c) Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   (d) Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

57.3 Gradient tree boosting

Gradient boosting is typically used with decision trees (especially CART trees) of a fixed size as base learners. For this special case Friedman proposes a modification to gradient boosting method which improves the quality of fit of each base learner.

Generic gradient boosting at the *m*-th step would fit a decision tree $h_m(x)$ to pseudo-residuals. Let J be the number of its leaves. The tree partitions the input space into J disjoint regions $R_{1m}, \ldots, R_{Jm}$ and predicts a constant value in each region. Using the indicator notation, the output of $h_m(x)$ for input *x* can be written as the sum:

$$h_m(x) = \sum_{j=1}^{J} b_{jm} I(x \in R_{jm}),$$

where b_{jm} is the value predicted in the region R_{jm} .[8]

Then the coefficients b_{jm} are multiplied by some value γ_m , chosen using line search so as to minimize the loss function, and the model is updated as follows:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \quad \gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma$$

Friedman proposes to modify this algorithm so that it chooses a separate optimal value γ_{jm} for each of the tree's regions, instead of a single γ_m for the whole tree. He calls the modified algorithm "TreeBoost". The coefficients b_{jm} from the tree-fitting procedure can be then simply discarded and the model update rule becomes:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm}), \quad \gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i$$

57.3.1 Size of trees

J, the number of terminal nodes in trees, is the method's parameter which can be adjusted for a data set at hand. It controls the maximum allowed level of interaction between variables in the model. With $J = 2$ (decision stumps), no interaction between variables is allowed. With $J = 3$ the model may include effects of the interaction between up to two variables, and so on.

Hastie et al.[7] comment that typically $4 \leq J \leq 8$ work well for boosting and results are fairly insensitive to the choice of J in this range, $J = 2$ is insufficient for many applications, and $J > 10$ is unlikely to be required.

57.4 Regularization

Fitting the training set too closely can lead to degradation of the model's generalization ability. Several so-called regularization techniques reduce this overfitting effect by constraining the fitting procedure.

One natural regularization parameter is the number of gradient boosting iterations *M* (i.e. the number of trees in the model when the base learner is a decision tree). Increasing *M* reduces the error on training set, but setting it too high may lead to overfitting. An optimal value of *M* is often selected by monitoring prediction error on a separate validation data set. Besides controlling *M*, several other regularization techniques are used.

57.4.1 Shrinkage

An important part of gradient boosting method is regularization by shrinkage which consists in modifying the update rule as follows:

$$F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x), \quad 0 < \nu \leq 1,$$

where parameter ν is called the "learning rate".

Empirically it has been found that using small learning rates (such as $\nu < 0.1$) yields dramatic improvements in model's generalization ability over gradient boosting without shrinking ($\nu = 1$).[7] However, it comes at the price of increasing computational time both during training and querying: lower learning rate requires more iterations.

57.4.2 Stochastic gradient boosting

Soon after the introduction of gradient boosting Friedman proposed a minor modification to the algorithm, motivated by Breiman's bagging method.[3] Specifically, he proposed that at each iteration of the algorithm, a base learner should be fit on a subsample of the training set drawn at random without replacement.[9] Friedman observed a substantial improvement in gradient boosting's accuracy with this modification.

Subsample size is some constant fraction *f* of the size of the training set. When *f* = 1, the algorithm is deterministic and identical to the one described above. Smaller values of *f* introduce randomness into the algorithm and help prevent overfitting, acting as a kind of regularization. The algorithm also becomes faster, because regression trees have to be fit to smaller datasets at each iteration. Friedman[3] obtained that $0.5 \leq f \leq 0.8$ leads to good results for small and moderate sized training sets. Therefore, *f* is typically set to 0.5, meaning that one half of the training set is used to build each base learner.

Also, like in bagging, subsampling allows one to define an out-of-bag estimate of the prediction performance improvement by evaluating predictions on those observations which were not used in the building of the next base learner. Out-of-bag estimates help avoid the need for an independent validation dataset, but often underestimate actual performance improvement and the optimal number of iterations.[10]

57.4.3 Number of observations in leaves

Gradient tree boosting implementations often also use regularization by limiting the minimum number of observations in trees' terminal nodes (this parameter is called n.minobsinnode in the R gbm package[10]). It is used in the tree building process by ignoring any splits that lead to nodes containing fewer than this number of training set instances.

Imposing this limit helps to reduce variance in predictions at leaves.

57.4.4 Penalize Complexity of Tree

Another useful regularization techniques for gradient boosted trees is to penalize model complexity of the learned model. [11] The model complexity can be defined proportional number of leaves in the learned trees. The jointly optimization of loss and model complexity corresponds to a post-pruning algorithm to remove branches that fail to reduce the loss by a threshold. Other kinds of regularization such as l2 penalty on the leave values can also be added to avoid overfitting.

57.5 Usage

Recently, gradient boosting has gained some popularity in the field of learning to rank. The commercial web search engines Yahoo[12] and Yandex[13] use variants of gradient boosting in their machine-learned ranking engines.

57.6 Names

The method goes by a variety of names. Friedman introduced his regression technique as a “Gradient Boosting Machine” (GBM).[2] Mason, Baxter et. el. described the generalized abstract class of algorithms as “functional gradient boosting”.[4][5]

A popular open-source implementation[10] for R calls it “Generalized Boosting Model”. Commercial implementations from Salford Systems use the names “Multiple Additive Regression Trees” (MART) and TreeNet, both trademarked.

57.7 See also

- AdaBoost
- Random forest

57.8 References

[1] Brieman, L. "Arcing The Edge" (June 1997)

[2] Friedman, J. H. "Greedy Function Approximation: A Gradient Boosting Machine." (February 1999)

[3] Friedman, J. H. "Stochastic Gradient Boosting." (March 1999)

[4] Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (1999). “Boosting Algorithms as Gradient Descent” (PDF). In S.A. Solla and T.K. Leen and K. Müller. *Advances in Neural Information Processing Systems 12*. MIT Press. pp. 512–518.

[5] Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (May 1999). *Boosting Algorithms as Gradient Descent in Function Space* (PDF).

[6] Cheng Li. “A Gentle Introduction to Gradient Boosting” (PDF). Northeastern University. Retrieved 19 August 2014.

[7] Hastie, T.; Tibshirani, R.; Friedman, J. H. (2009). “10. Boosting and Additive Trees”. *The Elements of Statistical Learning* (2nd ed.). New York: Springer. pp. 337–384. ISBN 0-387-84857-6.

[8] Note: in case of usual CART trees, the trees are fitted using least-squares loss, and so the coefficient b_{jm} for the region R_{jm} is equal to just the value of output variable, averaged over all training instances in R_{jm} .

[9] Note that this is different from bagging, which samples with replacement because it uses samples of the same size as the training set.

[10] Ridgeway, Greg (2007). Generalized Boosted Models: A guide to the gbm package.

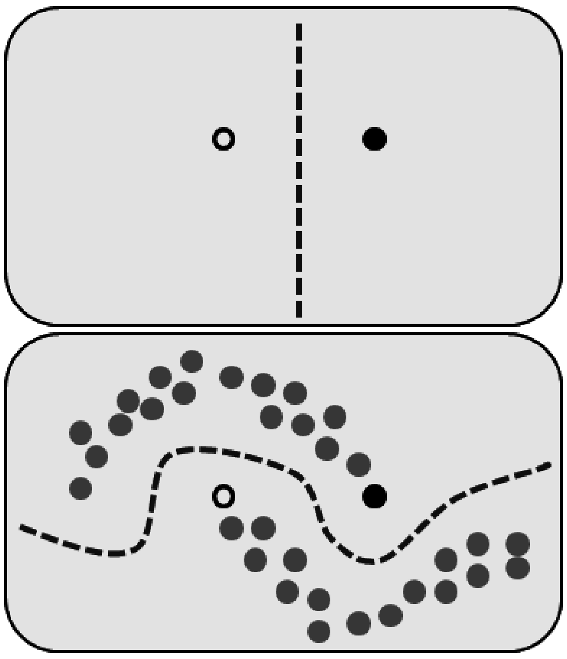
[11] Tianqi Chen. Introduction to Boosted Trees

[12] Cossock, David and Zhang, Tong (2008). Statistical Analysis of Bayes Optimal Subset Ranking, page 14.

[13] Yandex corporate blog entry about new ranking model “Snezhinsk” (in Russian)

Chapter 58

Semi-supervised learning



An example of the influence of unlabeled data in semi-supervised learning. The top panel shows a decision boundary we might adopt after seeing only one positive (white circle) and one negative (black circle) example. The bottom panel shows a decision boundary we might adopt if, in addition to the two labeled examples, we were given a collection of unlabeled data (gray circles). This could be viewed as performing clustering and then labeling the clusters with the labeled data, pushing the decision boundary away from high-density regions, or learning an underlying one-dimensional manifold where the data reside.

Semi-supervised learning is a class of supervised learning tasks and techniques that also make use of unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent (e.g. to transcribe an audio segment) or a physical experiment (e.g. determining the 3D structure of a protein or determining whether there is oil at a particular location). The cost associated with the labeling process thus may render a fully labeled training set infeasible, whereas acquisition of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value. Semi-supervised learning is also of theoretical interest in machine learning and as a model for human learning.

As in the supervised learning framework, we are given a set of l independently identically distributed examples $x_1, \ldots, x_l \in X$ with corresponding labels $y_1, \ldots, y_l \in Y$. Additionally, we are given u unlabeled examples $x_{l+1}, \ldots, x_{l+u} \in X$. Semi-supervised learning attempts to make use of this combined information to surpass the classification performance that could be obtained either by discarding the unlabeled data and doing supervised learning or by discarding the labels and doing unsupervised learning.

Semi-supervised learning may refer to either transductive learning or inductive learning. The goal of transductive learning is to infer the correct labels for the given unlabeled data $x_{l+1}, \ldots, x_{l+u}$ only. The goal of inductive learning is to infer the correct mapping from X to Y .

Intuitively, we can think of the learning problem as an exam and labeled data as the few example problems that the teacher solved in class. The teacher also provides a set of unsolved problems. In the transductive setting, these unsolved problems are a take-home exam and you want to do well on them in particular. In the inductive setting, these are practice problems of the sort you will encounter on the in-class exam.

It is unnecessary (and, according to Vapnik's principle, imprudent) to perform transductive learning by way of inferring a classification rule over the entire input space; however, in practice, algorithms formally designed for transduction or induction are often used interchangeably.

58.1 Assumptions used in semi-supervised learning

In order to make any use of unlabeled data, we must assume some structure to the underlying distribution of data. Semi-supervised learning algorithms make use of at least one of the following assumptions. [1]

58.1.1 Smoothness assumption

Points which are close to each other are more likely to share a label. This is also generally assumed in supervised learning and yields a preference for geometrically simple decision boundaries. In the case of semi-supervised learning, the smoothness assumption additionally yields a preference for decision boundaries in low-density regions, so that there are fewer points close to each other but in different classes.

58.1.2 Cluster assumption

The data tend to form discrete clusters, and points in the same cluster are more likely to share a label (although data sharing a label may be spread across multiple clusters). This is a special case of the smoothness assumption and gives rise to feature learning with clustering algorithms.

58.1.3 Manifold assumption

The data lie approximately on a manifold of much lower dimension than the input space. In this case we can attempt to learn the manifold using both the labeled and unlabeled data to avoid the curse of dimensionality. Then learning can proceed using distances and densities defined on the manifold.

The manifold assumption is practical when high-dimensional data are being generated by some process that may be hard to model directly, but which only has a few degrees of freedom. For instance, human voice is controlled by a few vocal folds,[2] and images of various facial expressions are controlled by a few muscles. We would like in these cases to use distances and smoothness in the natural space of the generating problem, rather than in the space of all possible acoustic waves or images respectively.

58.2 History

The heuristic approach of *self-training* (also known as *self-learning* or *self-labeling*) is historically the oldest approach to semi-supervised learning,[1] with examples of applications starting in the 1960s (see for instance Scudder (1965)[3]).

The transductive learning framework was formally introduced by Vladimir Vapnik in the 1970s.[4] Interest in inductive learning using generative models also began in the 1970s. A *probably approximately correct* learning bound for semi-supervised learning of a Gaussian mixture was demonstrated by Ratsaby and Venkatesh in 1995 [5]

Semi-supervised learning has recently become more popular and practically relevant due to the variety of problems for which vast quantities of unlabeled data are available—e.g. text on websites, protein sequences, or images. For a review of recent work see a survey article by Zhu (2008).[6]

58.3 Methods for semi-supervised learning

58.3.1 Generative models

Generative approaches to statistical learning first seek to estimate $p(x|y)$, the distribution of data points belonging to each class. The probability $p(y|x)$ that a given point x has label y is then proportional to $p(x|y)p(y)$ by Bayes' rule. Semi-supervised learning with generative models can be viewed either as an extension of supervised learning (classification plus information about $p(x)$) or as an extension of unsupervised learning (clustering plus some labels).

Generative models assume that the distributions take some particular form $p(x|y,\theta)$ parameterized by the vector θ . If these assumptions are incorrect, the unlabeled data may actually decrease the accuracy of the solution relative to what would have been obtained from labeled data alone. [7] However, if the assumptions are correct, then the unlabeled data necessarily improves performance.[5]

The unlabeled data are distributed according to a mixture of individual-class distributions. In order to learn the mixture distribution from the unlabeled data, it must be identifiable, that is, different parameters must yield different summed distributions. Gaussian mixture distributions are identifiable and commonly used for generative models.

The parameterized joint distribution can be written as $p(x,y|\theta) = p(y|\theta)p(x|y,\theta)$ by using the Chain rule. Each parameter vector θ is associated with a decision function $f_\theta(x) = \underset{y}{\operatorname{argmax}}\ p(y|x,\theta)$. The parameter is then chosen based on fit to both the labeled and unlabeled data, weighted by λ :

$$\underset{\Theta}{\operatorname{argmax}} \left(\log p(\{x_i, y_i\}_{i=1}^{l}|\theta) + \lambda \log p(\{x_i\}_{i=l+1}^{l+u}|\theta) \right)$$

[8]

58.3.2 Low-density separation

Another major class of methods attempts to place boundaries in regions where there are few data points (labeled or unlabeled). One of the most commonly used algorithms is the transductive support vector machine, or TSVM (which, despite its name, may be used for inductive learning as well). Whereas support vector machines for supervised learning seek a decision boundary with maximal margin over the labeled data, the goal of TSVM is a labeling of the unlabeled data such that the decision boundary has maximal margin over all of the data. In addition to the standard hinge loss $(1 - yf(x))_+$ for labeled data, a loss function $(1 - |f(x)|)_+$ is introduced over the unlabeled data by letting $y = \operatorname{sign} f(x)$. TSVM then selects $f^*(x) = h^*(x) + b$ from a reproducing kernel Hilbert space $\mathcal{H}$ by minimizing the regularized empirical risk:

$$f^* = \underset{f}{\operatorname{argmin}} \left(\sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 ||h||^2_{\mathcal{H}} + \lambda_2 \sum_{i=l+1}^{l+u} (1 - |f(x_i)|)_+ \right)$$

An exact solution is intractable due to the non-convex term $(1 - |f(x)|)_+$, so research has focused on finding useful approximations.[8]

Other approaches that implement low-density separation include Gaussian process models, information regularization, and entropy minimization (of which TSVM is a special case).

58.3.3 Graph-based methods

Graph-based methods for semi-supervised learning use a graph representation of the data, with a node for each labeled and unlabeled example. The graph may be constructed using domain knowledge or similarity of examples; two common methods are to connect each data point to its k nearest neighbors or to examples within some distance ϵ . The weight W_{ij} of an edge between x_i and x_j is then set to $e^{\frac{-||x_i - x_j||^2}{\epsilon}}$.

Within the framework of *manifold regularization*, [9] [10] the graph serves as a proxy for the manifold. A term is added to the standard Tikhonov regularization problem to enforce smoothness of the solution relative to the manifold (in the intrinsic space of the problem) as well as relative to the ambient input space. The minimization problem becomes

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \left(\frac{1}{l} \sum_{i=1}^{l} V(f(x_i), y_i) + \lambda_A ||f||^2_{\mathcal{H}} + \lambda_I \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} f(x)||^2 dp(x) \right)$$

[8]

where $\mathcal{H}$ is a reproducing kernel Hilbert space and $\mathcal{M}$ is the manifold on which the data lie. The regularization parameters λ_A and λ_I control smoothness in the ambient and intrinsic spaces respectively. The graph is used to approximate the intrinsic regularization term. Defining the graph Laplacian $L = D - W$ where $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$ and $\mathbf{f}$ the vector $[f(x_1) \dots f(x_{l+u})]$, we have

$$\mathbf{f}^T L \mathbf{f} = \sum_{i,j=1}^{l+u} W_{ij}(f_i - f_j)^2 \approx \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} f(x)||^2 dp(x)$$

The Laplacian can also be used to extend the supervised learning algorithms: regularized least squares and support vector machines (SVM) to semi-supervised versions Laplacian regularized least squares and Laplacian SVM.

58.3.4 Heuristic approaches

Some methods for semi-supervised learning are not intrinsically geared to learning from both unlabeled and labeled data, but instead make use of unlabeled data within a supervised learning framework. For instance, the labeled and unlabeled examples $x_1, \dots, x_{l+u}$ may inform a choice of representation, distance metric, or kernel for the data in an unsupervised first step. Then supervised learning proceeds from only the labeled examples.

Self-training is a wrapper method for semi-supervised learning. First a supervised learning algorithm is used to select a classifier based on the labeled data only. This classifier is then applied to the unlabeled data to generate more labeled examples as input for another supervised learning problem. Generally only the labels the classifier is most confident of are added at each step.

Co-training is an extension of self-training in which multiple classifiers are trained on different (ideally disjoint) sets of features and generate labeled examples for one another.

58.4 Semi-supervised learning in human cognition

Human responses to formal semi-supervised learning problems have yielded varying conclusions about the degree of influence of the unlabeled data (for a summary see [11]). More natural learning problems may also be viewed as instances of semi-supervised learning. Much of human concept learning involves a small amount of direct instruction (e.g. parental labeling of objects during childhood) combined with large amounts of unlabeled experience (e.g. observation of objects without naming or counting them, or at least without feedback).

Human infants are sensitive to the structure of unlabeled natural categories such as images of dogs and cats or male and female faces.[12] More recent work has shown that infants and children take into account not only the unlabeled examples available, but the sampling process from which labeled examples arise.[13][14]

58.5 See also

- PU learning

58.6 References

[1] Chapelle, Olivier; Schölkopf, Bernhard; Zien, Alexander (2006). *Semi-supervised learning*. Cambridge, Mass.: MIT Press. ISBN 978-0-262-03358-9.

[2] Stevens, K.N.(2000), Acoustic Phonetics, MIT Press, ISBN 0-262-69250-3, 978-0-262-69250-2

[3] Scudder, H.J. Probability of Error of Some Adaptive Pattern-Recognition Machines. IEEE Transaction on Information Theory, 11:363–371 (1965). Cited in Chapelle et al. 2006, page 3.

[4] Vapnik, V. and Chervonenkis, A. Theory of Pattern Recognition [in Russian]. Nauka, Moscow (1974). Cited in Chapelle et al. 2006, page 3.

[5] Ratsaby, J. and Venkatesh, S. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 412-417 (1995). Cited in Chapelle et al. 2006, page 4.

[6] Zhu, Xiaojin. Semi-supervised learning literature survey. Computer Sciences, University of Wisconsin-Madison (2008).

[7] Cozman, F. and Cohen, I. Risks of semi-supervised learning: how unlabeled data can degrade performance of generative classifiers. In: Chapelle et al. (2006).

[8] Zhu, Xiaojin. Semi-Supervised Learning University of Wisconsin-Madison.

[9] M. Belkin, P. Niyogi (2004). "Semi-supervised Learning on Riemannian Manifolds". *Machine Learning* **56** (Special Issue on Clustering): 209–239.

[10] M. Belkin, P. Niyogi, V. Sindhwani. On Manifold Regularization. AISTATS 2005.

[11] Zhu, Xiaojin; Goldberg, Andrew B. (2009). *Introduction to semi-supervised learning*. Morgan & Claypool. ISBN 9781598295481.

[12] Younger, B. A. and Fearing, D. D. (1999), Parsing Items into Separate Categories: Developmental Change in Infant Categorization. Child Development, 70: 291–303.

[13] Xu, F. and Tenenbaum, J. B. (2007). "Sensitivity to sampling in Bayesian word learning. Developmental Science" **10**. pp. 288–297. doi:10.1111/j.1467-7687.2007.00590.x.

[14] Gweon, H., Tenenbaum J.B., and Schulz L.E (2010). "Infants consider both the sample and the sampling process in inductive generalization". *Proc Natl Acad Sci U S A*. **107** (20): 9066–71.

58.7 External links

- A freely available MATLAB implementation of the graph-based semi-supervised algorithms Laplacian support vector machines and Laplacian regularized least squares.

Chapter 59

Perceptron

"Perceptrons" redirects here. For the book of that title, see Perceptrons (book).

In machine learning, the **perceptron** is an algorithm for supervised learning of binary classifiers: functions that can decide whether an input (represented by a vector of numbers) belong to one class or another.[1] It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time.

The perceptron algorithm dates back to the late 1950s; its first implementation, in custom hardware, was one of the first artificial neural networks to be produced.

59.1 History

See also: History of artificial intelligence, AI winter

The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt,[2] funded by the United States Office of Naval Research.[3] The perceptron was intended to be a machine, rather than a program, and while its first implementation was in software for the IBM 704, it was subsequently implemented in custom-built hardware as the "Mark 1 perceptron". This machine was designed for image recognition: it had an array of 400 photocells, randomly connected to the "neurons". Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors.[4]:193

In a 1958 press conference organized by the US Navy, Rosenblatt made statements about the perceptron that caused a heated controversy among the fledgling AI community; based on Rosenblatt's statements, *The New York Times* reported the perceptron to be "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."[3]

Although the perceptron initially seemed promising, it was quickly proved that perceptrons could not be trained to recognise many classes of patterns. This led to the field of neural network research stagnating for many years, before it was recognised that a feedforward neural network with two or more layers (also called a multilayer perceptron) had far greater processing power than perceptrons with one layer (also called a single layer perceptron). Single layer perceptrons are only capable of learning linearly separable patterns; in 1969 a famous book entitled *Perceptrons* by Marvin Minsky and Seymour Papert showed that it was impossible for these classes of network to learn an XOR function. It is often believed that they also conjectured (incorrectly) that a similar result would hold for a multi-layer perceptron network. However, this is not true, as both Minsky and Papert already knew that multi-layer perceptrons were capable of producing an XOR function. (See the page on *Perceptrons (book)* for more information.) Three years later Stephen Grossberg published a series of papers introducing networks capable of modelling differential, contrast-enhancing and XOR functions. (The papers were published in 1972 and 1973, see e.g.:Grossberg (1973). "Contour enhancement, short-term memory, and constancies in reverberating neural networks" (PDF). *Studies in Applied Mathematics* **52**: 213–257.). Nevertheless the often-miscited Minsky/Papert text caused a significant decline in interest and funding of neural network research. It took ten more years until neural network research experienced a resurgence in the 1980s. This text was reprinted in 1987 as "Perceptrons - Expanded Edition" where some errors in the original text are shown and corrected.

The kernel perceptron algorithm was already introduced in 1964 by Aizerman et al.[5] Margin bounds guarantees were given for the Perceptron algorithm in the general non-separable case first by Freund and Schapire (1998),[1] and more recently by Mohri and Rostamizadeh (2013) who extend previous results and give new L1 bounds.[6]

59.2 Definition

In the modern sense, the perceptron is an algorithm for learning a binary classifier: a function that maps its input

x (a real-valued vector) to an output value $f(x)$ (a single binary value):

$$f(x) = \begin{cases} 1 & \text{if} w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

where w is a vector of real-valued weights, $w \cdot x$ is the dot product $\sum_i w_i x_i$, and b is the *bias*, a term that shifts the decision boundary away from the origin and does not depend on any input value.

The value of $f(x)$ (0 or 1) is used to classify x as either a positive or a negative instance, in the case of a binary classification problem. If b is negative, then the weighted combination of inputs must produce a positive value greater than $|b|$ in order to push the classifier neuron over the 0 threshold. Spatially, the bias alters the position (though not the orientation) of the decision boundary. The perceptron learning algorithm does not terminate if the learning set is not linearly separable. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly. The most famous example of the perceptron's inability to solve problems with linearly nonseparable vectors is the Boolean exclusive-or problem. The solution spaces of decision boundaries for all binary functions and learning behaviors are studied in the reference.[7]

In the context of neural networks, a perceptron is an artificial neuron using the Heaviside step function as the activation function. The perceptron algorithm is also termed the **single-layer perceptron**, to distinguish it from a multilayer perceptron, which is a misnomer for a more complicated neural network. As a linear classifier, the single-layer perceptron is the simplest feedforward neural network.

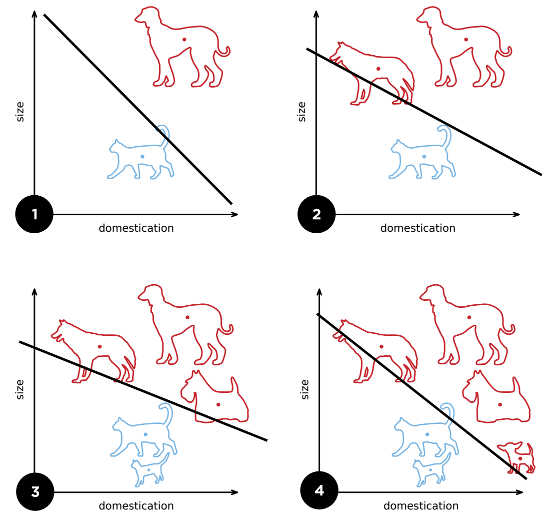
59.3 Learning algorithm

Below is an example of a learning algorithm for a (single-layer) perceptron. For multilayer perceptrons, where a hidden layer exists, more sophisticated algorithms such as backpropagation must be used. Alternatively, methods such as the delta rule can be used if the function is non-linear and differentiable, although the one below will work as well.

When multiple perceptrons are combined in an artificial neural network, each output neuron operates independently of all the others; thus, learning each output can be considered in isolation.

59.3.1 Definitions

We first define some variables:



A diagram showing a perceptron updating its linear boundary as more training examples are added.

- $y = f(\mathbf{z})$ denotes the *output* from the perceptron for an input vector $\mathbf{z}$.
- b is the *bias* term, which in the example below we take to be 0.
- $D = \{(\mathbf{x}_1, d_1), \dots, (\mathbf{x}_s, d_s)\}$ is the *training set* of s samples, where:
    - $\mathbf{x}_j$ is the n -dimensional input vector.
    - d_j is the desired output value of the perceptron for that input.

We show the values of the features as follows:

- $x_{j,i}$ is the value of the i th feature of the j th training *input vector*.
- $x_{j,0} = 1$.

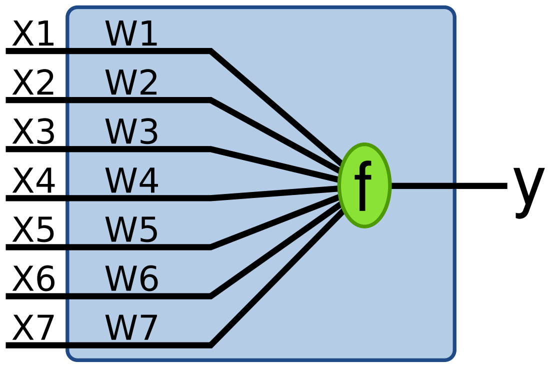
To represent the weights:

- w_i is the i th value in the *weight vector*, to be multiplied by the value of the i th input feature.
- Because $x_{j,0} = 1$, the w_0 is effectively a learned bias that we use instead of the bias constant b .

To show the time-dependence of $\mathbf{w}$, we use:

- $w_i(t)$ is the weight i at time t .
- α is the *learning rate*, where $0 < \alpha \leq 1$.

Too high a learning rate makes the perceptron periodically oscillate around the solution unless additional steps are taken.



The appropriate weights are applied to the inputs, and the resulting weighted sum passed to a function that produces the output y.

59.3.2 Steps

1. Initialize the weights and the threshold. Weights may be initialized to 0 or to a small random value. In the example below, we use 0.

2. For each example j in our training set D, perform the following steps over the input $\mathbf{x}_j$ and desired output d_j:

2a. Calculate the actual output:

$y_j(t) = f[\mathbf{w}(t)\cdot\mathbf{x}_j] = f[w_0(t)+w_1(t)x_{j,1}+w_2(t)x_{j,2}+\cdots$

2b. Update the weights:

$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i}$, for all feature $0 \leq i \leq n$.

3. For offline learning, the step 2 may be repeated until the iteration error $\frac{1}{s}\sum_{j=1}^{s}|d_j - y_j(t)|$ is less than a user-specified error threshold γ, or a predetermined number of iterations have been completed.

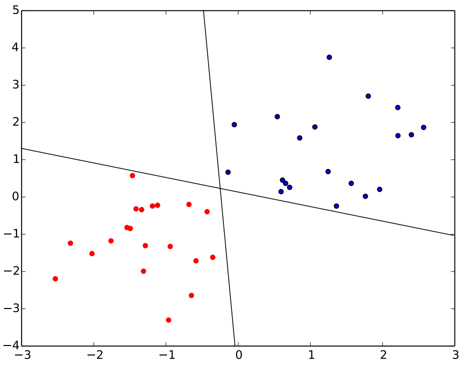
The algorithm updates the weights after steps 2a and 2b. These weights are immediately applied to a pair in the training set, and subsequently updated, rather than waiting until all pairs in the training set have undergone these steps.

59.3.3 Convergence

The perceptron is a linear classifier, therefore it will never get to the state with all the input vectors classified correctly if the training set D is not linearly separable, i.e. if the positive examples can not be separated from the negative examples by a hyperplane. In this case, no "approximate" solution will be gradually approached under the standard learning algorithm, but instead learning will fail completely. Hence, if linear separability of the training set is not known a priori, one of the training variants below should be used.

But if the training set *is* linearly separable, then the perceptron is guaranteed to converge, and there is an upper bound on the number of times the perceptron will adjust its weights during the training.

Suppose that the input vectors from the two classes can be separated by a hyperplane with a margin γ, i.e. there exists a weight vector $\mathbf{w}$, $||\mathbf{w}|| = 1$, and a bias term b such that $\mathbf{w}\cdot\mathbf{x}_j + b > \gamma$ for all $j : d_j = 1$ and $\mathbf{w}\cdot\mathbf{x}_j + b < -\gamma$ for all $j : d_j = 0$. And also let R denote the maximum norm of an input vector. Novikoff (1962) proved that in this case the perceptron algorithm converges after making $O(R^2/\gamma^2)$ updates. The idea of the proof is that the weight vector is always adjusted by a bounded amount in a direction that it has a negative dot product with, and thus can be bounded above by $O(\sqrt{t})$ where *t* is the number of changes to the weight vector. But it can also be bounded below by $O(t)$ because if there exists an (unknown) satisfactory weight vector, then every change makes progress in this (unknown) direction by a positive amount that depends only on the input vector.



Two classes of points, and two of the infinitely many linear boundaries that separate them. Even though the boundaries are at nearly right angles to one another, the perceptron algorithm has no way of choosing between them.

While the perceptron algorithm is guaranteed to converge on *some* solution in the case of a linearly separable training set, it may still pick *any* solution and problems may admit many solutions of varying quality.[8] The *perceptron of optimal stability*, nowadays better known as the linear support vector machine, was designed to solve this problem.

The decision boundary of a perceptron is invariant with respect to scaling of the weight vector; that is, a perceptron trained with initial weight vector $\mathbf{w}$ and learning rate α behaves identically to a perceptron trained with initial weight vector $\mathbf{w}/\alpha$ and learning rate 1. Thus, since the initial weights become irrelevant with increasing number of iterations, the learning rate does not matter in the case of the perceptron and is usually just set to 1.

59.4 Variants

The pocket algorithm with ratchet (Gallant, 1990) solves the stability problem of perceptron learning by keeping the best solution seen so far "in its pocket". The pocket algorithm then returns the solution in the pocket, rather than the last solution. It can be used also for non-separable data sets, where the aim is to find a perceptron with a small number of misclassifications. However, these solutions appear purely stochastically and hence the pocket algorithm neither approaches them gradually in the course of learning, nor are they guaranteed to show up within a given number of learning steps.

The Maxover algorithm (Wendemuth, 1995) [9] is "robust" in the sense that it will converge regardless of (prior) knowledge of linear separability of the data set. In the linear separable case, it will solve the training problem - if desired, even with optimal stability (maximum margin between the classes). For non-separable data sets, it will return a solution with a small number of misclassifications. In all cases, the algorithm gradually approaches the solution in the course of learning, without memorizing previous states and without stochastic jumps. Convergence is to global optimality for separable data sets and to local optimality for non-separable data sets.

In separable problems, perceptron training can also aim at finding the largest separating margin between the classes. The so-called perceptron of optimal stability can be determined by means of iterative training and optimization schemes, such as the Min-Over algorithm (Krauth and Mezard, 1987)[10] or the AdaTron (Anlauf and Biehl, 1989)) .[11] AdaTron uses the fact that the corresponding quadratic optimization problem is convex. The perceptron of optimal stability, together with the kernel trick, are the conceptual foundations of the support vector machine.

The α -perceptron further used a pre-processing layer of fixed random weights, with thresholded output units. This enabled the perceptron to classify analogue patterns, by projecting them into a binary space. In fact, for a projection space of sufficiently high dimension, patterns can become linearly separable.

For example, consider the case of having to classify data into two classes. Here is a small such data set, consisting of points coming from two Gaussian distributions.

- Two-class Gaussian data
- A linear classifier operating on the original space
- A linear classifier operating on a high-dimensional projection

A linear classifier can only separate points with a hyperplane, so no linear classifier can classify all the points here perfectly. On the other hand, the data can be projected into a large number of dimensions. In our example, a random matrix was used to project the data linearly to a 1000-dimensional space; then each resulting data point was transformed through the hyperbolic tangent function. A linear classifier can then separate the data, as shown in the third figure. However the data may still not be completely separable in this space, in which the perceptron algorithm would not converge. In the example shown, stochastic steepest gradient descent was used to adapt the parameters.

Another way to solve nonlinear problems without using multiple layers is to use higher order networks (sigma-pi unit). In this type of network, each element in the input vector is extended with each pairwise combination of multiplied inputs (second order). This can be extended to an *n*-order network.

It should be kept in mind, however, that the best classifier is not necessarily that which classifies all the training data perfectly. Indeed, if we had the prior constraint that the data come from equi-variant Gaussian distributions, the linear separation in the input space is optimal, and the nonlinear solution is overfitted.

Other linear classification algorithms include Winnow, support vector machine and logistic regression.

59.5 Example

A perceptron learns to perform a binary NAND function on inputs x_1 and x_2 .

Inputs: x_0 , x_1 , x_2 , with input x_0 held constant at 1.

Threshold (t): 0.5

Bias (b): 1

Learning rate (r): 0.1

Training set, consisting of four samples: $\{((1,0,0),1),((1,0,1),1),((1,1,0),1),((1,1,1),0)\}$

In the following, the final weights of one iteration become the initial weights of the next. Each cycle over all the samples in the training set is demarcated with heavy lines.

This example can be implemented in the following Python code.

threshold = 0.5 learning_rate = 0.1 weights = [0, 0, 0] training_set = [((1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0), 1), ((1, 1, 1), 0)] def dot_product(values, weights): return sum(value * weight for value, weight in zip(values, weights)) while True: print('-' * 60) error_count = 0 for input_vector, desired_output in training_set: print(weights) result = dot_product(input_vector, weights) > threshold error = desired_output - result if error != 0: error_count += 1 for index, value in enumerate(input_vector): weights[index] += learning_rate * error * value if error_count == 0: break

59.6 Multiclass perceptron

Like most other techniques for training linear classifiers, the perceptron generalizes naturally to multiclass classification. Here, the input x and the output y are drawn from arbitrary sets. A feature representation function $f(x, y)$ maps each possible input/output pair to a finite-dimensional real-valued feature vector. As before, the feature vector is multiplied by a weight vector w , but now the resulting score is used to choose among many possible outputs:

$$\hat{y} = \operatorname{argmax}_y f(x, y) \cdot w.$$

Learning again iterates over the examples, predicting an output for each, leaving the weights unchanged when the predicted output matches the target, and changing them when it does not. The update becomes:

$$w_{t+1} = w_t + f(x, y) - f(x, \hat{y}).$$

This multiclass formulation reduces to the original perceptron when x is a real-valued vector, y is chosen from $\{0, 1\}$, and $f(x, y) = yx$.

For certain problems, input/output representations and features can be chosen so that $\operatorname{argmax}_y f(x, y) \cdot w$ can be found efficiently even though y is chosen from a very large or even infinite set.

In recent years, perceptron training has become popular in the field of natural language processing for such tasks as part-of-speech tagging and syntactic parsing (Collins, 2002).

59.7 References

[1] Freund, Y.; Schapire, R. E. (1999). “Large margin classification using the perceptron algorithm” (PDF). *Machine Learning* **37** (3): 277–296. doi:10.1023/A:1007662407062.

[2] Rosenblatt, Frank (1957), The Perceptron--a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.

[3] Mikel Olazaran (1996). “A Sociological Study of the Official History of the Perceptrons Controversy”. *Social Studies of Science* **26** (3). JSTOR 285702.

[4] Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[5] Aizerman, M. A.; Braverman, E. M.; Rozonoer, L. I. (1964). “Theoretical foundations of the potential function method in pattern recognition learning”. *Automation and Remote Control* **25**: 821–837.

[6] Mohri, Mehryar and Rostamizadeh, Afshin (2013). Perceptron Mistake Bounds arXiv:1305.0208, 2013.

[7] Liou, D.-R.; Liou, J.-W.; Liou, C.-Y. (2013). “Learning Behaviors of Perceptron”. *ISBN 978-1-477554-73-9. iConcept Press.*

[8] Bishop, Christopher M. “Chapter 4. Linear Models for Classification”. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC. p. 194. ISBN 978-0387-31073-2.

[9] A. Wendemuth. Learning the Unlearnable. J. of Physics A: Math. Gen. 28: 5423-5436 (1995)

[10] W. Krauth and M. Mezard. Learning algorithms with optimal stabilty in neural networks. J. of Physics A: Math. Gen. 20: L745-L752 (1987)

[11] J.K. Anlauf and M. Biehl. The AdaTron: an Adaptive Perceptron algorithm. Europhysics Letters 10: 687-692 (1989)

- Aizerman, M. A. and Braverman, E. M. and Lev I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, 25:821–837, 1964.
- Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408. doi:10.1037/h0042519.
- Rosenblatt, Frank (1962), Principles of Neurodynamics. Washington, DC:Spartan Books.
- Minsky M. L. and Papert S. A. 1969. *Perceptrons*. Cambridge, MA: MIT Press.
- Gallant, S. I. (1990). Perceptron-based learning algorithms. IEEE Transactions on Neural Networks, vol. 1, no. 2, pp. 179–191.
- Mohri, Mehryar and Rostamizadeh, Afshin (2013). Perceptron Mistake Bounds arXiv:1305.0208, 2013.
- Novikoff, A. B. (1962). On convergence proofs on perceptrons. Symposium on the Mathematical Theory of Automata, 12, 615-622. Polytechnic Institute of Brooklyn.
- Widrow, B., Lehr, M.A., “30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation,” *Proc. IEEE*, vol 78, no 9, pp. 1415–1442, (1990).
- Collins, M. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '02).
- Yin, Hongfeng (1996), Perceptron-Based Algorithms and Analysis, Spectrum Library, Concordia University, Canada

59.8 External links

- A Perceptron implemented in MATLAB to learn binary NAND function
- Chapter 3 Weighted networks - the perceptron and chapter 4 Perceptron learning of *Neural Networks - A Systematic Introduction* by Raúl Rojas (ISBN 978-3-540-60505-8)
- Explanation of the update rule by Charles Elkan
- History of perceptrons
- Mathematics of perceptrons

Chapter 60

Support vector machine

Not to be confused with Secure Virtual Machine.

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**[1]) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

60.1 Definition

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

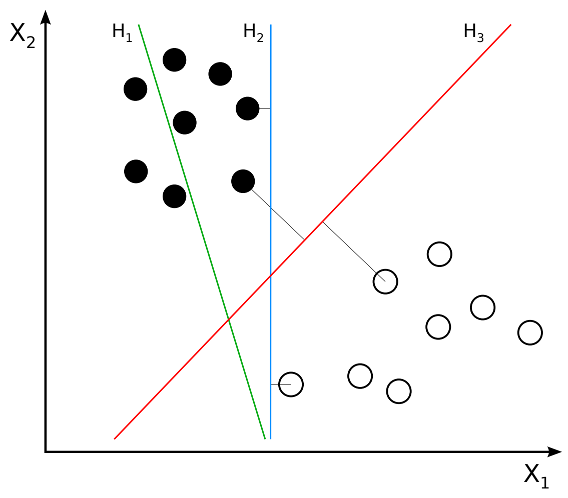
Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem.[2] The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vectors x_i that occur in the data base. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation: $\sum_i \alpha_i k(x_i, x) = \mathrm{constant}.$ Note that if $k(x, y)$ becomes small as y grows further away from x , each term in the sum measures the degree of closeness of the test point x to the corresponding data base point x_i . In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets which are not convex at all in the original space.

60.2 History

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes.[3] The current standard incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995.[1]

60.3 Motivation

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a *new* data point will be in. In the case of support vector machines, a data point is viewed as a p -dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$ -dimensional hyperplane. This is called a linear classifier. There are many hyperplanes



H_1 does not separate the classes. H_2 does, but only with a small margin. H_3 separates them with the maximum margin.

that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the *maximum-margin hyperplane* and the linear classifier it defines is known as a *maximum margin classifier*; or equivalently, the *perceptron of optimal stability.*

60.4 Linear SVM

Given some training data $\mathcal{D}$, a set of n points of the form

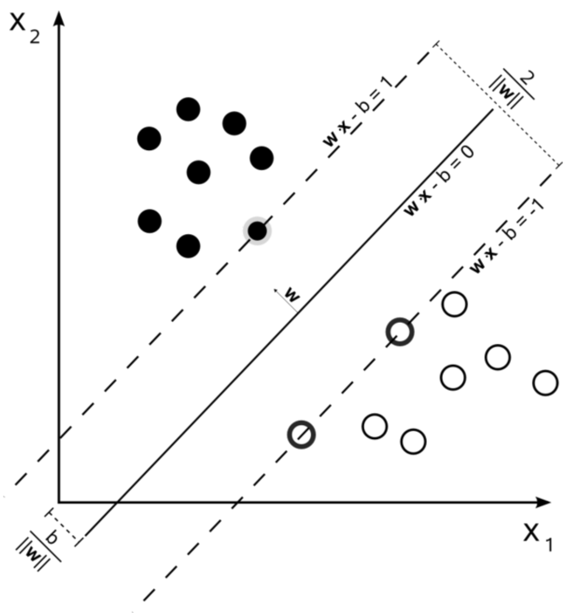
$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1,1\}\}_{i=1}^{n}$$

where the y_i is either 1 or −1, indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p -dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0,$$

where $\cdot$ denotes the dot product and $\mathbf{w}$ the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called “the margin”. These hyperplanes can be described by the equations



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

By using geometry, we find the distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we want to minimize $\|\mathbf{w}\|$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i$$

or

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i$$

This can be rewritten as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{all for } 1 \leq i \leq n. \qquad (1)$$

We can put this together to get the optimization problem:

Minimize (in $\mathbf{w}, b$)

$$\|\mathbf{w}\|$$

subject to (for any $i = 1, \ldots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

60.4.1 Primal form

The optimization problem presented in the preceding section is difficult to solve because it depends on $\|\mathbf{w}\|$, the norm of $\mathbf{w}$, which involves a square root. Fortunately it is possible to alter the equation by substituting $\|\mathbf{w}\|$ with $\frac{1}{2}\|\mathbf{w}\|^2$ (the factor of $\frac{1}{2}$ being used for mathematical convenience) without changing the solution (the minimum of the original and the modified equation have the same $\mathbf{w}$ and b). This is a quadratic programming optimization problem. More clearly:

$$\arg\min_{(\mathbf{w},b)} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to (for any $i = 1, \dots, n$)

$$y_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1.$$

By introducing Lagrange multipliers $\boldsymbol{\alpha}$, the previous constrained problem can be expressed as

$$\arg\min_{\mathbf{w},b}\max_{\boldsymbol{\alpha}\geq 0}\left\{\frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x_i} - b) - 1]\right\}$$

that is we look for a saddle point. In doing so all the points which can be separated as $y_i(\mathbf{w}\cdot\mathbf{x_i} - b) - 1 > 0$ do not matter since we must set the corresponding α_i to zero.

This problem can now be solved by standard quadratic programming techniques and programs. The "stationary" Karush–Kuhn–Tucker condition implies that the solution can be expressed as a linear combination of the training vectors

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x_i}.$$

Only a few α_i will be greater than zero. The corresponding $\mathbf{x_i}$ are exactly the *support vectors*, which lie on the margin and satisfy $y_i(\mathbf{w}\cdot\mathbf{x_i} - b) = 1$. From this one can derive that the support vectors also satisfy

$$\mathbf{w}\cdot\mathbf{x_i} - b = \frac{1}{y_i} = y_i \iff b = \mathbf{w}\cdot\mathbf{x_i} - y_i$$

which allows one to define the offset b . The b depends on y_i and x_i , so it will vary for each data point in the sample. In practice, it is more robust to average over all N_{SV} support vectors, since the average over the sample is an unbiased estimator of the population mean:

$$b = \frac{1}{N_{SV}}\sum_{i=1}^{N_{SV}}(\mathbf{w}\cdot\mathbf{x_i} - y_i)$$

60.4.2 Dual form

Writing the classification rule in its unconstrained dual form reveals that the *maximum-margin hyperplane* and therefore the classification task is only a function of the *support vectors*, the subset of the training data that lie on the margin.

Using the fact that $\|\mathbf{w}\|^2 = \mathbf{w}^T\cdot\mathbf{w}$ and substituting $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x_i}$, one can show that the dual of the SVM reduces to the following optimization problem:

Maximize (in α_i)

$$\tilde{L}(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j$$

subject to (for any $i = 1, \dots, n$)

$$\alpha_i \geq 0,$$

and to the constraint from the minimization in b

$$\sum_{i=1}^{n}\alpha_i y_i = 0.$$

Here the kernel is defined by $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i\cdot\mathbf{x}_j$.

W can be computed thanks to the α terms:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i.$$

60.4.3 Biased and unbiased hyperplanes

For simplicity reasons, sometimes it is required that the hyperplane pass through the origin of the coordinate system. Such hyperplanes are called *unbiased*, whereas general hyperplanes not necessarily passing through the origin are called *biased*. An unbiased hyperplane can be enforced by setting $b = 0$ in the primal optimization problem. The corresponding dual is identical to the dual given above without the equality constraint

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

60.5 Soft margin

In 1995, Corinna Cortes and Vladimir N. Vapnik suggested a modified maximum margin idea that allows for mislabeled examples.[1] If there exists no hyperplane that can split the "yes" and "no" examples, the *Soft Margin* method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces non-negative slack variables, ξ_i , which measure the degree of misclassification of the data x_i

$$y_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i \quad 1\leq i\leq n. \qquad (2)$$

The objective function is then increased by a function which penalizes non-zero ξ_i , and the optimization becomes a trade off between a large margin and a small error penalty. If the penalty function is linear, the optimization problem becomes:

$$\underset{\mathbf{w},\xi,b}{\arg\min}\left\{\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\xi_i\right\}$$

subject to (for any $i=1,\ldots n$)

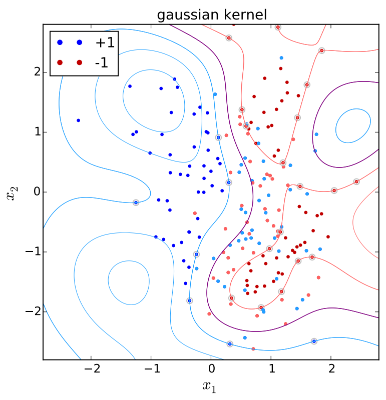
$$y_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i, \quad \xi_i\geq 0$$

Using the hinge function notation like that in MARS, this optimization problem can be rewritten as $\sum_i[1-y_i(w\cdot x_i+b)]_+ +\lambda\|w\|^2$, wherein let $[1-y_i(w\cdot x_i+b)]_+ = [\xi_i]_+=\xi_i, \quad \lambda=1/2C$.

This constraint in (2) along with the objective of minimizing $\|\mathbf{w}\|$ can be solved using Lagrange multipliers as done above. One then has to solve the following problem:

$$\underset{\mathbf{w},\xi,b}{\arg\min}\,\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\max}\left\{\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\xi_i-\sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x_i}-b)-$$

with $\alpha_i,\beta_i\geq 0$.



An example for a result of soft-margin SVM

60.5.1 Dual form

Maximize (in α_i)

$$\tilde{L}(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_jy_iy_jk(\mathbf{x}_i,\mathbf{x}_j)$$

subject to (for any $i=1,\ldots,n$)

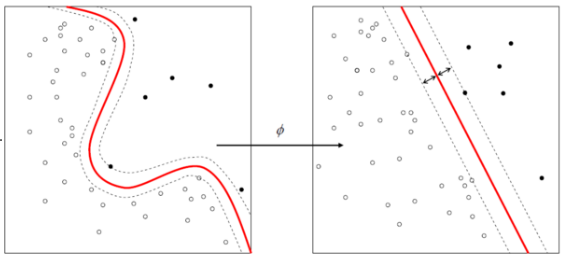
$$0\leq\alpha_i\leq C,$$

and

$$\sum_{i=1}^{n}\alpha_iy_i=0.$$

The key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant *C* appearing only as an additional constraint on the Lagrange multipliers. For the above formulation and its huge impact in practice, Cortes and Vapnik received the 2008 ACM Paris Kanellakis Award.[4] Nonlinear penalty functions have been used, particularly to reduce the effect of outliers on the classifier, but unless care is taken the problem becomes non-convex, and thus it is considerably more difficult to find a global solution.

60.6 Nonlinear classification



Kernel machine

The original optimal hyperplane algorithm proposed by Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick (originally proposed by Aizerman et al.[5]) to maximum-margin hyperplanes.[6] The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimensions. Maximum margin classifiers are well regularized, and previously it was widely believed that the infinite dimensions do not spoil the results. However, it has been shown that higher dimensions do increase the generalization error, although the amount is bounded.[7]

Some common kernels include:

- Polynomial (homogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j})^d$
- Polynomial (inhomogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$
- Gaussian radial basis function: $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$, for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/2\sigma^2$
- Hyperbolic tangent: $k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$, for some (not every) $\kappa > 0$ and $c < 0$

The kernel is related to the transform $\varphi(\mathbf{x_i})$ by the equation $k(\mathbf{x_i}, \mathbf{x_j}) = \varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x_j})$. The value **w** is also in the transformed space, with $\mathbf{w} = \sum_i \alpha_i y_i \varphi(\mathbf{x}_i)$. Dot products with **w** for classification can again be computed by the kernel trick, i.e. $\mathbf{w} \cdot \varphi(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$. However, there does not in general exist a value **w'** such that $\mathbf{w} \cdot \varphi(\mathbf{x}) = k(\mathbf{w}', \mathbf{x})$.

60.7 Properties

SVMs belong to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron. They can also be considered a special case of Tikhonov regularization. A special property is that they simultaneously minimize the empirical *classification error* and maximize the *geometric margin*; hence they are also known as **maximum margin classifiers**.

A comparison of the SVM to other classifiers has been made by Meyer, Leisch and Hornik.[8]

60.7.1 Parameter selection

The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C. A common choice is a Gaussian kernel, which has a single parameter γ . The best combination of C and γ is often selected by a grid search with exponentially growing sequences of C and γ , for example, $C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}$; $\gamma \in \{2^{-15}, 2^{-13}, \dots, 2^{1}, 2^{3}\}$. Typically, each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are picked. Alternatively, recent work in Bayesian optimization can be used to select C and γ , often requiring the evaluation of far fewer parameter combinations than grid search. The final model, which is used for testing and for classifying new data, is then trained on the whole training set using the selected parameters.[9]

60.7.2 Issues

Potential drawbacks of the SVM are the following three aspects:

- Uncalibrated class membership probabilities
- The SVM is only directly applicable for two-class tasks. Therefore, algorithms that reduce the multi-class task to several binary problems have to be applied; see the multi-class SVM section.
- Parameters of a solved model are difficult to interpret.

60.8 Extensions

60.8.1 Multiclass SVM

Multiclass SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements.

The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems.[10] Common methods for such reduction include:[10] [11]

- Building binary classifiers which distinguish between (i) one of the labels and the rest (*one-versus-all*) or (ii) between every pair of classes (*one-versus-one*). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.
- Directed acyclic graph SVM (DAGSVM)[12]
- Error-correcting output codes[13]

Crammer and Singer proposed a multiclass SVM method which casts the multiclass classification problem into a single optimization problem, rather than decomposing it into multiple binary classification problems.[14] See also Lee, Lin and Wahba.[15][16]

60.8.2 Transductive support vector machines

Transductive support vector machines extend SVMs in that they could also treat partially labeled data in semi-supervised learning by following the principles of transduction. Here, in addition to the training set $\mathcal{D}$, the learner is also given a set

$$\mathcal{D}^\star = \{\mathbf{x}_i^\star | \mathbf{x}_i^\star \in \mathbb{R}^p\}_{i=1}^k$$

of test examples to be classified. Formally, a transductive support vector machine is defined by the following primal optimization problem:[17]

Minimize (in $\mathbf{w}, b, \mathbf{y}^\star$)

$$\frac{1}{2}\|\mathbf{w}\|^2$$

subject to (for any $i = 1, \dots, n$ and any $j = 1, \dots, k$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1,$$

$$y_j^\star(\mathbf{w} \cdot \mathbf{x_j^\star} - b) \geq 1,$$

and

$$y_j^\star \in \{-1, 1\}.$$

Transductive support vector machines were introduced by Vladimir N. Vapnik in 1998.

60.8.3 Structured SVM

SVMs have been generalized to structured SVMs, where the label space is structured and of possibly infinite size.

60.8.4 Regression

A version of SVM for regression was proposed in 1996 by Vladimir N. Vapnik, Harris Drucker, Christopher J. C. Burges, Linda Kaufman and Alexander J. Smola.[18] This method is called support vector regression (SVR). The model produced by support vector classification (as described above) depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. Another SVM version known as least squares support vector machine (LS-SVM) has been proposed by Suykens and Vandewalle.[19]

Training the original SVR means solving[20]

$$\frac{1}{2}\|w\|^2$$

$$\begin{cases} y_i - \langle w, x_i \rangle - b \leq \epsilon \\ \langle w, x_i \rangle + b - y_i \leq \epsilon \end{cases}$$

where x_i is a training sample with target value y_i . The inner product plus intercept $\langle w, x_i \rangle + b$ is the prediction for that sample, and ϵ is a free parameter that serves as a threshold: all predictions have to be within an ϵ range of the true predictions. Slack variables are usually added into the above to allow for errors and to allow approximation in the case the above problem is infeasible.

60.9 Interpreting SVM models

The SVM algorithm has been widely applied in the biological and other sciences. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models.[21][22] Support vector machine weights have also been used to interpret SVM models in the past.[23] Posthoc interpretation of support vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

60.10 Implementation

The parameters of the maximum-margin hyperplane are derived by solving the optimization. There exist several specialized algorithms for quickly solving the QP problem that arises from SVMs, mostly relying on heuristics for breaking the problem down into smaller, more-manageable chunks. A common method is Platt's sequential minimal optimization (SMO) algorithm, which breaks the problem down into 2-dimensional sub-problems that may be solved analytically, eliminating the need for a numerical optimization algorithm.[24]

Another approach is to use an interior point method that uses Newton-like iterations to find a solution of the Karush–Kuhn–Tucker conditions of the primal and dual problems.[25] Instead of solving a sequence of broken down problems, this approach directly solves the problem altogether. To avoid solving a linear system involving the large kernel matrix, a low rank approximation to the matrix is often used in the kernel trick.

The special case of linear support vector machines can be solved more efficiently by the same kind of algorithms used to optimize its close cousin, logistic regression; this class of algorithms includes sub-gradient descent (e.g., PEGASOS[26]) and coordinate descent (e.g., LIBLINEAR[27]). LIBLINEAR has some attractive training time properties. Each convergence iteration takes time linear in the time taken to read the train data and the iterations also have a Q-Linear Convergence property, making the algorithm extremely fast.

The general kernel SVMs can also be solved more efficiently using sub-gradient descent (e.g. P-packSVM[28]), especially when parallelization is allowed.

Kernel SVMs are available in many machine learning toolkits, including LIBSVM, MATLAB, SVMlight, kernlab, scikit-learn, Shogun, Weka, Shark, JKernelMachines and others.

60.11 Applications

SVMs can be used to solve various real world problems:

- SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.
- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.
- SVMs are also useful in medical science to classify proteins with up to 90% of the compounds classified correctly.
- Hand-written characters can be recognized using SVM.

60.12 See also

- In situ adaptive tabulation
- Kernel machines
- Fisher kernel
- Platt scaling
- Polynomial kernel
- Predictive analytics
- Regularization perspectives on support vector machines
- Relevance vector machine, a probabilistic sparse kernel model identical in functional form to SVM
- Sequential minimal optimization
- Winnow (algorithm)

60.13 References

[1] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". *Machine Learning* **20** (3): 273. doi:10.1007/BF00994018.

[2] • Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, B. P. (2007). "Section 16.5. Support Vector Machines". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

[3] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992). "A training algorithm for optimal margin classifiers". *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. p. 144. doi:10.1145/130385.130401. ISBN 089791497X.

[4] ACM Website, Press release of March 17th 2009. http://www.acm.org/press-room/news-releases/awards-08-groupa

[5] Aizerman, Mark A.; Braverman, Emmanuel M.; and Rozonoer, Lev I. (1964). "Theoretical foundations of the potential function method in pattern recognition learning". *Automation and Remote Control* **25**: 821–837.

[6] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992). "A training algorithm for optimal margin classifiers". *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. p. 144. doi:10.1145/130385.130401. ISBN 089791497X.

[7] Jin, Chi; Wang, Liwei (2012). *Dimensionality dependent PAC-Bayes margin bound*. Advances in Neural Information Processing Systems.

[8] Meyer, D.; Leisch, F.; Hornik, K. (2003). "The support vector machine under test". *Neurocomputing* **55**: 169. doi:10.1016/S0925-2312(03)00431-4.

[9] Hsu, Chih-Wei; Chang, Chih-Chung; and Lin, Chih-Jen (2003). *A Practical Guide to Support Vector Classification* (PDF) (Technical report). Department of Computer Science and Information Engineering, National Taiwan University.

[10] Duan, K. B.; Keerthi, S. S. (2005). "Which Is the Best Multiclass SVM Method? An Empirical Study". *Multiple Classifier Systems* (PDF). LNCS **3541**. pp. 278–285. doi:10.1007/11494683_28. ISBN 978-3-540-26306-7.

[11] Hsu, Chih-Wei; and Lin, Chih-Jen (2002). "A Comparison of Methods for Multiclass Support Vector Machines". *IEEE Transactions on Neural Networks*.

[12] Platt, John; Cristianini, N.; and Shawe-Taylor, J. (2000). "Large margin DAGs for multiclass classification". In Solla, Sara A.; Leen, Todd K.; and Müller, Klaus-Robert; eds. *Advances in Neural Information Processing Systems* (PDF). MIT Press. pp. 547–553.

[13] Dietterich, Thomas G.; and Bakiri, Ghulum; Bakiri (1995). "Solving Multiclass Learning Problems via Error-Correcting Output Codes" (PDF). *Journal of Artificial Intelligence Research, Vol. 2* **2**: 263–286. arXiv:cs/9501101. Bibcode:1995cs........1101D.

[14] Crammer, Koby; and Singer, Yoram (2001). "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines" (PDF). *J. of Machine Learning Research* **2**: 265–292.

[15] Lee, Y.; Lin, Y.; and Wahba, G. (2001). "Multicategory Support Vector Machines" (PDF). *Computing Science and Statistics 33*.

[16] Lee, Y.; Lin, Y.; Wahba, G. (2004). "Multicategory Support Vector Machines". *Journal of the American Statistical Association* **99** (465): 67. doi:10.1198/016214504000000098.

[17] Joachims, Thorsten; "Transductive Inference for Text Classification using Support Vector Machines", Proceedings of the 1999 International Conference on Machine Learning (ICML 1999), pp. 200-209.

[18] Drucker, Harris; Burges, Christopher J. C.; Kaufman, Linda; Smola, Alexander J.; and Vapnik, Vladimir N. (1997); "Support Vector Regression Machines", in *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155–161, MIT Press.

[19] Suykens, Johan A. K.; Vandewalle, Joos P. L.; *Least squares support vector machine classifiers*, Neural Processing Letters, vol. 9, no. 3, Jun. 1999, pp. 293–300.

[20] Smola, Alex J.; Schölkopf, Bernhard (2004). "A tutorial on support vector regression" (PDF). *Statistics and Computing* **14** (3): 199–222.

[21] Bilwaj Gaonkar, Christos Davatzikos Analytic estimation of statistical significance maps for support vector machine based multi-variate image analysis and classification

[22] R. Cuingnet, C. Rosso, M. Chupin, S. Lehéricy, D. Dormont, H. Benali, Y. Samson and O. Colliot, Spatial regularization of SVM for the detection of diffusion alterations associated with stroke outcome, Medical Image Analysis, 2011, 15 (5): 729-737

[23] Statnikov, A., Hardin, D., & Aliferis, C. (2006). Using SVM weight-based methods to identify causally relevant and non-causally relevant variables. sign, 1, 4.

[24] John C. Platt (1999). *Using Analytic QP and Sparseness to Speed Training of Support Vector Machines* (PDF). NIPS.

[25] Ferris, M. C.; Munson, T. S. (2002). "Interior-Point Methods for Massive Support Vector Machines". *SIAM Journal on Optimization* **13** (3): 783. doi:10.1137/S1052623400374379.

[26] Shai Shalev-Shwartz; Yoram Singer; Nathan Srebro (2007). *Pegasos: Primal Estimated sub-GrAdient SOlver for SVM* (PDF). ICML.

[27] R.-E. Fan; K.-W. Chang; C.-J. Hsieh; X.-R. Wang; C.-J. Lin (2008). "LIBLINEAR: A library for large linear classification". *Journal of Machine Learning Research* **9**: 1871–1874.

[28] Zeyuan Allen Zhu et al. (2009). *P-packSVM: Parallel Primal grAdient desCent Kernel SVM* (PDF). ICDM.

60.14 External links

- www.support-vector.net The key book about the method, "An Introduction to Support Vector Machines" with online software
- Burges, Christopher J. C.; A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2:121–167, 1998
- www.kernel-machines.org *(general information and collection of research papers)*
- www.support-vector-machines.org *(Literature, Review, Software, Links related to Support Vector Machines — Academic Site)*
- videolectures.net *(SVM-related video lectures)*
- Karatzoglou, Alexandros et al.; Support Vector Machines in R, Journal of Statistical Software April 2006, Volume 15, Issue 9.
- libsvm LIBSVM is a popular library of SVM learners
- liblinear liblinear is a library for large linear classification including some SVMs
- Shark Shark is a C++ machine learning library implementing various types of SVMs
- dlib dlib is a C++ library for working with kernel methods and SVMs
- SVM light is a collection of software tools for learning and classification using SVM.
- SVMJS live demo is a GUI demo for Javascript implementation of SVMs
- Gesture Recognition Toolkit contains an easy to use wrapper for libsvm

60.15 Bibliography

- Theodoridis, Sergios; and Koutroumbas, Konstantinos; "Pattern Recognition", 4th Edition, Academic Press, 2009, ISBN 978-1-59749-272-0
- Cristianini, Nello; and Shawe-Taylor, John; *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000. ISBN 0-521-78019-5 *(SVM Book)*
- Huang, Te-Ming; Kecman, Vojislav; and Kopriva, Ivica (2006); *Kernel Based Algorithms for Mining Huge Data Sets*, in *Supervised, Semi-supervised, and Unsupervised Learning*, Springer-Verlag, Berlin, Heidelberg, 260 pp. 96 illus., Hardcover, ISBN 3-540-31681-7

- Kecman, Vojislav; *Learning and Soft Computing — Support Vector Machines, Neural Networks, Fuzzy Logic Systems*, The MIT Press, Cambridge, MA, 2001.

- Schölkopf, Bernhard; and Smola, Alexander J.; *Learning with Kernels*, MIT Press, Cambridge, MA, 2002. ISBN 0-262-19475-9

- Schölkopf, Bernhard; Burges, Christopher J. C.; and Smola, Alexander J. (editors); *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge, MA, 1999. ISBN 0-262-19416-3.

- Shawe-Taylor, John; and Cristianini, Nello; *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004. ISBN 0-521-81397-2 *(Kernel Methods Book)*

- Steinwart, Ingo; and Christmann, Andreas; *Support Vector Machines*, Springer-Verlag, New York, 2008. ISBN 978-0-387-77241-7 *(SVM Book)*

- Tan, Peter Jing; and Dowe, David L. (2004); *MML Inference of Oblique Decision Trees*, Lecture Notes in Artificial Intelligence (LNAI) 3339, Springer-Verlag, pp1082-1088. (This paper uses minimum message length (MML) and actually incorporates probabilistic support vector machines in the leaves of decision trees.)

- Vapnik, Vladimir N.; *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995. ISBN 0-387-98780-0

- Vapnik, Vladimir N.; and Kotz, Samuel; *Estimation of Dependences Based on Empirical Data*, Springer, 2006. ISBN 0-387-30865-2, 510 pages [this is a reprint of Vapnik's early book describing philosophy behind SVM approach. The 2006 Appendix describes recent development].

- Fradkin, Dmitriy; and Muchnik, Ilya; *Support Vector Machines for Classification* in Abello, J.; and Carmode, G. (Eds); *Discrete Methods in Epidemiology*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 70, pp. 13–20, 2006. . Succinctly describes theoretical ideas behind SVM.

- Bennett, Kristin P.; and Campbell, Colin; *Support Vector Machines: Hype or Hallelujah?*, SIGKDD Explorations, 2, 2, 2000, 1–13. . Excellent introduction to SVMs with helpful figures.

- Ivanciuc, Ovidiu; *Applications of Support Vector Machines in Chemistry*, in *Reviews in Computational Chemistry*, Volume 23, 2007, pp. 291–400. Reprint available:

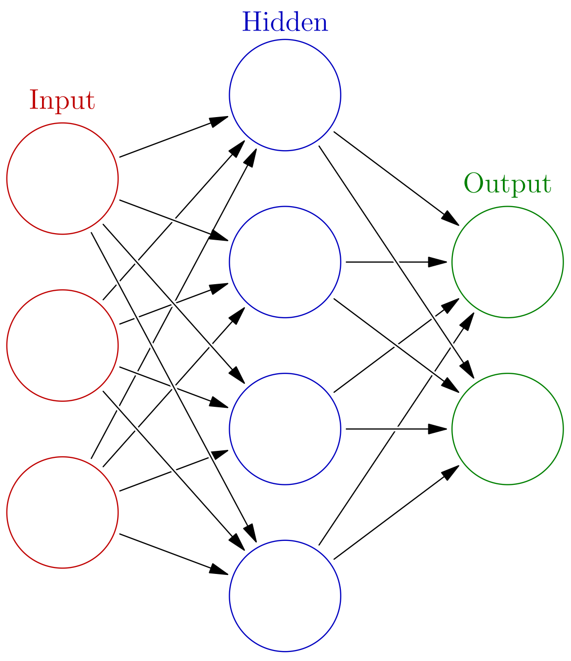
- Catanzaro, Bryan; Sundaram, Narayanan; and Keutzer, Kurt; *Fast Support Vector Machine Training and Classification on Graphics Processors*, in *International Conference on Machine Learning*, 2008

- Campbell, Colin; and Ying, Yiming; *Learning with Support Vector Machines*, 2011, Morgan and Claypool. ISBN 978-1-60845-616-1.

Chapter 61

Artificial neural network

"Neural network" redirects here. For networks of living neurons, see Biological neural network. For the journal, see Neural Networks (journal). For the evolutionary concept, see Neutral network (evolution).



An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

In machine learning and cognitive science, **artificial neural networks** (**ANNs**) are a family of statistical learning models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which send messages to each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

For example, a neural network for handwriting recognition is defined by a set of input neurons which may be activated by the pixels of an input image. After being weighted and transformed by a function (determined by the network's designer), the activations of these neurons are then passed on to other neurons. This process is repeated until finally, an output neuron is activated. This determines which character was read.

Like other machine learning methods - systems that learn from data - neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition.

61.1 Background

Examinations of humans' central nervous systems inspired the concept of artificial neural networks. In an artificial neural network, simple artificial nodes, known as "neurons", "neurodes", "processing elements" or "units", are connected together to form a network which mimics a biological neural network.

There is no single formal definition of what an artificial neural network is. However, a class of statistical models may commonly be called "Neural" if it possesses the following characteristics:

1. contains sets of adaptive weights, i.e. numerical parameters that are tuned by a learning algorithm, and
2. capability of approximating non-linear functions of their inputs.

The adaptive weights can be thought of as connection strengths between neurons, which are activated during training and prediction.

Neural networks are similar to biological neural networks in the performing of functions collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which individual units are assigned. The term "neural network" usually refers to models employed in statistics, cognitive psychology and artificial intelligence. Neural network models which emulate the central nervous system are part of theoretical neuroscience and computational neuroscience.

In modern software implementations of artificial neural networks, the approach inspired by biology has been largely abandoned for a more practical approach based on statistics and signal processing. In some of these systems, neural networks or parts of neural networks (like artificial neurons) form components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such systems is more suitable for real-world problem solving, it has little to do with the traditional, artificial intelligence connectionist models. What they do have in common, however, is the principle of non-linear, distributed, parallel and local processing and adaptation. Historically, the use of neural network models marked a directional shift in the late eighties from high-level (symbolic) AI, characterized by expert systems with knowledge embodied in *if-then* rules, to low-level (sub-symbolic) machine learning, characterized by knowledge embodied in the parameters of a dynamical system.

61.2 History

Warren McCulloch and Walter Pitts[1] (1943) created a computational model for neural networks based on mathematics and algorithms called threshold logic. This model paved the way for neural network research to split into two distinct approaches. One approach focused on biological processes in the brain and the other focused on the application of neural networks to artificial intelligence.

In the late 1940s psychologist Donald Hebb[2] created a hypothesis of learning based on the mechanism of neural plasticity that is now known as Hebbian learning. Hebbian learning is considered to be a 'typical' unsupervised learning rule and its later variants were early models for long term potentiation. Researchers started applying these ideas to computational models in 1948 with Turing's B-type machines.

Farley and Wesley A. Clark[3] (1954) first used computational machines, then called "calculators," to simulate a Hebbian network at MIT. Other neural network computational machines were created by Rochester, Holland, Habit, and Duda[4] (1956).

Frank Rosenblatt[5] (1958) created the perceptron, an algorithm for pattern recognition based on a two-layer computer learning network using simple addition and subtraction. With mathematical notation, Rosenblatt also described circuitry not in the basic perceptron, such as the exclusive-or circuit, a circuit whose mathematical computation could not be processed until after the backpropagation algorithm was created by Paul Werbos[6] (1975).

Neural network research stagnated after the publication of machine learning research by Marvin Minsky and Seymour Papert[7] (1969), who discovered two key issues with the computational machines that processed neural networks. The first was that single-layer neural networks were incapable of processing the exclusive-or circuit. The second significant issue was that computers didn't have enough processing power to effectively handle the long run time required by large neural networks. Neural network research slowed until computers achieved greater processing power. Another key advance that came later was the backpropagation algorithm which effectively solved the exclusive-or problem (Werbos 1975).[6]

The parallel distributed processing of the mid-1980s became popular under the name connectionism. The textbook by David E. Rumelhart and James McClelland[8] (1986) provided a full exposition of the use of connectionism in computers to simulate neural processes.

Neural networks, as used in artificial intelligence, have traditionally been viewed as simplified models of neural processing in the brain, even though the relation between this model and the biological architecture of the brain is debated; it's not clear to what degree artificial neural networks mirror brain function.[9]

Support vector machines and other, much simpler methods such as linear classifiers gradually overtook neural networks in machine learning popularity. But the advent of deep learning in the late 2000s sparked renewed interest in neural nets.

61.2.1 Improvements since 2006

Computational devices have been created in CMOS, for both biophysical simulation and neuromorphic computing. More recent efforts show promise for creating nanodevices[10] for very large scale principal components analyses and convolution. If successful, these efforts could usher in a new era of neural computing[11] that is a step beyond digital computing, because it depends on learning rather than programming and because it is fundamentally analog rather than digital even though the first instantiations may in fact be with CMOS digital devices.

Between 2009 and 2012, the recurrent neural networks and deep feedforward neural networks developed in the research group of Jürgen Schmidhuber at the Swiss AI Lab IDSIA have won eight international competitions in pattern recognition and machine learning.[12][13] For example, the bi-directional and multi-dimensional long short term memory (LSTM)[14][15][16][17] of Alex Graves et al. won three competitions in connected handwriting recognition at the 2009 International Conference on Document Analysis and Recognition (ICDAR), without any prior knowledge about the three different languages to be learned.

Fast GPU-based implementations of this approach by

Dan Ciresan and colleagues at IDSIA have won several pattern recognition contests, including the IJCNN 2011 Traffic Sign Recognition Competition,[18][19] the ISBI 2012 Segmentation of Neuronal Structures in Electron Microscopy Stacks challenge,[20] and others. Their neural networks also were the first artificial pattern recognizers to achieve human-competitive or even superhuman performance[21] on important benchmarks such as traffic sign recognition (IJCNN 2012), or the MNIST handwritten digits problem of Yann LeCun at NYU.

Deep, highly nonlinear neural architectures similar to the 1980 neocognitron by Kunihiko Fukushima[22] and the "standard architecture of vision",[23] inspired by the simple and complex cells identified by David H. Hubel and Torsten Wiesel in the primary visual cortex, can also be pre-trained by unsupervised methods[24][25] of Geoff Hinton's lab at University of Toronto.[26][27] A team from this lab won a 2012 contest sponsored by Merck to design software to help find molecules that might lead to new drugs.[28]

61.3 Models

Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function $f : X \to Y$ or a distribution over X or both X and Y , but sometimes models are also intimately associated with a particular learning algorithm or learning rule. A common use of the phrase "ANN model" is really the definition of a *class* of such functions (where members of the class are obtained by varying parameters, connection weights, or specifics of the architecture such as the number of neurons or their connectivity).

61.3.1 Network function

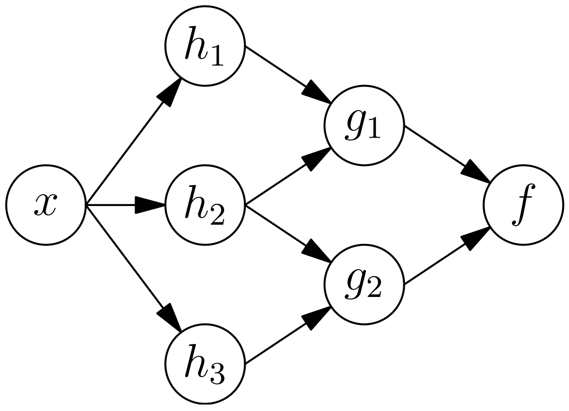
See also: Graphical models

The word *network* in the term 'artificial neural network' refers to the inter–connections between the neurons in the different layers of each system. An example system has three layers. The first layer has input neurons which send data via synapses to the second layer of neurons, and then via more synapses to the third layer of output neurons. More complex systems will have more layers of neurons, some having increased layers of input neurons and output neurons. The synapses store parameters called "weights" that manipulate the data in the calculations.

An ANN is typically defined by three types of parameters:

1. The interconnection pattern between the different layers of neurons
2. The learning process for updating the weights of the interconnections
3. The activation function that converts a neuron's weighted input to its output activation.

Mathematically, a neuron's network function $f(x)$ is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the *nonlinear weighted sum*, where $f(x) = K\left(\sum_i w_i g_i(x)\right)$, where K (commonly referred to as the activation function[29]) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$.



ANN dependency graph

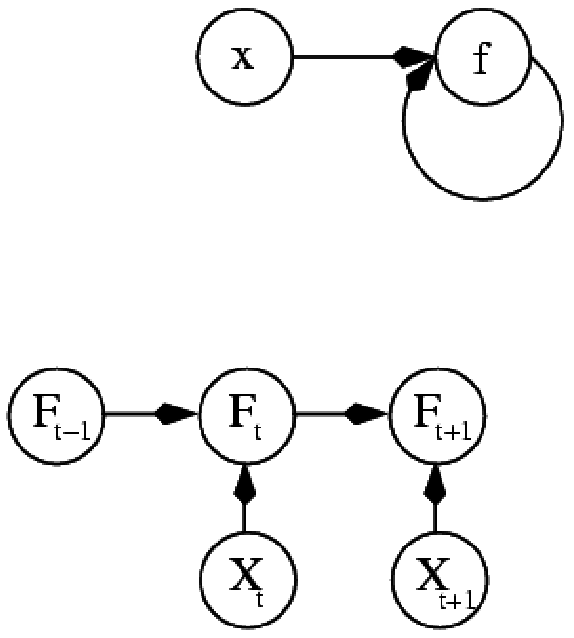
This figure depicts such a decomposition of f , with dependencies between variables indicated by arrows. These can be interpreted in two ways.

The first view is the functional view: the input x is transformed into a 3-dimensional vector h , which is then transformed into a 2-dimensional vector g , which is finally transformed into f . This view is most commonly encountered in the context of optimization.

The second view is the probabilistic view: the random variable $F = f(G)$ depends upon the random variable $G = g(H)$, which depends upon $H = h(X)$, which depends upon the random variable X . This view is most commonly encountered in the context of graphical models.

The two views are largely equivalent. In either case, for this particular network architecture, the components of individual layers are independent of each other (e.g., the components of g are independent of each other given their input h). This naturally enables a degree of parallelism in the implementation.

Networks such as the previous one are commonly called feedforward, because their graph is a directed acyclic graph. Networks with cycles are commonly called recurrent. Such networks are commonly depicted in the



Two separate depictions of the recurrent ANN dependency graph

manner shown at the top of the figure, where f is shown as being dependent upon itself. However, an implied temporal dependence is not shown.

61.3.2 Learning

What has attracted the most interest in neural networks is the possibility of *learning*. Given a specific *task* to solve, and a *class* of functions F , learning means using a set of *observations* to find $f^* \in F$ which solves the task in some *optimal* sense.

This entails defining a cost function $C : F \to \mathbb{R}$ such that, for the optimal solution f^* , $C(f^*) \leq C(f)$ $\forall f \in F$ – i.e., no solution has a cost less than the cost of the optimal solution (see Mathematical optimization).

The cost function C is an important concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Learning algorithms search through the solution space to find a function that has the smallest possible cost.

For applications where the solution is dependent on some data, the cost must necessarily be a *function of the observations*, otherwise we would not be modelling anything related to the data. It is frequently defined as a statistic to which only approximations can be made. As a simple example, consider the problem of finding the model f , which minimizes $C = E\left[(f(x) - y)^2\right]$, for data pairs (x, y) drawn from some distribution $\mathcal{D}$. In practical situations we would only have N samples from $\mathcal{D}$ and thus, for the above example, we would only minimize $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i)-y_i)^2$. Thus, the cost is minimized over a sample of the data rather than the entire distribution generating the data.

When $N \to \infty$ some form of online machine learning must be used, where the cost is partially minimized as each new example is seen. While online machine learning is often used when $\mathcal{D}$ is fixed, it is most useful in the case where the distribution changes slowly over time. In neural network methods, some form of online machine learning is frequently used for finite datasets.

See also: Mathematical optimization, Estimation theory and Machine learning

Choosing a cost function

While it is possible to define some arbitrary ad hoc cost function, frequently a particular cost will be used, either because it has desirable properties (such as convexity) or because it arises naturally from a particular formulation of the problem (e.g., in a probabilistic formulation the posterior probability of the model can be used as an inverse cost). Ultimately, the cost function will depend on the desired task. An overview of the three main categories of learning tasks is provided below:

61.3.3 Learning paradigms

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning

In supervised learning, we are given a set of example pairs $(x, y), x \in X, y \in Y$ and the aim is to find a function $f : X \to Y$ in the allowed class of functions that matches the examples. In other words, we wish to *infer* the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error, which tries to minimize the average squared error between the network's output, $f(x)$, and the target value y over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called multilayer perceptrons, one obtains the common and well-known backpropagation algorithm for training neural networks.

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a "teacher", in the form of a function that provides continuous feedback on the quality of solutions obtained thus far.

Unsupervised learning

In unsupervised learning, some data x is given and the cost function to be minimized, that can be any function of the data x and the network's output, f .

The cost function is dependent on the task (what we are trying to model) and our *a priori* assumptions (the implicit properties of our model, its parameters and the observed variables).

As a trivial example, consider the model $f(x) = a$ where a is a constant and the cost $C = E[(x - f(x))^2]$. Minimizing this cost will give us a value of a that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: for example, in compression it could be related to the mutual information between x and $f(x)$, whereas in statistical modeling, it could be related to the posterior probability of the model given the data (note that in both of those examples those quantities would be maximized rather than minimized).

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

Reinforcement learning

In reinforcement learning, data x are usually not given, but generated by an agent's interactions with the environment. At each point in time t , the agent performs an action y_t and the environment generates an observation x_t and an instantaneous cost c_t , according to some (usually unknown) dynamics. The aim is to discover a *policy* for selecting actions that minimizes some measure of a long-term cost, e.g., the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

More formally the environment is modeled as a Markov decision process (MDP) with states $s_1, ..., s_n \in S$ and actions $a_1, ..., a_m \in A$ with the following probability distributions: the instantaneous cost distribution $P(c_t|s_t)$, the observation distribution $P(x_t|s_t)$ and the transition $P(s_{t+1}|s_t, a_t)$, while a policy is defined as the conditional distribution over actions given the observations. Taken together, the two then define a Markov chain (MC). The aim is to discover the policy (i.e., the MC) that minimizes the cost.

ANNs are frequently used in reinforcement learning as part of the overall algorithm.[30][31] Dynamic programming has been coupled with ANNs (Neuro dynamic programming) by Bertsekas and Tsitsiklis[32] and applied to multi-dimensional nonlinear problems such as those involved in vehicle routing,[33] natural resources management[34][35] or medicine[36] because of the ability of ANNs to mitigate losses of accuracy even when reducing the discretization grid density for numerically approximating the solution of the original control problems.

Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

See also: dynamic programming and stochastic control

61.3.4 Learning algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation.

Most of the algorithms used in training artificial neural networks employ some form of gradient descent, using backpropagation to compute the actual gradients. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. The backpropagation training algorithms are usually classified into three categories: steepest descent (with variable learning rate, with variable learning rate and momentum, resilient backpropagation), quasi-Newton (Broyden-Fletcher-Goldfarb-Shanno, one step secant, Levenberg-Marquardt) and conjugate gradient (Fletcher-Reeves update, Polak-Ribiére update, Powell-Beale restart, scaled conjugate gradient). [37]

Evolutionary methods,[38] gene expression programming,[39] simulated annealing,[40] expectation-maximization, non-parametric methods and particle swarm optimization[41] are some commonly used methods for training neural networks.

See also: machine learning

61.4 Employing artificial neural networks

Perhaps the greatest advantage of ANNs is their ability to be used as an arbitrary function approximation mechanism that 'learns' from observed data. However, using them is not so straightforward, and a relatively good understanding of the underlying theory is essential.

- Choice of model: This will depend on the data representation and the application. Overly complex models tend to lead to problems with learning.

- Learning algorithm: There are numerous trade-offs between learning algorithms. Almost any algorithm will work well with the *correct hyperparameters* for training on a particular fixed data set. However, selecting and tuning an algorithm for training on unseen data requires a significant amount of experimentation.

- Robustness: If the model, cost function and learning algorithm are selected appropriately the resulting ANN can be extremely robust.

With the correct implementation, ANNs can be used naturally in online learning and large data set applications. Their simple implementation and the existence of mostly local dependencies exhibited in the structure allows for fast, parallel implementations in hardware.

61.5 Applications

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical.

61.5.1 Real-life applications

The tasks artificial neural networks are applied to tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.

- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.

- Data processing, including filtering, clustering, blind source separation and compression.

- Robotics, including directing manipulators, prosthesis.

- Control, including Computer numerical control.

Application areas include the system identification and control (vehicle control, process control, natural resources management), quantum chemistry,[42] game-playing and decision making (backgammon, chess, poker), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications (e.g. automated trading systems), data mining (or knowledge discovery in databases, “KDD”), visualization and e-mail spam filtering.

Artificial neural networks have also been used to diagnose several cancers. An ANN based hybrid lung cancer detection system named HLND improves the accuracy of diagnosis and the speed of lung cancer radiology.[43] These networks have also been used to diagnose prostate cancer. The diagnoses can be used to make specific models taken from a large group of patients compared to information of one given patient. The models do not depend on assumptions about correlations of different variables. Colorectal cancer has also been predicted using the neural networks. Neural networks could predict the outcome for a patient with colorectal cancer with more accuracy than the current clinical methods. After training, the networks could predict multiple patient outcomes from unrelated institutions.[44]

61.5.2 Neural networks and neuroscience

Theoretical and computational neuroscience is the field concerned with the theoretical analysis and the computational modeling of biological neural systems. Since neural systems are intimately related to cognitive processes and behavior, the field is closely related to cognitive and behavioral modeling.

The aim of the field is to create models of biological neural systems in order to understand how biological systems work. To gain this understanding, neuroscientists strive to make a link between observed biological processes (data), biologically plausible mechanisms for neural processing and learning (biological neural network models) and theory (statistical learning theory and information theory).

Types of models

Many models are used in the field, defined at different levels of abstraction and modeling different aspects of neural systems. They range from models of the short-term behavior of individual neurons (e.g. [45]), models of how the dynamics of neural circuitry arise from interactions between individual neurons and finally to models of how behavior can arise from abstract neural modules that represent complete subsystems. These include models of the long-term, and short-term plasticity, of neural systems and their relations to learning and memory from the individual neuron to the system level.

Memory networks

Integrating external memory components with artificial neural networks has a long history dating back to early research in distributed representations [46] and self-organizing maps. E.g. in sparse distributed memory the patterns encoded by neural networks are used as memory addresses for content-addressable memory, with “neurons” essentially serving as address encoders and decoders.

More recently deep learning was shown to be useful in semantic hashing[47] where a deep graphical model the word-count vectors[48] obtained from a large set of documents. Documents are mapped to memory addresses in such a way that semantically similar documents are located at nearby addresses. Documents similar to a query document can then be found by simply accessing all the addresses that differ by only a few bits from the address of the query document.

Neural Turing Machines[49] developed by Google DeepMind extend the capabilities of deep neural networks by coupling them to external memory resources, which they can interact with by attentional processes. The combined system is analogous to a Turing Machine but is differentiable end-to-end, allowing it to be efficiently trained with gradient descent. Preliminary results demonstrate that Neural Turing Machines can infer simple algorithms such as copying, sorting, and associative recall from input and output examples.

Memory Networks[50] is another extension to neural networks incorporating long-term memory which was developed by Facebook research. The long-term memory can be read and written to, with the goal of using it for prediction. These models have been applied in the context of question answering (QA) where the long-term memory effectively acts as a (dynamic) knowledge base, and the output is a textual response.

61.6 Neural network software

Main article: Neural network software

Neural network software is used to simulate, research, develop and apply artificial neural networks, biological neural networks and, in some cases, a wider array of adaptive systems.

61.7 Types of artificial neural networks

Main article: Types of artificial neural networks

Artificial neural network types vary from those with only one or two layers of single direction logic, to complicated multi–input many directional feedback loops and layers. On the whole, these systems use algorithms in their programming to determine control and organization of their functions. Most systems use “weights” to change the parameters of the throughput and the varying connections to the neurons. Artificial neural networks can be autonomous and learn by input from outside “teachers” or even self-teaching from written-in rules.

61.8 Theoretical properties

61.8.1 Computational power

The multi-layer perceptron (MLP) is a universal function approximator, as proven by the universal approximation theorem. However, the proof is not constructive regarding the number of neurons required or the settings of the weights.

Work by Hava Siegelmann and Eduardo D. Sontag has provided a proof that a specific recurrent architecture with rational valued weights (as opposed to full precision real number-valued weights) has the full power of a Universal Turing Machine[51] using a finite number of neurons and standard linear connections. Further, it has been shown that the use of irrational values for weights results in a machine with super-Turing power.[52]

61.8.2 Capacity

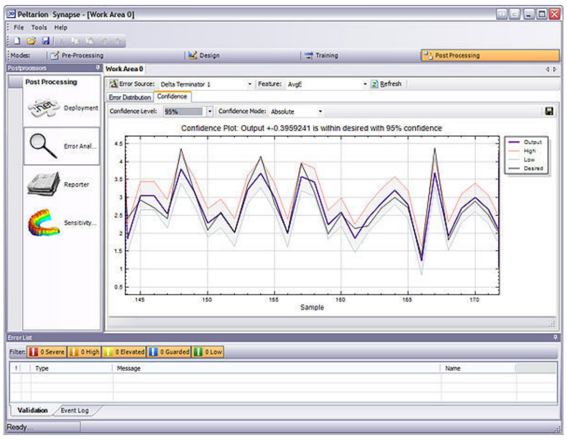
Artificial neural network models have a property called 'capacity', which roughly corresponds to their ability to model any given function. It is related to the amount of information that can be stored in the network and to the notion of complexity.

61.8.3 Convergence

Nothing can be said in general about convergence since it depends on a number of factors. Firstly, there may exist many local minima. This depends on the cost function and the model. Secondly, the optimization method used might not be guaranteed to converge when far away from a local minimum. Thirdly, for a very large amount of data or parameters, some methods become impractical. In general, it has been found that theoretical guarantees regarding convergence are an unreliable guide to practical application.

61.8.4 Generalization and statistics

In applications where the goal is to create a system that generalizes well in unseen examples, the problem of over-training has emerged. This arises in convoluted or over-specified systems when the capacity of the network significantly exceeds the needed free parameters. There are two schools of thought for avoiding this problem: The first is to use cross-validation and similar techniques to check for the presence of overtraining and optimally select hyperparameters such as to minimize the generalization error. The second is to use some form of *regularization*. This is a concept that emerges naturally in a probabilistic (Bayesian) framework, where the regularization can be performed by selecting a larger prior probability over simpler models; but also in statistical learning theory, where the goal is to minimize over two quantities: the 'empirical risk' and the 'structural risk', which roughly corresponds to the error over the training set and the predicted error in unseen data due to overfitting.


Peltarion Synapse - [Work Area 0]
File Tools Help
Modes: Pre-Processing Design Training Post Processing
Postprocessors
Work Area 0
Post Processing
Deployment
Error Anal...
Reporter
Sensitivity...
Error Source: Delta Terminator 1
Feature: AvgE
Refresh
Error Distribution Confidence
Confidence Level: 95%
Confidence Mode: Absolute
Confidence Plot: Output +-0.3959241 is within desired with 95% confidence
4.5
4
3.5
3
2.5
2
1.5
1
0.5
145
150
155
160
165
170
Sample
Output
High
Low
Desired
Error List
Filter: 0 Severe 0 High 0 Elevated 0 Guarded 0 Low
! Type Message Name
Validation Event Log
Ready...

Confidence analysis of a neural network

Supervised neural networks that use a mean squared error (MSE) cost function can use formal statistical methods to determine the confidence of the trained model. The MSE on a validation set can be used as an estimate for variance. This value can then be used to calculate the confidence interval of the output of the network, assuming a normal distribution. A confidence analysis made this way is statistically valid as long as the output probability distribution stays the same and the network is not modified.

By assigning a softmax activation function, a generalization of the logistic function, on the output layer of the neural network (or a softmax component in a component-based neural network) for categorical target variables, the outputs can be interpreted as posterior probabilities. This is very useful in classification as it gives a certainty measure on classifications.

The softmax activation function is:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}}$$

61.9 Controversies

61.9.1 Training issues

A common criticism of neural networks, particularly in robotics, is that they require a large diversity of training for real-world operation . This is not surprising, since any learning machine needs sufficient representative examples in order to capture the underlying structure that allows it to generalize to new cases. Dean Pomerleau, in his research presented in the paper "Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving," uses a neural network to train a robotic vehicle to drive on multiple types of roads (single lane, multi-lane, dirt, etc.). A large amount of his research is devoted to (1) extrapolating multiple training scenarios from a single training experience, and (2) preserving past training diversity so that the system does not become overtrained (if, for example, it is presented with a series of right turns – it should not learn to always turn right). These issues are common in neural networks that must decide from amongst a wide variety of responses, but can be dealt with in several ways, for example by randomly shuffling the training examples, by using a numerical optimization algorithm that does not take too large steps when changing the network connections following an example, or by grouping examples in so-called mini-batches.

A. K. Dewdney, a former *Scientific American* columnist, wrote in 1997, "Although neural nets do solve a few toy problems, their powers of computation are so limited that I am surprised anyone takes them seriously as a general problem-solving tool." (Dewdney, p. 82)

61.9.2 Hardware issues

To implement large and effective software neural networks, considerable processing and storage resources need to be committed . While the brain has hardware tailored to the task of processing signals through a graph of neurons, simulating even a most simplified form on Von Neumann technology may compel a neural network designer to fill many millions of database rows for its connections – which can consume vast amounts of computer memory and hard disk space. Furthermore, the designer of neural network systems will often need to simulate the transmission of signals through many of these connections and their associated neurons – which must often be matched with incredible amounts of CPU processing power and time. While neural networks often yield *effective* programs, they too often do so at the cost of *efficiency* (they tend to consume considerable amounts of time and money).

Computing power continues to grow roughly according to Moore's Law, which may provide sufficient resources to accomplish new tasks. Neuromorphic engineering addresses the hardware difficulty directly, by constructing non-Von-Neumann chips with circuits designed to implement neural nets from the ground up.

61.9.3 Practical counterexamples to criticisms

Arguments against Dewdney's position are that neural networks have been successfully used to solve many complex and diverse tasks, ranging from autonomously flying aircraft[53] to detecting credit card fraud .

Technology writer Roger Bridgman commented on Dewdney's statements about neural nets:

> Neural networks, for instance, are in the dock not only because they have been hyped to high heaven, (what hasn't?) but also because you could create a successful net without understanding how it worked: the bunch of numbers that captures its behaviour would in all probability be "an opaque, unreadable table...valueless as a scientific resource".
>
> In spite of his emphatic declaration that science is not technology, Dewdney seems here to pillory neural nets as bad science when most of those devising them are just trying to be good engineers. An unreadable table that a useful machine could read would still be well worth having.[54]

Although it is true that analyzing what has been learned by an artificial neural network is difficult, it is much easier to do so than to analyze what has been learned by a biological neural network. Furthermore, researchers involved in exploring learning algorithms for neural networks are gradually uncovering generic principles which allow a learning machine to be successful. For example, Bengio and LeCun (2007) wrote an article regarding local vs non-local learning, as well as shallow vs deep architecture.[55]

61.9.4 Hybrid approaches

Some other criticisms come from advocates of hybrid models (combining neural networks and symbolic approaches), who believe that the intermix of these two approaches can better capture the mechanisms of the human mind.[56][57]

61.10 Gallery

- A single-layer feedforward artificial neural network. Arrows originating from are omitted for clarity. There are p inputs to this network and q outputs. In this system, the value of the qth output, would be calculated as
- A two-layer feedforward artificial neural network.
-
-

61.11 See also

- 20Q
- ADALINE
- Adaptive resonance theory
- Artificial life
- Associative memory
- Autoencoder
- Backpropagation
- BEAM robotics
- Biological cybernetics
- Biologically inspired computing
- Blue brain
- Catastrophic interference
- Cerebellar Model Articulation Controller
- Cognitive architecture
- Cognitive science
- Convolutional neural network (CNN)
- Connectionist expert system
- Connectomics
- Cultured neuronal networks
- Deep learning
- Digital morphogenesis
- Encog
- Fuzzy logic
- Gene expression programming
- Genetic algorithm
- Group method of data handling
- Habituation
- In Situ Adaptive Tabulation
- Models of neural computation
- Multilinear subspace learning
- Neuroevolution
- Neural coding

- Neural gas
- Neural network software
- Neuroscience
- Ni1000 chip
- Nonlinear system identification
- Optical neural network
- Parallel Constraint Satisfaction Processes
- Parallel distributed processing
- Radial basis function network
- Recurrent neural networks
- Self-organizing map
- Spiking neural network
- Systolic array
- Tensor product network
- Time delay neural network (TDNN)

61.12 References

[1] McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics* **5** (4): 115–133. doi:10.1007/BF02478259.

[2] Hebb, Donald (1949). *The Organization of Behavior*. New York: Wiley.

[3] Farley, B.G.; W.A. Clark (1954). "Simulation of Self-Organizing Systems by Digital Computer". *IRE Transactions on Information Theory* **4** (4): 76–84. doi:10.1109/TIT.1954.1057468.

[4] Rochester, N.; J.H. Holland; L.H. Habit; W.L. Duda (1956). "Tests on a cell assembly theory of the action of the brain, using a large digital computer". *IRE Transactions on Information Theory* **2** (3): 80–93. doi:10.1109/TIT.1956.1056810.

[5] Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". *Psychological Review* **65** (6): 386–408. doi:10.1037/h0042519. PMID 13602029.

[6] Werbos, P.J. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*.

[7] Minsky, M.; S. Papert (1969). *An Introduction to Computational Geometry*. MIT Press. ISBN 0-262-63022-2.

[8] Rumelhart, D.E; James McClelland (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge: MIT Press.

[9] Russell, Ingrid. "Neural Networks Module". Retrieved 2012.

[10] Yang, J. J.; Pickett, M. D.; Li, X. M.; Ohlberg, D. A. A.; Stewart, D. R.; Williams, R. S. Nat. Nanotechnol. 2008, 3, 429–433.

[11] Strukov, D. B.; Snider, G. S.; Stewart, D. R.; Williams, R. S. *Nature* 2008, 453, 80–83.

[12] 2012 Kurzweil AI Interview with Jürgen Schmidhuber on the eight competitions won by his Deep Learning team 2009–2012

[13] http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions 2012 Kurzweil AI Interview with Jürgen Schmidhuber on the eight competitions won by his Deep Learning team 2009–2012

[14] Graves, Alex; and Schmidhuber, Jürgen; *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks*, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), *Advances in Neural Information Processing Systems 22 (NIPS'22), 7–10 December 2009, Vancouver, BC*, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552.

[15] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.

[16] Graves, Alex; and Schmidhuber, Jürgen; *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks*, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), *Advances in Neural Information Processing Systems 22 (NIPS'22), December 7th–10th, 2009, Vancouver, BC*, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552

[17] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.

[18] D. C. Ciresan, U. Meier, J. Masci, J. Schmidhuber. Multi-Column Deep Neural Network for Traffic Sign Classification. Neural Networks, 2012.

[19] D. C. Ciresan, U. Meier, J. Masci, J. Schmidhuber. Multi-Column Deep Neural Network for Traffic Sign Classification. Neural Networks, 2012.

[20] D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In Advances in Neural Information Processing Systems (NIPS 2012), Lake Tahoe, 2012.

[21] D. C. Ciresan, U. Meier, J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. IEEE Conf. on Computer Vision and Pattern Recognition CVPR 2012.

[22] Fukushima, K. (1980). “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. *Biological Cybernetics* **36** (4): 93–202. doi:10.1007/BF00344251. PMID 7370364.

[23] M Riesenhuber, T Poggio. Hierarchical models of object recognition in cortex. Nature neuroscience, 1999.

[24] Deep belief networks at Scholarpedia.

[25] Hinton, G. E.; Osindero, S.; Teh, Y. W. (2006). “A Fast Learning Algorithm for Deep Belief Nets” (PDF). *Neural Computation* **18** (7): 1527–1554. doi:10.1162/neco.2006.18.7.1527. PMID 16764513.

[26] http://www.scholarpedia.org/article/Deep_belief_networks /

[27] Hinton, G. E.; Osindero, S.; Teh, Y. (2006). “A fast learning algorithm for deep belief nets” (PDF). *Neural Computation* **18** (7): 1527–1554. doi:10.1162/neco.2006.18.7.1527. PMID 16764513.

[28] John Markoff (November 23, 2012). “Scientists See Promise in Deep-Learning Programs”. *New York Times*.

[29] “The Machine Learning Dictionary”.

[30] Dominic, S., Das, R., Whitley, D., Anderson, C. (July 1991). “Genetic reinforcement learning for neural networks”. *IJCNN-91-Seattle International Joint Conference on Neural Networks*. IJCNN-91-Seattle International Joint Conference on Neural Networks. Seattle, Washington, USA: IEEE. doi:10.1109/IJCNN.1991.155315. ISBN 0-7803-0164-1. Retrieved 29 July 2012.

[31] Hoskins, J.C.; Himmelblau, D.M. (1992). “Process control via artificial neural networks and reinforcement learning”. *Computers & Chemical Engineering* **16** (4): 241–251. doi:10.1016/0098-1354(92)80045-B.

[32] Bertsekas, D.P., Tsitsiklis, J.N. (1996). *Neuro-dynamic programming*. Athena Scientific. p. 512. ISBN 1-886529-10-8.

[33] Secomandi, Nicola (2000). “Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands”. *Computers & Operations Research* **27** (11–12): 1201–1225. doi:10.1016/S0305-0548(99)00146-X.

[34] de Rigo, D., Rizzoli, A. E., Soncini-Sessa, R., Weber, E., Zenesi, P. (2001). “Neuro-dynamic programming for the efficient management of reservoir networks” (PDF). *Proceedings of MODSIM 2001, International Congress on Modelling and Simulation*. MODSIM 2001, International Congress on Modelling and Simulation. Canberra, Australia: Modelling and Simulation Society of Australia and New Zealand. doi:10.5281/zenodo.7481. ISBN 0-867405252. Retrieved 29 July 2012.

[35] Damas, M., Salmeron, M., Diaz, A., Ortega, J., Prieto, A., Olivares, G. (2000). “Genetic algorithms and neuro-dynamic programming: application to water supply networks”. *Proceedings of 2000 Congress on Evolutionary Computation*. 2000 Congress on Evolutionary Computation. La Jolla, California, USA: IEEE. doi:10.1109/CEC.2000.870269. ISBN 0-7803-6375-2. Retrieved 29 July 2012.

[36] Deng, Geng; Ferris, M.C. (2008). “Neuro-dynamic programming for fractionated radiotherapy planning”. *Springer Optimization and Its Applications* **12**: 47–70. doi:10.1007/978-0-387-73299-2_3.

[37] M. Forouzanfar, H. R. Dajani, V. Z. Groza, M. Bolic, and S. Rajan, (July 2010). *Comparison of Feed-Forward Neural Network Training Algorithms for Oscillometric Blood Pressure Estimation* (PDF). 4th Int. Workshop Soft Computing Applications. Arad, Romania: IEEE.

[38] de Rigo, D., Castelletti, A., Rizzoli, A.E., Soncini-Sessa, R., Weber, E. (January 2005). “A selective improvement technique for fastening Neuro-Dynamic Programming in Water Resources Network Management”. In Pavel Zítek. *Proceedings of the 16th IFAC World Congress – IFAC-PapersOnLine*. 16th IFAC World Congress **16**. Prague, Czech Republic: IFAC. doi:10.3182/20050703-6-CZ-1902.02172. ISBN 978-3-902661-75-3. Retrieved 30 December 2011.

[39] Ferreira, C. (2006). “Designing Neural Networks Using Gene Expression Programming” (PDF). In A. Abraham, B. de Baets, M. Köppen, and B. Nickolay, eds., Applied Soft Computing Technologies: The Challenge of Complexity, pages 517–536, Springer-Verlag.

[40] Da, Y., Xiurun, G. (July 2005). T. Villmann, ed. *An improved PSO-based ANN with simulated annealing technique*. New Aspects in Neurocomputing: 11th European Symposium on Artificial Neural Networks. Elsevier. doi:10.1016/j.neucom.2004.07.002.

[41] Wu, J., Chen, E. (May 2009). Wang, H., Shen, Y., Huang, T., Zeng, Z., ed. *A Novel Nonparametric Regression Ensemble for Rainfall Forecasting Using Particle Swarm Optimization Technique Coupled with Artificial Neural Network*. 6th International Symposium on Neural Networks, ISNN 2009. Springer. doi:10.1007/978-3-642-01513-7_6. ISBN 978-3-642-01215-0.

[42] Roman M. Balabin, Ekaterina I. Lomakina (2009). “Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies”. *J. Chem. Phys.* **131** (7): 074104. doi:10.1063/1.3206326. PMID 19708729.

[43] Ganesan, N. “Application of Neural Networks in Diagnosing Cancer Disease Using Demographic Data” (PDF). International Journal of Computer Applications.

[44] Bottaci, Leonardo. “Artificial Neural Networks Applied to Outcome Prediction for Colorectal Cancer Patients in Separate Institutions” (PDF). The Lancet.

[45] Forrest MD (April 2015). “Simulation of alcohol action upon a detailed Purkinje neuron model and a simpler surrogate model that runs >400 times faster”. *BMC Neuroscience* **16** (27). doi:10.1186/s12868-015-0162-6.

[46] Hinton, Geoffrey E. “Distributed representations.” (1984)

[47] Salakhutdinov, Ruslan, and Geoffrey Hinton. “Semantic hashing.” International Journal of Approximate Reasoning 50.7 (2009): 969-978.

[48] Le, Quoc V., and Tomas Mikolov. "Distributed representations of sentences and documents." arXiv preprint arXiv:1405.4053 (2014).

[49] Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines." arXiv preprint arXiv:1410.5401 (2014).

[50] Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).

[51] Siegelmann, H.T.; Sontag, E.D. (1991). "Turing computability with neural nets" (PDF). *Appl. Math. Lett.* **4** (6): 77–80. doi:10.1016/0893-9659(91)90080-F.

[52] Balcázar, José (Jul 1997). "Computational Power of Neural Networks: A Kolmogorov Complexity Characterization". *Information Theory, IEEE Transactions on* **43** (4): 1175–1183. doi:10.1109/18.605580. Retrieved 3 November 2014.

[53] NASA - Dryden Flight Research Center - News Room: News Releases: NASA NEURAL NETWORK PROJECT PASSES MILESTONE. Nasa.gov. Retrieved on 2013-11-20.

[54] Roger Bridgman's defence of neural networks

[55] http://www.iro.umontreal.ca/~{ }lisa/publications2/index.php/publications/show/4

[56] Sun and Bookman (1990)

[57] Tahmasebi; Hezarkhani (2012). "A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation". *Computers & Geosciences* **42**: 18–27. doi:10.1016/j.cageo.2012.02.004.

61.13 Bibliography

- Bhadeshia H. K. D. H. (1999). "Neural Networks in Materials Science" (PDF). *ISIJ International* **39** (10): 966–979. doi:10.2355/isijinternational.39.966.
- Bishop, C.M. (1995) *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press. ISBN 0-19-853849-9 (hardback) or ISBN 0-19-853864-2 (paperback)
- Cybenko, G.V. (1989). Approximation by Superpositions of a Sigmoidal function, *Mathematics of Control, Signals, and Systems*, Vol. 2 pp. 303–314. electronic version
- Duda, R.O., Hart, P.E., Stork, D.G. (2001) *Pattern classification (2nd edition)*, Wiley, ISBN 0-471-05669-3
- Egmont-Petersen, M., de Ridder, D., Handels, H. (2002). "Image processing with neural networks – a review". *Pattern Recognition* **35** (10): 2279–2301. doi:10.1016/S0031-3203(01)00178-9.
- Gurney, K. (1997) *An Introduction to Neural Networks* London: Routledge. ISBN 1-85728-673-1 (hardback) or ISBN 1-85728-503-4 (paperback)
- Haykin, S. (1999) *Neural Networks: A Comprehensive Foundation*, Prentice Hall, ISBN 0-13-273350-1
- Fahlman, S, Lebiere, C (1991). *The Cascade-Correlation Learning Architecture*, created for National Science Foundation, Contract Number EET-8716324, and Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976 under Contract F33615-87-C-1499. electronic version
- Hertz, J., Palmer, R.G., Krogh. A.S. (1990) *Introduction to the theory of neural computation*, Perseus Books. ISBN 0-201-51560-1
- Lawrence, Jeanette (1994) *Introduction to Neural Networks*, California Scientific Software Press. ISBN 1-883157-00-5
- Masters, Timothy (1994) *Signal and Image Processing with Neural Networks*, John Wiley & Sons, Inc. ISBN 0-471-04963-8
- Ripley, Brian D. (1996) *Pattern Recognition and Neural Networks*, Cambridge
- Siegelmann, H.T. and Sontag, E.D. (1994). Analog computation via neural networks, *Theoretical Computer Science*, v. 131, no. 2, pp. 331–360. electronic version
- Sergios Theodoridis, Konstantinos Koutroumbas (2009) "Pattern Recognition", 4th Edition, Academic Press, ISBN 978-1-59749-272-0.
- Smith, Murray (1993) *Neural Networks for Statistical Modeling*, Van Nostrand Reinhold, ISBN 0-442-01310-8
- Wasserman, Philip (1993) *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, ISBN 0-442-00461-3
- *Computational Intelligence: A Methodological Introduction* by Kruse, Borgelt, Klawonn, Moewes, Steinbrecher, Held, 2013, Springer, ISBN 9781447150121
- *Neuro-Fuzzy-Systeme* (3rd edition) by Borgelt, Klawonn, Kruse, Nauck, 2003, Vieweg, ISBN 9783528252656

61.14 External links

- Neural Networks at DMOZ

- A brief introduction to Neural Networks (PDF), illustrated 250p textbook covering the common kinds of neural networks (CC license).

Chapter 62

Deep learning

For deep versus shallow learning in educational psychology, see Student approaches to learning

Deep learning (*deep machine learning*, or *deep structured learning*, or *hierarchical learning*, or sometimes *DL*) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using model architectures, with complex structures or otherwise, composed of multiple non-linear transformations.[1](p198)[2][3][4]

Deep learning is part of a broader family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc.. Some representations make it easier to learn tasks (e.g., face recognition or facial expression recognition[5]) from examples. One of the promises of deep learning is replacing handcrafted features with efficient algorithms for unsupervised or semi-supervised feature learning and hierarchical feature extraction.[6]

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data. Some of the representations are inspired by advances in neuroscience and are loosely based on interpretation of information processing and communication patterns in a nervous system, such as neural coding which attempts to define a relationship between the stimulus and the neuronal responses and the relationship among the electrical activity of the neurons in the brain.[7]

Various deep learning architectures such as deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks have been applied to fields like computer vision, automatic speech recognition, natural language processing, audio recognition and bioinformatics where they have been shown to produce state-of-the-art results on various tasks.

Alternatively, *deep learning* has been characterized as a buzzword, or a rebranding of neural networks.[8][9]

62.1 Introduction

62.1.1 Definitions

There are a number of ways that the field of deep learning has been characterized. Deep learning is a class of machine learning algorithms that[1](pp199–200)

- use a cascade of many layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. The algorithms may be supervised or unsupervised and applications include pattern analysis (unsupervised) and classification (supervised).
- are based on the (unsupervised) learning of multiple levels of features or representations of the data. Higher level features are derived from lower level features to form a hierarchical representation.
- are part of the broader machine learning field of learning representations of data.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

These definitions have in common (1) multiple layers of nonlinear processing units and (2) the supervised or unsupervised learning of feature representations in each layer, with the layers forming a hierarchy from low-level to high-level features.[1](p200) The composition of a layer of nonlinear processing units used in a deep learning algorithm depends on the problem to be solved. Layers that have been used in deep learning include hidden layers of an artificial neural network and sets of complicated propositional formulas.[2] They may also include latent variables organized layer-wise in deep generative models such as the nodes in Deep Belief Networks and Deep Boltzmann Machines.

Deep learning algorithms are contrasted with shallow learning algorithms by the number of parameterized transformations a signal encounters as it propagates from the input layer to the output layer, where a parameterized transformation is a processing unit that has trainable parameters, such as weights and thresholds.[4](p6) A chain of transformations from input to output is a *credit assignment path* (CAP). CAPs describe potentially causal connections between input and output and may vary in length. For a feedforward neural network, the depth of the CAPs, and thus the depth of the network, is the number of hidden layers plus one (the output layer is also parameterized). For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP is potentially unlimited in length. There is no universally agreed upon threshold of depth dividing shallow learning from deep learning, but most researchers in the field agree that deep learning has multiple nonlinear layers (CAP > 2) and Schmidhuber considers CAP > 10 to be very deep learning.[4](p7)

62.1.2 Fundamental concepts

Deep learning algorithms are based on distributed representations. The underlying assumption behind distributed representations is that observed data is generated by the interactions of many different factors on different levels. Deep learning adds the assumption that these factors are organized into multiple levels, corresponding to different levels of abstraction or composition. Varying numbers of layers and layer sizes can be used to provide different amounts of abstraction.[3]

Deep learning algorithms in particular exploit this idea of hierarchical explanatory factors. Different concepts are learned from other concepts, with the more abstract, higher level concepts being learned from the lower level ones. These architectures are often constructed with a greedy layer-by-layer method that models this idea. Deep learning helps to disentangle these abstractions and pick out which features are useful for learning.[3]

For supervised learning tasks where label information is readily available in training, deep learning promotes a principle which is very different than traditional methods of machine learning. That is, rather than focusing on feature engineering which is often labor-intensive and varies from one task to another, deep learning methods are focused on end-to-end learning based on raw features. In other words, deep learning moves away from feature engineering to a maximal extent possible. To accomplish end-to-end optimization starting with raw features and ending in labels, layered structures are often necessary. From this perspective, we can regard the use of layered structures to derive intermediate representations in deep learning as a natural consequence of raw-feature-based end-to-end learning.[1] Understanding the connection between the above two aspects of deep learning is important to appreciate its use in several application areas, all involving supervised learning tasks (e.g., supervised speech and image recognition), as to be discussed in a later part of this article.

Many deep learning algorithms are framed as unsupervised learning problems. Because of this, these algorithms can make use of the unlabeled data that supervised algorithms cannot. Unlabeled data is usually more abundant than labeled data, making this an important benefit of these algorithms. The deep belief network is an example of a deep structure that can be trained in an unsupervised manner.[3]

62.2 History

Deep learning architectures, specifically those built from artificial neural networks (ANN), date back at least to the Neocognitron introduced by Kunihiko Fukushima in 1980.[10] The ANNs themselves date back even further. In 1989, Yann LeCun et al. were able to apply the standard backpropagation algorithm, which had been around since 1974,[11] to a deep neural network with the purpose of recognizing handwritten ZIP codes on mail. Despite the success of applying the algorithm, the time to train the network on this dataset was approximately 3 days, making it impractical for general use.[12] Many factors contribute to the slow speed, one being due to the so-called vanishing gradient problem analyzed in 1991 by Sepp Hochreiter.[13][14]

While such neural networks by 1991 were used for recognizing isolated 2-D hand-written digits, 3-D object recognition by 1991 used a 3-D model-based approach – matching 2-D images with a handcrafted 3-D object model. Juyang Weng *et al.*. proposed that a human brain does not use a monolithic 3-D object model and in 1992 they published Cresceptron,[15][16][17] a method for performing 3-D object recognition directly from cluttered scenes. Cresceptron is a cascade of many layers similar to Neocognitron. But unlike Neocognitron which required the human programmer to hand-merge features, Cresceptron fully *automatically* learned an open number of unsupervised features in each layer of the cascade where each feature is represented by a convolution kernel. In addition, Cresceptron also segmented each learned object from a cluttered scene through back-analysis through the network. Max-pooling, now often adopted by deep neural networks (e.g., ImageNet tests), was first used in Cresceptron to reduce the position resolution by a factor of (2x2) to 1 through the cascade for better generalization. Because of a great lack of understanding how the brain autonomously wire its biological networks and the computational cost by ANNs then, simpler models that use task-specific handcrafted features such as Gabor filter and support vector machines (SVMs) were of popular choice of the field in the 1990s and 2000s.

In the long history of speech recognition, both shallow form and deep form (e.g., recurrent nets) of artificial neural networks had been explored for many years.[18][19][20] But these methods never won over the non-uniform internal-handcrafting Gaussian mixture model/Hidden

Markov model (GMM-HMM) technology based on generative models of speech trained discriminatively.[21] A number of key difficulties had been methodologically analyzed, including gradient diminishing and weak temporal correlation structure in the neural predictive models.[22][23] All these difficulties were in addition to the lack of big training data and big computing power in these early days. Most speech recognition researchers who understood such barriers hence subsequently moved away from neural nets to pursue generative modeling approaches until the recent resurgence of deep learning that has overcome all these difficulties. Hinton et al. and Deng et al. reviewed part of this recent history about how their collaboration with each other and then with cross-group colleagues ignited the renaissance of neural networks and initiated deep learning research and applications in speech recognition.[24][25][26][27]

The term "deep learning" gained traction in the mid-2000s after a publication by Geoffrey Hinton and Ruslan Salakhutdinov showed how a many-layered feedforward neural network could be effectively pre-trained one layer at a time, treating each layer in turn as an unsupervised restricted Boltzmann machine, then using supervised backpropagation for fine-tuning.[28] In 1992, Schmidhuber had already implemented a very similar idea for the more general case of unsupervised deep hierarchies of recurrent neural networks, and also experimentally shown its benefits for speeding up supervised learning [29][30]

Since the resurgence of deep learning, it has become part of many state-of-the-art systems in different disciplines, particularly that of computer vision and automatic speech recognition (ASR). Results on commonly used evaluation sets such as TIMIT (ASR) and MNIST (image classification) as well as a range of large vocabulary speech recognition tasks are constantly being improved with new applications of deep learning.[24][31][32] Currently, it has been shown that deep learning architectures in the form of convolutional neural networks have been nearly best performing;[33][34] however, these are more widely used in computer vision than in ASR.

The real impact of deep learning in industry started in large-scale speech recognition around 2010. In late 2009, Geoff Hinton was invited by Li Deng to work with him and colleagues at Microsoft Research in Redmond to apply deep learning to speech recognition. They co-organized the 2009 NIPS Workshop on Deep Learning for Speech Recognition. The workshop was motivated by the limitations of deep generative models of speech, and the possibility that the big-compute, big-data era warranted a serious try of the deep neural net (DNN) approach. It was then (incorrectly) believed that pre-training of DNNs using generative models of deep belief net (DBN) would be the cure for the main difficulties of neural nets encountered during 1990's.[26] However, soon after the research along this direction started at Microsoft Research, it was discovered that when large amounts of training data are used and especially when DNNs are designed correspondingly with large, context-dependent output layers, dramatic error reduction occurred over the then-state-of-the-art GMM-HMM and more advanced generative model-based speech recognition systems without the need for generative DBN pre-training, the finding verified subsequently by several other major speech recognition research groups [24][35] Further, the nature of recognition errors produced by the two types of systems was found to be characteristically different,[25][36] offering technical insights into how to artfully integrate deep learning into the existing highly efficient, run-time speech decoding system deployed by all major players in speech recognition industry. The history of this significant development in deep learning has been described and analyzed in recent books.[1][37]

Advances in hardware have also been an important enabling factor for the renewed interest of deep learning. In particular, powerful graphics processing units (GPUs) are highly suited for the kind of number crunching, matrix/vector math involved in machine learning. GPUs have been shown to speed up training algorithms by orders of magnitude, bringing running times of weeks back to days.[38][39]

62.3 Deep learning in artificial neural networks

Some of the most successful deep learning methods involve artificial neural networks. Artificial neural networks are inspired by the 1959 biological model proposed by Nobel laureates David H. Hubel & Torsten Wiesel, who found two types of cells in the primary visual cortex: simple cells and complex cells. Many artificial neural networks can be viewed as cascading models [15][16][17][40] of cell types inspired by these biological observations.

Fukushima's Neocognitron introduced convolutional neural networks partially trained by unsupervised learning while humans directed features in the neural plane. Yann LeCun et al. (1989) applied supervised backpropagation to such architectures.[41] Weng et al. (1992) published convolutional neural networks Cresceptron[15][16][17] for 3-D object recognition from images of cluttered scenes and segmentation of such objects from images.

An obvious need for recognizing general 3-D objects is least shift invariance and tolerance to deformation. Max-pooling appeared to be first proposed by Cresceptron[15][16] to enable the network to tolerate small-to-large deformation in a hierarchical way while using convolution. Max-pooling helps, but still does not fully guarantee, shift-invariance at the pixel level.[17]

With the advent of the back-propagation algorithm in the 1970s, many researchers tried to train supervised deep artificial neural networks from scratch, initially with little success. Sepp Hochreiter's diploma thesis of 1991[42][43] formally identified the reason for this failure in the "vanishing gradient problem," which not only affects many-layered feedforward networks, but also recurrent neural networks. The latter are trained by unfolding them into very deep feedforward networks, where a new layer is created for each time step of an input sequence processed by the network. As errors propagate from layer to layer, they shrink exponentially with the number of layers.

To overcome this problem, several methods were proposed. One is Jürgen Schmidhuber's multi-level hierarchy of networks (1992) pre-trained one level at a time through unsupervised learning, fine-tuned through backpropagation.[29] Here each level learns a compressed representation of the observations that is fed to the next level.

Another method is the long short term memory (LSTM) network of 1997 by Hochreiter & Schmidhuber.[44] In 2009, deep multidimensional LSTM networks won three ICDAR 2009 competitions in connected handwriting recognition, without any prior knowledge about the three different languages to be learned.[45][46]

Sven Behnke relied only on the sign of the gradient (Rprop) when training his Neural Abstraction Pyramid[47] to solve problems like image reconstruction and face localization.

Other methods also use unsupervised pre-training to structure a neural network, making it first learn generally useful feature detectors. Then the network is trained further by supervised back-propagation to classify labeled data. The deep model of Hinton et al. (2006) involves learning the distribution of a high level representation using successive layers of binary or real-valued latent variables. It uses a restricted Boltzmann machine (Smolensky, 1986[48]) to model each new layer of higher level features. Each new layer guarantees an increase on the lower-bound of the log likelihood of the data, thus improving the model, if trained properly. Once sufficiently many layers have been learned the deep architecture may be used as a generative model by reproducing the data when sampling down the model (an "ancestral pass") from the top level feature activations.[49] Hinton reports that his models are effective feature extractors over high-dimensional, structured data.[50]

The Google Brain team led by Andrew Ng and Jeff Dean created a neural network that learned to recognize higher-level concepts, such as cats, only from watching unlabeled images taken from YouTube videos.[51] [52]

Other methods rely on the sheer processing power of modern computers, in particular, GPUs. In 2010 it was shown by Dan Ciresan and colleagues[38] in Jürgen Schmidhuber's group at the Swiss AI Lab IDSIA that despite the above-mentioned "vanishing gradient problem," the superior processing power of GPUs makes plain back-propagation feasible for deep feedforward neural networks with many layers. The method outperformed all other machine learning techniques on the old, famous MNIST handwritten digits problem of Yann LeCun and colleagues at NYU.

At about the same time, in late 2009, deep learning made inroad into speech recognition, as marked by the NIPS Workshop on Deep Learning for Speech Recognition. Intensive collaborative work between Microsoft Research and University of Toronto researchers had demonstrated by mid 2010 in Redmond that deep neural networks interfaced with a hidden Markov model with context-dependent states that define the neural network output layer can drastically reduce errors in large vocabulary speech recognition tasks such as voice search. The same deep neural net model was shown to scale up to Switchboard tasks about one year later at Microsoft Research Asia.

As of 2011, the state of the art in deep learning feedforward networks alternates convolutional layers and max-pooling layers,[53][54] topped by several pure classification layers. Training is usually done without any unsupervised pre-training. Since 2011, GPU-based implementations[53] of this approach won many pattern recognition contests, including the IJCNN 2011 Traffic Sign Recognition Competition,[55] the ISBI 2012 Segmentation of neuronal structures in EM stacks challenge,[56] and others.

Such supervised deep learning methods also were the first artificial pattern recognizers to achieve human-competitive performance on certain tasks.[57]

To break the barriers of weak AI represented by deep learning, it is necessary to go beyond the deep learning architectures because biological brains use both shallow and deep circuits as reported by brain anatomy[58] in order to deal with the wide variety of invariance that the brain displays. Weng[59] argued that the brain self-wires largely according to signal statistics and, therefore, a serial cascade cannot catch all major statistical dependencies. Fully guaranteed shift invariance for ANNs to deal with small and large natural objects in large cluttered scenes became true when the invariance went beyond shift, to extend to all ANN-learned concepts, such as location, type (object class label), scale, lighting, in the Developmental Networks (DNs)[60] whose embodiments are Where-What Networks, WWN-1 (2008)[61] through WWN-7 (2013).[62]

62.4 Deep learning architectures

There are huge number of different variants of deep architectures; however, most of them are branched from some original parent architectures. It is not always possible to compare the performance of multiple architectures all together, since they are not all implemented on the same data set. Deep learning is a fast-growing field so new architectures, variants, or algorithms may appear every few weeks.

62.4.1 Deep neural networks

A deep neural network (DNN) is an artificial neural network with multiple hidden layers of units between the input and output layers.[2][4] Similar to shallow ANNs, DNNs can model complex non-linear relationships. DNN architectures, e.g., for object detection and parsing generate compositional models where the object is expressed as layered composition of image primitives.[63] The extra layers enable composition of features from lower layers, giving the potential of modeling complex data with fewer units than a similarly performing shallow network.[2]

DNNs are typically designed as feedforward networks, but recent research has successfully applied the deep learning architecture to recurrent neural networks for applications such as language modeling.[64] Convolutional deep neural networks (CNNs) are used in computer vision where their success is well-documented.[65] More recently, CNNs have been applied to acoustic modeling for automatic speech recognition (ASR), where they have shown success over previous models.[34] For simplicity, a look at training DNNs is given here.

A DNN can be discriminatively trained with the standard backpropagation algorithm. The weight updates can be done via stochastic gradient descent using the following equation:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\frac{\partial C}{\partial w_{ij}}$$

Here, η is the learning rate, and C is the cost function. The choice of the cost function depends on factors such as the learning type (supervised, unsupervised, reinforcement, etc.) and the activation function. For example, when performing supervised learning on a multiclass classification problem, common choices for the activation function and cost function are the softmax function and cross entropy function, respectively. The softmax function is defined as $p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}$ where p_j represents the class probability and x_j and x_k represent the total input to units j and k respectively. Cross entropy is defined as $C = -\sum_j d_j \log(p_j)$ where d_j represents the target probability for output unit j and p_j is the probability output for j after applying the activation function.[66]

62.4.2 Issues with deep neural networks

As with ANNs, many issues can arise with DNNs if they are naively trained. Two common issues are overfitting and computation time.

DNNs are prone to overfitting because of the added layers of abstraction, which allow them to model rare dependencies in the training data. Regularization methods such as weight decay (ℓ_2 -regularization) or sparsity (ℓ_1 -regularization) can be applied during training to help combat overfitting.[67] A more recent regularization method applied to DNNs is *dropout* regularization. In dropout, some number of units are randomly omitted from the hidden layers during training. This helps to break the rare dependencies that can occur in the training data [68]

Backpropagation and gradient descent have been the preferred method for training these structures due to the ease of implementation and their tendency to converge to better local optima in comparison with other training methods. However, these methods can be computationally expensive, especially when being used to train DNNs. There are many training parameters to be considered with a DNN, such as the size (number of layers and number of units per layer), the learning rate and initial weights. Sweeping through the parameter space for optimal parameters may not be feasible due to the cost in time and computational resources. Various 'tricks' such as using mini-batching (computing the gradient on several training examples at once rather than individual examples) [69] have been shown to speed up computation. The large processing throughput of GPUs has produced significant speedups in training, due to the matrix and vector computations required being well suited for GPUs.[4] Radical alternatives to backprop such as Extreme Learning Machines,[70] “No-prop” networks [71] and Weightless neural networks [72] are gaining attention.

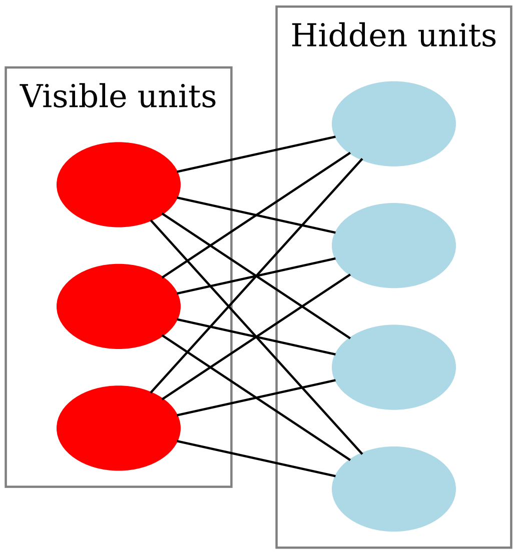
62.4.3 Deep belief networks

Main article: Deep belief network

A deep belief network (DBN) is a probabilistic, generative model made up of multiple layers of hidden units. It can be looked at as a composition of simple learning modules that make up each layer.[73]

A DBN can be used for generatively pre-training a DNN by using the learned weights as the initial weights. Backpropagation or other discriminative algorithms can then be applied for fine-tuning of these weights. This is particularly helpful in situations where limited training data is available, as poorly initialized weights can have significant impact on the performance of the final model. These pre-trained weights are in a region of the weight space that is closer to the optimal weights (as compared to just random initialization). This allows for both improved modeling capability and faster convergence of the fine-tuning phase.[74]

A DBN can be efficiently trained in an unsupervised, layer-by-layer manner where the layers are typically made of restricted Boltzmann machines (RBM). A description of training a DBN via RBMs is provided below. An RBM



A restricted Boltzmann machine (RBM) with fully connected visible and hidden units. Note there are no hidden-hidden or visible-visible connections.

is an undirected, generative energy-based model with an input layer and single hidden layer. Connections only exist between the visible units of the input layer and the hidden units of the hidden layer; there are no visible-visible or hidden-hidden connections.

The training method for RBMs was initially proposed by Geoffrey Hinton for use with training "Product of Expert" models and is known as contrastive divergence (CD).[75] CD provides an approximation to the maximum likelihood method that would ideally be applied for learning the weights of the RBM.[69][76]

In training a single RBM, weight updates are performed with gradient ascent via the following equation: $\Delta w_{ij}(t+1) = w_{ij}(t) + \eta\frac{\partial \log(p(v))}{\partial w_{ij}}$. Here, $p(v)$ is the probability of a visible vector, which is given by $p(v) = \frac{1}{Z}\sum_h e^{-E(v,h)}$. Z is the partition function (used for normalizing) and $E(v,h)$ is the energy function assigned to the state of the network. A lower energy indicates the network is in a more "desirable" configuration. The gradient $\frac{\partial \log(p(v))}{\partial w_{ij}}$ has the simple form $\langle v_i h_j\rangle_\text{data} - \langle v_i h_j\rangle_\text{model}$ where $\langle\cdots\rangle_p$ represent averages with respect to distribution p . The issue arises in sampling $\langle v_i h_j\rangle_\text{model}$ as this requires running alternating Gibbs sampling for a long time. CD replaces this step by running alternating Gibbs sampling for n steps (values of $n = 1$ have empirically been shown to perform well). After n steps, the data is sampled and that sample is used in place of $\langle v_i h_j\rangle_\text{model}$. The CD procedure works as follows:[69]

1. Initialize the visible units to a training vector.
2. Update the hidden units in parallel given the visible units: $p(h_j = 1 \mid \mathbf{V}) = \sigma(b_j + \sum_i v_i w_{ij})$. σ represents the sigmoid function and b_j is the bias of h_j .
3. Update the visible units in parallel given the hidden units: $p(v_i = 1 \mid \mathbf{H}) = \sigma(a_i + \sum_j h_j w_{ij})$. a_i is the bias of v_i . This is called the "reconstruction" step.
4. Reupdate the hidden units in parallel given the reconstructed visible units using the same equation as in step 2.
5. Perform the weight update: $\Delta w_{ij} \propto \langle v_i h_j\rangle_\text{data} - \langle v_i h_j\rangle_\text{reconstruction}$.

Once an RBM is trained, another RBM can be "stacked" atop of it to create a multilayer model. Each time another RBM is stacked, the input visible layer is initialized to a training vector and values for the units in the already-trained RBM layers are assigned using the current weights and biases. The final layer of the already-trained layers is used as input to the new RBM. The new RBM is then trained with the procedure above, and then this whole process can be repeated until some desired stopping criterion is met.[2]

Despite the approximation of CD to maximum likelihood being very crude (CD has been shown to not follow the gradient of any function), empirical results have shown it to be an effective method for use with training deep architectures.[69]

62.4.4 Convolutional neural networks

Main article: Convolutional neural network

A CNN is composed of one or more convolutional layers with fully connected layers (matching those in typical artificial neural networks) on top. It also uses tied weights and pooling layers. This architecture allows CNNs to take advantage of the 2D structure of input data. In comparison with other deep architectures, convolutional neural networks are starting to show superior results in both image and speech applications. They can also be trained with standard backpropagation. CNNs are easier to train than other regular, deep, feed-forward neural networks and have many fewer parameters to estimate, making them a highly attractive architecture to use.[77]

62.4.5 Convolutional Deep Belief Networks

A recent achievement in deep learning is from the use of convolutional deep belief networks (CDBN). A CDBN is very similar to normal Convolutional neural network in terms of its structure. Therefore, like CNNs they are also able to exploit the 2D structure of images combined with the advantage gained by pre-training in Deep belief network. They provide a generic structure which can be used in many image and signal processing tasks and can be trained in a way similar to that for Deep Belief Networks. Recently, many benchmark results on standard image datasets like CIFAR [78] have been obtained using CDBNs.[79]

62.4.6 Deep Boltzmann Machines

A *Deep Boltzmann Machine* (DBM) is a type of binary pairwise Markov random field (undirected probabilistic graphical models) with multiple layers of hidden random variables. It is a network of symmetrically coupled stochastic binary units. It comprises a set of visible units $\boldsymbol{\nu} \in \{0,1\}^D$, and a series of layers of hidden units $\boldsymbol{h}^{(1)} \in \{0,1\}^{F_1}, \boldsymbol{h}^{(2)} \in \{0,1\}^{F_2}, \ldots, \boldsymbol{h}^{(L)} \in \{0,1\}^{F_L}$. There is no connection between the units of the same layer (like RBM). For the DBM, we can write the probability which is assigned to vector ν as:

$$p(\boldsymbol{\nu}) = \frac{1}{Z} \sum_h e^{\sum_{ij} W_{ij}^{(1)} \nu_i h_j^{(1)} + \sum_{jl} W_{jl}^{(2)} h_j^{(1)} h_l^{(2)} + \sum_{lm} W_{lm}^{(3)} h_l^{(2)} h_m^{(3)}},$$

where $\boldsymbol{h} = \{\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\}$ are the set of hidden units, and $\theta = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}\}$ are the model parameters, representing visible-hidden and hidden-hidden *symmetric interaction*, since they are undirected links. As it is clear by setting $\boldsymbol{W}^{(2)} = 0$ and $\boldsymbol{W}^{(3)} = 0$ the network becomes the well-known Restricted Boltzmann machine.[80]

There are several reasons which motivate us to take advantage of deep Boltzmann machine architectures. Like DBNs, they benefit from the ability of learning complex and abstract internal representations of the input in tasks such as object or speech recognition, with the use of *limited number* of *labeled* data to fine-tune the representations built based on a large *supply* of *unlabeled* sensory input data. However, unlike DBNs and *deep convolutional* neural networks, they adopt the inference and training procedure in both directions, bottom-up and top-down pass, which enable the DBMs to better unveil the representations of the ambiguous and complex input structures,[81] .[82]

Since the *exact maximum likelihood* learning is intractable for the DBMs, we may perform the *approximate maximum likelihood* learning. There is another possibility, to use *mean-field* inference to estimate data-dependent expectations, incorporation with a *Markov chain Monte Carlo (MCMC)* based stochastic approximation technique to approximate the expected *sufficient statistics* of the model.[80]

We can see the difference between DBNs and DBM. In DBNs, the top two layers form a restricted Boltzmann machine which is an undirected graphical model, but the lower layers form a directed generative model.

Apart from all the advantages of DBMs discussed so far, they have a crucial disadvantage which limits the performance and functionality of this kind of architecture. The approximate inference, which is based on mean-field method, is about 25 to 50 times slower than a single bottom-up pass in DBNs. This time consuming task make the joint optimization, quite impractical for large data sets, and seriously restricts the use of DBMs in tasks such as feature representations (the mean-field inference have to be performed for each new test input).[83]

62.4.7 Stacked (Denoising) Auto-Encoders

The auto encoder idea is motivated by the concept of *good* representation. For instance for the case of classifier it is possible to define that a *good representation is one that will yield a better performing classifier*.

An *encoder* is referred to a deterministic mapping f_θ that transforms an input vector $\boldsymbol{x}$ into hidden representation $\boldsymbol{y}$, where $\theta = \{\boldsymbol{W}, b\}$, $\boldsymbol{W}$ is the weight matrix and **b** is an offset vector (bias). On the contrary a *decoder* maps back the hidden representation **y** to the reconstructed input $\boldsymbol{z}$ via g_θ . The whole process of auto encoding is to compare this reconstructed input to the original and try to minimize this error to make the reconstructed value as close as possible to the original.

In *stacked denoising auto encoders*, the partially corrupted output is cleaned (*denoised*). This fact has been introduced in [84] with a specific approach to *good* representation, a *good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input.* Implicit in this definition are the ideas of

- The higher level representations are relatively stable and robust to the corruption of the input;
- It is required to extract features that are useful for representation of the input distribution.

The algorithm consists of multiple steps; starts by a stochastic mapping of $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}}$ through $q_D(\tilde{\boldsymbol{x}}|\boldsymbol{x})$, this is the corrupting step. Then the corrupted input $\tilde{\boldsymbol{x}}$ passes through a basic auto encoder process and is mapped to a hidden representation $\boldsymbol{y} = f_\theta(\tilde{\boldsymbol{x}}) = s(\boldsymbol{W}\tilde{\boldsymbol{x}} + b)$. From this hidden representation we can reconstruct $\boldsymbol{z} = g_\theta(\boldsymbol{y})$. In the last stage a minimization algorithm is done in order to have a z as close as possible to uncorrupted input $\boldsymbol{x}$. The reconstruction error $L_H(\boldsymbol{x}, \boldsymbol{z})$ might be either the cross-entropy loss with an affine-sigmoid decoder, or the squared error loss with an affine decoder.[84]

In order to make a deep architecture, auto encoders stack one on top of another. Once the encoding function f_θ of the first denoising auto encoder is learned and used to uncorrupt the input (corrupted input), we can train the second level.[84]

Once the stacked auto encoder is trained, its output might be used as the input to a supervised learning algorithm such as support vector machine classifier or a multiclass logistic regression.[84]

62.4.8 Deep Stacking Networks

One of the deep architectures recently introduced in[85] which is based on building hierarchies with blocks of simplified neural network modules, is called *deep convex network*. They are called “convex” because of the formulation of the weights learning problem, which is a convex optimization problem with a closed-form solution. The network is also called the *deep stacking network (DSN)*,[86] emphasizing on this fact that a similar mechanism as the *stacked generalization* is used.[87]

The DSN blocks, each consisting of a simple, easy-to-learn module, are stacked to form the overall deep network. It can be trained block-wise in a supervised fashion without the need for back-propagation for the entire blocks.[88]

As designed in [85] each block consists of a simplified MLP with a single hidden layer. It comprises a weight matrix $\boldsymbol{U}$ as the connection between the logistic sigmoidal units of the hidden layer $\boldsymbol{h}$ to the linear output layer $\boldsymbol{y}$, and a weight matrix $\boldsymbol{W}$ which connects each input of the blocks to their respective hidden layers. If we assume that the target vectors $\boldsymbol{t}$ be arranged to form the columns of $\boldsymbol{T}$ (the target matrix), let the input data vectors $\boldsymbol{x}$ be arranged to form the columns of $\boldsymbol{X}$, let $\boldsymbol{H} = \sigma(\boldsymbol{W}^T\boldsymbol{X})$ denote the matrix of hidden units, and assume the lower-layer weights $\boldsymbol{W}$ are known (training layer-by-layer). The function performs the element-wise logistic sigmoid operation. Then learning the upper-layer weight matrix $\boldsymbol{U}$ given other weights in the network can be formulated as a convex optimization problem:

$$\min_{U^T} f = ||\boldsymbol{U}^T\boldsymbol{H} - \boldsymbol{T}||_F^2,$$

which has a closed-form solution. The input to the first block $\boldsymbol{X}$ only contains the original data, however in the upper blocks in addition to this original (raw) data there is a copy of the lower-block(s) output y.

In each block an estimate of the same final label class y is produced, then this estimated label concatenated with original input to form the *expanded input* for the upper block. In contrast with other deep architectures, such as DBNs, the goal is not to discover the transformed feature representation. Regarding the structure of the hierarchy of this kind of architecture, it makes the parallel training straightforward as the problem is naturally a batch-mode optimization one. In purely discriminative tasks DSN performance is better than the conventional DBN.[86]

62.4.9 Tensor Deep Stacking Networks (T-DSN)

This architecture is an extension of the DSN. It improves the DSN in two important ways, using the higher order information by means of covariance statistics and transforming the non-convex problem of the lower-layer to a convex sub-problem of the upper-layer.[89]

Unlike the DSN, the covariance statistics of the data is employed using a bilinear mapping from two distinct sets of hidden units in the same layer to predictions via a third-order tensor.

The scalability and parallelization are the two important factors in the learning algorithms which are not considered seriously in the conventional DNNs.[90][91][92] All the learning process for the DSN (and TDSN as well) is done on a batch-mode basis so as to make the parallelization possible on a cluster of CPU or GPU nodes.[85][86] Parallelization gives the opportunity to scale up the design to larger (deeper) architectures and data sets.

The basic architecture is suitable for diverse tasks such as classification and regression.

62.4.10 Spike-and-Slab RBMs (ssRBMs)

The need for real-valued inputs which are employed in Gaussian RBMs (GRBMs), motivates scientists seeking new methods. One of these methods is the *spike and slab* RBM (*ss*RBMs), which models continuous-valued inputs with strictly binary latent variables.[93]

Similar to basic RBMs and its variants, the spike and slab RBM is a bipartite graph. Like GRBM, the visible units (input) are real-valued. The difference arises in the hidden layer, where each hidden unit come along with a binary spike variable and real-valued slab variable. These terms (spike and slab) come from the statistics literature,[94] and refer to a prior including a mixture of two components. One is a discrete probability mass at zero called spike, and the other is a density over continuous domain.[95][95]

There is also an extension of the ssRBM model, which is called µ-ssRBM. This variant provides extra modeling capacity to the architecture using additional terms in the energy function. One of these terms enable model to form a conditional distribution of the spike variables by means of marginalizing out the slab variables given an observation.

62.4.11 Compound Hierarchical-Deep Models

The class architectures called *compound HD models*, where HD stands for *Hierarchical-Deep* are structured as a composition of non-parametric Bayesian models with deep networks. The features, learned by deep architectures such as DBNs,[96] DBMs,[81] deep auto encoders,[97] convolutional variants,[98][99] ssRBMs,[95] deep coding network,[100] DBNs with sparse feature learning,[101] recursive neural networks,[102] conditional DBNs,[103] denoising auto encoders,[104] are able to provide better representation for more rapid and accurate classification tasks with high-dimensional training data sets. However, they are not quite powerful in learning novel classes with few examples, themselves. In these architectures, all units through the network are involved in the representation of the input (*distributed representations*), and they have to be adjusted together (high degree of freedom). However, if we limit the degree of freedom, we make it easier for the model to learn new classes out of few training samples (less parameters to learn). *Hierarchical Bayesian (HB)* models, provide learning from few examples, for example [105][106][107][108][109] for computer vision, statistics, and cognitive science.

Compound HD architectures try to integrate both characteristics of HB and deep networks. The compound HDP-DBM architecture, a *hierarchical Dirichlet process (HDP)* as a hierarchical model, incorporated with DBM architecture. It is a full generative model, generalized from abstract concepts flowing through the layers of the model, which is able to synthesize new examples in novel classes that look *reasonably natural*. Note that all the levels are learned jointly by maximizing a joint log-probability score.[110]

Consider a DBM with three hidden layers, the probability of a visible input $\boldsymbol{\nu}$ is:

$$p(\boldsymbol{\nu}, \psi) = \frac{1}{Z} \sum_h e^{\sum_{ij} W_{ij}^{(1)} \nu_i h_j^1 + \sum_{jl} W_{jl}^{(2)} h_j^1 h_l^2 + \sum_{lm} W_{lm}^{(3)} h_l^2 h_m^3}$$

where $\boldsymbol{h} = \{\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\}$ are the set of hidden units, and $\psi = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}\}$ are the model parameters, representing visible-hidden and hidden-hidden symmetric interaction terms.

After a DBM model has been learned, we have an undirected model that defines the joint distribution $P(\nu, h^1, h^2, h^3)$. One way to express what has been learned is the conditional model $P(\nu, h^1, h^2|h^3)$ and a prior term $P(h^3)$.

The part $P(\nu, h^1, h^2|h^3)$, represents a *conditional* DBM model, which can be viewed as a two-layer DBM but with bias terms given by the states of h^3 :

$$P(\nu, h^1, h^2|h^3) = \frac{1}{Z(\psi, h^3)} e^{\sum_{ij} W_{ij}^{(1)} \nu_i h_j^1 + \sum_{jl} W_{jl}^{(2)} h_j^1 h_l^2 + \sum_{lm} W_{lm}^{(3)} h_l^2 h_m^3}$$

62.4.12 Deep Coding Networks

There are several advantages to having a model which can *actively* update itself to the context in data. One of these methods arises from the idea to have a model which is able to adjust its prior knowledge dynamically according to the context of the data. Deep coding network (DPCN) is a predictive coding scheme where top-down information is used to empirically adjust the priors needed for the bottom-up inference procedure by means of a deep locally connected generative model. This is based on extracting sparse features out of time-varying observations using a linear dynamical model. Then, a pooling strategy is employed in order to learn invariant feature representations. Similar to other deep architectures, these blocks are the building elements of a deeper architecture where greedy layer-wise unsupervised learning are used. Note that the layers constitute a kind of Markov chain such that the states at any layer are only dependent on the succeeding and preceding layers.

Deep predictive coding network (DPCN)[111] predicts the representation of the layer, by means of a top-down approach using the information in upper layer and also temporal dependencies from the previous states, it is called

It is also possible to extend the DPCN to form a convolutional network.[111]

62.4.13 Multilayer Kernel Machine

The *Multilayer Kernel Machine (MKM)* as introduced in [112] is a way of learning highly nonlinear functions with the iterative applications of weakly nonlinear kernels. They use the *kernel principal component analysis (KPCA)*, in,[113] as method for unsupervised greedy layer-wise pre-training step of the deep learning architecture.

Layer $l + 1$ -th learns the representation of the previous layer l , extracting the n_l principal component (PC) of the projection layer l output in the feature domain induced by the kernel. For the sake of dimensionality reduction of the updated representation in each layer, a supervised strategy is proposed to select the best informative features among the ones extracted by KPCA. The process is:

- ranking the n_l features according to their mutual information with the class labels;
- for different values of K and $m_l \in \{1, \ldots, n_l\}$, compute the classification error rate of a *K-nearest neighbor (K-NN)* classifier using only the m_l most informative features on a validation set;
- the value of m_l with which the classifier has reached the lowest error rate determines the number of features to retain.

There are some drawbacks in using the KPCA method as the building cells of an MKM.

Another, more straightforward method of integrating kernel machine into the deep learning architecture was developed by Microsoft researchers for spoken language understanding applications.[114] The main idea is to use a kernel machine to approximate a shallow neural net with an infinite number of hidden units, and then to use the stacking technique to splice the output of the kernel machine and the raw input in building the next, higher level of the kernel machine. The number of the levels in this kernel version of the deep convex network is a hyper-parameter of the overall system determined by cross validation.

62.4.14 Deep Q-Networks

This is the latest class of deep learning models targeted for reinforcement learning, published in February 2015 in Nature[115] The application discussed in this paper is limited to ATARI gaming, but the implications for other potential applications are profound.

62.4.15 Memory networks

Integrating external memory component with artificial neural networks has a long history dating back to early research in distributed representations [116] and self-organizing maps. E.g. in sparse distributed memory or HTM the patterns encoded by neural networks are used as memory addresses for *content-addressable memory*, with "neurons" essentially serving as address encoders and decoders.

In the 1990s and 2000s, there was a lot of related work with differentiable long-term memories. For example:

- Differentiable push and pop actions for alternative memory networks called *neural stack machines*[117][118]
- Memory networks where the control network's external differentiable storage is in the fast weights of another network [119]
- The LSTM *"forget gates"* [120]
- Self-referential RNNs with special output units for addressing and rapidly manipulating each of the RNN's own weights in differentiable fashion (so the external storage is actually internal) [121][122]

More recently deep learning was shown to be useful in semantic hashing[123] where a deep graphical model the word-count vectors[124] obtained from a large set of documents. Documents are mapped to memory addresses in such a way that semantically similar documents are located at nearby addresses. Documents similar to a query document can then be found by simply accessing all the addresses that differ by only a few bits from the address of the query document.

Neural Turing Machines[125] developed by Google DeepMind extend the capabilities of deep neural networks by coupling them to external memory resources, which they can interact with by attentional processes. The combined system is analogous to a Turing Machine but is differentiable end-to-end, allowing it to be efficiently trained with gradient descent. Preliminary results demonstrate that Neural Turing Machines can infer simple algorithms such as copying, sorting, and associative recall from input and output examples.

Memory Networks[126] is another extension to neural networks incorporating long-term memory which was developed by Facebook research. The long-term memory can be read and written to, with the goal of using it for prediction. These models have been applied in the context of question answering (QA) where the long-term memory effectively acts as a (dynamic) knowledge base, and the output is a textual response.

62.5 Applications

62.5.1 Automatic speech recognition

The results shown in the table below are for automatic speech recognition on the popular TIMIT data set. This is a common data set used for initial evaluations of deep learning architectures. The entire set contains 630 speakers from eight major dialects of American English, with each speaker reading 10 different sentences.[127] Its small size allows many different configurations to be tried effectively with it. More importantly, the TIMIT task concerns phone-sequence recognition, which, unlike word-sequence recognition, permits very weak "language models" and thus the weaknesses in acoustic modeling aspects of speech recognition can be more easily analyzed. It was such analysis on TIMIT contrasting the GMM (and other generative models of speech) vs. DNN models carried out by Li Deng and collaborators around 2009-2010 that stimulated early industrial investment on deep learning technology for speech recognition from small to large scales,[25][36] eventually leading to pervasive and dominant uses of deep learning in speech recognition industry. That analysis was carried out with comparable performance (less than 1.5% in error rate) between discriminative DNNs and generative models. The error rates presented below, including these early results and measured as percent phone error rates (PER), have been summarized over a time span of the past 20 years:

Extension of the success of deep learning from TIMIT to large vocabulary speech recognition occurred in 2010 by industrial researchers, where large output layers of the DNN based on context dependent HMM states constructed by decision trees were adopted.[130][131] See comprehensive reviews of this development and of the state of the art as of October 2014 in the recent Springer book from Microsoft Research.[37] See also the related background of automatic speech recognition and the impact of various machine learning paradigms including notably deep learning in a recent overview article.[132]

One fundamental principle of deep learning is to do away with hand-crafted feature engineering and to use raw features. This principle was first explored successfully in the architecture of deep autoencoder on the "raw" spectrogram or linear filter-bank features,[133] showing its superiority over the Mel-Cepstral features which contain a few stages of fixed transformation from spectrograms. The true "raw" features of speech, waveforms, have more recently been shown to produce excellent larger-scale speech recognition results.[134]

Since the initial successful debut of DNNs for speech recognition around 2009-2011, there has been huge progress made. This progress (as well as future directions) has been summarized into the following eight major areas:[1][27][37] 1) Scaling up/out and speedup DNN training and decoding; 2) Sequence discriminative training of DNNs; 3) Feature processing by deep models with solid understanding of the underlying mechanisms; 4) Adaptation of DNNs and of related deep models; 5) Multi-task and transfer learning by DNNs and related deep models; 6) Convolution neural networks and how to design them to best exploit domain knowledge of speech; 7) Recurrent neural network and its rich LSTM variants; 8) Other types of deep models including tensor-based models and integrated deep generative/discriminative models.

Large-scale automatic speech recognition is the first and the most convincing successful case of deep learning in the recent history, embraced by both industry and academic across the board. Between 2010 and 2014, the two major conferences on signal processing and speech recognition, IEEE-ICASSP and Interspeech, have seen near exponential growth in the numbers of accepted papers in their respective annual conference papers on the topic of deep learning for speech recognition. More importantly, all major commercial speech recognition systems (e.g., Microsoft Cortana, Xbox, Skype Translator, Google Now, Apple Siri, Baidu and iFlyTek voice search, and a range of Nuance speech products, etc.) nowadays are based on deep learning methods.[1][135][136] See also the recent media interview with the CTO of Nuance Communications.[137]

The wide-spreading success in speech recognition achieved by 2011 was followed shortly by large-scale image recognition described next.

62.5.2 Image recognition

A common evaluation set for image classification is the MNIST database data set. MNIST is composed of handwritten digits and includes 60000 training examples and 10000 test examples. Similar to TIMIT, its small size allows multiple configurations to be tested. A comprehensive list of results on this set can be found in.[138] The current best result on MNIST is an error rate of 0.23%, achieved by Ciresan et al. in 2012.[139]

The real impact of deep learning in image or object recognition, one major branch of computer vision, was felt in the fall of 2012 after the team of Geoff Hinton and his students won the large-scale ImageNet competition by a significant margin over the then-state-of-the-art shallow machine learning methods. The technology is based on 20-year-old deep convolutional nets, but with much larger scale on a much larger task, since it had been learned that deep learning works quite well on large-scale speech recognition. In 2013 and 2014, the error rate on the ImageNet task using deep learning was further reduced at a rapid pace, following a similar trend in large-scale speech recognition.

As in the ambitious moves from automatic speech recognition toward automatic speech translation and understanding, image classification has recently been extended to the more ambitious and challenging task of automatic image captioning, in which deep learning is the essential underlying technology. [140] [141] [142] [143]

One example application is a car computer said to be trained with deep learning, which may be able to let cars interpret 360° camera views.[144]

62.5.3 Natural language processing

Neural networks have been used for implementing language models since the early 2000s.[145] Key techniques in this field are negative sampling[146] and word embedding. A word embedding, such as *word2vec*, can be thought of as a representational layer in a deep learning architecture transforming an atomic word into a positional representation of the word relative to other words in the dataset; the position is represented as a point in a vector space. Using a word embedding as an input layer to a recursive neural network (RNN) allows for the training of the network to parse sentences and phrases using an effective *compositional vector grammar*. A compositional vector grammar can be thought of as probabilistic context free grammar (PCFG) implemented by a recursive neural network.[147] Recursive autoencoders built atop word embeddings have been trained to assess sentence similarity and detect paraphrasing.[147] Deep neural architectures have achieved state-of-the-art results in many tasks in natural language processing, such as constituency parsing,[148] sentiment analysis,[149] information retrieval,[150] [151] machine translation, [152] [153] contextual entity linking, [154] and other areas of NLP. [155]

62.5.4 Drug discovery and toxicology

The pharmaceutical industry faces the problem that a large percentage of candidate drugs fail to reach the market. These failures of chemical compounds are caused by insufficient efficacy on the biomolecular target (on-target effect), undetected and undesired interactions with other biomolecules (off-target effects), or unanticipated toxic effects.[156][157] In 2012 a team led by George Dahl won the "Merck Molecular Activity Challenge" using multi-task deep neural networks to predict the biomolecular target of a compound.[158][159] In 2014 Sepp Hochreiter's group used Deep Learning to detect off-target and toxic effects of environmental chemicals in nutrients, household products and drugs and won the "Tox21 Data Challenge" of NIH, FDA and NCATS.[160][161] These impressive successes show Deep Learning may be superior to other virtual screening methods.[162][163] Researchers from Google and Stanford enhanced Deep Learning for drug discovery by combining data from a variety of sources.[164]

62.5.5 Customer relationship management

Recently success has been reported with application of deep reinforcement learning in direct marketing settings, illustrating suitability of the method for CRM automation. A neural network was used to approximate the value of possible direct marketing actions over the customer state space, defined in terms of RFM variables. The estimated value function was shown to have a natural interpretation as CLV (customer lifetime value).[165]

62.6 Deep learning in the human brain

Computational deep learning is closely related to a class of theories of brain development (specifically, neocortical development) proposed by cognitive neuroscientists in the early 1990s.[166] An approachable summary of this work is Elman, et al.'s 1996 book "Rethinking Innateness" [167] (see also: Shrager and Johnson;[168] Quartz and Sejnowski [169]). As these developmental theories were also instantiated in computational models, they are technical predecessors of purely computationally-motivated deep learning models. These developmental models share the interesting property that various proposed learning dynamics in the brain (e.g., a wave of nerve growth factor) conspire to support the self-organization of just the sort of inter-related neural networks utilized in the later, purely computational deep learning models; and such computational neural networks seem analogous to a view of the brain's neocortex as a hierarchy of filters in which each layer captures some of the information in the operating environment, and then passes the remainder, as well as modified base signal, to other layers further up the hierarchy. This process yields a self-organizing stack of transducers, well-tuned to their operating environment. As described in The New York Times in 1995: "...the infant's brain seems to organize itself under the influence of waves of so-called trophic-factors ... different regions of the brain become connected sequentially, with one layer of tissue maturing before another and so on until the whole brain is mature." [170]

The importance of deep learning with respect to the evolution and development of human cognition did not escape the attention of these researchers. One aspect of human development that distinguishes us from our nearest primate neighbors may be changes in the timing of development.[171] Among primates, the human brain remains relatively plastic until late in the post-natal period, whereas the brains of our closest relatives are more completely formed by birth. Thus, humans have greater access to the complex experiences afforded by being out in the world during the most formative period of brain development. This may enable us to "tune in" to rapidly changing features of the environment that other animals, more constrained by evolutionary structuring of their brains, are unable to take account of. To the extent that these changes are reflected in similar timing changes in hypothesized wave of cortical development, they may also lead to changes in the extraction of information from the stimulus environment during the early self-organization of the brain. Of course, along with this flexibility comes an extended period of immaturity, during which we are dependent upon our caretakers and our community for both support and training. The theory of deep learning therefore sees the coevolution of culture and cognition as a fundamental condition of human evolution.[172]

62.7 Commercial activities

Deep learning is often presented as a step towards realising strong AI[173] and thus many organizations have become interested in its use for particular applications. Most recently, in December 2013, Facebook announced that it hired Yann LeCun to head its new artificial intelligence (AI) lab that will have operations in California, London, and New York. The AI lab will be used for developing deep learning techniques that will help Facebook do tasks such as automatically tagging uploaded pictures with the names of the people in them.[174]

In March 2013, Geoffrey Hinton and two of his graduate students, Alex Krizhevsky and Ilya Sutskever, were hired by Google. Their work will be focused on both improving existing machine learning products at Google and also help deal with the growing amount of data Google has. Google also purchased Hinton's company, DNNresearch.

In 2014 Google also acquired DeepMind Technologies, a British start-up that developed a system capable of learning how to play Atari video games using only raw pixels as data input.

Also in 2014, Microsoft established The Deep Learning Technology Center in its MSR division, amassing deep learning experts for application-focused activities.

And Baidu hired Andrew Ng to head their new Silicon

Valley based research lab focusing on deep learning.

62.8 Criticism and comment

Given the far-reaching implications of artificial intelligence coupled with the realization that deep learning is emerging as one of its most powerful techniques, the subject is understandably attracting both criticism and comment, and in some cases from outside the field of computer science itself.

A main criticism of deep learning concerns the lack of theory surrounding many of the methods. Most of the learning in deep architectures is just some form of gradient descent. While gradient descent has been understood for a while now, the theory surrounding other algorithms, such as contrastive divergence is less clear (i.e., Does it converge? If so, how fast? What is it approximating?). Deep learning methods are often looked at as a black box, with most confirmations done empirically, rather than theoretically.

Others point out that deep learning should be looked at as a step towards realizing strong AI, not as an all-encompassing solution. Despite the power of deep learning methods, they still lack much of the functionality needed for realizing this goal entirely. Research psychologist Gary Marcus has noted that:

“Realistically, deep learning is only part of the larger challenge of building intelligent machines. Such techniques lack ways of representing causal relationships (...) have no obvious ways of performing logical inferences, and they are also still a long way from integrating abstract knowledge, such as information about what objects are, what they are for, and how they are typically used. The most powerful A.I. systems, like Watson (...) use techniques like deep learning as just one element in a very complicated ensemble of techniques, ranging from the statistical technique of Bayesian inference to deductive reasoning.” [175]

To the extent that such a viewpoint implies, without intending to, that deep learning will ultimately constitute nothing more than the primitive discriminatory levels of a comprehensive future machine intelligence, a recent pair of speculations regarding art and artificial intelligence[176] offers an alternative and more expansive outlook. The first such speculation is that it might be possible to train a machine vision stack to perform the sophisticated task of discriminating between “old master” and amateur figure drawings; and the second is that such a sensitivity might in fact represent the rudiments of a non-trivial machine empathy. It is suggested, moreover, that such an eventuality would be in line with both anthropology, which identifies a concern with aesthetics as a key element of behavioral modernity, and also with a current school of thought which suspects that the allied phenomenon of consciousness, formerly thought of as a purely high-order phenomenon, may in fact have roots deep within the structure of the universe itself.

In further reference to the idea that a significant degree of artistic sensitivity might inhere within relatively low levels, whether biological or digital, of the cognitive hierarchy, there has recently been published a series of graphic representations of the internal states of deep (20-30 layers) neural networks attempting to discern within essentially random data the images on which they have been trained,[177] and these show a striking degree of what can only be described as visual creativity. This work, moreover, has captured a remarkable level of public attention, with the original research notice receiving well in excess of one thousand comments, and The Guardian coverage[178] achieving the status of most frequently accessed article on that newspaper’s web site.

Some currently popular and successful deep learning architectures display certain problematical behaviors[179] (e.g. confidently classifying random data as belonging to a familiar category of nonrandom images;[180] and misclassifying miniscule perturbations of correctly classified images [181]). The creator of OpenCog, Ben Goertzel hypothesized [179] that these behaviors are tied with limitations in the internal representations learned by these architectures, and that these same limitations would inhibit integration of these architectures into heterogeneous multi-component AGI architectures. It is suggested that these issues can be worked around by developing deep learning architectures that internally form states homologous to image-grammar [182] decompositions of observed entities and events.[179] Learning a grammar (visual or linguistic) from training data would be equivalent to restricting the system to commonsense reasoning which operates on concepts in terms of production rules of the grammar, and is a basic goal of both human language acquisition [183] and A.I. (Also see Grammar induction [184])

62.9 Deep learning software libraries

- Torch - An open source software library for machine learning based on the Lua programming language.
- Theano - An open source machine learning library for Python.
- H2O.ai - An open source machine learning platform written in Java with a parallel architecture.
- Deeplearning4j - An open source deep learning libray written for Java. It provides parallelization with CPUs and GPUs.
- OpenNN - An open source C++ library which implements deep neural networks and provides parallelization with CPUs.

- NVIDIA cuDNN - A GPU-accelerated library of primitives for deep neural networks.
- DeepLearnToolbox - A Matlab/Octave toolbox for deep learning.
- convnetjs - A Javascript library for training deep learning models. It contains online demos.
- Gensim - A toolkit for natural language processing implemented in the Python programming language.
- Caffe - A deep learning framework .
- Apache SINGA[185] - A deep learning platform developed for scalability, usability and extensibility.

62.10 See also

- Sparse coding
- Compressed Sensing
- Connectionism
- Self-organizing map
- Applications of artificial intelligence
- List of artificial intelligence projects
- Reservoir computing
- Liquid state machine
- Echo state network

62.11 References

[1] L. Deng and D. Yu (2014) “Deep Learning: Methods and Applications” http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-039.pdf

[2] Bengio, Yoshua (2009). “Learning Deep Architectures for AI” (PDF). *Foundations and Trends in Machine Learning* **2** (1).

[3] Y. Bengio, A. Courville, and P. Vincent., “Representation Learning: A Review and New Perspectives,” *IEEE Trans. PAMI, special issue Learning Deep Architectures*, 2013

[4] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview” http://arxiv.org/abs/1404.7828, 2014

[5] Patrick Glauner (2015), *Comparison of Training Methods for Deep Neural Networks*, arXiv:1504.06825

[6] Song, Hyun Ah, and Soo-Young Lee. “Hierarchical Representation Using NMF.” Neural Information Processing. Springer Berlin Heidelberg, 2013.

[7] Olshausen, Bruno A. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images.” Nature 381.6583 (1996): 607-609.

[8] Ronan Collobert (May 6, 2011). “Deep Learning for Efficient Discriminative Parsing”. *videolectures.net*. Ca. 7:45.

[9] Gomes, Lee (20 October 2014). “Machine-Learning Maestro Michael Jordan on the Delusions of Big Data and Other Huge Engineering Efforts”. *IEEE Spectrum*.

[10] Fukushima, K. (1980). “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. *Biol. Cybern* **36**: 193–202. doi:10.1007/bf00344251.

[11] P. Werbos., “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences,” *PhD thesis, Harvard University*, 1974.

[12] LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Computation*, 1, pp. 541–551, 1989.

[13] S. Hochreiter., "Untersuchungen zu dynamischen neuronalen Netzen," *Diploma thesis. Institut f. Informatik, Technische Univ. Munich. Advisor: J. Schmidhuber*, 1991.

[14] S. Hochreiter *et al.*, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” *In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press*, 2001.

[15] J. Weng, N. Ahuja and T. S. Huang, "Cresceptron: a self-organizing neural network which grows adaptively," *Proc. International Joint Conference on Neural Networks*, Baltimore, Maryland, vol I, pp. 576-581, June, 1992.

[16] J. Weng, N. Ahuja and T. S. Huang, "Learning recognition and segmentation of 3-D objects from 2-D images," *Proc. 4th International Conf. Computer Vision*, Berlin, Germany, pp. 121-128, May, 1993.

[17] J. Weng, N. Ahuja and T. S. Huang, "Learning recognition and segmentation using the Cresceptron," *International Journal of Computer Vision*, vol. 25, no. 2, pp. 105-139, Nov. 1997.

[18] Morgan, Bourlard, Renals, Cohen, Franco (1993) “Hybrid neural network/hidden Markov model systems for continuous speech recognition. ICASSP/IJPRAI”

[19] T. Robinson. (1992) A real-time recurrent error propagation network word recognition system, ICASSP.

[20] Waibel, Hanazawa, Hinton, Shikano, Lang. (1989) “Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech and Signal Processing.”

[21] Baker, J.; Deng, Li; Glass, Jim; Khudanpur, S.; Lee, C.-H.; Morgan, N.; O'Shaughnessy, D. (2009). “Research Developments and Directions in Speech Recognition and Understanding, Part 1”. *IEEE Signal Processing Magazine* **26** (3): 75–80. doi:10.1109/msp.2009.932166.

[22] Y. Bengio (1991). "Artificial Neural Networks and their Application to Speech/Sequence Recognition," Ph.D. thesis, McGill University, Canada.

[23] Deng, L.; Hassanein, K.; Elmasry, M. (1994). "Analysis of correlation structure for a neural predictive model with applications to speech recognition". *Neural Networks* **7** (2): 331–339. doi:10.1016/0893-6080(94)90027-2.

[24] Hinton, G.; Deng, L.; Yu, D.; Dahl, G.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.; Kingsbury, B. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition --- The shared views of four research groups". *IEEE Signal Processing Magazine* **29** (6): 82–97. doi:10.1109/msp.2012.2205597.

[25] Deng, L.; Hinton, G.; Kingsbury, B. (2013). "New types of deep neural network learning for speech recognition and related applications: An overview (ICASSP)".

[26] Keynote talk: Recent Developments in Deep Neural Networks. ICASSP, 2013 (by Geoff Hinton).

[27] Keynote talk: "Achievements and Challenges of Deep Learning - From Speech Analysis and Recognition To Language and Multimodal Processing," Interspeech, September 2014.

[28] G. E. Hinton., "Learning multiple layers of representation," *Trends in Cognitive Sciences*, 11, pp. 428–434, 2007.

[29] J. Schmidhuber., "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, 4, pp. 234–242, 1992.

[30] J. Schmidhuber., "My First Deep Learning System of 1991 + Deep Learning Timeline 1962–2013."

[31] http://research.microsoft.com/apps/pubs/default.aspx?id=189004

[32] L. Deng et al. Recent Advances in Deep Learning for Speech Research at Microsoft, ICASSP, 2013.

[33] L. Deng, O. Abdel-Hamid, and D. Yu, A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion, ICASSP, 2013.

[34] T. Sainath *et al.*, "Convolutional neural networks for LVCSR," *ICASSP*, 2013.

[35] D. Yu, L. Deng, G. Li, and F. Seide (2011). "Discriminative pretraining of deep neural networks," U.S. Patent Filing.

[36] NIPS Workshop: Deep Learning for Speech Recognition and Related Applications, Whistler, BC, Canada, Dec. 2009 (Organizers: Li Deng, Geoff Hinton, D. Yu).

[37] Yu, D.; Deng, L. (2014). "Automatic Speech Recognition: A Deep Learning Approach (Publisher: Springer)".

[38] D. C. Ciresan *et al.*, "Deep Big Simple Neural Nets for Handwritten Digit Recognition," *Neural Computation*, 22, pp. 3207–3220, 2010.

[39] R. Raina, A. Madhavan, A. Ng., "Large-scale Deep Unsupervised Learning using Graphics Processors," *Proc. 26th Int. Conf. on Machine Learning*, 2009.

[40] Riesenhuber, M; Poggio, T. "Hierarchical models of object recognition in cortex". *Nature Neuroscience* **1999** (11): 1019–1025.

[41] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition.* Neural Computation, 1(4):541–551, 1989.

[42] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut f. Informatik, Technische Univ. Munich, 1991. Advisor: J. Schmidhuber

[43] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.

[44] Hochreiter, Sepp; and Schmidhuber, Jürgen; *Long Short-Term Memory*, Neural Computation, 9(8):1735–1780, 1997

[45] Graves, Alex; and Schmidhuber, Jürgen; *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks*, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), *Advances in Neural Information Processing Systems 22 (NIPS'22), December 7th–10th, 2009, Vancouver, BC*, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552

[46] Graves, A.; Liwicki, M.; Fernandez, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31** (5): 2009.

[47] Sven Behnke (2003). *Hierarchical Neural Networks for Image Interpretation.* (PDF). Lecture Notes in Computer Science **2766**. Springer.

[48] Smolensky, P. (1986). *Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* **1**. pp. 194–281.

[49] Hinton, G. E.; Osindero, S.; Teh, Y. (2006). "A fast learning algorithm for deep belief nets" (PDF). *Neural Computation* **18** (7): 1527–1554. doi:10.1162/neco.2006.18.7.1527. PMID 16764513.

[50] Hinton, G. (2009). "Deep belief networks". *Scholarpedia* **4** (5): 5947. doi:10.4249/scholarpedia.5947.

[51] John Markoff (25 June 2012). "How Many Computers to Identify a Cat? 16,000.". *New York Times.*

[52] Ng, Andrew; Dean, Jeff (2012). "Building High-level Features Using Large Scale Unsupervised Learning" (PDF).

[53] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber. Flexible, High Performance Convolutional Neural Networks for Image Classification. International Joint Conference on Artificial Intelligence (IJCAI-2011, Barcelona), 2011.

[54] Martines, H.; Bengio, Y.; Yannakakis, G. N. (2013). "Learning Deep Physiological Models of Affect". *IEEE Computational Intelligence* **8** (2): 20.

[55] D. C. Ciresan, U. Meier, J. Masci, J. Schmidhuber. Multi-Column Deep Neural Network for Traffic Sign Classification. Neural Networks, 2012.

[56] D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In Advances in Neural Information Processing Systems (NIPS 2012), Lake Tahoe, 2012.

[57] D. C. Ciresan, U. Meier, J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. IEEE Conf. on Computer Vision and Pattern Recognition CVPR 2012.

[58] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, 1, pp. 1-47, 1991.

[59] J. Weng, "Natural and Artificial Intelligence: Introduction to Computational Brain-Mind," BMI Press, ISBN 978-0985875725, 2012.

[60] J. Weng, "Why Have We Passed `Neural Networks Do not Abstract Well'?," *Natural Intelligence: the INNS Magazine*, vol. 1, no.1, pp. 13-22, 2011.

[61] Z. Ji, J. Weng, and D. Prokhorov, "Where-What Network 1: Where and What Assist Each Other Through Top-down Connections," *Proc. 7th International Conference on Development and Learning (ICDL'08)*, Monterey, CA, Aug. 9-12, pp. 1-6, 2008.

[62] X. Wu, G. Guo, and J. Weng, "Skull-closed Autonomous Development: WWN-7 Dealing with Scales," *Proc. International Conference on Brain-Mind*, July 27–28, East Lansing, Michigan, pp. +1-9, 2013.

[63] Szegedy, Christian, Alexander Toshev, and Dumitru Erhan. "Deep neural networks for object detection." Advances in Neural Information Processing Systems. 2013.

[64] T. Mikolov *et al.*, "Recurrent neural network based language model," *Interspeech*, 2010.

[65] LeCun, Y. et al. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE* **86** (11): 2278–2324. doi:10.1109/5.726791.

[66] G. E. Hinton *et al.*., "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, pp. 82–97, November 2012.

[67] Y. Bengio *et al.*., "Advances in optimizing recurrent networks," *ICASSP*, 2013.

[68] G. Dahl *et al.*., "Improving DNNs for LVCSR using rectified linear units and dropout," *ICASSP*, 2013.

[69] G. E. Hinton., "A Practical Guide to Training Restricted Boltzmann Machines," *Tech. Rep. UTML TR 2010-003, Dept. CS., Univ. of Toronto*, 2010.

[70] Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: theory and applications." Neurocomputing 70.1 (2006): 489-501.

[71] Widrow, Bernard, et al. "The no-prop algorithm: A new learning algorithm for multilayer neural networks." Neural Networks 37 (2013): 182-188.

[72] Aleksander, Igor, et al. "A brief introduction to Weightless Neural Systems." ESANN. 2009.

[73] Hinton, G.E. "Deep belief networks". *Scholarpedia* **4** (5): 5947. doi:10.4249/scholarpedia.5947.

[74] H. Larochelle *et al.*., "An empirical evaluation of deep architectures on problems with many factors of variation," *in Proc. 24th Int. Conf. Machine Learning*, pp. 473–480, 2007.

[75] G. E. Hinton., "Training Product of Experts by Minimizing Contrastive Divergence," *Neural Computation*, 14, pp. 1771–1800, 2002.

[76] A. Fischer and C. Igel. Training Restricted Boltzmann Machines: An Introduction. Pattern Recognition 47, pp. 25-39, 2014

[77] http://ufldl.stanford.edu/tutorial/index.php/Convolutional_Neural_Network

[78]

[79]

[80] Hinton, Geoffrey; Salakhutdinov, Ruslan (2012). "A better way to pretrain deep Boltzmann machines" (PDF). *Advances in Neural* **3**: 1–9.

[81] Hinton, Geoffrey; Salakhutdinov, Ruslan (2009). "Efficient Learning of Deep Boltzmann Machines" (PDF) **3**. pp. 448–455.

[82] Bengio, Yoshua; LeCun, Yann (2007). "Scaling Learning Algorithms towards AI" (PDF) **1**. pp. 1–41.

[83] Larochelle, Hugo; Salakhutdinov, Ruslan (2010). "Efficient Learning of Deep Boltzmann Machines" (PDF). pp. 693–700.

[84] Vincent, Pascal; Larochelle, Hugo; Lajoie, Isabelle; Bengio, Yoshua; Manzagol, Pierre-Antoine (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". *The Journal of Machine Learning Research* **11**: 3371–3408.

[85] Deng, Li; Yu, Dong (2011). "Deep Convex Net: A Scalable Architecture for Speech Pattern Classification" (PDF). *Proceedings of the Interspeech*: 2285–2288.

[86] Deng, Li; Yu, Dong; Platt, John (2012). "Scalable stacking and learning for building deep architectures". *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*: 2133–2136.

[87] David, Wolpert (1992). "Stacked generalization". *Neural networks* **5(2)**: 241–259. doi:10.1016/S0893-6080(05)80023-1.

[88] Bengio, Yoshua (2009). "Learning deep architectures for AI". *Foundations and trends in Machine Learning* **2(1)**: 1–127.

[89] Hutchinson, Brian; Deng, Li; Yu, Dong (2012). "Tensor deep stacking networks". *IEEE transactions on pattern analysis and machine intelligence* **1–15**.

[90] Hinton, Geoffrey; Salakhutdinov, Ruslan (2006). "Reducing the Dimensionality of Data with Neural Networks". *Science* **313**: 504–507. doi:10.1126/science.1127647. PMID 16873662.

[91] Dahl, G.; Yu, D.; Deng, L.; Acero, A. (2012). "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition". *Audio, Speech, and ...* 20(1): 30–42.

[92] Mohamed, Abdel-rahman; Dahl, George; Hinton, Geoffrey (2012). "Acoustic Modeling Using Deep Belief Networks". *IEEE Transactions on Audio, Speech, and Language Processing*. 20(1): 14–22.

[93] Courville, Aaron; Bergstra, James; Bengio, Yoshua (2011). "A Spike and Slab Restricted Boltzmann Machine" (PDF). *International . . .* **15**: 233–241.

[94] Mitchell, T; Beauchamp, J (1988). "Bayesian Variable Selection in Linear Regression". *Journal of the American Statistical Association*. 83 (404): 1023–1032. doi:10.1080/01621459.1988.10478694.

[95] Courville, Aaron; Bergstra, James; Bengio, Yoshua (2011). "Unsupervised Models of Images by Spike-and-Slab RBMs" (PDF). *Proceedings of the . . .* **10**: 1–8.

[96] Hinton, Geoffrey; Osindero, Simon; Teh, Yee-Whye (2006). "A Fast Learning Algorithm for Deep Belief Nets". *Neural Computation* **1554**: 1527–1554.

[97] Larochelle, Hugo; Bengio, Yoshua; Louradour, Jerdme; Lamblin, Pascal (2009). "Exploring Strategies for Training Deep Neural Networks". *The Journal of Machine Learning Research* **10**: 1–40.

[98] Coates, Adam; Carpenter, Blake (2011). "Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning". pp. 440–445.

[99] Lee, Honglak; Grosse, Roger (2009). "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations". *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*: 1–8.

[100] Lin, Yuanqing; Zhang, Tong (2010). "Deep Coding Network" (PDF). *Advances in Neural . . .*: 1–9.

[101] Ranzato, Marc Aurelio; Boureau, Y-Lan (2007). "Sparse Feature Learning for Deep Belief Networks" (PDF). *Advances in neural information . . .*: 1–8.

[102] Socher, Richard; Lin, Clif (2011). "Parsing Natural Scenes and Natural Language with Recursive Neural Networks" (PDF). *Proceedings of the . . .*

[103] Taylor, Graham; Hinton, Geoffrey (2006). "Modeling Human Motion Using Binary Latent Variables" (PDF). *Advances in neural . . .*

[104] Vincent, Pascal; Larochelle, Hugo (2008). "Extracting and composing robust features with denoising autoencoders". *Proceedings of the 25th international conference on Machine learning - ICML '08*: 1096–1103.

[105] Kemp, Charles; Perfors, Amy; Tenenbaum, Joshua (2007). "Learning overhypotheses with hierarchical Bayesian models". *Developmental science*. 10(3): 307–21. doi:10.1111/j.1467-7687.2007.00585.x. PMID 17444972.

[106] Xu, Fei; Tenenbaum, Joshua (2007). "Word learning as Bayesian inference". *Psychol Rev*. 114(2): 245–72. doi:10.1037/0033-295X.114.2.245. PMID 17500627.

[107] Chen, Bo; Polatkan, Gungor (2011). "The Hierarchical Beta Process for Convolutional Factor Analysis and Deep Learning" (PDF). *Machine Learning . . .*

[108] Fei-Fei, Li; Fergus, Rob (2006). "One-shot learning of object categories". *IEEE Trans Pattern Anal Mach Intell*. 28(4): 594–611. doi:10.1109/TPAMI.2006.79. PMID 16566508.

[109] Rodriguez, Abel; Dunson, David (2008). "The Nested Dirichlet Process". *Journal of the American Statistical Association*. 103(483): 1131–1154. doi:10.1198/016214508000000553.

[110] Ruslan, Salakhutdinov; Joshua, Tenenbaum (2012). "Learning with Hierarchical-Deep Models". *IEEE transactions on pattern analysis and machine intelligence*: 1–14. PMID 23267196.

[111] Chalasani, Rakesh; Principe, Jose (2013). "Deep Predictive Coding Networks". *arXiv preprint arXiv*: 1–13.

[112] Cho, Youngmin (2012). "Kernel Methods for Deep Learning" (PDF). pp. 1–9.

[113] Scholkopf, B; Smola, Alexander (1998). "Nonlinear component analysis as a kernel eigenvalue problem". *Neural computation* **(44)**.

[114] L. Deng, G. Tur, X. He, and D. Hakkani-Tur. "Use of Kernel Deep Convex Networks and End-To-End Learning for Spoken Language Understanding," *Proc. IEEE Workshop on Spoken Language Technologies*, 2012

[115] Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning" (PDF) **518**. pp. 529–533.

[116] Hinton, Geoffrey E. "Distributed representations." (1984)

[117] S. Das, C.L. Giles, G.Z. Sun, “Learning Context Free Grammars: Limitations of a Recurrent Neural Network with an External Stack Memory,” Proc. 14th Annual Conf. of the Cog. Sci. Soc., p. 79, 1992.

[118] Mozer, M. C., & Das, S. (1993). A connectionist symbol manipulator that discovers the structure of context-free languages. NIPS 5 (pp. 863-870).

[119] J. Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. Neural Computation, 4(1):131-139, 1992

[120] F. Gers, N. Schraudolph, J. Schmidhuber. Learning precise timing with LSTM recurrent networks. JMLR 3:115-143, 2002.

[121] J. Schmidhuber. An introspective network that can learn to run its own weight change algorithm. In Proc. of the Intl. Conf. on Artificial Neural Networks, Brighton, pages 191-195. IEE, 1993.

[122] Hochreiter, Sepp; Younger, A. Steven; Conwell, Peter R. (2001). “Learning to Learn Using Gradient Descent”. ICANN 2001, 2130: 87–94.

[123] Salakhutdinov, Ruslan, and Geoffrey Hinton. “Semantic hashing.” International Journal of Approximate Reasoning 50.7 (2009): 969-978.

[124] Le, Quoc V., and Tomas Mikolov. “Distributed representations of sentences and documents.” arXiv preprint arXiv:1405.4053 (2014).

[125] Graves, Alex, Greg Wayne, and Ivo Danihelka. “Neural Turing Machines.” arXiv preprint arXiv:1410.5401 (2014).

[126] Weston, Jason, Sumit Chopra, and Antoine Bordes. “Memory networks.” arXiv preprint arXiv:1410.3916 (2014).

[127] *TIMIT Acoustic-Phonetic Continuous Speech Corpus* Linguistic Data Consortium, Philadelphia.

[128] Abdel-Hamid, O. et al. (2014). “Convolutional Neural Networks for Speech Recognition”. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **22** (10): 1533–1545. doi:10.1109/taslp.2014.2339736.

[129] Deng, L.; Platt, J. (2014). “Ensemble Deep Learning for Speech Recognition”. *Proc. Interspeech*.

[130] Yu, D.; Deng, L. (2010). “Roles of Pre-Training and Fine-Tuning in Context-Dependent DBN-HMMs for Real-World Speech Recognition”. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

[131] Deng L., Li, J., Huang, J., Yao, K., Yu, D., Seide, F. et al. Recent Advances in Deep Learning for Speech Research at Microsoft. ICASSP, 2013.

[132] Deng, L.; Li, Xiao (2013). “Machine Learning Paradigms for Speech Recognition: An Overview”. *IEEE Transactions on Audio, Speech, and Language Processing*.

[133] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. Hinton (2010) Binary Coding of Speech Spectrograms Using a Deep Auto-encoder. Interspeech.

[134] Z. Tuske, P. Golik, R. Schlüter and H. Ney (2014). Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR. Interspeech.

[135] McMillan, R. “How Skype Used AI to Build Its Amazing New Language Translator”, Wire, Dec. 2014.

[136] Hannun et al. (2014) “Deep Speech: Scaling up end-to-end speech recognition”, arXiv:1412.5567.

[137] Ron Schneiderman (2015) “Accuracy, Apps Advance Speech Recognition --- Interview with Vlad Sejnoha and Li Deng”, IEEE Signal Processing Magazine, Jan, 2015.

[138] http://yann.lecun.com/exdb/mnist/.

[139] D. Ciresan, U. Meier, J. Schmidhuber., “Multi-column Deep Neural Networks for Image Classification,” *Technical Report No. IDSIA-04-12', 2012.*

[140] Vinyals et al. (2014)."Show and Tell: A Neural Image Caption Generator,” arXiv:1411.4555.

[141] Fang et al. (2014)."From Captions to Visual Concepts and Back,” arXiv:1411.4952.

[142] Kiros et al. (2014)."Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models,” arXiv: 1411.2539.

[143] Zhong, S.; Liu, Y.; Liu, Y. “Bilinear Deep Learning for Image Classification”. *Proceedings of the 19th ACM International Conference on Multimedia* **11**: 343–352.

[144] Nvidia Demos a Car Computer Trained with “Deep Learning” (2015-01-06), David Talbot, *MIT Technology Review*

[145] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin., “A Neural Probabilistic Language Model,” *Journal of Machine Learning Research 3 (2003) 1137–1155', 2003.*

[146] Goldberg, Yoav; Levy, Omar. “word2vec Explained: Deriving Mikolov et al.’s Negative-Sampling Word-Embedding Method” (PDF). *Arxiv*. Retrieved 26 October 2014.

[147] Socher, Richard; Manning, Christopher. “Deep Learning for NLP” (PDF). Retrieved 26 October 2014.

[148] Socher, Richard; Bauer, John; Manning, Christopher; Ng, Andrew (2013). “Parsing With Compositional Vector Grammars” (PDF). *Proceedings of the ACL 2013 conference*.

[149] Socher, Richard (2013). “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank” (PDF). *EMNLP 2013*.

[150] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil (2014) " A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval,” Proc. CIKM.

[151] P. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck (2013) “Learning Deep Structured Semantic Models for Web Search using Clickthrough Data,” Proc. CIKM.

[152] I. Sutskever, O. Vinyals, Q. Le (2014) “Sequence to Sequence Learning with Neural Networks,” Proc. NIPS.

[153] J. Gao, X. He, W. Yih, and L. Deng(2014) “Learning Continuous Phrase Representations for Translation Modeling,” Proc. ACL.

[154] J. Gao, P. Pantel, M. Gamon, X. He, L. Deng (2014) “Modeling Interestingness with Deep Neural Networks,” Proc. EMNLP.

[155] J. Gao, X. He, L. Deng (2014) “Deep Learning for Natural Language Processing: Theory and Practice (Tutorial),” CIKM.

[156] Arrowsmith, J; Miller, P (2013). “Trial watch: Phase II and phase III attrition rates 2011-2012”. *Nature Reviews Drug Discovery* **12** (8): 569. doi:10.1038/nrd4090. PMID 23903212.

[157] Verbist, B; Klambauer, G; Vervoort, L; Talloen, W; The Qstar, Consortium; Shkedy, Z; Thas, O; Bender, A; Göhlmann, H. W.; Hochreiter, S (2015). “Using transcriptomics to guide lead optimization in drug discovery projects: Lessons learned from the QSTAR project”. *Drug Discovery Today*. doi:10.1016/j.drudis.2014.12.014. PMID 25582842.

[158] “Announcement of the winners of the Merck Molecular Activity Challenge” https://www.kaggle.com/c/MerckActivity/details/winners.

[159] Dahl, G. E.; Jaitly, N.; & Salakhutdinov, R. (2014) “Multi-task Neural Networks for QSAR Predictions,” ArXiv, 2014.

[160] “Toxicology in the 21st century Data Challenge” https://tripod.nih.gov/tox21/challenge/leaderboard.jsp

[161] “NCATS Announces Tox21 Data Challenge Winners” http://www.ncats.nih.gov/news-and-events/features/tox21-challenge-winners.html

[162] Unterthiner, T.; Mayr, A.; Klambauer, G.; Steijaert, M.; Ceulemans, H.; Wegner, J. K.; & Hochreiter, S. (2014) “Deep Learning as an Opportunity in Virtual Screening”. Workshop on Deep Learning and Representation Learning (NIPS2014).

[163] Unterthiner, T.; Mayr, A.; Klambauer, G.; & Hochreiter, S. (2015) “Toxicity Prediction using Deep Learning”. ArXiv, 2015.

[164] Ramsundar, B.; Kearnes, S.; Riley, P.; Webster, D.; Konerding, D.;& Pande, V. (2015) “Massively Multitask Networks for Drug Discovery”. ArXiv, 2015.

[165] Tkachenko, Yegor. Autonomous CRM Control via CLV Approximation with Deep Reinforcement Learning in Discrete and Continuous Action Space. (April 8, 2015). arXiv.org: http://arxiv.org/abs/1504.01840

[166] Utgoff, P. E.; Stracuzzi, D. J. (2002). “Many-layered learning”. *Neural Computation* **14**: 2497–2529. doi:10.1162/08997660260293319.

[167] J. Elman, *et al.*, “Rethinking Innateness,” 1996.

[168] Shrager, J.; Johnson, MH (1996). “Dynamic plasticity influences the emergence of function in a simple cortical array”. *Neural Networks* **9** (7): 1119–1129. doi:10.1016/0893-6080(96)00033-0.

[169] Quartz, SR; Sejnowski, TJ (1997). “The neural basis of cognitive development: A constructivist manifesto”. *Behavioral and Brain Sciences* **20** (4): 537–556. doi:10.1017/s0140525x97001581.

[170] S. Blakeslee., “In brain’s early growth, timetable may be critical,” *The New York Times, Science Section*, pp. B5–B6, 1995.

[171] {BUFILL} E. Bufill, J. Agusti, R. Blesa., “Human neoteny revisited: The case of synaptic plasticity,” *American Journal of Human Biology*, 23 (6), pp. 729–739, 2011.

[172] J. Shrager and M. H. Johnson., “Timing in the development of cortical function: A computational approach,” *In B. Julesz and I. Kovacs (Eds.), Maturational windows and adult cortical plasticity*, 1995.

[173] D. Hernandez., “The Man Behind the Google Brain: Andrew Ng and the Quest for the New AI,” *http://www.wired.com/wiredenterprise/2013/05/neuro-artificial-intelligence/all/. Wired*, 10 May 2013.

[174] C. Metz., “Facebook’s 'Deep Learning' Guru Reveals the Future of AI,” *http://www.wired.com/wiredenterprise/2013/12/facebook-yann-lecun-qa/. Wired*, 12 December 2013.

[175] G. Marcus., “Is “Deep Learning” a Revolution in Artificial Intelligence?" *The New Yorker*, 25 November 2012.

[176] Smith, G. W. (March 27, 2015). “Art and Artificial Intelligence”. ArtEnt. Retrieved March 27, 2015.

[177] Alexander Mordvintsev, Christopher Olah, and Mike Tyka (June 17, 2015). “Inceptionism: Going Deeper into Neural Networks”. Google Research Blog. Retrieved June 20, 2015.

[178] Alex Hern (June 18, 2015). “Yes, androids do dream of electric sheep”. The Guardian. Retrieved June 20, 2015.

[179] Ben Goertzel. Are there Deep Reasons Underlying the Pathologies of Today’s Deep Learning Algorithms? (2015) Url: http://goertzel.org/DeepLearning_v1.pdf

[180] Nguyen, Anh, Jason Yosinski, and Jeff Clune. “Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images.” arXiv preprint arXiv:1412.1897 (2014).

[181] Szegedy, Christian, et al. “Intriguing properties of neural networks.” arXiv preprint arXiv:1312.6199 (2013).

[182] Zhu, S.C.; Mumford, D. “A stochastic grammar of images”. *Found. Trends. Comput. Graph. Vis.* **2** (4): 259–362. doi:10.1561/0600000018.

[183] Miller, G. A., and N. Chomsky. “Pattern conception.” Paper for Conference on pattern detection, University of Michigan. 1957.

[184] Jason Eisner, Deep Learning of Recursive Structure: Grammar Induction, http://techtalks.tv/talks/deep-learning-of-recursive-structure-grammar-induction/58089/

[185] http://singa.incubator.apache.org/

62.12 Text and image sources, contributors, and licenses

62.12.1 Text

Kri, Gareth E Kegg, King of Hearts, Chobot, Theo Pardilla, Garas, Gwernol, Tone, Interested~enwiki, YurikBot, Wavelength, Pspoulin, Sceptre, Cabiria, Jimp, Charles Gaudette, RussBot, Arado, Sillybilly, TheDoober, Piet Delport, SnoopY~enwiki, DanMS, Tsch81, Hydrargyrum, Bill52270, Gaius Cornelius, CambridgeBayWeather, Pseudomonas, Thane, CarlHewitt, LauriO~enwiki, ENeville, Wiki alf, Boneheadmx, Grafen, Tailpig, ImGz, Introgressive, Srinivasasha, Darker Dreams, John Newbury, RabidDeity, Thiseye, Robert McClenon, Nick, Banes, Daniel Mietchen, Jpbowen, Larry laptop, Aldux, Iyerakshay, MarkAb, Tony1, Supten, Syrthiss, Linweizhen, Gadget850, DeadEyeArrow, Gzabers, Dv82matt, Searchme, Richardcavell, Zaklog, Pawyilee, Jvoegele, K.Nevelsteen, Zzuuzz, Nikkimaria, Arthur Rubin, ColinMcMillen, Xaxafrad, Jogers, Saizai, GraemeL, CWenger, HereToHelp, ArielGold, Dublinclontarf, Hamster Sandwich., Allens, Banus, Ásgeir IV.~enwiki, Anniepoo, NeilN, Kingboyk, GrinBot~enwiki, Jonathan Métillon, Bo Jacoby, D Monack, DVD R W, Teo64x, Oldhamlet, SmackBot, Mmernex, Shannonbonannon, Haymaker, Moxon, Mneser, 1dragon, InverseHypercube, Inego~enwiki, Hydrogen Iodide, McGeddon, Martinp, Jared555, Bbewsdirector, Davewild, Pkirlin, CommodiCast, Cacuija, Vilerage, Took, Edgar181, Alsandro, Moralis, W3bj3d1, Gilliam, Betacommand, Rpmorrow, ERcheck, Grokmoo, Mirokado, Chris the speller, Shaggorama, NCurse, Thumperward, Stevage, J. Spencer, DHN-bot~enwiki, Darth Panda, Zsinj, Pegua, Can't sleep, clown will eat me, Erayman64, Onorem, MJBurrage, JonHarder, EvelinaB, Nunocordeiro, RedHillian, Edivorce, Normxxx, Memming, COMPFUNK2, Freek Verkerk, PrometheusX303, Jaibe, Shadow1, Richard001, Taggart Transcontinental, NapoleonB, Freemarket, Lacatosias, StephenReed, Jon Awbrey, Metamagician3000, Salamurai, LeoNomis, Curly Turkey, Ck lostsword, Pilotguy, Kukini, Ohconfucius, Dankonikolic, William conway bcc, Nishkid64, Baby16, Harryboyles, Titus III, John, Jonrgrover, AmiDaniel, Geoinline, Yan Kuligin, Madhukaleeckal, Tazmaniacs, Gobonobo, Disavian, NewTestLeper79, ChaoticLogic, Chessmaniac, Goodnightmush, Ripe, Mr. Vernon, Newkidd11, Special-T, WMod-NS, Owlbuster, Mr Stephen, Dicklyon, Emurph, Daphne A, Dee Jay Randall, AdultSwim, Condem, Glen Pepicelli, Zapvet, Digsdirt, Nabeth, Peyre, Caiaffa, Andreworkney, Hu12, Burto88, Levineps, Vespine, Mjgilson, Clarityfiend, Sierkejd, Joseph Solis in Australia, Ralf Klinkenberg, Snoutholder, Igoldste, Rnb, Wfructose, Marysunshine, Dream land2080, Tawkerbot2, Dlohcierekim, GerryWolff, Eastlaw, Myncknm, 8754865, Rahuldharmani, CRGreathouse, CmdrObot, Eggman64, Wafulz, Makeemlighter, Discordant~enwiki, Mudd1, Ruslik0, El aprendelenguas, ShelfSkewed, Pgr94, MeekMark, Casper2k3, Fordmadoxfraud, Myasuda, Livingston7, Abdullahazzam, Gregbard, Shanoman, Cydebot, Peripitus, Abeg92, Besieged, Beta Trom, Peterdjones, Kaldosh, Gogo Dodo, Corpx, Ivant, Rzwitserloot, JamesLucas, Julian Mendez, Josephorourke, LaserBeams, WikiNing, Robertinventor, Bubba2323, Bryan Seecrets, Bitsmart, Gimmetrow, PamD, Michael Fourman, Thijs!bot, Epbr123, Mercury~enwiki, ConceptExp, Daniel, Hervegirod, Sagaciousuk, N5iln, Mojo Hand, Atamyrat~enwiki, Marek69, Zzthex, GTof~enwiki, Mailseth, I already forgot, David D., Mschures, Mac-steele, AntiVandalBot, Tabortiger, Luna Santin, Seaphoto, Opelio, Prolog, 17Drew, Nosirrom, Jj137, Ykalyan, TurntableX, Science History, Theropod-X, Spinningobo, Myanw, Steelpillow, MikeLynch, JAnDbot, Jimothytrotter, MattBan, MTuffield, Barek, Sarnholm, MER-C, CosineKitty, Mrsolutions, Schlegel, The Transhumanist, Davespice, 100110100, Greensburger, TAnthony, .anacondabot, Acroterion, Wasell, Coffee2theorems, Magioladitis, Bongwarrior, VoABot II, Cobratom, Farquaadhnchmn, Gamkiller, Redaktor, Tedickey, Johannes Simon, Tremilux, Rootxploit, Cic, Crunchy Numbers, Nposs, BatteryIncluded, Robotman1974, ArthurWeasley, Johnkoza1992, Jroudh, Allstarecho, Adamreiswig, Wookiee cheese, Joydurgin, Siddharthsk2000, Talon Artaine, DerHexer, Tommy Herbert, TheRanger, Francob, MartinBot, IanElmore, Bradgib, ARC Gritt, Naohiro19, Rettetast, Bissinger, Mschel, Kostisl, CommonsDelinker, Amareshjoshi, KTo288, Lilac Soul, LedgendGamer, Mausy5043, Pacas, J.delanoy, JuniorMonkey, Trusilver, Anandcv, Gustavo1255, Richiekim, Jiuguang Wang, Uncle Dick, Public Menace, Yonidebot, Laurusnobilis, Jkaplan, Dispenser, Eric Mathews Technology, DarkGhost08, Ignatzmice, Touisiau, Tdewey, Pogowitwiz, Mikael Häggström, Ypetrachenko, Tarotcards, Topazxx, ColinClark, AntiSpamBot, Zubenelgenubi, Tobias Kuhn, Lewblack, The Transhumanist (AWB), Hthth, Bobianite, Jorfer, Sbennett4, Sunderland06, Chuckino, Wesleymurphy, Burzmali, Remember the dot, Walter Fritz, Riccardopoli, Andy Marchbanks, Inwind, Useight, Leontolstoy2, Halmstad, RJASE1, Funandtrvl, Idarin, VolkovBot, Paranoid600, Stephen G Graham, Jeff G., Maghnus, Gogarburn, Philip Trueman, Kramlovesmovies, Sweetness46, TXiKiBoT, Coder Dan, Vrossi1, Master Jaken, Java7837, Alan Rockefeller, Technopat, Olinga, Lordvolton, BarryList, Una Smith, Sirkad, DonutGuy, Eroark, Seb az86556, Cremepuff222, TheSix, Jeeny, The Divine Fluffalizer, Bugone, CO, Eubulides, Kurowoofwoof111, Andy Dingley, Dirkbb, Haseo9999, Graymornings, Falcon8765, MCTales, Cvdwalt, Harderm, Dmcq, LittleBenW, Pjoef, AlleborgoBot, Symane, Szeldich, Darthali, Iceworks, SieBot, RHodnett, Frans Fowler, JamesA, Tiddly Tom, WereSpielChequers, Paradoctor, Blackshadow153, Viskonsas, WBTtheFROG, Yintan, Soler97, Grundle2600, Purbo T, DavidBourguignon, Yungjibbz, Mr.Z-bot, Wadeduck, Flyer22, Darth Chyrsaor, Jojalozzo, Taemyr, Arthur Smart, Oxymoron83, Ioverka, Scorpion451, Ddxc, Harry~enwiki, MiNombreDeGuerra, Steven Zhang, Lightmouse, Yankee-Bravo, Xobritbabeox10, RyanParis, Svick, Smallclone2, CharlesGillingham, Macduffman, Wyckster, Voices cray, Searchmaven, Cheesefondue, Neo., Denisarona, Finetooth, Emptymountains, ImageRemovalBot, Hpdl, Elassint, ClueBot, Fribbler, Rumping, Prohlep, Avenged Eightfold, The Thing That Should Not Be, Logperson, Ezavarei, Ndenison, Unbuttered Parsnip, Willingandable, Grantbow, Drmies, VQuakr, Mild Bill Hiccup, Malignedtruth, Boing! said Zebedee, Mayfly may fly, Nappy1020, Arne Heise, Neverquick, Thomas Kist, Yakota21, Pooya.babahajyani, Time for action, Jdzarlino, Excirial, Gnome de plume, Hezarfenn, Vivio Testarossa, Thunderhippo, Nayanraut, Xklsv, NuclearWarfare, Jotterbot, Iohannes Animosus, Sebaldusadam, Renamed user 3, Mr.Sisgay, Mullhawk, Chaosdruid, Aitias, EPIC MASTER, Ranjithsutari, Scalhotrod, Johnuniq, HumphreyW, Apparition11, Sparkygravity, MasterOfHisOwnDomain, Escientist, XLinkBot, AgnosticPreachersKid, Basploeger, Sgunteratparamus.k12.nj.us, Spitfire, Pichpich, Alex naish, Rror, Ost316, Libcub, Dheer7c, Noctibus, Thede, Truthnlove, D.M. from Ukraine, Addbot, Barsoomian, DOI bot, Guoguo12, Kimveale, Betterusername, Ashish krazzy, Coolcatfish, AndrewHZ, DougsTech, Elsendero, TutterMouse, CanadianLinuxUser, Leszek Jańczuk, Fluffernutter, Damiens.rf, Me pras, Sebastian scha., MrOllie, Aykantspel, LaaknorBot, Nuclear Treason, The world deserves the truth, Glane23, AndersBot, Favonian, Kyle1278, LinkFA-Bot, Rbalcke, AgadaUrbanit, Tassedethe, Brianjfox, Jan eissfeldt, Ricvelozo, Jarble, KarenEdda, Ben Ben, Luckas-bot, PetroKonashevich, Yobot, Worldbruce, Themfromspace, Ptbotgourou, Fraggle81, Twexcom, Obscuranym, Leoneris, Rinea, Mmxx, Taxisfolder, THEN WHO WAS PHONE?, Vini 17bot5, MHLU, PluniAlmoni, Pravdaverita, AnomieBOT, Macilnar, Floquenbeam, 1exec1, Ai24081983, Frans-w1, Jim1138, IRP, Galoubet, Fraziergeorge122, Palace of the Purple Emperor, AdjustShift, ChristopheS, Glenfarclas, Flewis, Materialscientist, Citation bot, Kjellmikal, Devantheryv, Vivohobson, Eumolpo, Ankank, ArthurBot, Xqbot, TinucherianBot II, AVBAI, Roesslerj, Capricorn42, The Magnificent Clean-keeper, Staberind, Renaissancee, Taylormas229, Oxwil, Wyklety, The Evil IP address, Turk oğlan, Isheden, S0aasdf2sf, Crzer07, A157247, Hi878, IntellectToday, J04n, GrouchoBot, Armbrust, Wizardist, Diogeneselcinico42, Omnipaedista, Shirik, RibotBOT, Chris.urs-o, Mathonius, Shadowjams, Chicarelli, WhatisFeelings?, AlGreen00, Dougofborg, Fillepa, FrescoBot, FalconL, Hobsonlane, Recognizance, Nojan, Ilcmuchas, Zero Thrust, HJ Mitchell, Steve Quinn, Yoyosocool, Lightbound, Juno, Spectatorbot13, Intrealm, Philapathy, HamburgerRadio, Citation bot 1, DeStilaDo, RCPayne, Lylodo, MacMed, Pinethicket, WaveRunner85, Jonesey95, Rameshngbot, Skyerise, Staflorin, Agemoi, Farmer21, Bubwater, Fartherred, Reconsider the static, IJBall, Cnwilliams, Bqdemon, JCAILLAT, TobeBot, Puzl bustr, SchreyP, Compvis, LogAntiLog, ItsZippy, Lotje, Javierito92, Emarus, Fox Wilson, Vrenator, Lynn Wilbur, Robot8888, Gregman2, Fricanod, Stroppolo, Wikireviewer99, BrightBlackHeaven, DARTH SIDIOUS 2, Chucks91, Mean as custard, RjwilmsiBot, Humanrobo, Иъ Лю Ха, Theyer, BertSeghers, Rollins83, Nistra, DASHBot, EmausBot, John of Reading, Orphan Wiki, WikitanvirBot, Syncategoremata, GoingBatty, Implements, Wikicolleen, Olof nord, Gimmetoo, Tommy2010, K6ka, Azlan Iqbal, Thecheesykid, Werieth, Evanh2008, AvicBot, Vfrias, AVGavrilov, Josve05a, Shuipzv3, Lateg, Annonnimus, Habstinat, Unreal7, Foryourinfo, Tolly4bolly, Thine Antique

Pen, Staszek Lem, Δ, Coasterlover1994, Westwood25, L Kensington, Casia wyq, Rangoon11, Keidax, Lilinjing cas, Terraflorin, JonRichfield, ClueBot NG, Xzbobzx, Tillander, ClaretAsh, Astrobase36, MelbourneStar, Satellizer, Mostlymark, James childs, Robiminer, Tideflat, Aloisdimpflmoser, Prakashavinash, Delusion23, Widr, Danim, Heldrain, Helpful Pixie Bot, Adam Roush, SecretStory, Whitehead3, IrishStephen, Calabe1992, Jeraphine Gryphon, Technical 13, BG19bot, Virtualerian, Jwchong, Starship.paint, Juro2351, Mr.TAMER.Shlash, MangoWong, ElphiBot, MusikAnimal, Josvebot, Piguy101, Treva26, Reckiz, Cadiomals, Drift chambers, Altaïr, Cassianp, Wiki-uoft, Snow Blizzard, TheProfessor, NotWith, Mschuldt, Glacialfox, Peter Baum, Devonmann101, Avery.mabojie, Wannabemodel, Maarten Rail, BattyBot, StarryGrandma, Simbara, E prosser, Varagrawal, Cyberbot II, Ideafarmcity, Brookeweiner, Fireboy3015, Psych2012Joordens, Marktindal, IjonTichyIjonTichy, RichardKPSun, Dexbot, Oritnk, Acrognale, Lugia2453, Sly1993, Iskand26, SFK2, Dineshhi, Yassu66, Henrifrischborno, Reatlas, Samee, Joeinwiki, Faizan, ChassonR, Epicgenius, Newtesteditor, Allaroundtheworld123, Amandapan1, Snatoon, Sonĝanto, Melonkelon, Kingfaj, Silas Ropac, Weightlessjapan1337, Satassi, Jiayuliu, Tentinator, Rosenblumb1, Kogmaw, Syntaxerrorz, Msundqvist, Shrikarsan, Dustin V. S., Mamadoutadioukone, Laper1900, Gaurang2904, Riddhi-gupta, Babitaarora, Megerler, IWishIHadALife, Orion dude, Mrm7171, MarinMersenne, □□□□, Aniruddhmanishpatel, Mandruss, Ginsuloft, Mintiematt, Aubreybardo, Francois-Pier, Healybd, Cows12345, Zourcilleux, Fixuture, Mellon2030, Lakun.patra, Balaji4894, FelixRosch, Wyn.junior, JacwanaKL, □□□□, 22merlin, MadScientistX11, Hnchinmaya, Monkbot, Lucyloo10, SantiLak, Jfishel, TheQ Editor, YdJ, T lanuza, BioMeow, Madan Pokharel, Pbravo2, Dodynamic, Evolutionvisions, Wstevens2090, Zeus000000, Charmingpianist, ChamithN, Feynman1918, Shrestha sumit, Yoon Aris, Dorianluparu, Junbibi, Pythagoros, SpRu01, Comp-heur-intel, Praneshyadav1, Kmp302, Devwebtel, KayleighwilliamsLS, Easeit7, Illiteration, Spectrum629, Miguelsnchz723, Newwikieditor678, KenTancwell, Demdim0, Eobasanya, KasparBot, Pianophage, Csisawesome, Pmaiden, Συντάκτης Βικιλεξικό, Ray engh 302, Skyshines, Andyjin2002, Aviartm and Anonymous: 1608

- **Information theory** *Source:* https://en.wikipedia.org/wiki/Information_theory?oldid=668259571 *Contributors:* AxelBoldt, Brion VIBBER, Timo Honkasalo, Ap, Graham Chapman, XJaM, Toby Bartels, Hannes Hirzel, Edward, D, PhilipMW, Michael Hardy, Isomorphic, Kku, Bobby D. Bryant, Varunrebel, Vinodmp, AlexanderMalmberg, Karada, (, Iluvcapra, Minesweeper, StephanWehner, Ahoerstemeier, Angela, LittleDan, Kevin Baas, Poor Yorick, Andres, Novum, Charles Matthews, Guaka, Bemoeial, Ww, Dysprosia, The Anomebot, Munford, Hyacinth, Ann O'nyme, Shizhao, AnonMoos, MH~enwiki, Robbot, Fredrik, Rvollmert, Seglea, Chancemill, Securiger, SC, Lupo, Wile E. Heresiarch, ManuelGR, Ancheta Wis, Giftlite, Lee J Haywood, COMPATT, KuniShiro~enwiki, SoWhy, Andycjp, Ynh~enwiki, Pcarbonn, OverlordQ, L353a1, Rdsmith4, APH, Cihan, Elektron, Creidieki, Neuromancien, Rakesh kumar, Picapica, D6, Jwdietrich2, Imroy, CALR, Noisy, Rich Farmbrough, Guanabot, NeuronExMachina, Ericamick, Xezbeth, MisterSheik, Crunchy Frog, El C, Spoon!, Simon South, Bobo192, Smalljim, Rbj, Maurreen, Nothingmuch, Photonique, Andrewbadr, Haham hanuka, Pearle, Mpeisenbr, Mdd, Msh210, Uncle Bill, Pouya, BryanD, PAR, Cburnett, Jheald, Geraldshields11, Kusma, DV8 2XL, Oleg Alexandrov, FrancisTyers, Velho, Woohookitty, Linas, Mindmatrix, Ruud Koot, Eatsaq, Eyreland, SeventyThree, Kanenas, Graham87, Josh Parris, Mayumashu, SudoMonas, Arunkumar, HappyCamper, Alejo2083, Chris Pressey, Mathbot, Annacoder, Nabarry, Srleffler, Chobot, Commander Nemet, FrankTobia, Siddhant, YurikBot, Wavelength, RobotE, RussBot, Michael Slone, Loom91, Grubber, ML, Yahya Abdal-Aziz, Raven4x4x, Moe Epsilon, DanBri, Allchopin, Light current, Mceliece, Arthur Rubin, Lyrl, GrinBot~enwiki, Sardanaphalus, Lordspaz, SmackBot, Imz, Henri de Solages, Incnis Mrsi, Reedy, InverseHypercube, Cazort, Gilliam, Metacomet, Octahedron80, Spellchecker, Colonies Chris, Jahiegel, Unnikrishnan.am, LouScheffer, Calbaer, EPM, Djcmackay, Michael Ross, Tyrson, Jon Awbrey, Het, Bidabadi~enwiki, Chungc, SashatoBot, Nick Green, Harryboyles, Sina2, Lachico, Almkglor, Bushsf, Sir Nicholas de Mimsy-Porpington, FreezBee, Dicklyon, E-Kartoffel, Wizard191, Matthew Verey, Isvish, ScottHolden, CapitalR, Gnome (Bot), Tawkerbot2, Marty39, Daggerstab, CRGreathouse, Thermochap, Ale jrb, Thomas Keyes, Mct mht, Pulkitgrover, Grgarza, Maria Vargas, Roman Cheplyaka, Hpalaiya, Vanished User jdksfajlasd, Nearfar, Heidijane, Thijs!bot, N5iln, WikiIT, James086, PoulyM, Edchi, D.H, Jvstone, HSRT, JAnDbot, BenjaminGittins, RainbowCrane, Jthomp4338, Buettcher, MetsBot, David Eppstein, Pax:Vobiscum, MartinBot, Tamer ih~enwiki, Sigmundg, Jargon777, Policron, Useight, VolkovBot, Joeoettinger, JohnBlackburne, Jimmaths, Constant314, Starrymessenger, Kjells, Magmi, AllGloryToTheHypnotoad, Bemba, Radagast3, Newbyguesses, SieBot, Ivan Štambuk, Robert Loring, Masgatotkaca, Pcontrop, Algorithms, Anchor Link Bot, Melcombe, ClueBot, Fleem, Ammarsakaji, Estirabot, 7&6=thirteen, Oldrubbie, Vegetator, Singularity42, Dziewa, Lambtron, Johnuniq, SoxBot III, HumphreyW, Vanished user uih38riiw4hjlsd, Mitch Ames, Addbot, Deepmath, Peerc, Eweinber, Sun Ladder, C9900, L.exsteens, Xlasne, Egoistorms, Luckas-bot, Quadrescence, Yobot, TaBOT-zerem, Taxisfolder, Carleas, Twohoos, Cassandra Cathcart, Dbln, Materialscientist, Informationtheory, Jingluolaodao, Expooz, Raysonik, Xqbot, Isheden, Informationtricks, Dani.gomezdp, PHansen, Masrudin, FrescoBot, Nageh, Tiramisoo, Sanpitch, Gnomehacker, Pinethicket, Momergil, SkyMachine, SchreyP, Lotje, Miracle Pen, Vanadiumho, Kastchei, Djjr, EmausBot, WikitanvirBot, Jmencisom, Bethnim, Quondum, Henriqueroscoe, Terra Novus, ClueBot NG, MelbourneStar, BarrelProof, TimeOfDei, Frietjes, Thepigdog, Pzrq, MrJosiahT, Lawsonstu, Helpful Pixie Bot, Leopd, BG19bot, Wiki13, Trevayne08, CitationCleanerBot, Brad7777, Schafer510, BattyBot, Bankmichael1, SFK2, Jochen Burghardt, Limit-theorem, Szzoli, 314Username, Roastliras, Eigentensor, Comp.arch, SakeUPenn, Logan.dunbar, Prof. Michael Bank, DanBalance, JellydPuppy, KasparBot and Anonymous: 262
- **Computational science** *Source:* https://en.wikipedia.org/wiki/Computational_science?oldid=668832468 *Contributors:* SebastianHelm, Charles Matthews, Fredrik, Mpiper, Ancheta Wis, Behnam, Hugh Mason, Alsocal, Discospinster, Cjb88, Andrewwall, Mykej, Wayne Schroeder, Oleg Alexandrov, Ruud Koot, Vegaswikian, Meawoppl, Raikkonen, JJL, SmackBot, Peter Sloot, Chris the speller, Janm67, LuchoX, Beetstra, Rhebus, Hu12, Haus, CmdrObot, Pgr94, Aajaja, Mentifisto, VoABot II, Jwagnerhki, Mrseacow, SharkD, Antony-22, JonMcLoone, Arithma, Sargursrihari, Lordvolton, Mcsmom, Douglas256, Kayvan45622, ClueBot, Lgstarn, Eurekaman, Tomtzigt, Qwfp, Zoidbergmd, T68492, Nicoguaro, Superzoulou, Addbot, Wordsoup, Ekojekoj, Jarble, سعی, Yobot, Caracho, AnomieBOT, Materialscientist, P99am, Isheden, CES1596, FrescoBot, Kiefer.Wolfowitz, Gryllida, Willihans, Lotje, Persian knight shiraz, Almarklein, Ouji-fin, Beatnik8983, Super48paul, Vilietha, RockMagnetist, Vincehradil, ClueBot NG, Aiwing, Dslice00, Northamerica1000, Nsda, Anbu121, EricEnfermero, Ema--or, Compsim, Ice666, Shtamy, Schimmdog, MariSo87, Jaime hoyos b, Yakitawa and Anonymous: 58
- **Exploratory data analysis** *Source:* https://en.wikipedia.org/wiki/Exploratory_data_analysis?oldid=670352181 *Contributors:* Michael Hardy, PuzzletChung, Schutz, Aetheling, Astaines, Cutler, Filemon, Giftlite, Jason Quinn, Khalid hassani, Bender235, Elwikipedista~enwiki, Ejrrjs, BlueNovember, Mdd, Oleg Alexandrov, David Haslam, Btyner, Holek, Larman, Rjwilmsi, Chobot, Dppl, YurikBot, Wavelength, Morphh, Closedmouth, Zvika, SmackBot, DCDuring, Jtneill, CommodiCast, Nbarth, Gragus, Sergio.ballestrero, Abmac, Ft93110, Smallpond, Talgalili, JAnDbot, Magioladitis, Jim Douglas, Johnbibby, Rlsheehan, Luc.girardin, Ignacio Icke, STBotD, Botx, Farcaster, Msrasnw, Melcombe, Delphis wk, Linforest, Bea68~enwiki, DragonBot, Ezsdgxrfhcv, Stealth500, BOTarate, JamesXinzhiLi, Qwfp, Tayste, Addbot, Bruce rennes, Kapilghodawat, NjardarBot, Srylesmor, Visnut, Delaszk, Yobot, AlotToLearn, Pablo07, Dpoduval, Citation bot, ArthurBot, GrouchoBot, Omnipaedista, Decisal, Thehelpfulbot, Rdledesma, Boxplot, Kiefer.Wolfowitz, Dmitry St, DixonDBot, Mean as custard, Valeropedro, Macseven, RockMagnetist, Mathstat, WikiMSL, BG19bot, Therealjoefaith, TwoTwoHello, Naabou, Municca, Mkhambaty, Dave Braunschweig, Story645, Fratilias, Alakzi, KasparBot, Olgreenwood and Anonymous: 61
- **Predictive analytics** *Source:* https://en.wikipedia.org/wiki/Predictive_analytics?oldid=669931704 *Contributors:* The Anome, Edward, Michael Hardy, Kku, Karada, Ronz, Avani~enwiki, Giftlite, SWAdair, Thorwald, Bender235, Rubicon, Calair, Rajah, Mdd, Andrew-

pmk, Stephen Turner, Dominic, Oleg Alexandrov, OwenX, HughJorgan, RussBot, Leighblackall, Aeusoes1, Gloumouth1, DeadEyeArrow, EAderhold, Zzuuzz, Talyian, Allens, Zvika, Yvwv, SmackBot, CommodiCast, Mcld, Gilliam, EncMstr, Eudaemonic3, S2rikanth, Onorem, JonHarder, Krexer, Lpgeffen, Doug Bell, Kuru, IronGargoyle, JHunterJ, Ralf Klinkenberg, CmdrObot, Van helsing, Requestion, Pgr94, Myasuda, LeoHeska, Dancter, Talgalili, Scientio, Marek69, JustAGal, Batra, Mr. Darcy, AntiVandalBot, MER-C, Fbooth, VoABot II, Baccyak4H, Bellemichelle, Sweet2, Gregheth, Apdevries, Sudheervaishnav, Ekotkie, Dontdoit, Jfroelich, Trusilver, Dvdpwiki, Ramkumar.krishnan, Atama, Bonadea, BernardZ, Cyricx, GuyRo, Deanabb, Dherman652, Arpabr, Selain03, Hherbert, Cuttysc, Vikx1, Ralftgehrig, Rpm698, MaynardClark, Drakedirect, Chrisguyot, Melcombe, Maralia, Into The Fray, Kai-Hendrik, Ahyeek, Howie Goodell, Sterdeus, SpikeToronto, Stephen Milborrow, Jlamro, Isthisthingworking, SchreiberBike, Bateni, Cookiehead, Qwfp, Angoss, Sunsetsky, Vianello, MystBot, Vaheterdu, BizAnalyst, Addbot, MrOllie, Download, Yobot, SOMart, AnomieBOT, IRP, Nosperantos, Citation bot, Jtamad, BlaineKohl, Phy31277, CorporateM, GESICC, FrescoBot, Boxplot, I dream of horses, Triplestop, Dmitry St, SpaceFlight89, Jackverr, Σ, Peter.borissow, Ethansdad, Cambridgeblue1971, Pamparam, Kmettler, Glenn Maddox, Vrenator, Crysb, DARTH SIDIOUS 2, Onel5969, WikitanvirBot, Sugarfoot1001, Jssgator, Chire, MainFrame, Synecticsgroup, ClueBot NG, Thirdharringtonskier, Stefanomazzalai, Ricekido, Widr, Luke145, Mikeono, Helpful Pixie Bot, WhartonCAI, JonasJSchreiber, Wbm1058, BG19bot, Rafaelgmonteiro, Lisasolomonsalford, TLAN38, Lynnlangit, Flaticida, MrBill3, MC Wapiti, HHinman, BattyBot, Raspabill, Jeremy Kolb, TwoTwoHello, Andrux, Mkhambaty, Cmdima, HeatherMKCampbell, Tommycarney, Tentinator, Sgolestanian, Gbtodd29, Brishtikonna, Thisaccountisbs, Mitchki.nj, Jvn mht, JaconaFrere, Pablodim91, Cbuyle, Monkbot, AmandaJohnson2014, JSHorwitz, Thomas Speidel, HappyVDZ, Wikiperson99, Justincahoon, Femiolajiga, Stevenfinlay, Rlm1188, Bildn, Nivedita1414, Frankecoker, Annaelison, Olosko, Vedanga Kumar, Gary2015, HelpUsStopSpam, Heinrichvk, Rodionos, Olavlaudy and Anonymous: 215

- **Business intelligence** *Source:* https://en.wikipedia.org/wiki/Business_intelligence?oldid=667392522 *Contributors:* Manning Bartlett, Ant, Chuq, Leandrod, Michael Hardy, Norm, Nixdorf, Kku, SebastianHelm, Ellywa, Ronz, Mkoval, Elvis, Mydogategodshat, Jay, Rednblu, Pedant17, Chuckrussell, Traroth, Robbot, ZimZalaBim, Mirv, Aetheling, Lupo, Wile E. Heresiarch, Mattflaschen, Psb777, Ianhowlett, Beardo, AlistairMcMillan, Intergalacticz9, Macrakis, Joelm, Khalid hassani, Alem~enwiki, Edcolins, Golbez, Lucky 6.9, Roc, Alexf, Beland, Bharatcit, Heirpixel, Karl-Henner, Gscshoyru, DMG413, Kadambarid, Guppyfinsoup, KeyStroke, Discospinster, Rhobite, Martpol, S.K., RJHall, Saturnight, Just zis Guy, you know?, Etz Haim, Tjic, Reinyday, John Vandenberg, Maurreen, MPerel, Nsaa, Mdd, Gwalarn, Alansohn, Gary, PaulHanson, Arthena, ABCD, Snowolf, Wtmitchell, Evil Monkey, Sciurinæ, Brookie, Stephen, Zntrip, Dr Gangrene, Woohookitty, Mindmatrix, TigerShark, Camw, Arcann, Jeff3000, GregorB, Liface, Stefanomione, DePiep, Hans Genten, DouglasGreen~enwiki, Ademkader, Slant, Alberrosidus, AndriuZ, ViriiK, M7bot, Danielsmith, Chrisvonsimson, Bgwhite, Wavelength, TexasAndroid, StuffOfInterest, RussBot, AVM, Bhny, DanMS, Manop, Grafen, Welsh, Joel7687, Aaron Brenneman, Muu-karhu, Mikeblas, Zwobot, Pamela Haas, Langbk01, Zzuuzz, Chase me ladies, I'm the Cavalry, Arthur Rubin, Nraden, Guillom, Katieh5584, Tom Morris, Veinor, Drcwright, SmackBot, Schniider~enwiki, McGeddon, MeiStone, Brick Thrower, CommodiCast, Eskimbot, Ohnoitsjamie, Folajimi, Jcarroll, Setti, Chris the speller, Bluebot, Stevage, Swells65, Nick Levine, TheKMan, Xyzzyplugh, Mitrius, Krich, Warren, Yasst8, Ohconfucius, Wikiolap, Eliyak, Kuru, Tomhubbard, Dreamrequest, ElixirTechnology, Beetstra, Ashil04, Frederikton, Blork-mtl, Larrymcp, Waggers, MTSbot~enwiki, Peyre, Aspandyar, Apolitano, AdjustablePliers, OnBeyondZebrax, Lancet75, IvanLanin, Az1568, Nhgaudreau, Codeculturist, HMishkoff, SkyWalker, Racecarradar, CmdrObot, ShelfSkewed, Nmourfield, Cryptblade, Dancter, X0lani, Roberta F., FrancoGG, Thijs!bot, Wernight, Qwyrxian, Czenek~enwiki, PerfectStorm, CharlesHoffman, Batra, QuiteUnusual, Prolog, Charlesmnicholls, Kbeneby, Lfstevens, JAnDbot, Sarnholm, MER-C, Rongou, YK Times, Entgroupzd, Technologyvoices, Supercactus, Magioladitis, VoABot II, Rajashekar iitm, Vanished user ty12kl89jq10, Nposs, Ionium, Peters72, WLU, Halfgoku, Cquels, Iamthenewno2, R'n'B, Gary a mason, Trusilver, Svetovid, DanDoughty, Siryendor, Extransit, A40220, Wxhat1, Sinotara, Srknet, Edit06, Mark Bosley, Naniwako, L'Aquatique, Islamomt, Wendecover, Priyank bolia, WinterSpw, Phani96, Seankenalty, Jeff G., Philip Trueman, TXiKiBoT, Blackstar138, Perohanych, Technopat, Rich Janis, Fredsmith2, Mcclarke, Bansipatel, Andy Dingley, JukoFF, Wikidan829, Ceranthor, Quantpole, Ermite~enwiki, Hazel77, Dwandelt, Moonriddengirl, SEOtools, Julianclark, Android Mouse Bot, Ireas, ObserverToSee, Corp Vision, Janner210, Aadeel, Bcarrdba, Ncw0617, Melcombe, Denisarona, Jvlock527, Ukpremier, Martarius, ClueBot, WriterListener, Natasha81, John ellenberger, Supertouch, Ryan Rasmussen, Chrisawiki, Niceguyedc, Rickybarron, LeoFrank, Srkview, Aexis, SethGrimes, Kit Berg, Jpnofseattle, Mymallandnews, Tompana82, Sierramadrid, DumZiBoT, Man koznar~enwiki, Jmkim dot com, Ejosse1, Writerguy71, Addbot, Butterwell, Mehtasanjay, Wsvlqc, Ronhjones, BlackLips, MrOllie, Glane23, Fauxstar, Lightbot, דוד שי, Pravisurabhi, Luckas-bot, Yobot, Fraggle81, Travis.a.buckingham, Evans1982, Becky Sayles, Coolpriyanka10, IW.HG, Sualfradique, Intelligentknowledgeyoudrive, AnomieBOT, Rubinbot, Jsmith1108, Piano non troppo, Materialscientist, Citation bot, Stationcall, Quebec99, Jehan21, BlaineKohl, Momotoshi, Wperdue, JVRudnick, Prazan, Euthenicsit, Wilcoxaj, RibotBOT, Urchandru, Mathonius, Lovedemon84, Shadowjams, Opagecrtr, Forceblue, Force88, นคเรศ, Mark Renier, Wiki episteme, Glaugh, D'ohBot, Greenboite, Pacific202, Pinethicket, Rayrubenstein, Qqppqqpp, Triplestop, Jim380, Serols, Dnedzel, Jandalhandler, Steelsabre, Ordnascrazy, Hyphen DJW, ITPerforms, Ethansdad, Genuinedifference, Wondigoma, Iaantheron, Ansumang, Crysb, Dr.apostrophe, Goyalaishwarya, Sulhan, Vasant Dhar, Navvod, RjwilmsiBot, Bonanjef, Ananthnats, Rollins83, ITtalker, Helwr, Logical Cowboy, Timtempleton, JEL888, Dewritech, Kellylautt, K6ka, AsceticRose, Jesaisca, Dnazip, Fæ, Jahub, TheWakeUpFactory, Ruislick0, Alpha Quadrant (alt), Eken7, Makecat, Erianna, Tjtyrrell, Openstrings, L Kensington, Yorkshiresoul, Bevelson, Alexandra31, Beroneous, Hanantaiber, Outbackprincess, ClueBot NG, CaveJohnson, AMJBIUser, This lousy T-shirt, Jaej, Qarakesek, Happyinmaine, Robiminer, Mathew105601, Widr, Pmresource, Helpful Pixie Bot, TimMulherin, Bpm123, Kaimu17, BG19bot, Vaulttech, Mr.Gaebrial, Joshua.pierce84, Xjengvyen, Jwcga, Chafe66, Loripiquet, Einsteinlebt, Reverend T. R. Malthus, Meclee, Sutanupaul, Y.Kondrykava, Khazar2, Dhavalp453, Jkofron4, Ivytrejo, Cwobeel, Meg2765, Mogism, Bpmbooks, XXN, Riyadmks, OnTheNet21, Michael.h.zimmerman, Ergoodell, Zkhall, Mangotron, HowardDresner, Mikevandeneijnden, Yanis ahmed, Dkrapohl, Ginsuloft, ReclaGroup, BIcurious3334, DauphineBI, Lakun.patra, Compprof9000, Mgt88drcr, Julep.hawthorne, Tastiera, Marc Schønwandt, TechnoTalk, Wiki-jonne, BrettofMoore, Vanished user 9j34rnfjemnrjnasj4, Generalcontributor, Clumsied, Mihaescu Constantin, Deever21, Xpansa, Galaktikasoft, Frankecoker, Gary2015, BrandonMcBride, Brendonritz, ThatKongregateGuy, Soheilmamani and Anonymous: 692

- **Analytics** *Source:* https://en.wikipedia.org/wiki/Analytics?oldid=670788887 *Contributors:* SimonP, Michael Hardy, Kku, Ronz, Julesd, Charles Matthews, Dysprosia, Kadambarid, Stephenpace, Visviva, Hanswaarle, Jeff3000, GrundyCamellia, Rjwilmsi, Intgr, Srleffler, Rick lightburn, DeadEyeArrow, MagneticFlux, SmackBot, C.Fred, CommodiCast, Ohnoitsjamie, BenAveling, PitOfBabel, Deli nk, Sergio.ballestrero, Wikiolap, Kuru, Ocatecir, 16@r, Beetstra, RichardF, IvanLanin, Hobophobe, Lamiot, Zgemignani, NishithSingh, Gogo Dodo, Barticus88, Brandoneus, QuiteUnusual, TFinn734, Magioladitis, Prabhu137, Elringo, Kimleonard, KylieTastic, Jevansen, VolkovBot, Trevorallred, Jimmaths, Tavix, BarryList, Bansipatel, Rpanigassi, Kerenb, Falcon8765, LittleBenW, Planbhups, Sanya r, Melcombe, Maralia, Aharol, Apptrain, Ottawahitech, Cyberjacob, GDibyendu, Deineka, Addbot, Mortense, Freakmighty, MrOllie, Glorydaze716, Luckas-bot, Yobot, Ptbotgourou, Freikorp, AnomieBOT, HikeBandit, Spugsley, BlaineKohl, Kerberus13, Omnipaedista, Emcien, FrescoBot, James Doehring, Ethansdad, Jonkerz, RjwilmsiBot, TjBot, DASHBot, Timtempleton, Kellylautt, Tmguru, Simplyuttam, Idea Farm, KyleAraujo, Gregory787, Paolo787, ClueBot NG, Networld1965, Helpful Pixie Bot, WhartonCAI, Wbm1058, Mkennedy1981, BG19bot, Pine, Jobin RV, Loripiquet, Analytically, MikeLampaBI, Clkim, Melenc, Cryptodd, TheAdamEvans, Vishal.dani, OnTheNet21, Municca, Dougs campbell, Gmid associates, Makvar, RupertLipton1986, I am One of Many, Edwinboothnyc, JuanCarlosBrandt, Lukebradford, Jenny Rankin, Prussonyc, Rzicari, Ramg iitk, Daph8, Filedelinkerbot, Raj Aradhyula, Aymanmogh, Bildn, Anecdotic, Vidyasnap, Abcdudtc and Anonymous: 110

- **Data mining** *Source:* https://en.wikipedia.org/wiki/Data_mining?oldid=670989267 *Contributors:* Dreamyshade, WojPob, Bryan Derksen, The Anome, Ap, Verloren, Andre Engels, Fcueto, Matusz, Deb, Boleslav Bobcik, Hefaistos, Mswake, N8chz, Michael Hardy, Confusss, Fred Bauder, Isomorphic, Nixdorf, Dhart, Ixfd64, Lament, Alfio, CesarB, Ahoerstemeier, Haakon, Ronz, Angela, Den fjättrade ankan~enwiki, Netsnipe, Jfitzg, Tristanb, Hike395, Mydogategodshat, Dcoetzee, Andrevan, Jay, Fuzheado, WhisperToMe, Epic~enwiki, Tpbradbury, Furrykef, Traroth, Nickshanks, Joy, Shantavira, Pakcw, Robbot, ZimZalaBim, Altenmann, Henrygb, Ojigiri~enwiki, Sunray, Aetheling, Apogr~enwiki, Wile E. Heresiarch, Tobias Bergemann, Filemon, Adam78, Alan Liefting, Giftlite, ShaunMacPherson, Sepreece, Philwelch, Tom harrison, Jkseppan, Simon Lacoste-Julien, Ianhowlett, Varlaam, LarryGilbert, Kainaw, Siroxo, Adam McMaster, Just Another Dan, Neilc, Comatose51, Chowbok, Gadfium, Pgan002, Bolo1729, SarekOfVulcan, Raand, Antandrus, Onco p53, OverlordQ, Gscshoyru, Urhixidur, Kadambarid, Mike Rosoft, Monkeyman, KeyStroke, Rich Farmbrough, Nowozin, Stephenpace, Vitamin b, Bender235, Flyskippy1, Marner, Aaronbrick, Etz Haim, Janna Isabot, Mike Schwartz, John Vandenberg, Maurreen, Ejrrjs, Nsaa, Mdd, Alansohn, Gary, Walter Görlitz, Denoir, Rd232, Jeltz, Jet57, Jamiemac, Malo, Compo, Caesura, Axeman89, Vonaurum, Oleg Alexandrov, Jefgodesky, Nuno Tavares, OwenX, Woohookitty, Mindmatrix, Katyare, TigerShark, LOL, David Haslam, Ralf Mikut, GregorB, Hynespm, Essjay, MarcoTolo, Joerg Kurt Wegner, Simsong, Lovro, Tslocum, Graham87, Deltabeignet, BD2412, Kbdank71, DePiep, CoderGnome, Chenxlee, Sjakkalle, Rjwilmsi, Gmelli, Lavishluau, Michal.burda, Bubba73, Bensin, GeorgeBills, GregAsche, HughJorgan, Twerbrou, FlaBot, Emarsee, AlexAnglin, Ground Zero, Mathbot, Jrtayloriv, Predictor, Bmicomp, Compuneo, Vonkje, Gurubrahma, BMF81, Chobot, DVdm, Bgwhite, The Rambling Man, YurikBot, Wavelength, NTBot~enwiki, H005, Phantomsteve, AVM, Hede2000, Splash, SpuriousQ, Ansell, RadioFan, Hydrargyrum, Gaius Cornelius, Philopedia, Bovineone, Zeno of Elea, EngineerScotty, NawlinWiki, Grafen, ONEder Boy, Mshecket, Aaron Brenneman, Jpbowen, Tony1, Dlyons493, DryaUnda, Bota47, Tlevine, Ripper234, Graciella, Deville, Zzuuzz, Lt-wiki-bot, Fang Aili, Pb30, Modify, GraemeL, Wikiant, JoanneB, LeonardoRob0t, ArielGold, Katieh5584, John Broughton, SkerHawx, Capitalist, Palapa, SmackBot, Looper5920, ThreeDee912, TestPilot, Unyoyega, Cutter, KocjoBot~enwiki, Bhikubhadwa, Thunderboltz, CommodiCast, Comp8956, Delldot, Eskimbot, Slhumph, Onebravemonkey, Ohnoitsjamie, Skizzik, Somewherepurple, Leo505, MK8, Thumperward, DHN-bot~enwiki, Tdelamater, Antonrojo, Differentview, Janvo, Can't sleep, clown will eat me, Sergio.ballestrero, Frap, Nixeagle, Serenity-Fr, Thefriedone, JonHarder, Propheci, Joinarnold, Bennose, Mackseem~enwiki, Radagast83, Nibuod, Daqu, DueSouth, Blake-, Krexer, Weregerbil, Vina-iwbot~enwiki, Andrei Stroe, Deepred6502, Spiritia, Lambiam, Wikiolap, Kuru, Bmhkim, Vgy7ujm, Calum MacÙisdean, Athernar, Burakordu, Feraudyh, 16@r, Beetstra, Mr Stephen, Jimmy Pitt, Julthep, Dicklyon, Waggers, Ctacmo, RichardF, Nabeth, Beefyt, Hu12, Enggakshat, Vijay.babu.k, Ft93110, Dagoldman, Veyklevar, Ralf Klinkenberg, JHP, IvanLanin, Paul Foxworthy, Adrian.walker, Linkspamremover, CRGreathouse, CmdrObot, Filip*, Van helsing, Shorespirit, Matt1299, Kushal one, CWY2190, Ipeirotis, Nilfanion, Cydebot, Valodzka, Gogo Dodo, Ar5144-06, Akhil joey, Martin Jensen, Pingku, Oli2140, Mikeputnam, Talgalili, Malleus Fatuorum, Thijs!bot, Barticus88, Nirvanalulu, Drowne, Scientio, Kxlai, Headbomb, Ubuntu2, AntiVandalBot, Seaphoto, Ajaysathe, Gwyatt-agastle, Onasraou, Spencer, Alphachimpbot, JAnDbot, Wiki0709, Barek, Sarnholm, MER-C, The Transhumanist, Bull3t, TFinn734, Andonic, Mkch, Hut 8.5, Leiluo, Jguthaaz, EntropyAS, SiobhanHansa, Timdew, Dmmd123, Connormah, Bongwarrior, VoABot II, Tedickey, Giggy, JJ Harrison, David Eppstein, Chivista~enwiki, Gomm, Pmbhagat, Fourthcourse, Kgfleischmann, RoboBaby, Quanticle, ERI employee, R'n'B, Jfroelich, Tgeairn, Pharaoh of the Wizards, Trusilver, Bongomatic, Roxy1984, Andres.santana, Shwapnil, DanDoughty, Foober, Ocarbone, RepubCarrier, Gzkn, AtholM, Salih, LordAnubisBOT, Starnestommy, Jmajeremy, A m sheldon, AntiSpamBot, LeighvsOptimvsMaximvs, Ramkumar.krishnan, Shoessss, Josephjthomas, Parikshit Basrur, Doug4, Cometstyles, DH85868993, DorganBot, Bonadea, WinterSpw, Mark.hornick, Andy Marchbanks, Yecril, BernardZ, RJASE1, Idiomabot, RonFredericks, Black Kite, Jeff G., Jimmaths, DataExp, Philip Trueman, Adamminstead, TXiKiBoT, Deleet, Udufruduhu, Deanabb, Valerie928, TyrantX, OlavN, Arpabr, Vlad.gerchikov, Don4of4, Raymondwinn, Mannafredo, 1yesfan, Bearian, Jkosik1, Wykypydya, Billinghurst, Atannir, Hadleywickham, Hherbert, Falcon8765, Sebastjanmm, Monty845, Pjoef, Mattelsen, AlleborgoBot, Burkeangirl, NHRHS2010, Rknasc, Pdfpdf, Equilibrioception, Calliopejen1, VerySmartNiceGuy, Euryalus, Dawn Bard, Estard, Srp33, Jerryobject, Kexpert, Mark Klamberg, Curuxz, Flyer22, Eikoku, JCLately, Powtroll, Jpcedenog, Strife911, Pyromaniaman, Oxymoron83, Gpswiki, Dodabe~enwiki, Gargvikram07, Mátyás, Fratrep, Chrisguyot, Odo Benus, Stfg, StaticGull, Sanya r, DixonD, Kjtobo, Melcombe, 48states, LaUs3r, Pinkadelica, Ypouliot, Denisarona, Sbacle, Kotsiantis, Loren.wilton, Sfan00 IMG, Nezza 4 eva, ClueBot, The Thing That Should Not Be, EoGuy, Supertouch, Kkarimi, Blanchardb, Edayapattiarun, Lbertolotti, Shaw76, Verticalsearch, Sebleouf, Hanifbbz, Abrech, Sterdeus, DrCroco, Nano5656, Aseld, Amossin, Dekisugi, SchreiberBike, DyingIce, Atallcostsky, 9Nak, Dank, Versus22, Katanada, Qwfp, DumZiBoT, Sunsetsky, XLinkBot, Articdawg, Cgfjpfg, Ecmalthouse, Little Mountain 5, WikHead, SilvonenBot, Badgernet, Foxyliah, Freestyle-69, Texterp, Addbot, DOI bot, Mabdul, Landon1980, Mhahsler, AndrewHZ, Elsendero, Matt90855, Jpoelma13, Cis411, Drkknightbatman, MrOllie, Download, RTG, M.r santosh kumar., Glane23, Delaszk, Chzz, Swift-Epic (Refectory), AtheWeatherman, Fauxstar, Jesuja, Luckas-bot, Yobot, Adelpine, Bunnyhop11, Ptbotgourou, Cflm001, Hulek, Alusayman, Ryanscraper, Carleas, Nallimbot, SOMart, Tiffany9027, AnomieBOT, Rjanag, Jim1138, JackieBot, Fahadsadah, OptimisticCynic, Dudukeda, Materialscientist, Citation bot, Schul253, Cureden, Capricorn42, Gtfjbl, Lark137, Liwaste, The Evil IP address, Tomwsulcer, BluePlateSpecial, Dr Oldekop, Rosannel, Rugaaad, RibotBOT, Charvest, Tareq300, Cmccormick8, Smallman12q, Andrzejrauch, Davgrig04, Stekre, Whizzdumb, Thehelpfulbot, Kyleamiller, OlafvanD, FrescoBot, Mark Renier, Ph92, W Nowicki, X7q, Colewaldron, Er.piyushkp, HamburgerRadio, Atlantia, Webzie, Citation bot 1, Killian441, Manufan 11, Rustyspatula, Pinethicket, Guerrerocarlos, Toohuman1, BRUTE, Elsevieredítormath, Stpasha, MastiBot, SpaceFlight89, Jackverr, UngerJ, Juliustch, Priyank782, TobeBot, Pamparam, Btcoal, Kmettler, Jonkerz, GregKaye, Glenn Maddox, Jayrde, Angelorf, Reaper Eternal, Chenzheruc, Pmauer, DARTH SIDIOUS 2, Mean as custard, RjwilmsiBot, Mike78465, D vandyke67, Ripchip Bot, Slon02, Aaronzat, Helwr, Ericmortenson, EmausBot, Acather96, BillyPreset, Fly by Night, WirlWhind, GoingBatty, Emilescheepers444, Stheodor, Lawrykid, Uploadvirus, Wikipelli, Dcirovic, Joanlofe, Anir1uph, Chire, Cronk28, Zedutchgandalf, Vangelis12, T789, Rick jens, Donner60, Terryholmsby, MainFrame, Phoglenix, Raomohsinkhan, ClueBot NG, Mathstat, Aiwing, Nuwanmenuka, Statethatiamin, CherryX, Candace Gillhoolley, Robiminer, Leonardo61, Twillisjr, Widr, WikiMSL, Luke145, EvaJamax, Debuntu, Helpful Pixie Bot, AlbertoBetulla, HMSSolent, Ngorman, Inoshika, Data.mining, ErinRea, BG19bot, Wanming149, PhnomPencil, Lisasolomonsalford, Uksas, Naeemmalik036, Chafe66, Onewhohelps, Netra Nahar, Aranea Mortem, Jasonem, Flaticida, Funkykeith777, Moshiurbd, Nathanashleywild, Anilkumar 0587, Mpaye, Rabarbaro70, Thundertide, BattyBot, Aacruzr, Warrenxu, IjonTichyIjonTichy, Harsh 2580, Dexbot, Webclient101, Mogism, TwoTwoHello, Frosty, Bradhill14, 7376a73b3bf0a490fa04bea6b76f4a4b, L8fortee, Dougs campbell, Mark viking, Cmartines, Epicgenius, THill182, Delafé, Melonkelon, Herpderp1235689999, Revengetechy, Amykam32, The hello doctor, Mimarios1, Huang cynthia, DavidLeighEllis, Gnust, Rbrandon87, Astigitana, Alihaghi, Philip Habing, Wccsnow, Jianhui67, Tahmina.tithi, Yeda123, Skr15081997, Charlotth, Jfrench7, Zjl9191, Davidhart007, Routerdecomposer, Augt.pelle, Justincahoon, Gstoel, Wiki-jonne, MatthewP42, 115ash, LiberumConsilium, Ran0512, Daniel Bachar, Galaktikasoft, Prof PD Hoy, GoldCoastPrior, Gary2015, KasparBot, Baharsahu, Hillbilly Dragon Farmer and Anonymous: 987
- **Big data** *Source:* https://en.wikipedia.org/wiki/Big_data?oldid=670390555 *Contributors:* William Avery, Heron, Kku, Samw, An-

drewman327, Ryuch, דוד, Topbanana, Paul W, F3meyer, Sunray, Giftlite, Langec, Erik Carson, Utcursch, Beland, Jeremykemp, David@scatter.com, Discospinster, Rich Farmbrough, Kdammers, ArnoldReinhold, Narsil, Viriditas, Lenov, Gary, Pinar, Tobych, Miranche, Broeni, Tomlzz1, Axeman89, Woohookitty, Pol098, Qwertyus, Rjwilmsi, ElKevbo, Jehochman, Nihiltres, Lumin~enwiki, Tedder, DVdm, SteveLoughran, Aeusoes1, Daniel Mietchen, Dimensionsix, Katieh5584, Henryyan, McGeddon, Od Mishehu, Gilliam, Ohnoitsjamie, Chris the speller, RDBrown, Pegua, Madman2001, Krexer, Kuru, Almaz~enwiki, Dl2000, The Letter J, Chris55, Yragha, Jac16888, Marc W. Abel, Cydebot, Matrix61312, Quibik, DumbBOT, Malleus Fatuorum, EdJohnston, Nick Number, Cowb0y, Lmusher, Josephmarty, Kforeman1, Rmyeid, OhanaUnited, Relyk, Wllm, Magioladitis, Nyq, Tedickey, Steven Walling, Thevoid00, Casieg, Jim.henderson, Tokyogirl79, MacShimi, McSly, NewEnglandYankee, Lamp90, Asefati, Pchackal, Mgualtieri, VolkovBot, JohnBlackburne, Vincent Lextrait, Philip Trueman, Ottb19, Billinghurst, Grinq, Scottywong, Luca Naso, Dawn Bard, Yintan, Jazzwang, Jojikiba, Eikoku, SPACKlick, CutOffTies, Mkbergman, Melcombe, Siskus, PabloStraub, Dilaila, Martarius, Sfan00 IMG, Faalagorn, Apptrain, Morrisjd1, Grantbow, Mild Bill Hiccup, Ottawahitech, Cirt, Auntof6, Lbertolotti, Gnome de plume, Resoru, Pablomendes, Saisdur, SchreiberBike, MPH007, Rui Gabriel Correia, Mymallandnews, XLinkBot, Ost316, Benboy00, MystBot, P.r.newman, Addbot, Mortense, Drevicko, Thomas888b, AndrewHZ, Tothwolf, Ronhjones, Moosehadley, MrOllie, Download, Jarble, Arbitrarily0, Luckas-bot, Yobot, Fraggle81, Manivannan pk, Elfix, Jean.julius, AnomieBOT, Jim1138, Babrodtk, Bluerasberry, Materialscientist, Citation bot, Xqbot, Marko Grobelnik, Bgold12, Anna Frodesiak, Tomwsulcer, Srich32977, Omnipaedista, Smallman12q, Joaquin008, Jugdev, FrescoBot, Jonathanchaitow, I42, PeterEastern, AtmosNews, B3t, I dream of horses, HRoestBot, Jonesey95, Jandalhandler, Mengxr, Ethansdad, Yzerman123, Msalganik, בן גרשון, Sideways713, Stuartzs, Jfmantis, Mean as custard, RjwilmsiBot, Ripchip Bot, Mm479flarok, Winchetan, Petermcelwee, DASHBot, EmausBot, John of Reading, Oliverlyc, Timtempleton, Dewritech, Peaceray, Radshashi, Cmlloyd1969, K6ka, HiW-Bot, Richard asr, ZéroBot, Checkingfax, BobGourley, Josve05a, Xtzou, Chire, Kilopi, Laurawilber, Rcsprinter123, Rick jens, Palosirkka, MainFrame, ChuispastonBot, Sean Quixote, Axelode, Mhiji, Helpsome, ClueBot NG, Behrad3d, Danielg922, Pramanicks, Jj1236, Widr, WikiMSL, Lawsonstu, Fvillanustre, Helpful Pixie Bot, Lowercase sigmabot, BG19bot, And Adoil Descended, Seppemans123, Jantana, Innocentantic, Northamerica1000, Asplanchna, MusikAnimal, AvocatoBot, Noelwclarke, Matt tubb, Jordanzhang, Bar David, InfoCmplx, Atlasowa, Fylbecatulous, Camberleybates, BattyBot, WH98, DigitalDev, Haroldpolo, Ryguyrg, Untioencolonia, Shirishnetke, Ampersandian, MarkTraceur, ChrisGualtieri, TheJJJunk, Khazar2, Vaibhav017, IjonTichyIjonTichy, Saturdayswiki, Mheikkurinen, Seherrell, Mjvaugh2, ChazzI73, Davidogm, Mherradora, Jkofron4, Stevebillings, Indianbusiness, Toopathfind, Jeremy Kolb, Frosty, Jamesx12345, OnTheNet21, BrighterTomorrow, Jacoblarsen net, Epicgenius, DavidKSchneider, Socratesplato9, Anirudhrata, Parasdoshiblog, Edwinboothnyc, JuanCarlosBrandt, Helenellis, MMeTrew, Warrenpd86, AuthorAnil, ViaJFK, Gary Simon, Bsc, FCA, FBCS, CITP, Mcioffi, Joe204, Caraconan, Evaluatorgroup, Hessmike, TJLaher123, Chengying10, IndustrialAutomationGuru, Dabramsdt, Prussonyc, Abhishek1605, Dilaila123, Willymomo, Rzicari, Mandruss, Mingminchi, BigDataGuru1, Sugamsha, Sysœp, Azra2013, Paul2520, Dudewhereismybike, Shahbazali101, Yeda123, Miakeay, Stamptrader, Accountdp, Morganmissen, JeanneHolm, Yourconnotation, JenniferAndy, Arcamacho, Amgauna, Bigdatavomit, Monkbot, Wikientg, Scottishweather, Textractor, Analytics ireland, Lspin01l, ForumOxford Online, Mansoor-siamak, Belasobral, Sightestrp, Jwdang4, Amortias, Wikiauthor22, Femiolajiga, Tttcraig, Lepro2, Mythfinder, DexterToo, Mr P. Kopee, Pablollopis, SVtechie, Deathmuncher19, Smaske, Greystoke1337, Prateekkeshari, Hmrv83, KaraHayes, Iqmc, Lalith269, Helloyoubum, Jakesher, IEditEncyclopedia, Rajsbhatta123, Ragnar Valgeirsson, Vedanga Kumar, Fgtyg78, Gary2015, EricVSiegel, Benedge46, Friafternoon, KasparBot, Adzzyman, Pmaiden, Spetrowski88, JuiAmale, Yasirsid and Anonymous: 330

- **Euclidean distance** *Source:* https://en.wikipedia.org/wiki/Euclidean_distance?oldid=669284157 *Contributors:* Damian Yerrick, AxelBoldt, XJaM, Boleslav Bobcik, Michael Hardy, Nikai, Epl18, AnthonyQBachler, Fredrik, Altenmann, MathMartin, Saforrest, Enochlau, Giftlite, BenFrantzDale, Bender235, Rgdboer, Bobo192, Dvogel, Obradovic Goran, Fawcett5, Oleg Alexandrov, Warbola, Ruud Koot, Isnow, Qwertyus, Unused007, Ckelloug, DVdm, Wavelength, Multichill, Number 57, StuRat, Arthur Rubin, Clams, SmackBot, ReverendSam, InverseHypercube, 127, Mcld, Oli Filth, Papa November, Octahedron80, Nbarth, Tsca.bot, OrphanBot, Bombshell, Lambiam, Delfinite, Aldarione, Jminguillona, Thijs!bot, JAnDbot, .anacondabot, Theunicyclegirl, Yesitsapril, Graeme.e.smith, Robertgreer, Cometstyles, JohnBlackburne, TXiKiBoT, FrederikHertzum, Tiddly Tom, Paolo.dL, Dattorro, Justin W Smith, DragonBot, Freebit50, Triathematician, Qwfp, Zik2, Ali Esfandiari, SilvonenBot, Addbot, Fgnievinski, AkhtaBot, Tanhabot, LaaknorBot, Favonian, West.andrew.g, Yobot, Ehaussecker, Nallimbot, Ciphers, Materialscientist, Xqbot, Simeon87, Erik9bot, Gleb.svechnikov, Sławomir Biały, RedBot, EmausBot, Rasim, RA0808, Cskudzu, Quondum, Kweckzilber, EdoBot, ClueBot NG, Wcherowi, Stultiwikia, Papadim.G, Ascoldcaves, Arrogantrobot, Soni, Jcarrete, ShuBraque, Erotemic, 7Sidz, Loraof and Anonymous: 69
- **Hamming distance** *Source:* https://en.wikipedia.org/wiki/Hamming_distance?oldid=669792039 *Contributors:* Damian Yerrick, Ap, Pit~enwiki, Kku, Kevin Baas, Poor Yorick, Dcoetzee, Silvonen, David Shay, Altenmann, Wile E. Heresiarch, Tosha, Giftlite, Seabhcan, BenFrantzDale, Markus Kuhn, CryptoDerk, Beland, Gene s, Mindspillage, Leibniz, Zaslav, Danakil, Aaronbrick, Blotwell, Flammifer, 3mta3, Awolsoldier, Obradovic Goran, ABCD, Pouya, Cburnett, Wsloand, Joepzander, Linas, Kasuga~enwiki, Ruud Koot, Zelse81, Qwertyus, Rjwilmsi, Tizio, Pentalith, Mathbot, Margosbot~enwiki, Quuxplusone, Bgwhite, YurikBot, Personman, Michael Slone, Armistej, Archelon, Alcides, Ttam, Zwobot, Attilios, SmackBot, BiT, Bluebot, DHN-bot~enwiki, Scray, Frap, Decltype, Slach~enwiki, SashatoBot, Lambiam, Shir Khan, Loadmaster, ThePacker, DagErlingSmørgrav, ChetTheGray, Eastlaw, CRGreathouse, Krauss, Thijs!bot, Headbomb, Wainson, Fulkkari~enwiki, Adma84, JAnDbot, Sterrys, JPG-GR, David Eppstein, JMyrleFuller, ANONYMOUS COWARD0xC0DE, Ksero, DorganBot, JohnBlackburne, Lixo2, Sue Rangell, Svick, Hhbruun, Thegeneralguy, TSylvester, DragonBot, Alexbot, SchreiberBike, Muro Bot, Cerireid, Oğuz Ergin, MystBot, Addbot, LaaknorBot, Gnorthup, Ramses68, Lightbot, Math Champion, Luckas-bot, Ptbotgourou, AnomieBOT, Joule36e5, Materialscientist, RibotBOT, Citation bot 1, Kiefer.Wolfowitz, Compvis, Ripchip Bot, Valyt, EmausBot, Olof nord, Froch514, Jnaranjo86, Sumanah, Jcarrete, Wkschwartz, Tiagofrepereira, Rubenaodom, ScrapIronIV, Some1Redirects4You and Anonymous: 78
- **Norm (mathematics)** *Source:* https://en.wikipedia.org/wiki/Norm_(mathematics)?oldid=667112150 *Contributors:* Zundark, The Anome, Tomo, Patrick, Michael Hardy, SebastianHelm, Selket, Zero0000, Robbot, Altenmann, MathMartin, Bkell, Tobias Bergemann, Tosha, Connelly, Giftlite, BenFrantzDale, Lethe, Fropuff, Sendhil, Dratman, Jason Quinn, Tomruen, Almit39, Urhixidur, Beau~enwiki, PhotoBox, Sperling, Paul August, Bender235, MisterSheik, EmilJ, Dalf, Bobo192, Army1987, Bestian~enwiki, HasharBot~enwiki, Ncik~enwiki, ABCD, Oleg Alexandrov, Linas, MFH, Nahabedere, Tlroche, HannsEwald, Mike Segal, Magidin, Mathbot, ChongDae, Jenny Harrison, Tardis, CiaPan, Chobot, Algebraist, Wavelength, Eraserhead1, Hairy Dude, KSmrq, JosephSilverman, VikC, Trovatore, Vanished user 1029384756, Crasshopper, David Pal, Tribaal, Fmccown, Arthur Rubin, TomJF, Killerandy, Lunch, That Guy, From That Show!, SmackBot, David Kernow, InverseHypercube, Melchoir, Mhss, Bluebot, Oli Filth, Silly rabbit, Nbarth, Sbharris, Tamfang, Cícero, Cybercobra, DMacks, Lambiam, Dicklyon, SimonD, CBM, Irritate, MaxEnt, Mct mht, Rudjek, Xtv, Thijs!bot, D4g0thur, Headbomb, Steve Kroon, Urdutext, Selvik, Heysan, JAnDbot, Magioladitis, Reminiscenza, Chutzpan, Sullivan.t.j, ANONYMOUS COWARD0xC0DE, JoergenB, Robin S, Allispaul, Pharaoh of the Wizards, Lucaswilkins, Singularitarian, Potatoswatter, Idioma-bot, Cerberus0, JohnBlackburne, PMajer, Don Quixote de la Mancha, Falcongl, Wikimorphism, Synthebot, Free0willy, Dan Polansky, RatnimSnave, Paolo.dL, MiNombreDeGuerra, JackSchmidt, ClueBot, Veromies, Baldphil, Mpd1989, Rockfang, Brews ohare, Hans Adler, Jaan Vajakas, Addbot, Saavek47, Zorrobot,

سعی, Luckas-bot, Yobot, TaBOT-zerem, Kan8eDie, Ziyuang, SvartMan, Citation bot, Jxramos, ArthurBot, DannyAsher, Bdmy, Dlazesz, Omnipaedista, RibotBOT, Shadowjams, Quartl, FrescoBot, Paine Ellsworth, Sławomir Biały, Pinethicket, Kiefer.Wolfowitz, NearSetAccount, Stpasha, RedBot, אביב~enwiki, Dmitri666, Datahaki, JumpDiscont, Weedwhacker128, Xnn, FoxRaweln, Tom Peleg, Jowa fan, EmausBot, Helptry, Effigies, KHamsun, ZéroBot, Midas02, Quondum, Bugmenot10, PerimeterProf, Sebjlan, Petrb, ClueBot NG, Wcherowi, Lovasoa, Snotbot, Helpful Pixie Bot, Rheyik, Aisteco, Deltahedron, Mgkrupa, Laiwoonsiu and Anonymous: 113

- **Regularization (mathematics)** *Source:* https://en.wikipedia.org/wiki/Regularization_(mathematics)?oldid=668286789 *Contributors:* The Anome, Fnielsen, Jitse Niesen, Giftlite, BenFrantzDale, 3mta3, Arcenciel, Oleg Alexandrov, Qwertyus, Patrick Gill, Gareth McCaughan, Eubot, RussBot, Alexmorgan, SmackBot, Took, Chris the speller, Memming, CBM, Headbomb, David Eppstein, Nigholith, ShambhalaFestival, Denisarona, Alexbot, Skbkekas, Addbot, Luckas-bot, AnomieBOT, Citation bot, Obersachsebot, FrescoBot, Kiefer.Wolfowitz, Jonesey95, Noblestats, Chire, Helpful Pixie Bot, Benelot, Illia Connell, Mark viking, Star767, Terrance26, AioftheStorm, Monkbot, Ddunn801 and Anonymous: 14
- **Loss function** *Source:* https://en.wikipedia.org/wiki/Loss_function?oldid=671232453 *Contributors:* The Anome, Michael Hardy, Karada, Delirium, CesarB, Ronz, Den fjättrade ankan~enwiki, A5, Benwing, Henrygb, Cutler, Giftlite, Lethe, Jason Quinn, Nova77, MarkSweep, Rich Farmbrough, Bender235, MisterSheik, 3mta3, John Quiggin, Jheald, Shoefly, Eclecticos, Qwertyus, Rjwilmsi, Chobot, Wavelength, KSmrq, Zvika, Chris the speller, Nbarth, Hongooi, Zvar, DavidBailey, Robofish, Kvng, Shirahadasha, Headbomb, Yellowbeard, Daniel5Ko, Kjtobo, Melcombe, Rumping, Koczy, El bot de la dieta, Qwfp, Addbot, Fgnievinski, Yobot, Pasmargo, AnomieBOT, J04n, X7q, Raise-the-Sail, Nabiw1, Trappist the monk, Duoduoduo, EmausBot, Bnegreve, Akats, Chire, ClueBot NG, BG19bot, Chrish42, Ejjordan, Bodormenta, Andy231987, Brirush, Limit-theorem, Mark viking, Loraof and Anonymous: 22
- **Least squares** *Source:* https://en.wikipedia.org/wiki/Least_squares?oldid=668060355 *Contributors:* The Anome, HelgeStenstrom, Enchanter, Maury Markowitz, Michael Hardy, Delirium, William M. Connolley, Snoyes, Mxn, Drz~enwiki, Charles Matthews, Dysprosia, Jitse Niesen, Robbot, Benwing, Muxxa, Gandalf61, Henrygb, Ashwin, Bkell, Giftlite, BenFrantzDale, Sreyan, Kusunose, DragonflySixtyseven, Frau Holle, Geof, Paul August, Elwikipedista~enwiki, JustinWick, El C, O18, Oyz, Landroni, Jumbuck, Anthony Appleyard, John Quiggin, Drbreznjev, WojciechSwiderski~enwiki, Oleg Alexandrov, Gmaxwell, Soultaco, Isnow, Male1979, GeLuxe, Salix alba, Nneonneo, Tardis, Hugeride, Bgwhite, YurikBot, Wavelength, Jlc46, KSmrq, Philopedia, Grafen, Deodar~enwiki, Witger, Bordaigorl, Scs, Syrthiss, Arthur Rubin, JahJah, Zvika, KnightRider~enwiki, SmackBot, Unyoyega, DanielPenfield, Tgdwyer, Oli Filth, Kostmo, DHN-bot~enwiki, Ladislav Mecir, Rludlow, Tamfang, Berland, Memming, G716, Lambiam, Derek farn, Dicklyon, Theswampman, Ichoran, Tudy77, CapitalR, Harold f, Tensheapz, Ezrakilty, Cydebot, Zzzmarcus, Talgalili, Thijs!bot, Naucer, Tolstoy the Cat, MattWatt, Daniel il, Escarbot, JEBrown87544, Woollymammoth, And4e, Daytona2, JAnDbot, Eromana, Gavia immer, Magioladitis, Albmont, Jllm06, Risujin, MyNameIsNeo, Livingthingdan, Jkjo, User A1, AllenDowney, Glrx, Pharaoh of the Wizards, Vi2, Jiuguang Wang, Baiusmc, DorganBot, PesoSwe, Larryisgood, Nevillerichards, Jmath666, Dfarrar, Forwardmeasure, Sbratu, Bpringlemeir, Petergans, BotMultichill, Zbvhs, MinorContributor, KoenDelaere, Lourakis, Mika.fischer, JackSchmidt, Water and Land, Melcombe, Oekaki, ClueBot, Vikasatkin, Turbojet, Cipherous, Lbertolotti, Geoeg, Uraza, Bender2k14, Sun Creator, Brews ohare, Muro Bot, ChrisHodgesUK, BOTarate, Qwfp, XLinkBot, Juliusllb, NellieBly, Addbot, Mmonks, AndrewHZ, Fgnievinski, MrOllie, EconoPhysicist, Publichealthguru, Glane23, LinkFA-Bot, Kruzmissile, Erutuon, Lightbot, Zorrobot, TeH nOmInAtOr, Meisam, Legobot, Yobot, Sked123, AnomieBOT, SomethingElseToSay, GLRenderer, Ciphers, Rubinbot, Materialscientist, HanPritcher, Citation bot, JmCor, Xqbot, Urbansuperstar~enwiki, Flavio Guitian, Dwlotter, RibotBOT, Hamamelis, FrescoBot, J6w5, Idfah, AstaBOTh15, Kiefer.Wolfowitz, Astropro, Stpasha, Emptiless, Sss41, Full-date unlinking bot, Jonkerz, Weedwhacker128, Elitropia, EmausBot, WikitanvirBot, Netheril96, Cfg1777, Manyu aditya, ZéroBot, Durka42, JonAWellner, JA(000)Davidson, AManWithNoPlan, Mayur, Zfeinst, Robin48gx, Sigma0 1, JaneCow, Mikhail Ryazanov, ClueBot NG, KlappCK, Demonsquirrel, Habil zare, Hikenstuff, Helpful Pixie Bot, Mythirdself, Koertefa, Abryhn, BG19bot, Benelot, Lxlxlx82, Vanangamudiyan, Op47, Manoguru, Simonfn, Kodiologist, ChrisGualtieri, IPWAI, Gameboy97q, Tschmidt23, Dexbot, Declaration1776, Dough34, RichardInMiami, Mgfbinae, Leegrc, Velvel2, Yilincau, Gowk, BTM912, KasparBot and Anonymous: 208
- **Newton's method** *Source:* https://en.wikipedia.org/wiki/Newton'{}s_method?oldid=663961763 *Contributors:* AxelBoldt, Lee Daniel Crocker, Zundark, Miguel~enwiki, Roadrunner, Formulax~enwiki, Hirzel, Pichai Asokan, Patrick, JohnOwens, Michael Hardy, Pit~enwiki, Dominus, Dcljr, Loisel, Minesweeper, Ejrh, Looxix~enwiki, Cyp, Poor Yorick, Pizza Puzzle, Hike395, Dcoetzee, Jitse Niesen, Kbk, Saltine, AaronSw, Robbot, Jaredwf, Fredrik, Wikibot, Giftlite, Rs2, BenFrantzDale, Neilc, MarkSweep, PDH, Torokun, Sam Hocevar, Kutulu, Fintor, Frau Holle, TheObtuseAngleOfDoom, Paul August, Pt, Aude, Iamunknown, Blotwell, Nk, Haham hanuka, LutzL, Borisblue, Jeltz, Laug, Olegalexandrov, Oleg Alexandrov, Tbsmith, Joriki, Shreevatsa, LOL, Decrease789, Jimbryho, Robert K S, JonBirge, GregorB, Casey Abell, Eyu100, Mathbot, Shultzc, Kri, Glenn L, Wikipedia is Nazism, Chobot, YurikBot, Wavelength, Laurentius, Swerty, JabberWok, KSmrq, Exir Kamalabadi, Tomisti, Alias Flood, Marquez~enwiki, SmackBot, RDBury, Selfworm, Adam majewski, Saravask, Tom Lougheed, InverseHypercube, Jagged 85, Dulcamara~enwiki, Commander Keane bot, Slaniel, Skizzik, Chris the speller, Berland, Rrburke, Earlh, ConMan, Jon Awbrey, Henning Makholm, Bdiscoe, Wvbailey, Coredesat, Jim.belk, Magmait, Gco, JRSpriggs, CRGreathouse, Jackzhp, David Cooke, Holycow958, Eric Le Bigot, Cydebot, Quibik, Christian75, Billtubbs, Talgalili, Thijs!bot, Epbr123, Nonagonal Spider, Headbomb, Martin Hedegaard, BigJohnHenry, Ben pcc, Seaphoto, CPMartin, JAnDbot, Coffee2theorems, VoABot II, JamesBWatson, Baccyak4H, Avicennasis, David Eppstein, User A1, GuidoGer, Arithmonic, Glrx, Pbroks13, Kawautar, Rankarana, Nedunuri, K.menin, Gombang, Chiswick Chap, Goingstuckey, Policron, Juliancolton, Homo logos, JohnBlackburne, Philip Trueman, TXiKiBoT, Anonymous Dissident, Broadbot, Aaron Rotenberg, Draconx, Pitel, Katzmik, Psymun747, SieBot, Gex999, Dawn Bard, Bentogoa, Flyer22, MinorContributor, Jasondet, Smarchesini, Redmarkviolinist, Dreamofthedolphin, Cyfal, PlantTrees, ClueBot, Metaprimer, Wysprgr2005, JP.Martin-Flatin, Mild Bill Hiccup, CounterVandalismBot, Tesspub, Chrisgolden, Annne furanku, Dekisugi, Xooll, Muro Bot, Jpginn, RMFan1, Galoisgroupie, Addbot, Some jerk on the Internet, Eweinber, Fluffernutter, Ckamas, Protonk, EconoPhysicist, LinkFA-Bot, Uscitizenjason, AgadaUrbanit, Numbo3-bot, Tide rolls, CountryBot, Xieyihui, Luckas-bot, Yobot, Estudiarme, AnomieBOT, 1exec1, Illegal604, Apau98, Пика Пика, Materialscientist, Zhurov, ArthurBot, PavelSolin, Capricorn42, Titolatif, CBoeckle, Nyirenda, Dlazesz, Point-set topologist, CnkALTDS, Shadowjams, FrescoBot, Pepper, DNA Games, Citation bot 1, Tkuvho, Gaba p, Pinethicket, Eyrryds, Kiefer.Wolfowitz, White Shadows, Vrenator, ClarkSims, Duoduoduo, KMic, Suffusion of Yellow, H.ehsaan, Jfmantis, Hyarmendacil, 123Mike456Winston789, EmausBot, TheJesterLaugh, KHamsun, Mmeijeri, Shuipzv3, D.Lazard, Eniagrom, U+003F, Bomazi, Chris857, Howard nyc, Kyle.drerup, Elvek, Mikhail Ryazanov, ClueBot NG, KlappCK, Aero-Plex, Chogg, Helpful Pixie Bot, EmadIV, Drift chambers, Toshiki, Scuchina, AWTom, RiabzevMichael, Electricmuffin11, Khazar2, Qxukhgiels, Dexbot, M.shahriarinia, I am One of Many, Lakshmi7977, Manoelramon, Ginsuloft, Mohit.del94, Blitztall, Rob Haelterman, Loraof, Akemdh and Anonymous: 276
- **Supervised learning** *Source:* https://en.wikipedia.org/wiki/Supervised_learning?oldid=643120523 *Contributors:* Damian Yerrick, LC~enwiki, Isomorph, Darius Bacon, Boleslav Bobcik, Michael Hardy, Oliver Pereira, Zeno Gantner, Chadloder, Alfio, Ahoerstemeier, Cyp, Snoyes, Rotem Dan, Cherkash, Mxn, Hike395, Shizhao, Topbanana, Unknown, Ancheta Wis, Giftlite, Markus Krötzsch, Duncharris, MarkSweep, APH, Gene s, Sam Hocevar, Violetriga, Skeppy, Denoir, Mscnln, Rrenaud, Oleg Alexandrov, Lloydd, Joerg Kurt Wegner, Marudubshinki, Qwertyus, Mathbot, Chobot, YurikBot, Wavelength, Jlc46, Ritchy, Tony1, Tribaal, BenBildstein, SmackBot,

Reedy, KnowledgeOfSelf, MichaelGasser, Zearin, Dfass, Beetstra, CapitalR, Domanix, Sad1225, Thijs!bot, Mailseth, Escarbot, Prolog, Peteymills, Robotman1974, 28421u2232nfenfcenc, A3nm, David Eppstein, Mange01, Paskari, VolkovBot, Naveen Sundar, Jamelan, Temporaluser, EverGreg, Yintan, Flyer22, Melcombe, Baosheng, Kotsiantis, Doloco, Skbkekas, Magdon~enwiki, DumZiBoT, Addbot, AndrewHZ, Anders Sandberg, EjsBot, MrOllie, Buster7, Numbo3-bot, Yobot, Twri, Fstonedahl, FrescoBot, LucienBOT, X7q, Mostafa mahdieh, Classifier1234, Erylaos, Zadroznyelkan, Fritq, BertSeghers, WikitanvirBot, Fly by Night, Sun116, Pintaio, Dappermuis, Tdietterich, WikiMSL, EvaJamax, BrutForce, Colbert Sesanker, J.Davis314, Citing, Ferrarisailor, ChrisGualtieri, YFdyh-bot, Alialamifard, Francisbach, Donjohn1 and Anonymous: 65

- **Linear regression** *Source:* https://en.wikipedia.org/wiki/Linear_regression?oldid=671189370 *Contributors:* The Anome, Taw, Ap, Danny, Miguel~enwiki, Rade Kutil, Edward, Patrick, Michael Hardy, GABaker, Shyamal, Kku, Tomi, TakuyaMurata, Den fjättrade ankan~enwiki, Kevin Baas, Rossami, Hike395, Jitse Niesen, Andrewman327, Taxman, Donarreiskoffer, Robbot, Jaredwf, Benwing, Gak, ZimZalaBim, Yelyos, Babbage, Henrygb, Jcoleff, Wile E. Heresiarch, Giftlite, BenFrantzDale, Fleminra, Alison, Duncharris, Jason Quinn, Ato, Utcursch, Pgan002, MarkSweep, Piotrus, Wurblzap~enwiki, Icairns, Urhixidur, Natrij, Discospinster, Rich Farmbrough, Pak21, Paul August, Bender235, Violetriga, Elwikipedista~enwiki, Gauge, MisterSheik, Spoon!, Perfecto, O18, Davidswelt, R. S. Shaw, Tobacman, Arcadian, NickSchweitzer, 99of9, Crust, Landroni, Storm Rider, Musiphil, Arthena, ABCD, Kotasik, Avenue, Snowolf, LFaraone, Forderud, Drummond, Oleg Alexandrov, Abanima, Tappancsa, Mindmatrix, BlaiseFEgan, Btyner, Joerg Kurt Wegner, Lacurus~enwiki, Graham87, Qwertyus, Rjwilmsi, Vegaswikian, Matt Deres, TeaDrinker, Chobot, Manscher, FrankTobia, Wavelength, RussBot, Gaius Cornelius, Bug42, Afelton, Thiseye, Cruise, Moe Epsilon, Voidxor, Dggoldst, Arch o median, Arthur Rubin, Drallim, Anarch21, SolarMcPanel, SmackBot, NickyMcLean, Quazar777, Prodego, InverseHypercube, Jtneill, DanielPenfield, Evanreyes, Commander Keane bot, Ohnoitsjamie, Hraefen, Afa86, Markush, Amatulic, Feinstein, Oli Filth, John Reaves, Berland, Wolf87, Cybercobra, Semanticprecision, G716, Unco, Theblackgecko, Lambiam, Jonas August, Vjeet a, Nijdam, Beetstra, Emurph, Hu12, Pjrm, LAlawMedMBA, JoeBot, Chris53516, AlainD, Jsorens, Tawkerbot2, CmdrObot, JRavn, CBM, Anakata, Chrike, Harej bot, Thomasmeeks, Neelix, Cassmus, Bumbulski, MaxEnt, 137 0, Pedrolapinto, Mmmooonnnsssttteeerrr, Farshidforouz~enwiki, FrancoGG, Talgalili, Thijs!bot, Epbr123, Tolstoy the Cat, Jfaller, Whooooooknows, Natalie Erin, Woollymammoth, Mack2, JAnDbot, MER-C, Jeff560, Ph.eyes, Hectorlamadrid, Magioladitis, Tripbeetle, MastCell, Albmont, Baccyak4H, Ddr~enwiki, David Eppstein, Joostw, Apal~enwiki, Yonaa, R'n'B, Noyder, Charlesmartin14, Kawautar, Mbhiii, J.delanoy, Scythe of Death, Salih, TomyDuby, Jaxha, HyDeckar, Jewzip, MrPaul84, Copsi, Llorenzi, VolkovBot, Smarty07, Muzzamo, Mfreund~enwiki, Jsd115, Zhenqinli, Greswik, P1h3r1e3d13, Ricardo MC, Jhedengren, Petergans, Karthik Sarma, Rlendog, Zsniew, Paolo.dL, OKBot, Water and Land, Scottyoak2, Melcombe, Gpap.gpap, Tanvir Ahmmed, ClueBot, HairyFotr, Cp111, Rhubbarb, Dromedario~enwiki, Alexbot, Ecov, Kaspar.jan, Tokorode~enwiki, Skbkekas, Stephen Milborrow, Diaa abdelmoneim, Qwfp, Bigoperm, Sunsetsky, XLinkBot, Tofallis, W82~enwiki, Tayste, Addbot, RPHv, Fgnievinski, Doronp, MrOllie, Download, Forich, Zorrobot, Ettrig, Luckas-bot, Yobot, Sked123, It's Been Emotional, AnomieBOT, Rubinbot, IRP, Materialscientist, HanPritcher, Citation bot, Lixiaoxu, Sketchmoose, Flavio Guitian, Gtfjbl, Istrill, Mstangeland, Aa77zz, Imran.fanaswala, Fstonedahl, PhysicsJoe, Nickruiz, FrescoBot, X7q, Citation bot 1, AstaBOTh15, Boxplot, Pinethicket, Elockid, Kiefer.Wolfowitz, Rdecker02, Jonesey95, Stpasha, Oldrrb, Trappist the monk, Wotnow, Duoduoduo, PAC2, Wombathammer, Diannaa, RjwilmsiBot, Elitropia, John of Reading, Sugarfoot1001, Julienbarlan, Wikieconometrician, Bkearb, NGPriest, BartlebytheScrivener, Chewings72, Esaintpierre, Manipande, ClueBot NG, Mathstat, Frietjes, BlueScreenD, Helpful Pixie Bot, Grandwgy, Daonng, Mark Arsten, MyWikiNik, ChrisGualtieri, Illia Connell, Dansbecker, Dexbot, Hkoslik, Sa publishers, Ossifragus, Bha100710, Drvikas74, Melonkelon, Asif usa, Pandadai, Bryanrutherford0, Tertius51, Logan.dunbar, Monkbot, Jpeterson1346, Bob nau, Moorshed, Velvel2, Whatfoxsays, 18trevor3695, Split97 and Anonymous: 373

- **Tikhonov regularization** *Source:* https://en.wikipedia.org/wiki/Tikhonov_regularization?oldid=670944714 *Contributors:* Gareth Owen, Edward, Michael Hardy, Willem, Charles Matthews, Dysprosia, Benwing, Wile E. Heresiarch, BenFrantzDale, Markus Kuhn, Sietse, Quadell, Sam Hocevar, Rich Farmbrough, Bender235, Billlion, Arcenciel, Jheald, Oleg Alexandrov, Simetrical, Shoyer, Btyner, BD2412, Qwertyus, Rjwilmsi, Gseryakov, EricCHill, Wavelength, RussBot, Bruguiea, Dtrebbien, Yahya Abdal-Aziz, Syrthiss, Caliprincess, Sharat sc, Zvika, SmackBot, CapitalSasha, JesseStone, Oli Filth, NickPenguin, Tarantola, Lavaka, David s graff, Shorespirit, Lklundin, A3nm, Pablodiazgutierrez, Lantonov, TomyDuby, Asjogren, Sigmundur, STBotD, Thiverus, FghIJklm, Melcombe, Skbkekas, Addbot, Legobot, Ptbotgourou, AnomieBOT, Angry bee, SassoBot, Richarddonkin, FrescoBot, Fortdj33, Citation bot 1, Wkretzsch, Duoduoduo, RjwilmsiBot, EmausBot, ZéroBot, Jotaf, Yagola, Koertefa, SciCompTeacher, Manoguru, BattyBot, Viraltux, Wsrosenthal, Evan Aad, Monkbot and Anonymous: 59

- **Regression analysis** *Source:* https://en.wikipedia.org/wiki/Regression_analysis?oldid=670597343 *Contributors:* Berek, Taw, ChangChienFu, Michael Hardy, Kku, Meekohi, Jeremymiles, Ronz, Den fjättrade ankan~enwiki, Hike395, Quickbeam, Jitse Niesen, Taxman, Samsara, Bevo, Mazin07, Benwing, Robinh, Giftlite, Bfinn, TomViza, BrendanH, Jason Quinn, Noe, Piotrus, APH, Israel Steinmetz, Urhixidur, Rich Farmbrough, Pak21, Paul August, Bender235, Bobo192, Cretog8, Arcadian, NickSchweitzer, Photonique, Mdd, Jérôme, Denoir, Arthena, Riana, Avenue, Emvee~enwiki, Nvrmnd, Gene Nygaard, Krubo, Oleg Alexandrov, Abanima, Lkinkade, Woohookitty, LOL, Marc K, Kosher Fan, BlaiseFEgan, Wayward, Btyner, Lacurus~enwiki, Gmelli, Salix alba, MZMcBride, Pruneau, Mathbot, Valermos, Goudzovski, King of Hearts, Chobot, Jdannan, Krishnavedala, Wavelength, Wimt, Afelton, Brian Crawford, DavidHouse~enwiki, DeadEyeArrow, Avraham, Jmchen, NorsemanII, Tribaal, Closedmouth, Arthur Rubin, Josh3580, Wikiant, Shawnc, □□□□ robot, Veinor, Doubleplusjeff, SmackBot, NickyMcLean, Deimos 28, Antro5, Cazort, Gilliam, Feinstein, Oli Filth, Nbarth, Ctbolt, DHN-bot~enwiki, Gruzd, Hve, Berland, EvelinaB, Radagast83, Cybercobra, Krexer, CarlManaster, Nrcprm2026, G716, Mwtoews, Cosmix, Tedjn, Friend of facts, Danilcha, John, FrozenMan, Tim bates, JorisvS, IronGargoyle, Beetstra, Dicklyon, AdultSwim, Kvng, Joseph Solis in Australia, Chris53516, AbsolutDan, Ioannes Pragensis, Markjoseph125, CBM, Thomasmeeks, GargoyleMT, Ravensfan5252, JohnInDC, Talgalili, Wikid77, Qwyrxian, Sagaciousuk, Tolstoy the Cat, N5iln, Carpentc, AntiVandalBot, Woollymammoth, Lcalc, JAnDbot, Goskan, Giler, QuantumEngineer, Ph.eyes, SiobhanHansa, DickStartz, JamesBWatson, Username550, Fleagle11, Marcelobbribeiro, David Eppstein, DerHexer, Apdevries, Thenightowl~enwiki, Mbhiii, Discott, Trippingpixie, Cpiral, Gzkn, Rod57, TomyDuby, Coppertwig, RenniePet, Policron, Bobianite, Blueharmony, Peepeedia, EconProf86, Qtea, BernardZ, TinJack, CardinalDan, HughD, DarkArcher, Gpeilon, TXiKiBoT, SueHay, Qxz, Gnomepirate, Sintaku, Antaltamas, JhsBot, Broadbot, Beusson, Cremepuff222, Zain Ebrahim111, Billinghurst, Kusyadi, Traderlion, Asjoseph, Petergans, Rlendog, BotMultichill, Statlearn, Gerakibot, Matthew Yeager, Timhowardriley, Strife911, Indianarhodes, Amitabha sinha, OKBot, Water and Land, AlanUS, Savedthat, Mangledorf, Randallbsmith, Amadas, Tesi1700, Melcombe, Denisarona, JL-Bot, Mrfebruary, Kotsiantis, Tdhaene, The Thing That Should Not Be, Sabri76, Auntof6, DragonBot, Sterdeus, Skbkekas, Stephen Milborrow, Cfn011, Crash D 0T0, SBemper, Qwfp, Antonwg, Sunsetsky, XLinkBot, Gerhardvalentin, Nomoskedasticity, Veryhuman, Piratejosh85, WikHead, SilvonenBot, Hess88, Addbot, Diegoful, Wootbag, Geced, MrOllie, LaaknorBot, Lightbot, Luckas-bot, Yobot, Themfromspace, TaBOT-zerem, Andresswift, KamikazeBot, Eaihua, Tempodivalse, AnomieBOT, Andypost, RandomAct, HanPritcher, Citation bot, Jyngyr, LilHelpa, Obersachsebot, Xqbot, Statisticsblog, TinucherianBot II, Ilikeed, J04n, GrouchoBot, BYZANTIVM, Fstonedahl, Bartonpoulson, D0kkaebi, Citation bot 1, Dmitronik~enwiki, Boxplot, Yuanfangdelang, Pinethicket, Kiefer.Wolfowitz, Tom.Reding, Stpasha, Di1000, Jonkerz, Duoduoduo, Diannaa, Tbhotch, RjwilmsiBot, EmausBot, RA0808, KHamsun, Fæ, Julienbarlan, Hypocritical~enwiki, Kgwet, Zfeinst, Bomazi, ChuispastonBot, 28bot, Rocketrod1960, ClueBot NG, Mathstat, MelbourneStar, Joel B. Lewis, CH-stat, Helpful Pixie Bot, BG19bot, Giogm2000, CitationCleanerBot, Hakimo99, Gprobins, Prof. Squirrel, Attleboro, Illia Connell, JYBot, Sinxvin, Francescapelusi, Lugia2453, SimonPerera, Lemnaminor, Infiniti4, EJM86, Francisbach, Eli the King, Monkbot, Bob nau, Moorshed k, Moorshed, KasparBot and Anonymous: 385

- **Statistical learning theory** *Source:* https://en.wikipedia.org/wiki/Statistical_learning_theory?oldid=655393279 *Contributors:* Michael Hardy, Hike395, Jitse Niesen, Bearcat, Tomchiukc, Klemen Kocjancic, Rajah, Qwertyus, Chris the speller, Geach, ClydeC, Katharineamy, VolkovBot, Melcombe, XLinkBot, Fgnievinski, MichalSylwester, Brightgalrs, RealityApologist, Callanecc, Zephyrus Tavvier, ClueBot NG, BG19bot, Aisteco, Atastorino, Francisbach, I3roly, Mgfbinae, Jala Daibajna, Slee325 and Anonymous: 15
- **Vapnik–Chervonenkis theory** *Source:* https://en.wikipedia.org/wiki/Vapnik%E2%80%93Chervonenkis_theory?oldid=666685396 *Contributors:* Awaterl, Michael Hardy, Hike395, Bearcat, Tomchiukc, Naddy, Gene s, Rajah, Male1979, Qwertyus, SmackBot, Vermorel, Momet, Vanisaac, JForget, Mmunoz.ag, Ph.eyes, Tremilux, Alfred Legrand, Nadiatalent, Gambette, Rei-bot, Guillaume2303, Melcombe, Pichpich, Addbot, Yobot, Qorilla, Xqbot, Alfredo ougaowen, Newyorkadam, BG19bot, Crispulop, Brk0 0, Mneykov and Anonymous: 8
- **Probably approximately correct learning** *Source:* https://en.wikipedia.org/wiki/Probably_approximately_correct_learning?oldid=601228074 *Contributors:* The Anome, SimonP, Michael Hardy, Kku, Hike395, Charles Matthews, Dysprosia, Populus, Andris, Daniel Brockman, APH, Gene s, Wimmerm, Mathbot, Zvika, Serg3d2, German name, John b cassel, A3nm, Martarius, MystBot, Addbot, Wzhx cc, JonathanWilliford, Twri, Rfgrove, RedBot, Ego White Tray, Zormit and Anonymous: 15
- **Algorithmic learning theory** *Source:* https://en.wikipedia.org/wiki/Algorithmic_learning_theory?oldid=593887386 *Contributors:* Michael Hardy, Hike395, Secretlondon, Onebyone, Mrdice, APH, Alansohn, Dsemeas, Nesbit, Cybercobra, Dreadstar, Jwfour, RichardF, Ferris37, Mattisse, James086, Melcombe, Yobot, AnomieBOT, John of Reading, Schulte.oliver, Jochen Burghardt and Anonymous: 7
- **Statistical hypothesis testing** *Source:* https://en.wikipedia.org/wiki/Statistical_hypothesis_testing?oldid=668899473 *Contributors:* The Anome, DavidSJ, Edward, Patrick, Michael Hardy, Gabbe, Dcljr, Ronz, Den fjättrade ankan~enwiki, Darkwind, Sir Paul, Poor Yorick, Cherkash, Trainspotter~enwiki, Dcoetzee, Robbot, Robbyjo~enwiki, Henrygb, Wile E. Heresiarch, Giftlite, J heisenberg, Jackol, Utcursch, Pgan002, Andycjp, Maneesh, Daniel11, Rich Farmbrough, AlexKepler, Bender235, Cyc~enwiki, Cretog8, Spalding, Johnkarp, Davidruben, Arcadian, NickSchweitzer, Nsaa, Mdd, Storm Rider, Alansohn, Gary, Guy Harris, Arthena, John Quiggin, Hu, Cburnett, Pixie, Ultramarine, InBalance, Oleg Alexandrov, Woohookitty, JonBirge, BlaiseFEgan, Btyner, Marudubshinki, Graham87, Rjwilmsi, Jake Wartenberg, Badjoby, Bdolicki, Pete.Hurd, Pstevens, Benlisquare, Adoniscik, YurikBot, Wavelength, RobotE, Xdenizen, Crasshopper, Bota47, Ott2, NormDor, TrickyTank, Digfarenough, Andrew73, Zvika, SmackBot, Reedy, Tom Lougheed, Mcld, Chris the speller, Feinstein, Jprg1966, Bazonka, Nbarth, Jyeee, Danielkueh, Jmnbatista, Robma, Radagast83, Cybercobra, Nakon, Richard001, G716, Salamurai, Lambiam, Tim bates, Nijdam, IronGargoyle, Slakr, Varuag doos, Meritus~enwiki, Hu12, Pejman47, Rory O'Kane, MystRivenExile, Levineps, K, Jason.grossman, AbsolutDan, Jackzhp, Mudd1, Thomasmeeks, Requestion, Pewwer42, Cydebot, Reywas92, B, Michael C Price, Ggchen, Mattisse, JamesAM, Talgalili, Thijs!bot, Epbr123, Wikid77, Crazy george, Adjespers, Philippe, Urdutext, AntiVandalBot, Lordmetroid, NYC2TLV, JAnDbot, Mikelove1, Andonic, SiobhanHansa, Acroterion, Magioladitis, VoABot II, Albmont, Pdbogen, Sullivan.t.j, Cydmab, Kateshortforbob, Mbhiii, Xiphosurus, Ulyssesmsu, Silas S. Brown, Coppertwig, Policron, Dhaluza, Juliancolton, DavidCBryant, DeFaultRyan, Speciate, Sam Blacketer, VolkovBot, LeilaniLad, Agricola44, FreeT, Someguy1221, Strategist333, Snowbot, Jcchurch, Finnrind, Zheric~enwiki, Statlearn, Krawi, Dailyknowledge, Bentogoa, Flyer22, Drakedirect, Ddxc, Larjohn~enwiki, Czap42, Kjtobo, Melcombe, Bradford rob, Sfan00 IMG, ClueBot, Aua, Shabbychef, Catraeus, Adamjslund, Skbkekas, The-tenth-zdog, Qwfp, DumZiBoT, Wolverineski, Terry0051, Libcub, Tayste, Addbot, Mortense, Some jerk on the Internet, Fgnievinski, Innv, דולב, Ryanblak, MrOllie, Protonk, CarsracBot, West.andrew.g, Tanath, Luckas-bot, Yobot, Andresswift, Brougham96, AnomieBOT, Materialscientist, Citation bot, Xqbot, Srich32977, J04n, Thosjleep, JonDePlume, A.amitkumar, FrescoBot, Jollyroger131, Citation bot 1, DrilBot, Boxplot, Wittygrittydude, RedBot, Ivancho.was.here, Bbarkley2, Thoytpt, Aurorion, Verlainer, Kastchei, J36miles, John of Reading, Philippe (WMF), GoingBatty, Andreim27, Tsujimasen, Eukaryote89, Hatshepsut1988, ClueBot NG, BWoodrum, Kkddkkdd, Psy1235, JimsMaher, Yg12, Rezabot, Widr, Trift, Helpful Pixie Bot, BG19bot, MasterMind5991, Mechnikov, AvocatoBot, Ryan.morton, Op47, Emanuele.olivetti, Manoguru, Leapsword, BattyBot, Attleboro, Viraltux, Tibbyshep, TheJJJunk, Illia Connell, Knappsych, Citrusccoconut, GargantuanDan, TejDham, Valcust, Jamesmcmahon0, Waynechew87, Nullhypothesistester, Penitence, Anrnusna, Kfudala, Tertius51, Monkbot, BethNaught, TedPSS, 1980na, Libarian, SolidPhase, Anon124, Zjillanz and Anonymous: 324
- **Bayesian inference** *Source:* https://en.wikipedia.org/wiki/Bayesian_inference?oldid=670136415 *Contributors:* The Anome, Fubar Obfusco, DavidSJ, Jinian, Edward, JohnOwens, Michael Hardy, Lexor, Karada, Ronz, Suisui, Den fjättrade ankan~enwiki, LouI, EdH, Jonik, Hike395, Novum, Timwi, WhisperToMe, Selket, SEWilco, Jose Ramos, Insightaction, Banno, Robbot, Kiwibird, Benwing, Meduz, Henrygb, AceMyth, Wile E. Heresiarch, Ancheta Wis, Giftlite, DavidCary, Dratman, Leonard G., JimD, Wmahan, Pcarbonn, MarkSweep, L353a1, FelineAvenger, APH, Sam Hocevar, Perey, Discospinster, Rich Farmbrough, Bender235, ZeroOne, Donsimon~enwiki, MisterSheik, El C, Edward Z. Yang, DimaDorfman, Cje~enwiki, John Vandenberg, LeonardoGregianin, Jung dalglish, Hooperbloob, Landroni, Arcenciel, Nurban, Avenue, Cburnett, Jheald, Facopad, Sjara, Oleg Alexandrov, Roylee, Joriki, Mindmatrix, BlaiseFEgan, Btyner, Magister Mathematicae, Tlroche, Rjwilmsi, Ravik, Jeffmcneill, Billjefferys, FlaBot, Brendan642, Kri, Chobot, Reetep, Gdrbot, Adoniscik, Wavelength, Pacaro, Gaius Cornelius, ENeville, Dysmorodrepanis~enwiki, Snek01, BenBildstein, Modify, Mastercampbell, Nothlit, NielsenGW, Mebden, Bo Jacoby, Cmglee, Boggie~enwiki, Harthacnut, SmackBot, Mmernex, Rtc, Mcld, Cunya, Gilliam, DoctorW, Nbarth, G716, Jbergquist, Turms, Bejnar, Gh02t, Wyxel, Josephsieh, JeonghunNoh, Thermochap, BoH, Basar, TheRegicider, Farzaneh, Lindsay658, Tdunning, Helgus, EdJohnston, Jvstone, Mack2, Lfstevens, Makohn, Stephanhartmannde, Comrade jo, Ph.eyes, Coffee2theorems, Ling.Nut, Charlesbaldo, DAGwyn, User A1, Tercer, STBot, Tobyr2, LittleHow, Policron, Jeffbadge, Bhepburn, Robcalver, James Kidd, VolkovBot, Thedjatclubrock, Maghnus, TXiKiBoT, Andrewaskew, GirasoleDE, SieBot, Doctorfree, Natta.d, Anchor Link Bot, Melcombe, Kvihill, Rfinchdavis, Smithpith, GeneCallahan, Krogstadt, Reovalis, Hussainshafqat, Charledl, ERosa, Qwfp, Tdslk, XLinkBot, Erreip, Addbot, K-MUS, Metagraph, LaaknorBot, Ozob, Legobot, Yobot, Gongshow, AnomieBOT, Citation bot, Shadak, Danielshin, VladimirReshetnikov, KingScot, JonDePlume, Thehelpfulbot, FrescoBot, Olexa Riznyk, WhatWasDone, Haeinous, JFK0502, Kiefer.Wolfowitz, 124Nick, Night Jaguar, Scientist2, Trappist the monk, Gnathan87, Philocentric, Jonkerz, Jowa fan, EmausBot, Blumehua, Montgolfière, Moswento, McPastry, Bagrowjp, SporkBot, Willy.pregliasco, Floombottle, Epdeloso, ClueBot NG, Mathstat, Bayes Puppy, Jj1236, Alberttttt, Thepigdog, Helpful Pixie Bot, Michael.d.larkin, Jeraphine Gryphon, Whyking thc, Intervallic, CitationCleanerBot, DaleSpam, Kaseton, Simonsm21, Danielribeirosilva, ChrisGualtieri, Alialamifard, Yongli Han, 90b56587, MittensR, Mark viking, Boomx09, Waynechew87, Hamoudafg, Promise her a definition, Abacenis, Engheta, Avehtari, SolidPhase, LadyLeodia, KasparBot and Anonymous: 228
- **Chi-squared distribution** *Source:* https://en.wikipedia.org/wiki/Chi-squared_distribution?oldid=666620239 *Contributors:* AxelBoldt, Bryan Derksen, The Anome, Ap, Michael Hardy, Stephen C. Carlson, Tomi, Mdebets, Ronz, Den fjättrade ankan~enwiki, Willem, Jitse Niesen, Hgamboa, Fibonacci, Zero0000, AaronSw, Robbot, Sander123, Seglea, Henrygb, Robinh, Isopropyl, Weialawaga~enwiki, Giftlite,

Dbenbenn, BenFrantzDale, Herbee, Sietse, MarkSweep, Gauss, Zfr, Fintor, Rich Farmbrough, Dbachmann, Paul August, Bender235, MisterSheik, O18, TheProject, NickSchweitzer, Iav, Jumbuck, B k, Kotasik, Sligocki, PAR, Cburnett, Shoefly, Oleg Alexandrov, Mindmatrix, Btyner, Rjwilmsi, Pahan~enwiki, Salix alba, FlaBot, Alvin-cs, Pstevens, Philten, Roboto de Ajvol, YurikBot, Wavelength, Schmock, Tony1, Zwobot, Jspacemen01-wiki, Reyk, Zvika, KnightRider~enwiki, SmackBot, Eskimbot, BiT, Afa86, Bluebot, TimBentley, Master of Puppets, Silly rabbit, Nbarth, AdamSmithee, Iwaterpolo, Eliezg, Robma, A.R., G716, Saippuakauppias, Rigadoun, Loodog, Mgiganteus1, Qiuxing, Funnybunny, Chris53516, Tawkerbot2, Jackzhp, CBM, Rflrob, Dgw, FilipeS, Blaisorblade, Talgalili, Thijs!bot, DanSoper, Lovibond, Pabristow, MER-C, Plantsurfer, Mcorazao, J-stan, Leotolstoy, Wasell, VoABot II, Jaekrystyn, User A1, TheRanger, MartinBot, STBot, Steve8675309, Neon white, Icseaturtles, It Is Me Here, TomyDuby, Mikael Häggström, Quantling, Policron, Nm420, HyDeckar, Sam Blacketer, DrMicro, LeilaniLad, Gaara144, AstroWiki, Notatoad, Johnlv12, Wesamuels, Tarkashastri, Quietbritishjim, Rlendog, Sheppa28, Phe-bot, Jason Goldstick, Tombomp, OKBot, Melcombe, Digisus, Volkan.cevher, Loren.wilton, Animeronin, ClueBot, Jdgilbey, MATThematical, UKoch, SamuelTheGhost, EtudiantEco, Bluemaster, Qwfp, XLinkBot, Knetlalala, MystBot, Paulginz, Fergikush, Tayste, Addbot, Fgnievinski, Fieldday-sunday, MrOllie, Download, LaaknorBot, Renatokeshet, Lightbot, Ettrig, Chaldor, Luckas-bot, Yobot, Wjastle, Johnlemartirao, AnomieBOT, Microball, MtBell, Materialscientist, Geek1337~enwiki, EOBarnett, DirlBot, LilHelpa, Lixiaoxu, Xqbot, Eliel Jimenez, Etoombs, Control.valve, NocturneNoir, GrouchoBot, RibotBOT, Entropeter, Shadowjams, Griffinofwales, Constructive editor, FrescoBot, Tom.Reding, Stpasha, MastiBot, Gperjim, Fergusq, Xnn, RjwilmsiBot, Kastchei, Alph Bot, Wassermann7, Markg0803, EmausBot, Yuzisee, Dai bach, Pet3ris, U+003F, Zephyrus Tavvier, Levdtrotsky, ChuispastonBot, Emilpohl, Brycehughes, ClueBot NG, BG19bot, Analytics447, Snouffy, Drhowey, Dlituiev, Minsbot, HelicopterLlama, Limit-theorem, Ameer diaa, Idoz he, Zjbranson, DonaghHorgan, Catalin.ghervase, BeyondNormality, Monkbot, Alakzi, Bderrett, Uceeylu and Anonymous: 238

- **Chi-squared test** *Source:* https://en.wikipedia.org/wiki/Chi-squared_test?oldid=667776610 *Contributors:* The Anome, Matusz, Michael Hardy, Tomi, Karada, Ronz, Ciphergoth, Jfitzg, Mxn, Silverfish, Crissov, Robbot, Giftlite, Andris, Matt Crypto, MarkSweep, Piotrus, Elektron, Rich Farmbrough, Cap'n Refsmmat, Kwamikagami, Smalljim, PAR, Stefan.karpinski, Spangineer, Wtmitchell, Falcorian, Bluemoose, Aatombomb, Strait, MZMcBride, Yar Kramer, JoseMires~enwiki, Intgr, Pstevens, YurikBot, Wavelength, Darker Dreams, DYLAN LENNON~enwiki, Avraham, Arthur Rubin, Reyk, SmackBot, Turadg, Nbarth, Scwlong, Whpq, Bowlhover, G716, Lambiam, Cronholm144, Loodog, Tim bates, Smith609, Beetstra, Chris53516, Usgnus, Dgw, Requestion, WeggeBot, Steel, Karuna8, Talgalili, Thijs!bot, Adjespers, Itsmejudith, AntiVandalBot, ReviewDude, Seaphoto, Johannes Simon, Ranger2006, Baccyak4H, KenyaSong, Serviscope Minor, MartinBot, Poeloq, Lbeaumont, Khatterj, JoshuaEyer, STBotD, VolkovBot, Pleasantville, Grotendeels Onschadelijk, Synthebot, Ignoscient, SieBot, Matthew Yeager, Quest for Truth, Svick, Melcombe, Digisus, Tuxa, Animeronin, ClueBot, Muhandes, Qwfp, Tdslk, WikHead, SilvonenBot, Prax54, Sindbad72, Tayste, Addbot, Luzingit, Doronp, MrOllie, Bhdavis1978, Legobot, Luckas-bot, Yobot, Amirobot, AnomieBOT, Walter Grassroot, Unara, Jtamad, KuRiZu, GrouchoBot, Joxemai, Thehelpfulbot, Pinethicket, I dream of horses, Madonius, Kastchei, EmausBot, Kgwet, Lolcatsdeamon13, Orange Suede Sofa, Levdtrotsky, ClueBot NG, Hyiltiz, Ion vasilief, Epfuerst, Helpful Pixie Bot, Evcifreo, Chafe66, Jf.alcover, Aymankamelwiki, Brirush, RichardMarioFratini, MNikulin, EJM86, BethNaught, Hannasnow, Iwilsonp, Pentaquark and Anonymous: 140
- **Goodness of fit** *Source:* https://en.wikipedia.org/wiki/Goodness_of_fit?oldid=664740099 *Contributors:* Khendon, Michael Hardy, Ronz, Den fjättrade ankan~enwiki, Benwing, David Edgar, Giftlite, ReallyNiceGuy, Army1987, NickSchweitzer, Keflavich, Alkarex, Btyner, Demian12358, YurikBot, Wavelength, Amakuha, Jon Olav Vik, Carlosguitar, Slashme, Kslays, Chris the speller, BostonMA, Mwtoews, Nutcracker, Dicklyon, Belizefan, Jayen466, Mr Gronk, Talgalili, Thijs!bot, Danger, Ph.eyes, Fjalokin, Glrx, Bonadea, SueHay, Llamabr, Tomaxer, Melcombe, ClueBot, Tomas e, Hoskee, Qwfp, DumZiBoT, Addbot, Fgnievinski, Renatokeshet, AnomieBOT, Gumlicks, Joxemai, Fortdj33, Kastchei, John of Reading, Dai bach, JordiGH, Mathstat, MerlIwBot, Bhaveshpatil04 and Anonymous: 51
- **Likelihood-ratio test** *Source:* https://en.wikipedia.org/wiki/Likelihood-ratio_test?oldid=668781783 *Contributors:* The Anome, Fnielsen, Torfason, Michael Hardy, Kku, Notheruser, Den fjättrade ankan~enwiki, Jfitzg, Cherkash, Unknown, Seglea, Meduz, Babbage, Henrygb, Elysdir, Robinh, Giftlite, Pgan002, MarkSweep, Corti, Bender235, El C, Arcadian, Seans Potato Business, Cburnett, Jheald, Oleg Alexandrov, Btyner, Graham87, NeoUrfahraner, Pete.Hurd, Thecurran, Adoniscik, YurikBot, Cancan101, Draeco, Robertvan1, RL0919, Nescio, Badgettrg, SmackBot, Tom Lougheed, Rajah9, Nbarth, Yimmieg, Moverly, Tim bates, Dchudz, Smith609, AnRtist, Jackzhp, RobDe68, AgentPeppermint, Guy Macon, Mack2, Kniwor, JamesBWatson, TomyDuby, Quantling, Cmcnicoll, AlleborgoBot, Arknascar44, Adismalscientist, Jeremiahrounds, Melcombe, Mild Bill Hiccup, Wildland, 1ForTheMoney, Qwfp, Jmac2222, Prax54, Jht4060, Tayste, Addbot, DOI bot, Nilayvaish, Legobot, Zaqrfv, AnomieBOT, Citation bot, Twri, LilHelpa, ArcadianOnUnsecuredLoc, Kristján Jónasson, Fortdj33, Vthesniper, Octonion, Aryan1989, HRoestBot, Ridgeback22, Madbix, Kastchei, Salvio giuliano, EmausBot, Wiki091005!!, Fanyavizuri, Frietjes, Masssly, Sboludo, Helpful Pixie Bot, Fayue1015, NaftaliHarris, BG19bot, Chafe66, Limesave, Chuk.plante, Dexbot, Nm160111, Penitence, FrB.TG, TedPSS and Anonymous: 88
- **Statistical classification** *Source:* https://en.wikipedia.org/wiki/Statistical_classification?oldid=666781901 *Contributors:* The Anome, Michael Hardy, GTBacchus, Hike395, Robbot, Benwing, Giftlite, Beland, Violetriga, Kierano, Jérôme, Anthony Appleyard, Denoir, Oleg Alexandrov, Bkkbrad, Qwertyus, Bgwhite, Roboto de Ajvol, YurikBot, Jrbouldin, Dtrebbien, Tiffanicita, Tobi Kellner, SmackBot, Object01, Mcld, Chris the speller, Nervexmachina, Can't sleep, clown will eat me, Memming, Cybercobra, Richard001, Bohunk, Beetstra, Hu12, Billgaitas@hotmail.com, Trauber, Juansempere, Thijs!bot, Prolog, Mack2, Peteymills, VoABot II, Robotman1974, Quocminh9, RJASE1, Jamelan, ThomHImself, Gdupont, Junling, Melcombe, WikiBotas, Agor153, Addbot, Giggly37, Fgnievinski, SpBot, Movado73, Yobot, Oleginger, AnomieBOT, Ashershow1, Verbum Veritas, FrescoBot, Gire 3pich2005, DrilBot, Classifier1234, Jonkerz, Fly by Night, Microfries, Chire, Sigma0 1, Rmashhadi, ClueBot NG, Girish280, MerlIwBot, Helpful Pixie Bot, Chyvve, Swsboarder366, Klilidiplomus, Ferrarisailor, Mark viking, Francisbach, Imphil, I Less than3 Maths, LdyBruin and Anonymous: 65
- **Binary classification** *Source:* https://en.wikipedia.org/wiki/Binary_classification?oldid=668840507 *Contributors:* The Anome, Michael Hardy, Kku, Nichtich~enwiki, Janka~enwiki, Henrygb, Sepreece, Wmahan, Dfrankow, Nonpareility, 3mta3, Oleg Alexandrov, Linas, Btyner, Qwertyus, Salix alba, FlaBot, Jaraalbe, DRosenbach, RG2, SmackBot, Chris the speller, Nbarth, Mauro Bieg, Ebraminio, Amit Moscovich, Coolhandscot, Mstillman, STBot, Rlsheehan, Salih, Mikael Häggström, HELvet, Dr.007, Synthesis88, Jamelan, Thefellswooper, Pkgx, Melcombe, Denisarona, Mild Bill Hiccup, Qwfp, AndrewHZ, Fgnievinski, Yobot, AnomieBOT, Twri, Saeidpourbabak, FrescoBot, Duoduoduo, MartinThoma, Pablo Picossa, Alishahss75ali, Sds57, SoledadKabocha, Alialamifard, Richard Kohar, DoctorTerrella, Loraof, Shirleyyoung0812 and Anonymous: 32
- **Maximum likelihood** *Source:* https://en.wikipedia.org/wiki/Maximum_likelihood?oldid=670023975 *Contributors:* The Anome, ChangChienFu, Patrick, Michael Hardy, Lexor, Dcljr, Karada, Ellywa, Den fjättrade ankan~enwiki, Cherkash, Hike395, Samsara, Phil Boswell, R3m0t, Guan, Henrygb, Robinh, Giftlite, DavidCary, BenFrantzDale, Chinasaur, Jason Quinn, Urhixidur, Rich Farmbrough, Rama, Chowells, Bender235, Maye, Violetriga, 3mta3, Arthena, Inky, PAR, Cburnett, Algocu, Ultramarine, Oleg Alexandrov, James I Hall, Rschulz, Btyner, Marudubshinki, Graham87, BD2412, Rjwilmsi, Koavf, Cjpuffin, Mathbot, Nivix, Jrtayloriv, Chobot, Reetep, YurikBot, Wavelength, Cancan101, Dysmorodrepanis~enwiki, Avraham, Saric, Bo Jacoby, XpXiXpY, Zvika, SolarMcPanel, SmackBot,

Royalguard11, Warren.cheung, Chris the speller, Nbarth, Hongooi, Juffi, Earlh, TedE, Dreadstar, G716, Qwerpoiu, Loodog, Lim Wei Quan, Rogerbrent, Hu12, Freeside3, Cbrown1023, Lavaka, Simo Kaupinmäki, 137 0, Travelbird, Headbomb, John254, Z10x, Nick Number, Binarybits, Alfalfahotshots, Magioladitis, Albmont, Livingthingdan, Baccyak4H, Julian Brown, A3nm, Cehc84, Algebraic, R'n'B, Lilac Soul, Samikrc, Rlsheehan, Gill110951, AntiSpamBot, Policron, MJamesCA, Mathuranathan, JeffreyRMiles, Slaunger, TXiKiBoT, Atabəy, JimJJewett, Jsd115, Ramiromagalhaes, Henrikholm, AlleborgoBot, Matt Gleeson, Logan, Zonuleofzinn, Quietbritishjim, BotMultichill, CurranH, Davidmosen, Svick, Melcombe, Classicalecon, Davyzhu, The Thing That Should Not Be, Drazick, Alexbot, NuclearWarfare, Qwfp, Agravier, Vitanyi, Ninja247, Addbot, DOI bot, Lucifer87, MrOllie, SpBot, Hawk8103, Westcoastr13, Luckas-bot, Yobot, Amirobot, Mathdrum, Zbodnar, AnomieBOT, DemocraticLuntz, RVS, Citation bot, ArthurBot, Xqbot, Flavio Guitian, Xappppp, Shadowjams, Af1523, FrescoBot, LucienBOT, Citation bot 1, EduardoValle, Kiefer.Wolfowitz, Jmc200, Stpasha, BPets, Casp11, Dimtsit, Jingyu.cui, RjwilmsiBot, Brandynwhite, Set theorist, Cal-linux, K6ka, Chadhoward, DavidMCEddy, Alexey.kudinkin, JA(000)Davidson, SporkBot, Zfeinst, Zueignung, Gjshisha, Mikhail Ryazanov, ClueBot NG, Gareth Griffith-Jones, MelbourneStar, Arthurcburigo, Frietjes, Delusion23, Nak9x, Helpful Pixie Bot, Tony Tan, TheMathAddict, Dlituiev, Khazar2, Illia Connell, JYBot, Oneway3124, Jamesmcmahon0, Deschwartz, Hamoudafg, Crispulop, LokeshRavindranathan, Monkbot, Engheta, Isambard Kingdom and Anonymous: 200

- **Linear classifier** *Source:* https://en.wikipedia.org/wiki/Linear_classifier?oldid=669802403 *Contributors:* The Anome, Hike395, WhisperToMe, Rls, Benwing, Bernhard Bauer, Wile E. Heresiarch, BenFrantzDale, Neilc, MarkSweep, Bobo192, Jung dalglish, Arcenciel, Oleg Alexandrov, Linas, Bluemoose, Qwertyus, Mathbot, Sderose, Daniel Mietchen, SmackBot, Hongooi, Mcswell, Marcuscalabresus, Thijs!bot, Camphor, AnAj, Dougher, BrotherE, TXiKiBoT, Qxz, Phe-bot, Melcombe, SPiNoZA, Jakarr, Addbot, Yobot, AnomieBOT, Sgoder and Anonymous: 21
- **Logistic regression** *Source:* https://en.wikipedia.org/wiki/Logistic_regression?oldid=671175842 *Contributors:* Twanvl, Michael Hardy, Tomi, Den fjättrade ankan~enwiki, Benwing, Gak, Giftlite, BrendanH, YapaTi~enwiki, Dfrankow, Pgan002, Bolo1729, Qef, Rich Farmbrough, Mani1, Bender235, Mdf, O18, NickSchweitzer, CarrKnight, Kierano, Arthena, Velella, Oleg Alexandrov, LOL, BlaiseFEgan, Qwertyus, Rjwilmsi, Ground Zero, Sderose, Shaggyjacobs, YurikBot, Wavelength, Cancan101, Johndburger, Rodolfo Hermans, Jtneill, D nath1, Lassefolkersen, Aldaron, G716, Esrever, RomanSpa, Nutcracker, Cbuckley, Ionocube, Kenkleinman, Markjoseph125, David s graff, Jjoseph, Olberd, Requestion, Future Perfect at Sunrise, Kallerdis, Neoforma, LachlanA, Tomixdf, Mack2, JAnDbot, Every Creek Counts, Sanchom, Owenozier, Magioladitis, Baccyak4H, Nszilard, Mbhiii, Lilac Soul, Kpmiyapuram, Djjrjr, Ronny Gunnarsson, Ypetrachenko, Dvdpwiki, Squids and Chips, Mobeets, Ktalon, TXiKiBoT, Harrelfe, Antaltamas, Aaport, Jamelan, Synthebot, Dvandeventer, Anameofmyveryown, Prakash Nadkarni, BAnstedt, Junling, AlanUS, Melcombe, Sphilbrick, Denisarona, Alpapad, Statone, SchreiberBike, Cmelan, Aprock, Qwfp, XLinkBot, WikHead, Tayste, Addbot, Kurttg~enwiki, New Image Uploader 929, Luckas-bot, Yobot, Secondsminutes, AnomieBOT, Ciphers, Materialscientist, Xqbot, Gtfjbl, FrescoBot, X7q, Orubt, Chenopodiaceous, Albertzeyer, Trappist the monk, Duoduoduo, Diannaa, RjwilmsiBot, Grumpfel, Strano.m, Mudx77, EmausBot, RMGunton, Alexey.kudinkin, Zephyrus Tavvier, Kchowdhary, Sigma0 1, DASHBotAV, Vldscore, Kjalarr, Mhsmith0, Timflutre, Helpful Pixie Bot, Ngocminh.oss, BG19bot, Martha6981, DPL bot, BattyBot, Guziran99, Eflatmajor7th, AndrewSmithQueen's, Jey42, JTravelman, SFK2, Yongli Han, Merespiz, Tentinator, Pkalczynski, Yoboho, Hayes.rachel, E8xE8, Tertius51, Nyashinski, ThuyNgocTran, Monkbot, SantiLak, Bluma.Gelley, Kamerondeckerharris, Srp54, P.thesling, Alexander Craig Russell, Екатерина Конь, Velvel2, Anuragsodhi, Hughkf, PushTheButton108, LockemyCock, Gzluyongxi and Anonymous: 182
- **Linear discriminant analysis** *Source:* https://en.wikipedia.org/wiki/Linear_discriminant_analysis?oldid=668696480 *Contributors:* The Anome, Fnielsen, XJaM, Edward, Michael Hardy, Kku, Den fjättrade ankan~enwiki, Hike395, Jihg, Deus~enwiki, Nonick, Giftlite, Duncharris, Dfrankow, Pgan002, 3mta3, Arcenciel, Forderud, Crackerbelly, Qwertyus, Rjwilmsi, Mathbot, Predictor, Adoniscik, YurikBot, Timholy, Doncram, Tcooke, Shawnc, SmackBot, Maksim-e~enwiki, Slashme, Mcld, Memming, Solarapex, Beetstra, Dicklyon, Lifeartist, StanfordProgrammer, Petrus Adamus, Sopoforic, Cydebot, Thijs!bot, AlexAlex, AnAj, Mack2, Cpl Syx, Stephenchou0722, Lwaldron, R'n'B, G.kunter~enwiki, Nechamayaniger, Daviddoria, SieBot, Ivan Štambuk, Mverleg, Jonomillin, OKBot, Melcombe, Produit, Statone, Calimo, Qwfp, Addbot, Mabdul, AndrewHZ, Lightbot, Yobot, Citation bot, Klisanor, Sylwia Ufnalska, Morten Isaksen, Olg wiki, SchnitzelMannGreek, Pcoat, FrescoBot, X7q, Citation bot 1, Wkretzsch, Heavy Joke, Www wwwjs1, Jfmantis, EmausBot, Dewritech, Radshashi, Manyu aditya, Marion.cuny, Vldscore, WikiMSL, Helpful Pixie Bot, BG19bot, CitationCleanerBot, Khazar2, Illia Connell, I am One of Many, Lcparra, Ashleyleia, ArtfulVampire, SJ Defender, Екатерина Конь, Degill, Olosko and Anonymous: 107
- **Naive Bayes classifier** *Source:* https://en.wikipedia.org/wiki/Naive_Bayes_classifier?oldid=669331569 *Contributors:* The Anome, Awaterl, Olivier, Michael Hardy, Bewildebeast, Zeno Gantner, Karada, Cyp, Den fjättrade ankan~enwiki, Hike395, Njoshi3, WhisperToMe, Toreau, Phil Boswell, RedWolf, Bkell, Wile E. Heresiarch, Giftlite, Akella, JimD, Bovlb, Macrakis, Neilc, Pgan002, MarkSweep, Gene s, Cagri, Anirvan, Trevor MacInnis, Thorwald, Splatty, Rich Farmbrough, Violetriga, Peterjoel, Smalljim, John Vandenberg, BlueNovember, Jason Davies, Caesura, Oleg Alexandrov, KKramer~enwiki, Btyner, Mandarax, Qwertyus, Rjwilmsi, Hgkamath, Johnnyw, Mathbot, Intgr, Sderose, YurikBot, Wavelength, PiAndWhippedCream, Cancan101, Bovineone, Arichnad, Karipuf, BOT-Superzerocool, Evryman, Johndburger, Mebden, XAVeRY, SmackBot, InverseHypercube, CommodiCast, Stimpy, ToddDeLuca, Gilliam, NickGarvey, Chris the speller, OrangeDog, PerVognsen, Can't sleep, clown will eat me, Memming, Mitar, Neshatian, Jklin, Ringger, WModNS, Tobym, Shorespirit, Mat1971, Dstanfor, Arauzo, Dantiston, Sytelus, Vera Rita~enwiki, Dkemper, Prolog, Ninjakannon, Jrennie, MSBOT, Coffee2theorems, Tremilux, Saurabh911, Robotman1974, David Eppstein, User A1, HebrewHammerTime, AllenDowney, Troos, AntiSpamBot, Newtman, STBotD, Mike V, RJASE1, VolkovBot, Maghnus, Anna Lincoln, Mbusux, Anders gorm, EverGreg, Fcady2007, Jojalozzo, Ddxc, Dchwalisz, AlanUS, Melcombe, Headlessplatter, Kotsiantis, Justin W Smith, Motmahp, Calimo, Dianegarey, Doobliebop, Alousybum, Sunsetsky, XLinkBot, Herlocker, Addbot, RPHv, Tsunanet, MrOllie, LaaknorBot, Yobot, TaBOT-zerem, Twexcom, AnomieBOT, Rubinbot, Smk65536, The Almighty Bob, Cantons-de-l'Est, گس‌تهم, FrescoBot, X7q, Proffviktor, Svourdroculed, Rickyphyllis, Jonesey95, Geoffrey I Webb, Classifier1234, Mwojnars, Wingiii, Larry.europe, Helwr, EmausBot, Orphan Wiki, Tommy2010, GarouDan, Joseagonzalez, ClueBot NG, Hofmic, NilsHaldenwang, Luoli2000, BG19bot, MusikAnimal, Chafe66, Kavishwar.wagholikar, Geduowenyang, Hipponix, Fcbarbi, Librawill, ChrisGualtieri, XMU zhangy, Alialamifard, CorvetteC6RVip, Jamesmcmahon0, Tonytonov, Jmagasin, ScienceRandomness, Qingyuanxingsi, Micpalmia, Sofia Koutsouveli, Yuchsiao, Mvdyck, Don neufeld, YoniSmolin, Rapanshi, Ananth.sankar.1963, Hmerzic and Anonymous: 184
- **Cross-validation (statistics)** *Source:* https://en.wikipedia.org/wiki/Cross-validation_(statistics)?oldid=668481251 *Contributors:* Michael Hardy, Shyamal, Delirium, Den fjättrade ankan~enwiki, Hike395, Phil Boswell, Cutler, Pgan002, Urhixidur, Discospinster, 3mta3, Bfg, Jörg Knappen~enwiki, GregorB, Btyner, Qwertyus, Rjwilmsi, Mattopia, BMF81, Wavelength, Rsrikanth05, Bruguiea, Saric, Zvika, Capitalist, SmackBot, Glvgfz, Stimpy, Nbarth, Iridescent, CmdrObot, Olaf Davis, Gogo Dodo, Rphirsch, Blaisorblade, Headbomb, Mgierdal, Fogeltje, Alanmalek, AnAj, Onasraou, JAnDbot, Olaf, Necroforest, Johnbibby, Paresnah, Jiuguang Wang, VolkovBot, Jamelan, SieBot, Anchor Link Bot, Melcombe, Headlessplatter, Calimo, Skbkekas, Rbeg, XLinkBot, MystBot, Addbot, Sohail stat, Fieldday-sunday, Josevellezcaldas, Movado73, Legobot, Yobot, Materialscientist, Citation bot, Xqbot, Georg Stillfried, WaysToEscape, Allion, X7q, Codemonkey87, Duoduoduo, Ulatekh, Noblestats, WikitanvirBot, Mjollnir82, ZéroBot, H3llBot, Donner60, Robertschulze, ClueBot NG, Widr,

Helpful Pixie Bot, Beaumont877, AdventurousSquirrel, ChrisGualtieri, Khazar2, Clevera, Rolf h nelson, Pandadai, Winlose378, Cosminstamate, Emmanuel-L.T, Degill, Mberming, Kouroshbehzadian and Anonymous: 79

- **Unsupervised learning** *Source:* https://en.wikipedia.org/wiki/Unsupervised_learning?oldid=660135356 *Contributors:* Michael Hardy, Kku, Alfio, Ahoerstemeier, Hike395, Ojigiri~enwiki, Gene s, Urhixidur, Alex Kosorukoff, Aaronbrick, Bobo192, 3mta3, Tablizer, Denoir, Nkour, Qwertyus, Rjwilmsi, Chobot, Roboto de Ajvol, YurikBot, Darker Dreams, Daniel Mietchen, SmackBot, CommodiCast, Trebor, DHN-bot~enwiki, Lambiam, CRGreathouse, Carstensen, Thijs!bot, Jaxelrod, AnAj, Peteymills, David Eppstein, Agentesegreto, Maheshbest, Timohonkela, Ng.j, EverGreg, Algorithms, Kotsiantis, Auntof6, PixelBot, Edg2103, Addbot, EjsBot, Yobot, Les boys, AnomieBOT, Salvamoreno, D'ohBot, Skyerise, Ranjan.acharyya, BertSeghers, EmausBot, Fly by Night, Rotcaeroib, Stheodor, Daryakav, Ida Shaw, Chire, Candace Gillhoolley, WikiMSL, Helpful Pixie Bot, Majidjanz and Anonymous: 40
- **Cluster analysis** *Source:* https://en.wikipedia.org/wiki/Cluster_analysis?oldid=667300175 *Contributors:* The Anome, Fnielsen, Nealmcb, Michael Hardy, Shyamal, Kku, Tomi, GTBacchus, Den fjättrade ankan~enwiki, Cherkash, BAxelrod, Hike395, Dbabbitt, Phil Boswell, Robbot, Gandalf61, Babbage, Aetheling, Giftlite, Lcgarcia, Cfp, BenFrantzDale, Soundray~enwiki, Ketil, Khalid hassani, Angelo.romano, Dfrankow, Gadfium, Pgan002, Gene s, EBB, Sam Hocevar, Pwaring, Jutta, Abdull, Bryan Barnard, Rich Farmbrough, Mathiasl26, NeuronExMachina, Yersinia~enwiki, Bender235, Alex Kosorukoff, Aaronbrick, John Vandenberg, Greenleaf~enwiki, Ahc, NickSchweitzer, 3mta3, Jonsafari, Jumbuck, Jérôme, Terrycojones, Denoir, Jnothman, Stefan.karpinski, Hazard, Oleg Alexandrov, Soultaco, Woohookitty, Linas, Uncle G, Borb, Ruud Koot, Tabletop, Male1979, Joerg Kurt Wegner, DESiegel, Ruziklan, Sideris, BD2412, Qwertyus, Rjwilmsi, Koavf, Salix alba, Michal.burda, Denis Diderot, Klonimus, FlaBot, Mathbot, BananaLanguage, Kcarnold, Payo, Jrtayloriv, Windharp, BMF81, Roboto de Ajvol, The Rambling Man, YurikBot, Wavelength, Argav, SpuriousQ, Pseudomonas, NawlinWiki, Gareth Jones, Bayle Shanks, TCrossland, JFD, Hirak 99, Zzuuzz, Rudrasharman, Zigzaglee, Closedmouth, Dontaskme, Kevin, Killerandy, Airconswitch, SmackBot, Drakyoko, Jtneill, Pkirlin, Object01, Mcld, Ohnoitsjamie, KaragouniS, Bryan Barnard1, MalafayaBot, Drewnoakes, Tenawy, DHN-bot~enwiki, Iwaterpolo, Zacronos, MatthewKarlsen, Krexer, Bohunk, MOO, Lambiam, Friend of facts, Benash, ThomasHofmann, Dfass, Beetstra, Ryulong, Nabeth, Hu12, Iridescent, Ralf Klinkenberg, Madla~enwiki, Alanbino, Origin415, Bairam, Ioannes Pragensis, Joaoluis, Megannnn, Nczempin, Harej bot, Slack---line, Playtime, Endpoint, Dgtized, Skittleys, DumbBOT, Talgalili, Thijs!bot, Barticus88, Vinoduec, Mailseth, Danhoppe, Phoolimin, Onasraou, Denaxas, AndreasWittenstein, Daytona2, MikeLynch, JAnDbot, Inverse.chi, .anacondabot, Magioladitis, Andrimirzal, Fallschirmjäger, JBIdF, David Eppstein, User A1, Eeera, Varun raptor, LedgendGamer, Jiuguang Wang, Sommersprosse, Koko90, Smite-Meister, McSly, Dvdpwiki, DavidCBryant, AStrathman, Camrn86, TXiKiBoT, Rnc000, Tamás Kádár, Mundhenk, Maxim, Winterschlaefer, Lamro, Wheatin, Arrenbas, Sesilbumfluff, Tomfy, Kerveros 99, Seemu, WRK, Drdan14, Harveydrone, Graham853, Wcdriscoll, Zwerglein~enwiki, Osian.h, FghIJklm, Melcombe, Kotsiantis, Freeman77, Victor Chmara, Kl4m, Mugvin, Manuel freire, Boing! said Zebedee, Tim32, PixelBot, Lartoven, Chaosdruid, Aprock, Practical321, Qwfp, FORTRANslinger, Sunsetsky, Ocean931, Phantom xxiii, XLinkBot, Pichpich, Gnowor, Sujaykoduri, WikHead, Addbot, Allenchue, DOI bot, Bruce rennes, Fgnievinski, Gangcai, MrOllie, FerrousTigrus, Delaszk, Tide rolls, Lightbot, PAvdK, Fjrohlf, Tobi, Luckas-bot, Yobot, Gulfera, Hungpuiki, AnomieBOT, Flamableconcrete, Materialscientist, Citation bot, Xqbot, Erud, Sylwia Ufnalska, Simeon87, Omnipaedista, Kamitsaha, Playthebass, FrescoBot, Sacomoto, D'ohBot, Dan Golding, JohnMeier, Slowmo0815, Atlantia, Citation bot 1, Boxplot, Edfox0714, MondalorBot, Lotje, E.V.Krishnamurthy, Capez1, Koozedine, Tbalius, RjwilmsiBot, Ripchip Bot, Jchemmanoor, GodfriedToussaint, Aaronzat, Helwr, EmausBot, John of Reading, Stheodor, Elixirrixile, BOUMEDJOUT, ZéroBot, Sgoder, Chire, Darthhappyface, Jucypsycho, RockMagnetist, Wakebrdkid, Fazlican, Anita5192, ClueBot NG, Marion.cuny, Ericfouh, Simeos, Poirel, Robiminer, Michael-stanton, Girish280, Helpful Pixie Bot, Novusuna, BG19bot, Cpkex0102, Wiki13, TimSwast, Cricetus, Douglas H Fisher, Mu.ting, ColanR, Cornelius3, Illia Connell, Compsim, Mogism, Frosty, Abewley, Mark viking, Metcalm, Ninjarua, Trouveur de faits, TCMemoire, ErezHartuv, Monkbot, Leegrc, Imsubhashjha, Екатерина Конь, Olosko, Angelababy00 and Anonymous: 327
- **Expectation–maximization algorithm** *Source:* https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm?oldid=671233060 *Contributors:* Rodrigob, Michael Hardy, Karada, Jrauser, BAxelrod, Hike395, Phil Boswell, Owenman, Robbyjo~enwiki, Benwing, Wile E. Heresiarch, Giftlite, Paisa, Vadmium, Onco p53, MarkSweep, Piotrus, Cataphract, Rama, MisterSheik, Alex Kosorukoff, O18, John Vandenberg, Jjmerelo~enwiki, 3mta3, Terrycojones, B k, Eric Kvaalen, Cburnett, Finfobia, Jheald, Forderud, Sergey Dmitriev, Igny, Bkkbrad, Bluemoose, Btyner, Qwertyus, Rjwilmsi, KYPark, Salix alba, Hild, Mathbot, Glopk, Kri, BradBeattie, YurikBot, Nils Grimsmo, Schmock, Régis B., Klutzy, Hakeem.gadi, Maechler, Ladypine, M.A.Dabbah, SmackBot, Mcld, Nbarth, Tekhnofiend, Iwaterpolo, Bilgrau, Joeyo, Raptur, Derek farn, Jrouquie, Dicklyon, Alex Selby, Saviourmachine, Lavaka, Requestion, Cydebot, A876, Kallerdis, Libro0, Blaisorblade, Skittleys, Andyrew609, Talgalili, Tiedyeina, Rusmike, Headbomb, RobHar, LachlanA, AnAj, Zzpmarco, Dekimasu, JamesBWatson, Richard Bartholomew, Livingthingdan, Nkwatra, User A1, Edratzer, Osquar F, Numbo3, Salih, GongYi, Douglas-Lanman, Bigredbrain, Market Efficiency, Lamro, Daviddoria, Pine900, Tambal, Mosaliganti1.1, Melcombe, Sitush, Pratx, Alexbot, Hbeigi, Jakarr, Jwmarck, XLinkBot, Jamshidian, Addbot, Sunjuren, Fgnievinski, LaaknorBot, Aanthony1243, Peni, Luckas-bot, Yobot, LeonardoWeiss, AnomieBOT, Citation bot, TechBot, Chuanren, FrescoBot, Nageh, Erhanbas, Nocheenlatierra, Qiemem, Kiefer.Wolfowitz, Jmc200, Stpasha, Jszymon, GeypycGn, Trappist the monk, Thái Nhi, Ismailari, Dropsciencenotbombs, RjwilmsiBot, Slon02, EmausBot, Mikealandewar, John of Reading, Ш, Chire, Statna, ClueBot NG, Rezabot, Meea, Qwerty9967, Helpful Pixie Bot, Rxnt, Bibcode Bot, BG19bot, Chafe66, Whym, Lvilnis, BattyBot, Yasuo2, Illia Connell, JYBot, Blegat, Yogtad, Tentinator, Marko0991, Ginsuloft, Wccsnow, Ronniemaor, Monkbot, Nboley, Faror91, DilumA, Rider ranger47, Velvel2, Crimsonslide, Megadata tensor, Surbut, Greatwave and Anonymous: 151
- **K-means clustering** *Source:* https://en.wikipedia.org/wiki/K-means_clustering?oldid=671161262 *Contributors:* Fnielsen, Michael Hardy, Ixfd64, Den fjättrade ankan~enwiki, Charles Matthews, Dbabbitt, Phil Boswell, Ashwin, Pengo, Giftlite, BenFrantzDale, Duncharris, Soren.harward, WorldsApart, Ratiocinate, Gazpacho, Rich Farmbrough, Mathiasl26, Greenleaf~enwiki, 3mta3, Jonsafari, Andkaha, Ricky81682, Jnothman, Alai, Robert K S, Qwertyus, Rjwilmsi, Hgkamath, Miserlou, Gringer, Mathbot, Mahlon, Chobot, Bgwhite, UkPaolo, YurikBot, Wavelength, SpuriousQ, Annabel, Hakkinen, SamuelRiv, Leishi, Killerandy, SmackBot, Zanetu, Mauls, Mcld, Memming, Cronholm144, Barabum, Denshade, Mauro Bieg, CBM, Chrike, Chrisahn, Talgalili, Thijs!bot, June8th, N5iln, Headbomb, Nick Number, Phoolimin, Sanchom, Charibdis, Smartcat, Magioladitis, David Eppstein, Kzafer, Gfxguy, Turketwh, Stimpak, Mati22081979, Alirn, JohnBlackburne, TXiKiBoT, FedeLebron, ChrisDing, Corvus cornix, Ostrouchov, Yannis1962, Billinghurst, Maxlittle2007, Erniepan, Illuminated, Strife911, Weston.pace, Ntvuok, AlanUS, Melcombe, PerryTachett, MenoBot, DEEJAY JPM, DragonBot, Alexbot, Pot, Tbmurphy, Rcalhoun, Agor153, Qwfp, Niteskate, Tavlos, Avoided, Addbot, DOI bot, Foma84, Fgnievinski, Homncruse, Wfolta, AndresH, Yobot, AnomieBOT, Jim1138, Materialscientist, Citation bot, LilHelpa, Honkkis, Gtfjbl, Gilo1969, Simeon87, Woolleynick, Wonderful597, Dpf90, Foobarhoge, FrescoBot, Dan Golding, Phillipe Israel, Jonesey95, Cincoutprabu, Amkilpatrick, NedLevine, Ranumao, Larry.europe, Helwr, EmausBot, John of Reading, Lessbread, Manyu aditya, ZéroBot, Sgoder, Chire, Toninowiki, 0sm0sm0, Helpsome, ClueBot NG, Mathstat, Jack Greenmaven, Railwaycat, BlueScreenD, Jsanchezalmeida, BG19bot, MusikAnimal, Mark Arsten, SciCompTeacher, Chmarkine, Utacsecd, Amritamaz, EdwardH, Sundirac, BattyBot, Illia Connell, MarkPundurs, MindAfterMath, Jamesx12345, MEmreCelebi, Jcallega, Watarok, E8xE8, Quenhitran, Anrnusna, MSheshera, Monkbot, Mazumdarparijat, Joma.huguet,

Chamale, Jddriessen, Albany NY, Magioladitis, Eleschinski2000, Avicennasis, David Eppstein, Mange01, Zangkannt, Policron, BernardZ, Maghnus, TXiKiBoT, ArzelaAscoli, Monty845, Hobartimus, Denisarona, HairyFotr, Meekywiki, Enmc, Addbot, Legobot, Fraggle81, Materialscientist, Shrishaster, Hhulzk, Rickproser, پووی, Eirik the Viking, X7q, Kiefer.Wolfowitz, A8UDI, JumpDiscont, Skakkle, Polariseke, John of Reading, Bernard Teo, Optimering, ZéroBot, Ziradkar, Chire, AManWithNoPlan, EdoBot, Swfung8, Petrb, ClueBot NG, ElphiBot, Makecat-bot, Lesser Cartographies, Scarlettail, Kuchayrameez, Srijanrshetty, Amaniitk, Boky90 and Anonymous: 109

- **Reinforcement learning** *Source:* https://en.wikipedia.org/wiki/Reinforcement_learning?oldid=670259161 *Contributors:* Wmorgan, Imran, Mrwojo, Michael Hardy, Togelius, DopefishJustin, Kku, Delirium, Hike395, Charles Matthews, Robbot, Altenmann, Giftlite, Dratman, Gene s, Juxi, Urhixidur, Bender235, Tobacman, Diego Moya, Nvrmnd, Oleg Alexandrov, Olethros, Qwertyus, Seliopou, Mathbot, Banazir, Kri, Chobot, Bgwhite, YurikBot, Wavelength, Masatran, Digfarenough, SmackBot, Fabrice.Rossi, Vermorel, Jcarroll, Chris the speller, Ash.dyer, DHN-bot~enwiki, Mitar, Beetstra, Flohack, Ceran, Janrpeters, XApple, ShelfSkewed, Perimosocordiae, Skittleys, Rev.bayes, Escarbot, Tremilux, Parunach, R'n'B, Wfu, Jiuguang Wang, Shyking, Kpmiyapuram, Qsung, Szepi~enwiki, Nedrutland, Mdchang, Sebastjanmm, MrinalKalakrishnan, Flyer22, Melcombe, Rinconsoleao, MBK004, XLinkBot, Addbot, DOI bot, MrOllie, Download, Mianarshad, Yobot, Maderlock, Citation bot, LilHelpa, DSisyphBot, J04n, Gosavia, FrescoBot, Fgpilot, Kartoun, Mr ashyash, D'ohBot, Citation bot 1, Albertzeyer, Wikinacious, Skyerise, Trappist the monk, Dpbert, Stuhlmueller, RjwilmsiBot, Claggierk, EmausBot, Macopema, Chire, Jcautilli, DrewNoakes, Correction45, Rlguy, ChuispastonBot, Mbdts, Dvir-ad, Albertttt, Uymj, Helpful Pixie Bot, BG19bot, Stephen Balaban, ChrisGualtieri, Rbabuska, Ra ules, Chrislgarry, Awliehr, Monkbot, SoloGen and Anonymous: 118
- **Decision tree learning** *Source:* https://en.wikipedia.org/wiki/Decision_tree_learning?oldid=671088225 *Contributors:* Michael Hardy, TheEternalVortex, Greenrd, Maximus Rex, Tschild, Populus, Khalid hassani, Pgan002, Raand, Dan aka jack, Andreas Kaufmann, Discospinster, John Vandenberg, Giraffedata, Mdd, Equinoxe, VeXocide, Bushytails, GregorB, Qwertyus, Rjwilmsi, Gmelli, Salix alba, Vonkje, SmackBot, Diegotorquemada, Mcld, Riedl, Zven, Mitar, Krexer, Kuru, Beetstra, Courcelles, Ceran, Pgr94, Yaris678, Talgalili, A3RO, Nobar, Martinkunev, Destynova, Jessicapierce, Dobi~enwiki, User A1, Jalaska13, A m sheldon, Xs2me, Polyextremophile, Semifinalist, Extabgrad, Dodabe~enwiki, Foxj, Stephen Milborrow, Dank, Pichpich, Addbot, AndrewHZ, MrOllie, Download, Peni, Yobot, TestEditBot, AnomieBOT, Jim1138, Royote, Citation bot, FrescoBot, Hobsonlane, X7q, Kelos omos1, Orchidbox, Citation bot 1, Boxplot, Thinking of England, Janez Demsar, Mwojnars, Gzorg, Wik-dt, Chad.burrus, Bethnim, ZéroBot, Chire, Liorrokach, Pxtreme75, ClueBot NG, Psorakis, Aristitleism, Frietjes, Widr, Helpful Pixie Bot, BG19bot, QualitycontrolUS, BendelacBOT, Mightyteja, Djplaner, CitationCleanerBot, A923812, Sboosali, Douglas H Fisher, Zhang1989cn, JYBot, Lizhengui, Declaration1776, Jey42, Mgibby5, Pimgd, SiraRaven, Slash1986, Monkbot, 00tau, HossPatrol, DizzyRebel and Anonymous: 88
- **Information gain in decision trees** *Source:* https://en.wikipedia.org/wiki/Information_gain_in_decision_trees?oldid=664264867 *Contributors:* Kku, Dcoetzee, Neilc, Andreas Kaufmann, Mathiasl26, Flammifer, Nulli~enwiki, Musiphil, Jheald, Nulli2, SmackBot, Mcld, Freelance Intellectual, Funnyfarmofdoom, Semifinalist, Mild Bill Hiccup, AndrewHZ, MattTait, Шахурик, Erik9bot, QualitycontrolUS, A923812, BattyBot, New Children of the Almighty, Monkbot and Anonymous: 24
- **Ensemble learning** *Source:* https://en.wikipedia.org/wiki/Ensemble_learning?oldid=667910483 *Contributors:* The Anome, Greenrd, Violetriga, 3mta3, Jheald, Qwertyus, Vegaswikian, Wavelength, Crasshopper, ToddDeLuca, Littenberg, Mickeyg13, Mandra Oleka, Magioladitis, Destynova, Salih, EverGreg, Headlessplatter, Kotsiantis, Calimo, Skbkekas, Cowwaw, Qwfp, Sameer0s, Addbot, AndrewHZ, Ettrig, Yobot, Erik9bot, Citation bot 1, John of Reading, Liorrokach, Zephyrus Tavvier, Helpful Pixie Bot, Laughsinthestocks, BG19bot, Rmasba, Monkbot, Tyler Streeter, Delibzr, Anshurm, Velvel2, Olosko, Meteozay and Anonymous: 26
- **Random forest** *Source:* https://en.wikipedia.org/wiki/Random_forest?oldid=670688623 *Contributors:* Michael Hardy, Willsmith, Zeno Gantner, Ronz, Den fjättrade ankan~enwiki, Hike395, Nstender, Giftlite, Neilc, Pgan002, Sam Hocevar, Urhixidur, Andreas Kaufmann, Rich Farmbrough, O18, Rajah, Ferkel, 3mta3, Knowledge Seeker, Rrenaud, Qwertyus, Rjwilmsi, Punk5, Nigosh, Mathbot, LuisPedroCoelho, Bgwhite, RussBot, Dsol, Diegotorquemada, Mcld, Bluebot, Eep1mp, Cybercobra, Mitar, Ben Moore, Shorespirit, Ninetyone, Innohead, Bumbulski, Jason Dunsmore, Talgalili, Thijs!bot, Tolstoy the Cat, Headbomb, Utopiah, Baccyak4H, Hue White, David Eppstein, Trusilver, Yogeshkumkar12, Dvdpwiki, Gerifalte~enwiki, WereSpielChequers, Melcombe, Headlessplatter, Jashley13, Xiawi, Alexbot, Dboehmer, MystBot, Addbot, AndrewHZ, Bastion Monk, MrOllie, Jperl, Legobot, Yobot, AnomieBOT, Randomexpert, Jim1138, Citation bot, Twri, V35b, Nippashish, Sgtf, X7q, Dront, Yurislator, Delmonde, John of Reading, ZéroBot, Chire, Jwollbold, Pokbot, V.cheplygina, Joel B. Lewis, EmmanuelleGouillart, Helpful Pixie Bot, BG19bot, QualitycontrolUS, Spaligo, Stevetihi, Schreckse, A923812, JoshuSasori, JimmyJimmereeno, ChrisGualtieri, Kosio.the.truthseeker, IOverThoughtThis, Bvlb, Svershin, Austrartsua, Monkbot, HossPatrol, Dongkyu Kim, Puxiao129, StudentDH and Anonymous: 87
- **Boosting (machine learning)** *Source:* https://en.wikipedia.org/wiki/Boosting_(machine_learning)?oldid=670591164 *Contributors:* The Anome, Michael Hardy, Cherkash, BAxelrod, Hike395, Phil Boswell, Seabhcan, Indelible~enwiki, Pgan002, Beland, MarkSweep, Collino, APH, Gene s, Urhixidur, Ehamberg, Kate, Nowozin, Violetriga, JustinWick, HasharBot~enwiki, Velella, GJeffery, Demiurg, Qwertyus, Rjwilmsi, Intgr, Bgwhite, YurikBot, Petri Krohn, KYN, Slaniel, COMPFUNK2, Mitar, Trifon Triantafillidis, Jminguillona, Nick Ottery, Edchi, Seaphoto, Matuag, Magioladitis, A3nm, Kent37, Kovianyo, Sebastjanmm, AaronArvey, Slett~enwiki, AlanUS, Hughpugh, Kotsiantis, ClueBot, Piastu, Xiawi, TarzanASG, Excirial, Calimo, Skbkekas, XLinkBot, Glane23, SpBot, Movado73, Legobot, Yobot, AnomieBOT, Hahahaha4, Cobranet, Yousian, X7q, Lreyzin, Boxplot, Danyaljj, Leopd, BG19bot, Striaten, Ianschillebeeckx, François Robere, Jthurst3, Polar Mermaid, Velvel2, Olosko and Anonymous: 57
- **Bootstrap aggregating** *Source:* https://en.wikipedia.org/wiki/Bootstrap_aggregating?oldid=664942138 *Contributors:* Fnielsen, Michael Hardy, Delirium, Den fjättrade ankan~enwiki, Hike395, Mtcv, Phil Boswell, Neilc, Pgan002, Gdm, Urhixidur, Violetriga, JustinWick, Alex Kosorukoff, BlueNovember, Blobglob, Rrenaud, Alai, Oleg Alexandrov, Demiurg, GregorB, Qwertyus, Koolkao, Vonkje, Splash, SmackBot, Minhtuanht~enwiki, Stimpy, Hongooi, Radagast83, Mitar, John, Kvng, Beefyt, Bumbulski, Tolstoy the Cat, Rubesam, Lawrencehu~enwiki, EagleFan, David Eppstein, Melcombe, Kotsiantis, Martarius, MystBot, Addbot, DOI bot, AndrewHZ, Movado73, AnomieBOT, Nippashish, X7q, Citation bot 1, Boxplot, ELAD3, Chire, ChuispastonBot, Helpful Pixie Bot, Wbm1058, BG19bot, Chafe66, Joeinwiki, Mark viking, Brendonboshell, Ealfaro, Tc325, Rmasba, Caozhu, Anshurm and Anonymous: 30
- **Gradient boosting** *Source:* https://en.wikipedia.org/wiki/Gradient_boosting?oldid=667898442 *Contributors:* Ronz, Topbanana, Benwing, Violetriga, Qwertyus, Hongooi, Dgianotti, Medovina, Andre.holzner, Jazzcat81, Semifinalist, JeffDonner, MrOllie, Davedev15, Yobot, Jtamad, LilHelpa, Sophus Bie, X7q, Yihucha166, Trappist the monk, CristiCbz, Alephnot, Chire, HHinman, Mark viking, Pprettenhofer, DerHessi, Crowwork, P.thesling, XQQ14, Gary2015 and Anonymous: 24
- **Semi-supervised learning** *Source:* https://en.wikipedia.org/wiki/Semi-supervised_learning?oldid=667482117 *Contributors:* Edward, Kku, Delirium, Furrykef, Benwing, Rajah, Arthena, Facopad, Soultaco, Bkkbrad, Ruud Koot, Qwertyus, Gmelli, Chobot, DaveWF, Cedar101, Jcarroll, Drono, Phoxhat, Rahimiali, Bookuser, Lamro, Tbmurphy, Addbot, MrOllie, Luckas-bot, Yobot, Gelbukh, AnomieBOT, Xqbot, Omnipaedista, Romainbrasselet, D'ohBot, Wokonen, EmausBot, Grisendo, Stheodor, Rahulkmishra, Pintaio, Helpful Pixie Bot, BG19bot, CarrieVS, AK456, Techerin, M.shahriarinia, Rcpt2 and Anonymous: 28

Durran65, MockDuck, JonathanWilliford, Materialscientist, Citation bot, Eumolpo, Twri, NFD9001, Isheden, J04n, Omnipaedista, Mark Schierbecker, RibotBOT, RoodyBeep, Gunjan verma81, FrescoBot, X7q, Ömer Cengiz Çelebi, Outback the koala, Citation bot 1, Tylor.Sampson, Jonesey95, Calmer Waters, Skyerise, Trappist the monk, Krassotkin, Cjlim, Fox Wilson, The Strategist, LilyKitty, Eparo, בן גרשון, Jfmantis, Mehdiabbasi, VernoWhitney, Wiknn, BertSeghers, DASHBot, EmausBot, Nacopt, Dzkd, Racerx11, Japs 88, GoingBatty, RaoInWiki, Roposeidon, Epsiloner, Stheodor, Benlansdell, Radshashi, K6ka, D'oh!, Thisisentchris87, Aavindraa, Chire, Glosser.ca, IGeMiNix, Donner60, Yoshua.Bengio, Shinosin, Venkatarun95, ChuckNorrisPwnedYou, Petrb, ClueBot NG, Raghith, Robiminer, Snotbot, Tideflat, Frietjes, Gms3591, Ryansandersuk, Widr, MerlIwBot, Helpful Pixie Bot, Trepier, BG19bot, Thwien, Adams7, Rahil2000, Chafe66, Michaelmalak, Compfreak7, Kirananils, Altaïr, J.Davis314, Attleboro, Pratyya Ghosh, JoshuSasori, Ferrarisailor, Eugenecheung, Mtschida, ChrisGualtieri, Dave2k6inthemix, Whebzy, APerson, JurgenNL, Oritnk, Stevebillings, Djfrost711, Sa publishers, ⍰, Mark viking, Markus.harz, Deeper Learning, Vinchaud20, Soueumxm, Toritris, Evolution and evolvability, Sboddhu, Sharva029, Paheld, Putting things straight, Rosario Berganza, Monkbot, Buggiehuggie, Santoshwriter, Likerhayter, Joma.huguet, Bclark401, Rahulpratapsingh06, Donkeychee, Michaelwine, Xsantostill, Jorge Guerra Pires, Wfwhitney, Loïc Bourgois, KasparBot and Anonymous: 497

- **Deep learning** *Source:* https://en.wikipedia.org/wiki/Deep_learning?oldid=671235450 *Contributors:* The Anome, Ed Poor, Michael Hardy, Meekohi, Glenn, Bearcat, Nandhp, Stesmo, Giraffedata, Jonsafari, Oleg Alexandrov, Justin Ormont, BD2412, Qwertyus, Rjwilmsi, Kri, Bgwhite, Tomdooner, Arado, Bhny, Malcolma, Arthur Rubin, Mebden, SeanAhern, Dicklyon, JHP, ChrisCork, Lfstevens, A3nm, R'n'B, Like.liberation, Popoki, Jshrager, Strife911, Bfx0, Daniel Hershcovich, Pinkpedaller, Dthomsen8, Addbot, Mamikonyana, Yobot, AnomieBOT, Jonesey95, Zabbarob, Wyverald, RjwilmsiBot, Larry.europe, Helwr, GoingBatty, Sergey WereWolf, SlowByte, Yoshua.Bengio, JuyangWeng, Renklauf, Bittenus, Widr, BG19bot, Lukas.tencer, Kareltje63, Synchronist, Gameboy97q, IjonTichyIjonTichy, Mogism, AlwaysCoding, Mark viking, Cagarie, Deeper Learning, Prisx, Wikiyant, Underflow42, GreyShields, Opokopo, Prof. Oundest, Gigavanti, Sevensharpnine, Yes deeper, Monkbot, Chieftains337, Samueldg89, Rober9876543210, Nikunj157, Engheta, Aspurdy, Velvel2, TeaLover1996, Deng629, Zhuikov, Stergioc, Jerodlycett, DragonbornXXL, Lzjpaul and Anonymous: 113

62.12.2 Images

- **File:2013-09-11_Bus_wrapped_with_SAP_Big_Data_parked_outside_IDF13_(9730051783).jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/8d/2013-09-11_Bus_wrapped_with_SAP_Big_Data_parked_outside_IDF13_%289730051783%29.jpg *License:* CC BY-SA 2.0 *Contributors:* Bus wrapped with SAP Big Data parked outside IDF13 *Original artist:* Intel Free Press
- **File:Ambox_important.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/b4/Ambox_important.svg *License:* Public domain *Contributors:* Own work, based off of Image:Ambox scales.svg *Original artist:* Dsmurat (talk · contribs)
- **File:Animation2.gif** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/c0/Animation2.gif *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* MG (talk · contribs)
- **File:Ann_dependency_(graph).svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/dd/Ann_dependency_%28graph%29.svg *License:* CC BY-SA 3.0 *Contributors:* Vector version of File:Ann dependency graph.png *Original artist:* Glosser.ca
- **File:Anscombe'{}s_quartet_3.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/ec/Anscombe%27s_quartet_3.svg *License:* CC BY-SA 3.0 *Contributors:*
- Anscombe.svg *Original artist:* Anscombe.svg: Schutz
- **File:ArtificialFictionBrain.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/17/ArtificialFictionBrain.png *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:Artificial_neural_network.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/e4/Artificial_neural_network.svg *License:* CC-BY-SA-3.0 *Contributors:* This vector image was created with Inkscape. *Original artist:* en:User:Cburnett
- **File:Automated_online_assistant.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/8b/Automated_online_assistant.png *License:* Attribution *Contributors:*

  The text is adapted from the Wikipedia merchandise page (this automated customer service itself, however, is fictional), and pasted into a page in Wikipedia:

  Original artist: Mikael Häggström
- **File:Bayes_icon.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/ed/Bayes_icon.svg *License:* CC0 *Contributors:* <a href='http://validator.w3.org/' data-x-rel='nofollow'><img alt='W3C' src='https://upload.wikimedia.org/wikipedia/commons/thumb/1/1a/Valid_SVG_1.1_%28green%29.svg/88px-Valid_SVG_1.1_%28green%29.svg.png' width='88' height='30' style='vertical-align: top' srcset='https://upload.wikimedia.org/wikipedia/commons/thumb/1/1a/Valid_SVG_1.1_%28green%29.svg/132px-Valid_SVG_1.1_%28green%29.svg.png 1.5x, https://upload.wikimedia.org/wikipedia/commons/thumb/1/1a/Valid_SVG_1.1_%28green%29.svg/176px-Valid_SVG_1.1_%28green%29.svg.png 2x' data-file-width='91' data-file-height='31' /></a>iThe source code of this SVG is <a data-x-rel='nofollow' class='external text' href='http://validator.w3.org/check?uri=https%3A%2F%2Fcommons.wikimedia.org%2Fwiki%2FSpecial%3AFilepath%2FBayes_icon.svg,<span>,&,</span>,ss=1#source'>valid</a>.

  Original artist: Mikhail Ryazanov
- **File:Bayes_theorem_visualisation.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bf/Bayes_theorem_visualisation.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Cmglee
- **File:Bayesian_inference_archaeology_example.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/6d/Bayesian_inference_archaeology_example.jpg *License:* CC0 *Contributors:* Own work *Original artist:* Gnathan87
- **File:Bayesian_inference_event_space.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/ad/Bayesian_inference_event_space.svg *License:* CC0 *Contributors:* Own work *Original artist:* Gnathan87
- **File:Bendixen_-_Carl_Friedrich_Gauß,_1828.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/3/33/Bendixen_-_Carl_Friedrich_Gau%C3%9F%2C_1828.jpg *License:* Public domain *Contributors:* published in “Astronomische Nachrichten” 1828 *Original artist:* Siegfried Detlev Bendixen
- **File:Big_data_cartoon_t_gregorius.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/b3/Big_data_cartoon_t_gregorius.jpg *License:* CC BY 2.0 *Contributors:* Cartoon: Big Data *Original artist:* Thierry Gregorius

- **File:Wikinews-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/24/Wikinews-logo.svg *License:* CC BY-SA 3.0 *Contributors:* This is a cropped version of Image:Wikinews-logo-en.png. *Original artist:* Vectorized by Simon 01:05, 2 August 2006 (UTC) Updated by Time3000 17 April 2007 to use official Wikinews colours and appear correctly on dark backgrounds. Originally uploaded by Simon.
- **File:Wikiquote-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fa/Wikiquote-logo.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Wikisource-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/4c/Wikisource-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Rei-artur *Original artist:* Nicholas Moreau
- **File:Wikiversity-logo-Snorky.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/1b/Wikiversity-logo-en.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Snorky
- **File:Wikiversity-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/91/Wikiversity-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Snorky (optimized and cleaned up by verdy_p) *Original artist:* Snorky (optimized and cleaned up by verdy_p)
- **File:Wiktionary-logo-en.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f8/Wiktionary-logo-en.svg *License:* Public domain *Contributors:* Vector version of Image:Wiktionary-logo-en.png. *Original artist:* Vectorized by Fvasconcellos (talk · contribs), based on original logo tossed together by Brion Vibber

62.12.3 Content license

- Creative Commons Attribution-Share Alike 3.0