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]

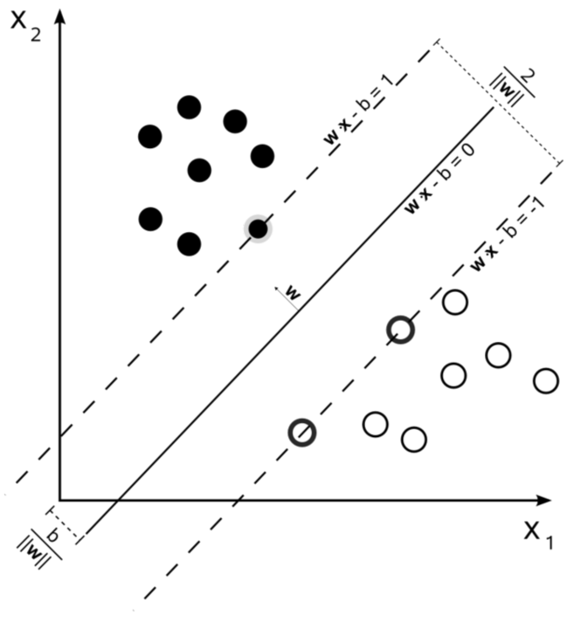
1.1.1 Types of problems and tasks

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

scikit-learn

scikit-learn (formerly **scikits.learn**) is an open source machine learning library for the Python programming language.[2] It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

2.1 Overview

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a “SciKit” (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[3] The original codebase was later extensively rewritten by other developers. Of the various scikits, scikit-learn as well as scikit-image were described as “well-maintained and popular” in November 2012.[4]

As of 2015, scikit-learn is under active development and is sponsored by INRIA, Telecom ParisTech and occasionally Google (through the Google Summer of Code).[5] Among its users are Evernote, which uses the library to distinguish recipes from other user posts through a naive Bayes classifier,[6] and Mendeley, which builds recommender systems from scikit-learn’s SGD regression algorithm.[7]

The scikit-learn API has been adopted by wise.io, who offer a proprietary implementation of random forests called wiseRF.[8][9] wise.io’s business partner Continuum IO claimed data throughput of up to 7.5 times that of scikit-learn’s implementation;[10] since then, the scikit-learn developers claim to have optimized their implementation to be competitive with wise.io’s, except in terms of memory use.[11]

2.2 Implementation

scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

2.3 See also

- mlpy
- Orange
- NLTK

2.4 References

[1] Andreas Müller. “scikit-learn 0.16.1”. *Python Package Index*.

[2] Fabian Pedregosa; Gaël Varoquaux; Alexandre Gramfort; Vincent Michel; Bertrand Thirion; Olivier Grisel; Mathieu Blondel; Peter Prettenhofer; Ron Weiss; Vincent Dubourg; Jake Vanderplas; Alexandre Passos; David Cournapeau (2011). “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research* **12**: 2825–2830.

[3] Dreijer, Janto. “scikit-learn”.

[4] Eli Bressert (2012). *SciPy and NumPy: an overview for developers*. O'Reilly. p. 43.

[5] “About Us”. http://scikit-learn.org. Retrieved 23 March 2015.

[6] Mark Ayzenshtat (22 January 2013). “Stay classified”. *Evernote Techblog*. Retrieved 4 May 2013.

[7] Mark Levy (2013). *Efficient Top-N Recommendation by Linear Regression*. ACM RecSys Large Scale Recommender System workshop.

[8] “wiserf”. wise.io. Retrieved 22 January 2014.

[9] Buitinck, Lars, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae *et al.* (2013). *API design for machine learning software: experiences from the scikit-learn project*. ECML PKDD Workshop on Languages for Machine Learning.

[10] Joseph W. Richards (27 November 2012). "wiseRF Use Cases and Benchmarks". Continuum IO. Retrieved 22 January 2014.

[11] Gaël Varoquaux (8 August 2013). "Scikit-learn 0.14 release: features and benchmarks". Retrieved 22 January 2014.

2.5 External links

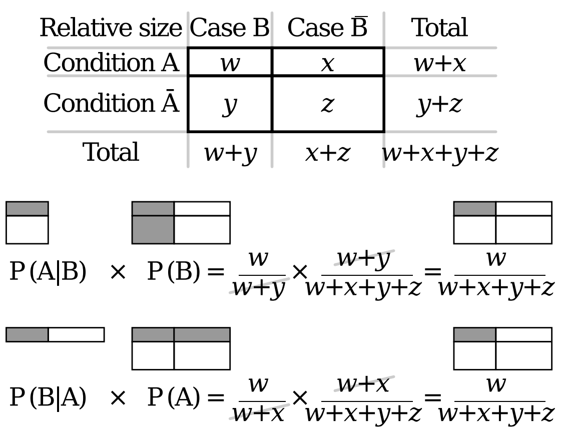
- Official website
- https://github.com/scikit-learn/scikit-learn

Chapter 3

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.

3.1 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

3.1.1 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 .

3.1.2 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".

3.1.3 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]

3.2 Formal description of Bayesian inference

3.2.1 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.

3.2.2 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”.

3.2.3 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.

3.3 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.

3.3.1 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]

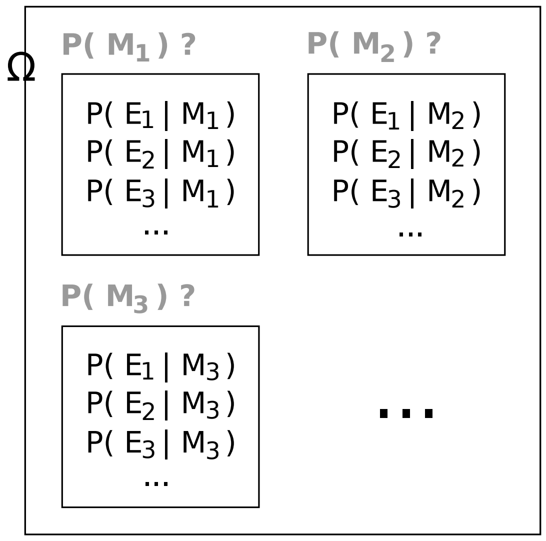


Diagram illustrating event space Ω in general formulation of Bayesian inference. Although this diagram shows discrete models and events, the continuous case may be visualized similarly using probability densities.

$$P(M \mid E) = \frac{P(E \mid M)}{\sum_m P(E \mid M_m) P(M_m)} \cdot P(M)$$

Upon observation of further evidence, this procedure may be repeated.

3.3.2 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.

3.3.3 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

$$p(\mathbf{E} \mid \theta, \alpha) = \prod_k p(e_k \mid \theta)$$

3.4 Mathematical properties

3.4.1 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.

3.4.2 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.

3.4.3 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.

3.4.4 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.

3.4.5 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

$$p(\tilde{x}|\mathbf{X}, \alpha) = \int_\theta p(\tilde{x}, \theta \mid \mathbf{X}, \alpha) \, d\theta = \int_\theta p(\tilde{x} \mid \theta) p(\theta \mid \mathbf{X}, \alpha) \, d\theta.$$

3.5 Examples

3.5.1 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?

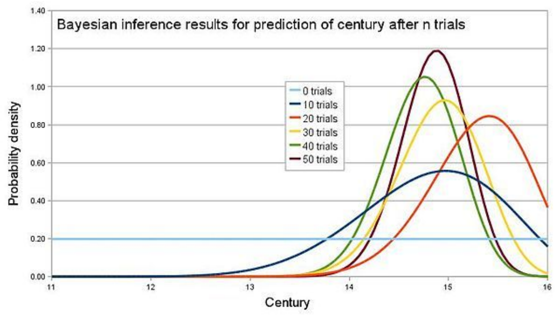
Intuitively, it seems clear that the answer should be more than a half, since there are more plain cookies in bowl #1. The precise answer is given by Bayes’ theorem. Let H_1 correspond to bowl #1, and H_2 to bowl #2. It is given that the bowls are identical from Fred’s point of view, thus $P(H_1) = P(H_2)$, 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.

3.5.2 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).

3.6 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]

3.6.1 Model selection

See Bayesian model selection

3.7 Applications

3.7.1 Computer applications

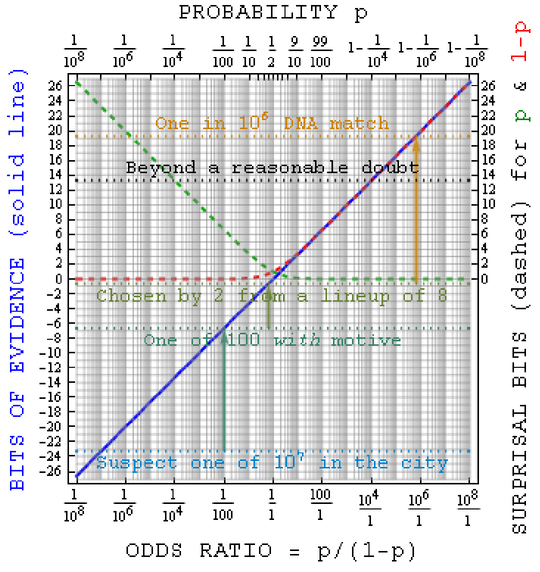
Bayesian inference has applications in artificial intelligence and expert systems. Bayesian inference techniques have been a fundamental part of computerized pattern recognition techniques since the late 1950s. There is also an ever growing connection between Bayesian methods and simulation-based Monte 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]

3.7.2 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.

3.7.3 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.

3.7.4 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

3.8 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.

3.9 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]

3.10 See also

- Bayes' theorem
- Bayesian hierarchical modeling
- Bayesian Analysis, the journal of the ISBA
- Inductive probability
- International Society for Bayesian Analysis (ISBA)
- Jeffreys prior

3.11 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.

3.12 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.

3.13 Further reading

3.13.1 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.

3.13.2 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

3.14 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 4

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.

4.1 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.

4.2 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]

4.2.1 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.

4.2.2 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.

4.2.3 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).

4.3 Example

4.3.1 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.

4.3.2 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.

4.3.3 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).

4.3.4 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: $c = 13$.

4.3.5 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.

4.4 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.

4.5 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]

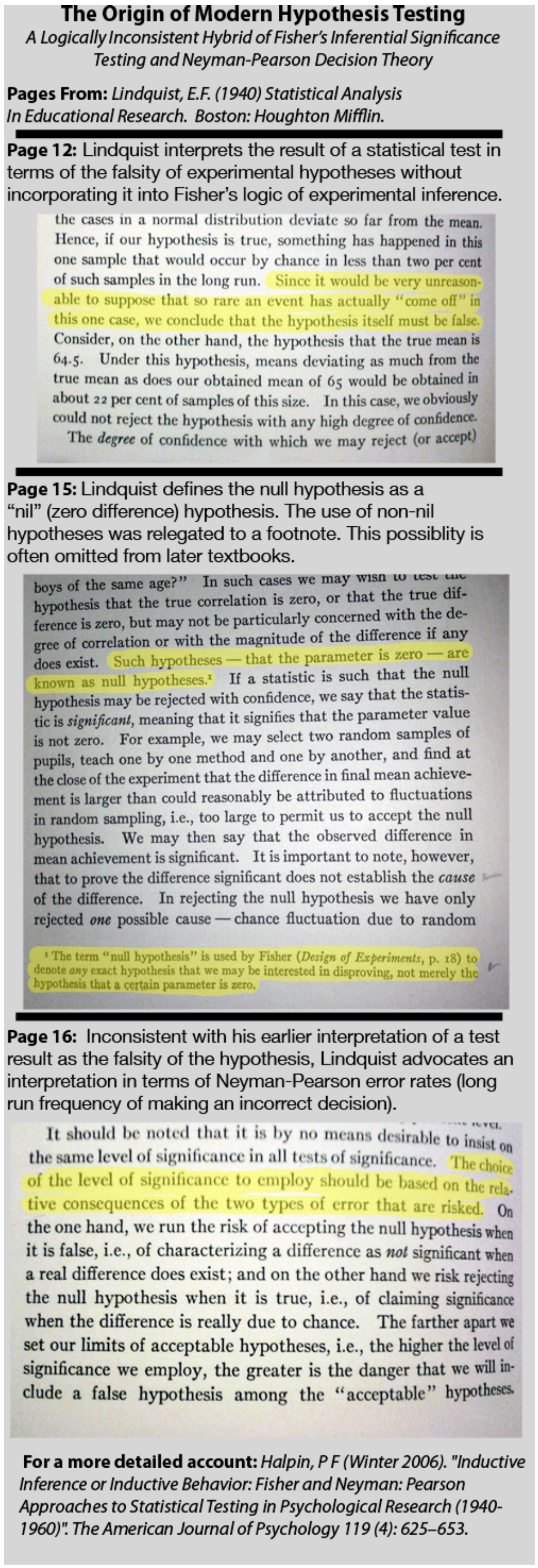
4.6 Origins and early controversy

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)

4.6.1 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]

4.7 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]

4.8 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]

4.9 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*.

4.10 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]

4.11 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]

4.12 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

4.13 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.

4.14 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.

4.15 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

4.15.1 Online calculators

- MBAStats confidence interval and hypothesis test calculators

Chapter 5

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.

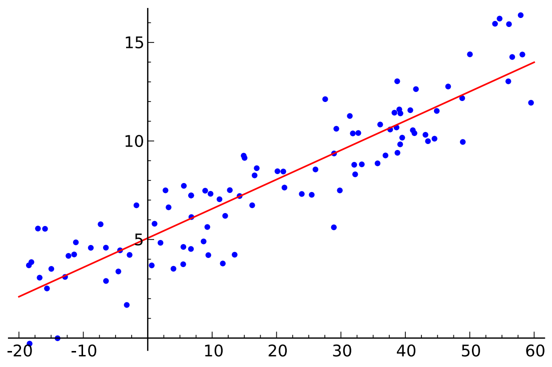
Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications.[4] This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and 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.

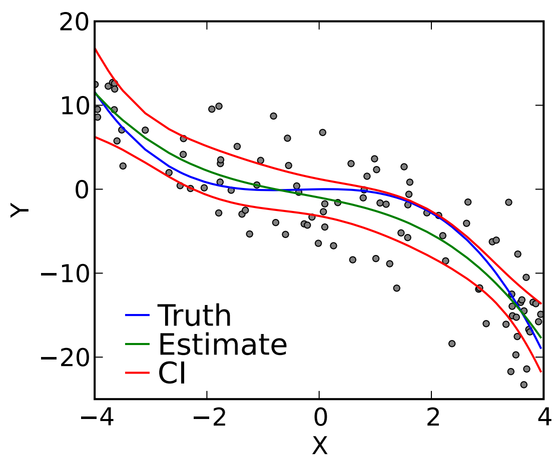
Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares loss function as 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.

5.1 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.

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\rm T}\boldsymbol\beta + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $^{\rm T}$ denotes the transpose, so that $\boldsymbol{xi}^{\rm T}\boldsymbol\beta$ 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

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\rm T} \\ \mathbf{x}_2^{\rm T} \\ \vdots \\ \mathbf{x}_n^{\rm T} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

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 $\mathbf{X}$ 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 $i = 1, ..., n$. 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

$$h_i = \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta} + \varepsilon_i.$$

5.1.1 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.

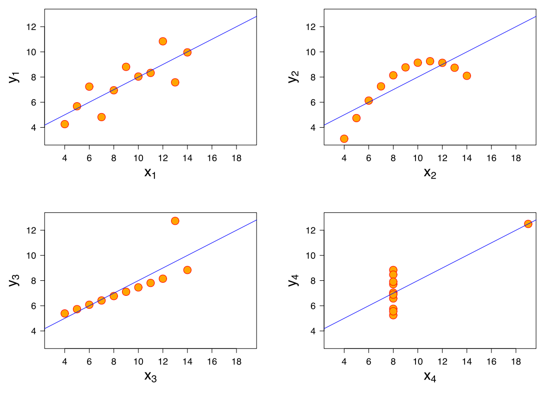
- **Lack of multicollinearity** in the predictors. For standard least squares estimation methods, the design matrix X must have full column rank p,; otherwise, we have a condition known as multicollinearity in the predictor variables. This can be triggered by having two or more perfectly correlated predictor 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.

5.1.2 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]

5.2 Extensions

Numerous extensions of linear regression have been developed, which allow some or all of the assumptions underlying the basic model to be relaxed.

5.2.1 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.

5.2.2 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."

5.2.3 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.

5.2.4 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]

5.2.5 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.

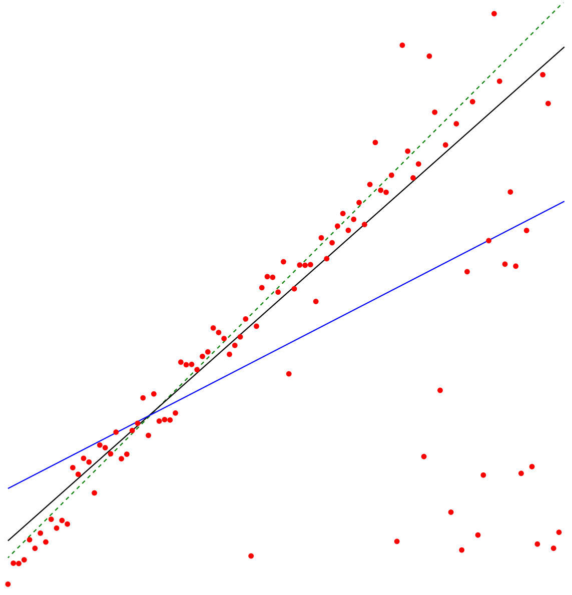
5.2.6 Errors-in-variables

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.

5.2.7 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.

5.3 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.

5.3.1 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 i^{th} case is inversely proportional to var(εi). This special case of GLS is called "weighted least squares". The GLS solution to estimation problem is

$$\hat{\beta} = (\mathbf{X}^{\mathrm{T}}\boldsymbol{\Omega}^{-1}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\Omega}^{-1}\mathbf{y},$$

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*εi] = 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.

5.3.2 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]

5.3.3 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.

5.3.4 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]

5.3.5 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 .

5.4 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.

5.4.1 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.

5.4.2 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.

5.4.3 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.

5.4.4 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]

5.4.5 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]

5.5 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

5.6 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; Wei (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. More than one of |author2= and |last2= specified (help); More than one of |author3= and |last3= specified (help)

[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

5.7 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).

5.8 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

5.9 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 6

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]

6.1 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.

6.2 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.

6.2.1 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.

6.2.2 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.

6.3 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.

6.4 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:

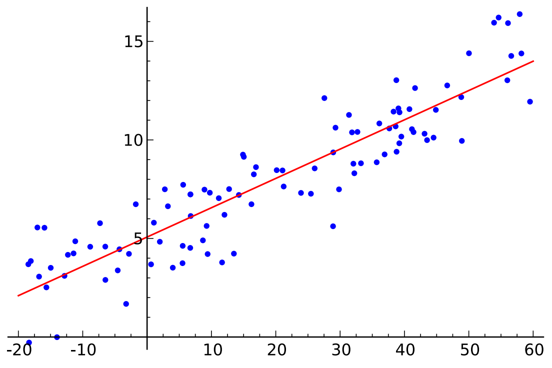


Illustration of linear regression on a data set.

$$\hat{\sigma}_\varepsilon^2 = \frac{SSE}{n-2}.$$

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.

6.4.1 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

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

6.4.2 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.

6.4.3 "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.

6.5 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).

6.6 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

6.7 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

$\frac{\log 1000}{\log 5} = 4.29$.

6.8 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]

6.9 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.

6.10 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

6.11 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.

6.12 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.

6.13 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 7

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.

7.1 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 $P(\theta)$.

7.2 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.

7.2.1 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]

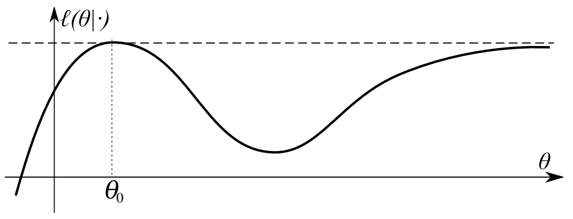
1. **Identification** of the model:

   $$\theta \neq \theta_0 \quad \Leftrightarrow \quad f(\cdot \mid \theta) \neq f(\cdot \mid \theta_0).$$

   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

   $$\big|\ln f(x \mid \theta)\big| < D(x) \quad \text{all for } \theta \in \Theta.$$

   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:

$$\sup_{\theta\in\Theta} \big\|\ \hat{\ell}(x \mid \theta) - \ell(\theta)\ \big\| \xrightarrow{\text{a.s.}} 0.$$

7.2.2 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. ∫ supθ∈N||∇$\theta f(x \mid \theta)$||dx < ∞, and ∫ supθ∈N||∇$\theta\theta f(x|\theta)$||dx < ∞;
4. I = E[∇θln$f(x \mid \theta_0)$ ∇θln$f(x|\theta_0)'$] exists and is non-singular;
5. E[supθ∈N||∇$\theta\theta$ln$f(x \mid \theta)$||] < ∞.

Then the maximum likelihood estimator has asymptotically normal distribution:

$$\sqrt{n}\big(\hat\theta_\mathrm{mle} - \theta_0\big)\ \xrightarrow{d}\ \mathcal{N}(0,\, I^{-1}).$$

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 $\theta = \theta_0$:

$$0 = \frac{1}{n}\sum_{i=1}^n \nabla_{\!\theta}\ln f(x_i \mid \theta_0) + \Bigg[\,\frac{1}{n}\sum_{i=1}^n \nabla_{\!\theta\theta}\ln f(x_i \mid \tilde\theta)\,\Bigg](\hat\theta - \theta_0),$$

where $\tilde\theta$ is some point intermediate between θ_0 and $\hat\theta$. From this expression we can derive that

$$\sqrt{n}(\hat{\theta}-\theta_0) = \Bigg[\,{-\frac{1}{n}\sum_{i=1}^n \nabla_{\!\theta\theta}\ln f(x_i \mid \tilde\theta)}\,\Bigg]^{-1} \frac{1}{\sqrt{n}}\sum_{i=1}^n \nabla_{\!\theta}\ln f(x_i \mid \theta_0)$$

Here the expression in square brackets converges in probability to H = E[−∇$\theta\theta$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}.

7.2.3 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.

7.2.4 Higher-order properties

The standard asymptotics tells that the maximum-likelihood estimator is $\sqrt{n}$-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:

$$I_{jk} = \operatorname{E}_X \left[-\frac{\partial^2 \ln f_{\theta_0}(X_t)}{\partial\theta_j\,\partial\theta_k} \right].$$

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]

$$b_s \equiv \operatorname{E}[(\hat{\theta}_{\text{mle}} - \theta_0)_s] = \frac{1}{n} \cdot I^{si} I^{jk} \big(\tfrac{1}{2} K_{ijk} + J_{j,ik} \big)$$

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

$$\tfrac{1}{2}K_{ijk} + J_{j,ik} = \operatorname{E} \left[\frac{1}{2} \frac{\partial^3 \ln f_{\theta_0}(x_t)}{\partial\theta_i\,\partial\theta_j\,\partial\theta_k} + \frac{\partial \ln f_{\theta_0}(x_t)}{\partial\theta_j} \frac{\partial^2 \ln f_{\theta_0}(x_t)}{\partial\theta_i\,\partial\theta_k} \right]$$

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.

7.3 Examples

7.3.1 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 $(n - 1)/2$.

7.3.2 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:

$$\Pr(\mathrm{H}=49 \mid p=1/3) = \binom{80}{49}(1/3)^{49}(1-1/3)^{31} \approx 0.000,$$

$$\Pr(\mathrm{H}=49 \mid p=1/2) = \binom{80}{49}(1/2)^{49}(1-1/2)^{31} \approx 0.012,$$

$$\Pr(\mathrm{H}=49 \mid p=2/3) = \binom{80}{49}(2/3)^{49}(1-2/3)^{31} \approx 0.054.$$

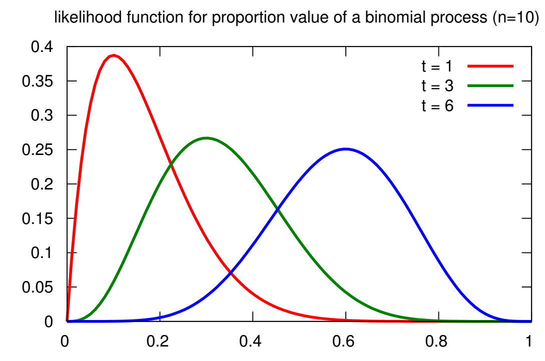
The likelihood is maximized when p = 2/3, and so this is the *maximum likelihood estimate* for p.

7.3.3 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’.

7.3.4 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.

7.4 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.

7.5 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]

7.6 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

7.7 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]

7.8 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.

7.9 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)

7.10 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.

7.11 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 8

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.

8.1 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*.

8.2 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.

8.3 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.

8.4 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.

8.5 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.

8.6 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.

8.7 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

8.8 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.

8.9 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

8.10 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

8.11 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.

8.12 External links

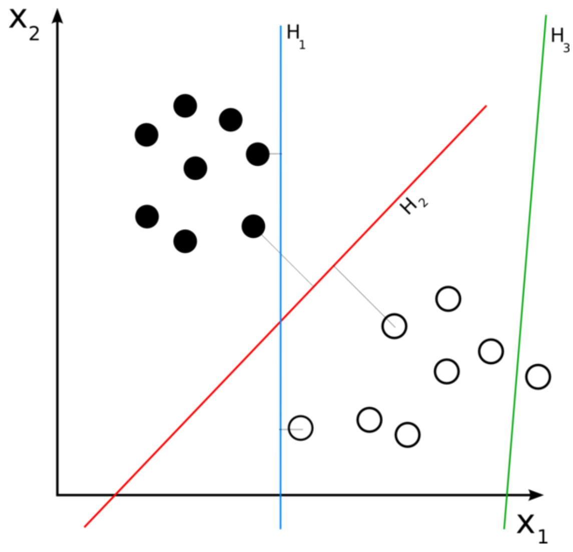
- Classifier showdown A practical comparison of classification algorithms.
- Statistical Pattern Recognition Toolbox for Matlab.
- TOOLDIAG Pattern recognition toolbox.
- Statistical classification software based on adaptive kernel density estimation.
- PAL Classification Suite written in Java.
- kNN 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 9

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]

9.1 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.

9.2 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.

9.2.1 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

- $\mathbf{w}$ 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.

9.3 See also

- Linear regression
- Winnow (algorithm)
- Quadratic classifier
- Support vector machines

9.4 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 10

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.

10.1 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.

10.2 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.

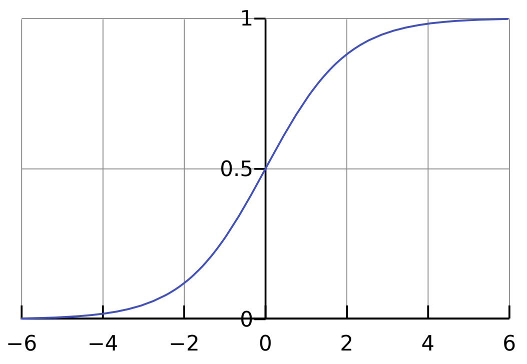


Figure 1. The logistic function $\sigma(t)$; note that $\sigma(t) \in [0,1]$ for all t .

10.3 Logistic function, odds, odds ratio, and logit

10.3.1 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]

10.3.2 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:

$$\frac{F(x)}{1 - F(x)} = e^{\beta_0 + \beta_1 x}.$$

10.3.3 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.

10.3.4 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:

$$\text{odds} = e^{\beta_0 + \beta_1 x}.$$

10.3.5 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]

10.3.6 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.

10.4 Model fitting

10.4.1 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

10.4.2 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.

10.5 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.

10.5.1 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.

10.5.2 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]

10.5.3 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.

10.6 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.

10.6.1 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.

10.6.2 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):

$$\Pr(Y_i = y_i \mid \mathbf{X}_i) = {p_i}^{y_i}(1-p_i)^{1-y_i} = \left(\frac{e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}{1+e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}\right)^{y_i} \left(1 - \frac{e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}{1+e^{\boldsymbol{\beta} \cdot \mathbf{X}_i}}\right)$$

10.6.3 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:

$$Y_i^* = \boldsymbol\beta \cdot \mathbf{X}_i + \varepsilon$$

where

$$\varepsilon \sim \text{Logistic}(0, 1)$$

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:

$$Y_i = \begin{cases} 1 & \text{if} Y_i^* > 0 \ \text{ i.e. } -\varepsilon < \boldsymbol\beta \cdot \mathbf{X}_i, \\ 0 & \text{otherwise.} \end{cases}$$

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.

$$\Pr(\varepsilon < x) = \text{logit}^{-1}(x)$$

Then:

$$\begin{aligned} \Pr(Y_i = 1 \mid \mathbf{X}_i) &= \Pr(Y_i^* > 0 \mid \mathbf{X}_i) \\ &= \Pr(\boldsymbol\beta \cdot \mathbf{X}_i + \varepsilon > 0) \\ &= \Pr(\varepsilon > -\boldsymbol\beta \cdot \mathbf{X}_i) \\ &= \Pr(\varepsilon < \boldsymbol\beta \cdot \mathbf{X}_i) && \text{symmetric) is distribution logisti} \\ &= \text{logit}^{-1}(\boldsymbol\beta \cdot \mathbf{X}_i) \\ &= p_i && \text{above) (see} \end{aligned}$$

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).

10.6.4 As a two-way latent-variable model

Yet another formulation uses two separate latent variables:

$$\begin{aligned} Y_i^{0*} &= \boldsymbol\beta_0 \cdot \mathbf{X}_i + \varepsilon_0 \\ Y_i^{1*} &= \boldsymbol\beta_1 \cdot \mathbf{X}_i + \varepsilon_1 \end{aligned}$$

where

$$\begin{aligned} \varepsilon_0 &\sim \text{EV}_1(0, 1) \\ \varepsilon_1 &\sim \text{EV}_1(0, 1) \end{aligned}$$

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

$$Y_i = \begin{cases} 1 & \text{if} Y_i^{1*} > Y_i^{0*}, \\ 0 & \text{otherwise.} \end{cases}$$

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:

$$\begin{aligned}
& \Pr(Y_i = 1 \mid \mathbf{X}_i) \\
= \ & \Pr(Y_i^{1\ast} > Y_i^{0\ast} \mid \mathbf{X}_i) \\
= \ & \Pr(Y_i^{1\ast} - Y_i^{0\ast} > 0 \mid \mathbf{X}_i) \\
= \ & \Pr(\boldsymbol\beta_1 \cdot \mathbf{X}_i + \varepsilon_1 - (\boldsymbol\beta_0 \cdot \mathbf{X}_i + \varepsilon_0) > 0) \\
= \ & \Pr((\boldsymbol\beta_1 \cdot \mathbf{X}_i - \boldsymbol\beta_0 \cdot \mathbf{X}_i) + (\varepsilon_1 - \varepsilon_0) > 0) \\
= \ & \Pr((\boldsymbol\beta_1 - \boldsymbol\beta_0) \cdot \mathbf{X}_i + (\varepsilon_1 - \varepsilon_0) > 0) \\
= \ & \Pr((\boldsymbol\beta_1 - \boldsymbol\beta_0) \cdot \mathbf{X}_i + \varepsilon > 0) && \text{(substitute}\varepsilon\text{above) as} \\
= \ & \Pr(\boldsymbol\beta \cdot \mathbf{X}_i + \varepsilon > 0) && \text{(substitute}\boldsymbol\beta\text{above) as} \\
= \ & \Pr(\varepsilon > -\boldsymbol\beta \cdot \mathbf{X}_i) && \text{model) above as same (now,} \\
= \ & \Pr(\varepsilon < \boldsymbol\beta \cdot \mathbf{X}_i) \\
= \ & \operatorname{logit}^{-1}(\boldsymbol\beta \cdot \mathbf{X}_i) \\
= \ & p_i
\end{aligned}$$

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.

10.6.5 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.

10.6.6 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:

$$\frac{\mathrm{d}y}{\mathrm{d}X} = y(1 - y)\frac{\mathrm{d}f}{\mathrm{d}X}.$$

10.6.7 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:

$$Y_i \ \sim \text{Bin}(n_i, p_i), \text{ for } i = 1, \dots, n$$

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

$$\text{logit}\left(\mathbb{E}\left[\left.\frac{Y_i}{n_i}\,\right|\,\mathbf{X}_i\right]\right) = \text{logit}(p_i) = \ln\left(\frac{p_i}{1 - p_i}\right) = \boldsymbol\beta\cdot\mathbf{X}_i,$$

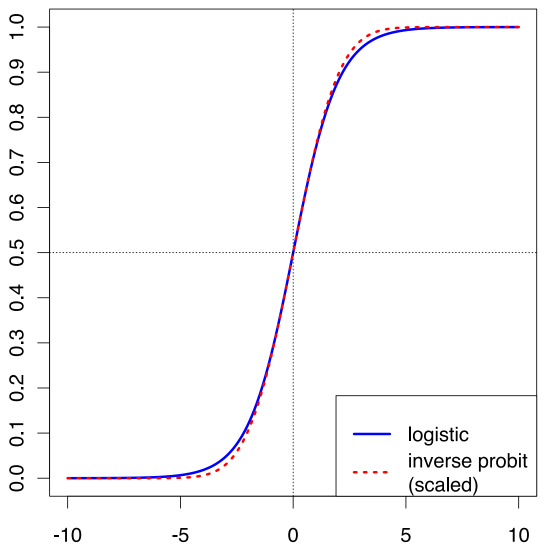
Or equivalently:

$$\Pr(Y_i = y_i \mid \mathbf{X}_i) = \binom{n_i}{y_i} p_i^{y_i}(1-p_i)^{n_i-y_i} = \binom{n_i}{y_i}\left(\frac{1}{1+e^{-\boldsymbol\beta\cdot\mathbf{X}_i}}\right)^{y_i}\left(1 - \frac{1}{1+e^{-\boldsymbol\beta\cdot\mathbf{X}_i}}\right)^{n_i-y_i}$$

This model can be fit using the same sorts of methods as the above more basic model.

10.7 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]

10.7.1 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.)

10.8 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.

10.9 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:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

$$\text{Precision} = \text{value predictive Positive} = \frac{TP}{TP+FP}$$

$$\text{value predictive Negative} = \frac{TN}{TN+FN}$$

$$\text{Recall} = \text{Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{Specificity} = \frac{TN}{TN+FP}$$

10.10 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

10.11 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.

10.12 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.

10.13 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 11

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]

11.1 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.

11.2 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.

11.3 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.

11.4 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.

11.5 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.

11.6 Applications

In addition to the examples given below, LDA is applied in positioning and product management.

11.6.1 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.

11.6.2 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*.

11.6.3 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.

11.6.4 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.

11.6.5 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]

11.7 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

11.8 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.

11.9 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.

11.10 External links

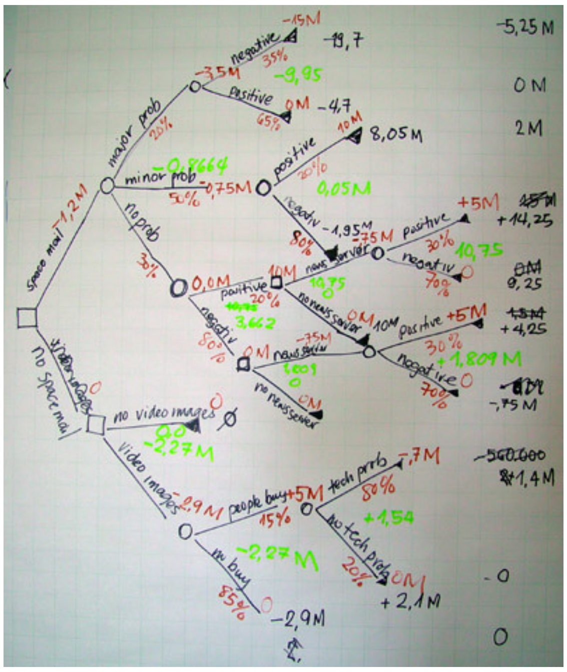
- ALGLIB contains open-source LDA implementation in C# / C++ / Pascal / VBA.
- Psychometrica.de open-source LDA implementation in Java
- LDA tutorial using MS Excel
- Biomedical statistics. Discriminant analysis

Chapter 12

Decision tree

This article is about decision trees in decision analysis. For the use of the term in machine learning, see Decision tree learning.

A **decision tree** is a decision support tool that uses a tree-



Traditionally, decision trees have been created manually.

like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm.

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal.

12.1 Overview

A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represents classification rules.

In decision analysis a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

A decision tree consists of 3 types of nodes:

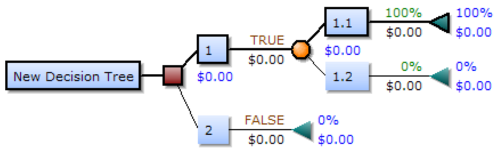
1. Decision nodes - commonly represented by squares
2. Chance nodes - represented by circles
3. End nodes - represented by triangles

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal. If in practice decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

Decision trees, influence diagrams, utility functions, and other decision analysis tools and methods are taught to undergraduate students in schools of business, health economics, and public health, and are examples of operations research or management science methods.

12.2 Decision tree building blocks

12.2.1 Decision tree elements



Drawn from left to right, a decision tree has only burst nodes (splitting paths) but no sink nodes (converging paths). Therefore, used manually, they can grow very big and are then often hard to draw fully by hand. Traditionally, decision trees have been created manually - as the aside example shows - although increasingly, specialized software is employed.

12.2.2 Decision rules

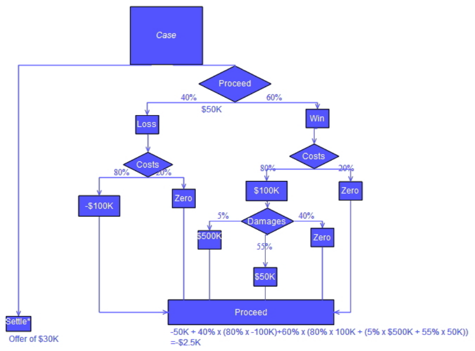
The decision tree can be linearized into **decision rules**,[1] where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. In general, the rules have the form:

> *if* condition1 *and* condition2 *and* condition3 *then* outcome.

Decision rules can also be generated by constructing association rules with the target variable on the right.

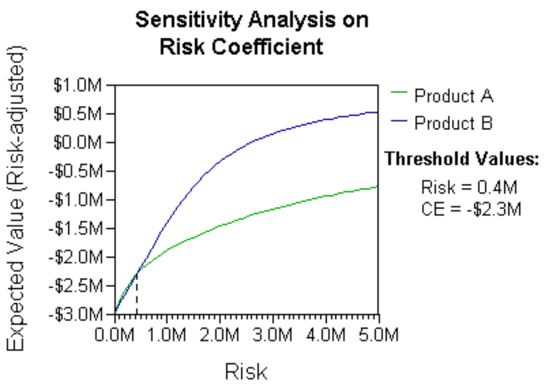
12.2.3 Decision tree using flowchart symbols

Commonly a decision tree is drawn using flowchart symbols as it is easier for many to read and understand.



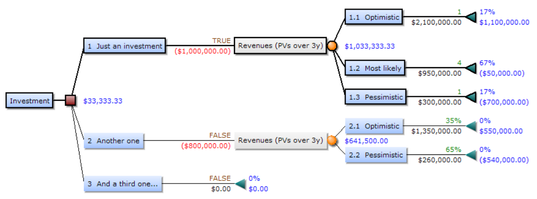
12.2.4 Analysis example

Analysis can take into account the decision maker's (e.g., the company's) preference or utility function, for example:



The basic interpretation in this situation is that the company prefers B's risk and payoffs under realistic risk preference coefficients (greater than $400K—in that range of risk aversion, the company would need to model a third strategy, "Neither A nor B").

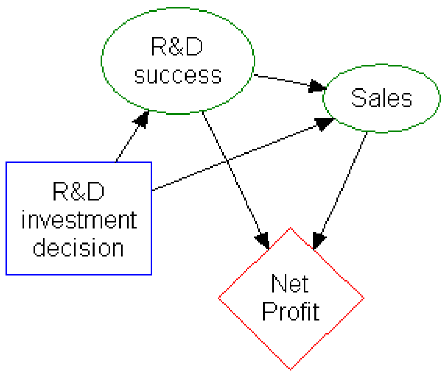
12.2.5 Another example



Decision trees can be used to optimize an investment portfolio. The following example shows a portfolio of 7 investment options (projects). The organization has $10,000,000 available for the total investment. Bold lines mark the best selection 1, 3, 5, 6, and 7, which will cost $9,750,000 and create a payoff of 16,175,000. All other combinations would either exceed the budget or yield a lower payoff.[2]

12.2.6 Influence diagram

Much of the information in a decision tree can be represented more compactly as an influence diagram, focusing attention on the issues and relationships between events.



The squares represent decisions, the ovals represent action, and the diamond represents results.

12.3 Advantages and disadvantages

Among decision support tools, decision trees (and influence diagrams) have several advantages. Decision trees:

- Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
- Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.
- Allow the addition of new possible scenarios
- Help determine worst, best and expected values for different scenarios
- Use a white box model. If a given result is provided by a model.
- Can be combined with other decision techniques.

Disadvantages of decision trees:

- For data including categorical variables with different number of levels, information gain in decision trees are biased in favor of those attributes with more levels.[3]
- Calculations can get very complex particularly if many values are uncertain and/or if many outcomes are linked.

12.4 See also

12.5 References

[1] Quinlan, J. R. (1987). "Simplifying decision trees". *International Journal of Man-Machine Studies* **27** (3): 221. doi:10.1016/S0020-7373(87)80053-6.

[2] Y. Yuan and M.J. Shaw, Induction of fuzzy decision trees. Fuzzy Sets and Systems 69 (1995), pp. 125–139

[3] 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).

12.6 Further reading

- Cha, Sung-Hyuk; Tappert, Charles C (2009). "A Genetic Algorithm for Constructing Compact Binary Decision Trees". *Journal of Pattern Recognition Research* **4** (1): 1–13. doi:10.13176/11.44.

12.7 External links

- SilverDecisions: a free and open source decision tree software
- Decision Tree Analysis mindtools.com
- Decision Analysis open course at George Mason University
- Extensive Decision Tree tutorials and examples

Chapter 13

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]

13.1 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.

13.2 Algorithm

13.2.1 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.

13.2.2 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.

13.2.3 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.

13.2.4 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]

13.3 Properties

13.3.1 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]

13.3.2 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]

13.4 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]

13.5 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]

13.6 See also

- Decision tree learning
- Gradient boosting
- Randomized algorithm
- Bootstrap aggregating (bagging)
- Ensemble learning
- Boosting
- Non-parametric statistics
- Kernel random forest

13.7 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.

13.8 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 14

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.

14.1 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.

14.2 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]

14.3 Common types of ensembles

14.3.1 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.

14.3.2 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]

14.3.3 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 .

14.3.4 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]

14.3.5 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.

14.3.6 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]

14.3.7 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]

14.4 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

14.5 Further reading

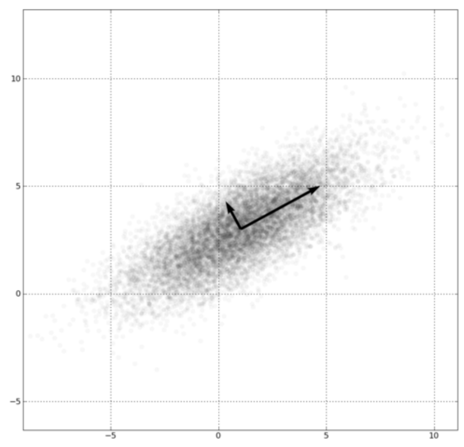
- Zhou Zhihua (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC. ISBN 978-1-439-83003-1.
- Robert Schapire; Yoav Freund (2012). *Boosting: Foundations and Algorithms*. MIT. ISBN 978-0-262-01718-3.

14.6 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 15

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 $\mathbf{X}^T\mathbf{X}$ 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]

15.1 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.

15.2 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.

15.2.1 First component

The first loading vector $\mathbf{w}_{(1)}$ thus has to satisfy

$$\mathbf{w}_{(1)} = \underset{\Vert\mathbf{w}\Vert=1}{\arg\max}\,\left\{\sum_i (t_1)^2_{(i)}\right\} = \underset{\Vert\mathbf{w}\Vert=1}{\arg\max}\,\left\{\sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w}\right)^2\right\}$$

Equivalently, writing this in matrix form gives

$$\mathbf{w}_{(1)} = \underset{\Vert\mathbf{w}\Vert=1}{\arg\max}\,\{\Vert\mathbf{Xw}\Vert^2\} = \underset{\Vert\mathbf{w}\Vert=1}{\arg\max}\,\left\{\mathbf{w}^T\mathbf{X}^T\mathbf{Xw}\right\}$$

Since $\mathbf{w}_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_{(1)} = \arg\max\,\left\{\frac{\mathbf{w}^T\mathbf{X}^T\mathbf{Xw}}{\mathbf{w}^T\mathbf{w}}\right\}$$

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, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}\}\ \mathbf{w}_{(1)}$.

15.2.2 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

$$\mathbf{w}_{(k)} = \underset{\Vert\mathbf{w}\Vert=1}{\arg\max}\,\left\{\Vert\hat{\mathbf{X}}_k\mathbf{w}\Vert^2\right\} = \arg\max\,\left\{\tfrac{\mathbf{w}^T\hat{\mathbf{X}}_k^T\hat{\mathbf{X}}_k\mathbf{w}}{\mathbf{w}^T\mathbf{w}}\right\}$$

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 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$

15.2.3 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 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$

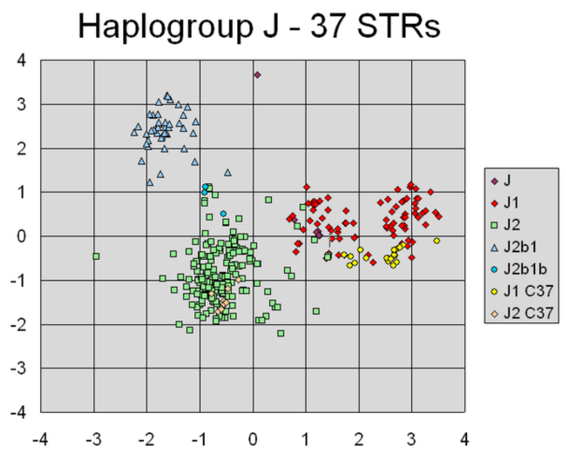
($\lambda_{(k)}$ being equal to the sum of the squares over the dataset associated with each component k: $\lambda_{(k)} = \Sigma i\ tk^2{}_{(i)} = \Sigma i\ (\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)})^2$)

15.2.4 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.

15.2.5 Singular value decomposition

The principal components transformation can also be associated with another matrix factorisation, the singular value decomposition (SVD) of **X**,

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{W}^T$$

Here $\boldsymbol{\Sigma}$ 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

$$\begin{aligned}\mathbf{X}^T\mathbf{X} &= \mathbf{W}\boldsymbol{\Sigma}\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{W}^T\\ &= \mathbf{W}\boldsymbol{\Sigma}^2\mathbf{W}^T\end{aligned}$$

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

$$\begin{aligned}\mathbf{T} &= \mathbf{X}\mathbf{W}\\ &= \mathbf{U}\boldsymbol{\Sigma}\mathbf{W}^T\mathbf{W}\\ &= \mathbf{U}\boldsymbol{\Sigma}\end{aligned}$$

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:

$$\mathbf{T}_L = \mathbf{U}_L\boldsymbol{\Sigma}_L = \mathbf{X}\mathbf{W}_L$$

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].

15.3 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 $\boldsymbol{\Sigma}$) 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.

15.4 Table of symbols and abbreviations

15.5 Properties and limitations of PCA

15.5.1 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: $\alpha_k'\alpha_k = 1, k = 1, \cdots, p$

15.5.2 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.

15.5.3 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.

15.6 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 $\mathbf{X}$:

$$\mathbf{Y} = \mathbb{KLT}\{\mathbf{X}\}$$

15.6.1 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 $n \times p$.

15.6.2 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 $p \times 1$.

$$u[j] = \frac{1}{n}\sum_{i=1}^{n} X[i,j]$$

15.6.3 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:

$$h[i] = 1 \qquad \text{for} i = 1, \ldots, n$$

15.6.4 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

15.6.5 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.

15.6.6 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.

15.6.7 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:

  $$g[j] = \sum_{k=1}^{j} D[k,k] \qquad \text{for} \qquad j = 1,\dots,p$$

15.6.8 Select a subset of the eigenvectors as basis vectors

- Save the first L columns of **V** as the $p \times L$ matrix **W**:

  $$W[k,l] = V[k,l] \qquad \text{for} \qquad k = 1,\dots,p \qquad l = 1,\dots,L$$

  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

  $$\frac{g[L]}{g[p]} \geq 0.9$$

15.6.9 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

15.6.10 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**.

15.7 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:

$$\begin{aligned} \mathrm{var}(PX) &= \mathbb{E}[PX\ (PX)^\dagger] \\ &= \mathbb{E}[PX\ X^\dagger P^\dagger] \\ &= P\ \mathbb{E}[XX^\dagger]P^\dagger \\ &= P\ \mathrm{var}(X)P^{-1} \end{aligned}$$

Hence $(*)$ holds if and only if $\mathrm{var}(X)$ 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.

15.7.1 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 **r**

This algorithm is simply an efficient way of calculating $\mathbf{X^T X}$ **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.

15.7.2 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 $\mathbf{w}_1{}^{\mathrm{T}}$ from **X**. The outer product, $\mathbf{t}_1\mathbf{w}_1{}^{\mathrm{T}}$ 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]

15.7.3 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]

15.8 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.

15.9 Applications

15.9.1 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.

15.10 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]

15.11 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.

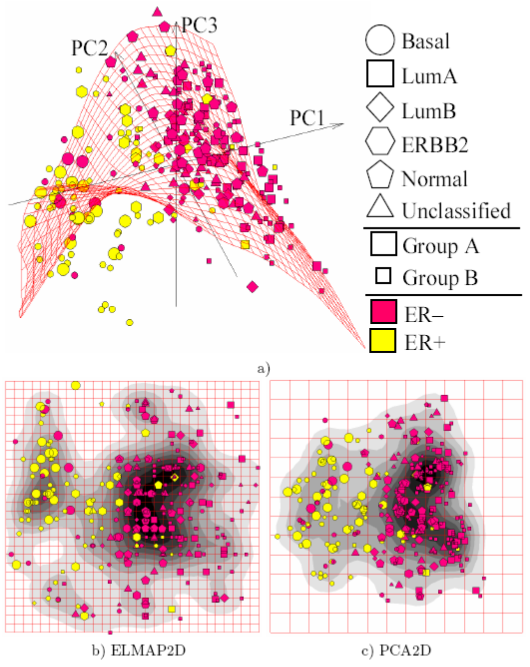
15.12 Correspondence analysis

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]

15.13 Generalizations

15.13.1 Nonlinear generalizations

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.



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]

15.13.2 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.

15.13.3 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.

15.13.4 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.

15.13.5 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.

15.13.6 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.

15.14 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.

15.15 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

15.16 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

15.17 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. Iisbn=978-2-7535-0938-2
- Pagès Jérôme (2014). *Multiple Factor Analysis by Example Using R*. Chapman & Hall/CRC The R Series London 272 p

15.18 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 16

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.

16.1 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]

16.2 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.

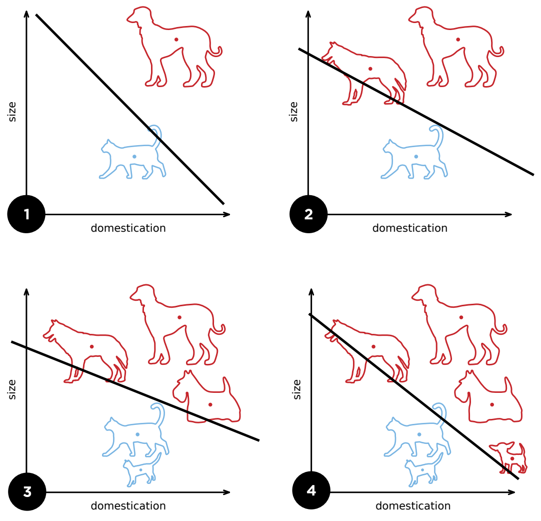
The value of $f(x)$ (0 or 1) is used to classify x as either a positive or a negative instance, in the case of a binary classification problem. If b is negative, then the weighted combination of inputs must produce a positive value greater than $|b|$ in order to push the classifier neuron over the 0 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.

16.3 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.



A diagram showing a perceptron updating its linear boundary as more training examples are added.

16.3.1 Definitions

We first define some variables:

- $y = f(\mathbf{z})$ 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*.
- $x_{j,0} = 1$.

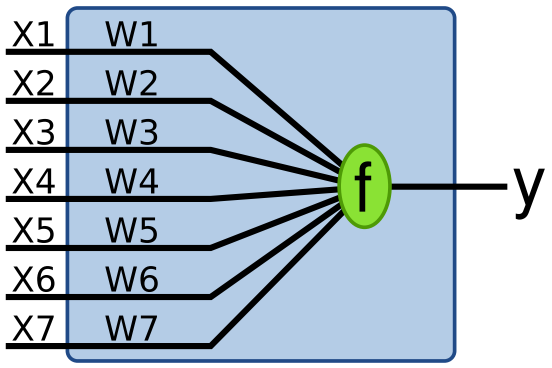
To represent the weights:

- 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.

16.3.2 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.

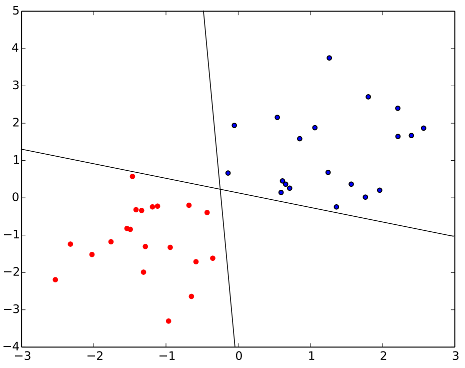
The algorithm updates the weights after steps 2a and 2b. These weights are immediately applied to a pair in the training set, and subsequently updated, rather than waiting until all pairs in the training set have undergone these steps.

16.3.3 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.

16.4 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.

16.5 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

16.6 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:

$$\hat{y} = \text{argmax}_y\, f(x,y) \cdot w.$$

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 $\text{argmax}_y f(x,y) \cdot w$ 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).

16.7 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

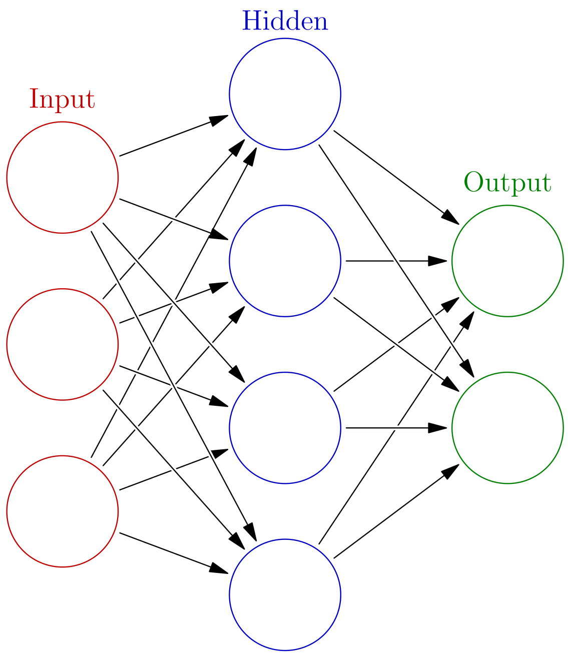
16.8 External links

- 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 17

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.

17.1 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.

17.2 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.

17.2.1 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]

17.3 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).

17.3.1 Network function

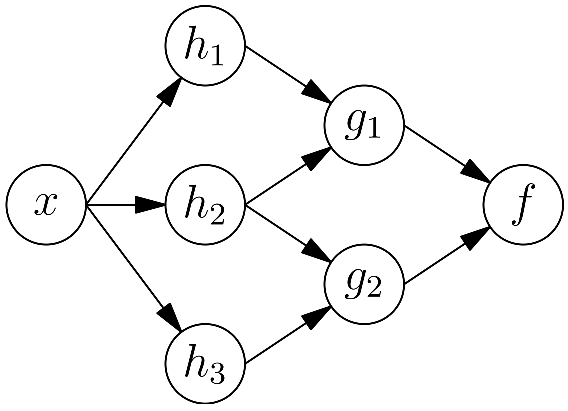
See also: Graphical models

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

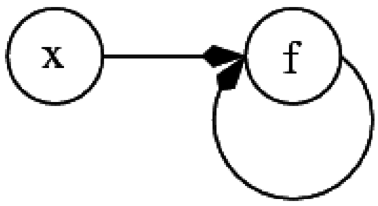
This figure depicts such a decomposition of f , with dependencies between variables indicated by arrows. These can be interpreted in two ways.

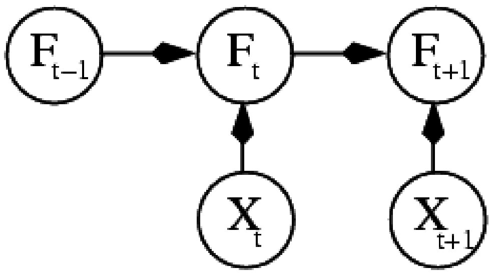
The first view is the functional view: the input x is transformed into a 3-dimensional vector h , which is then transformed into a 2-dimensional vector g , which is finally transformed into f . This view is most commonly encountered in the context of optimization.

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.

17.3.2 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:

17.3.3 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

17.3.4 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

17.4 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.

17.5 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.

17.5.1 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]

17.5.2 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.

17.6 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.

17.7 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.

17.8 Theoretical properties

17.8.1 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]

17.8.2 Capacity

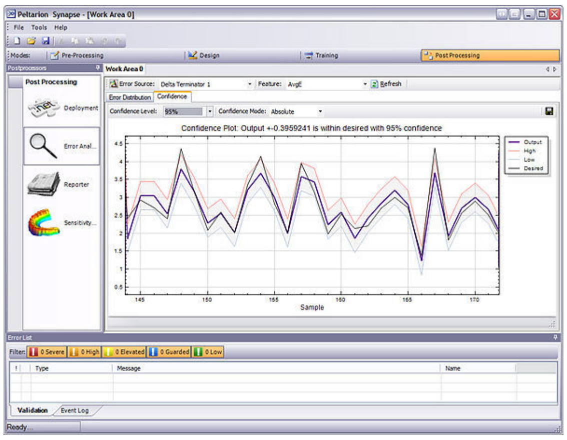
Artificial neural network models have a property called 'capacity', which roughly corresponds to their ability to model any given function. It is related to the amount of information that can be stored in the network and to the notion of complexity.

17.8.3 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.

17.8.4 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.



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:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}}$$

17.9 Controversies

17.9.1 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)

17.9.2 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.

17.9.3 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]

17.9.4 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]

17.10 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.
-
-

17.11 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)

17.12 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.

17.13 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

17.14 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 18

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]

18.1 Introduction

18.1.1 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)

18.1.2 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]

18.2 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]

18.3 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]

18.4 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.

18.4.1 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]

18.4.2 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.

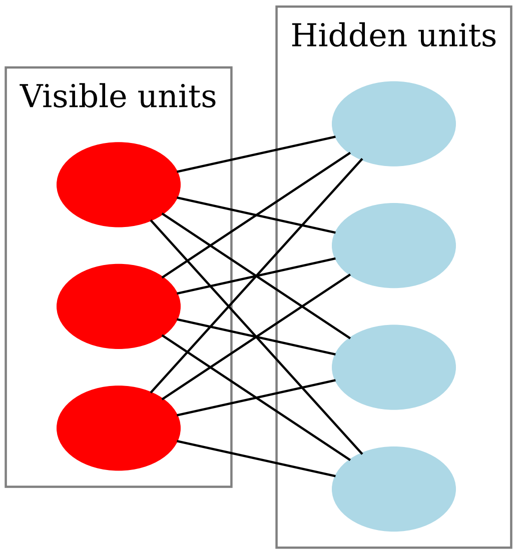
18.4.3 Deep belief networks

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]

18.4.4 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]

18.4.5 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]

18.4.6 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:

$p(\boldsymbol{\nu}) = \frac{1}{Z}\sum_h e^{\sum_{ij} W^{(1)}_{ij}\nu_i h^{(1)}_j + \sum_{jl} W^{(2)}_{jl} h^{(1)}_j h^{(2)}_l + \sum_{lm} W^{(3)}_{lm} h^{(2)}_l h^{(3)}_m}$

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]

18.4.7 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]

18.4.8 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]

18.4.9 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.

18.4.10 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.

18.4.11 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:

$$p(\boldsymbol{\nu},\psi) = \frac{1}{Z}\sum_h e^{\sum_{ij} W^{(1)}_{ij}\nu_i h^1_j + \sum_{jl} W^{(2)}_{jl} h^1_j h^2_l + \sum_{lm} W^{(3)}_{lm} h^2_l h^3_m}$$

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 :

$$P(\nu, h^1, h^2|h^3) = \frac{1}{Z(\psi,h^3)} e^{\sum_{ij} W^{(1)}_{ij}\nu_i h^1_j + \sum_{jl} W^{(2)}_{jl} h^1_j h^2_l + \sum_{lm} W^{(3)}_{lm} h^2_l h^3_m}$$

18.4.12 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]

18.4.13 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.

18.4.14 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.

18.4.15 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.

18.5 Applications

18.5.1 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.

18.5.2 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]

18.5.3 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]

18.5.4 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]

18.5.5 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]

18.6 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]

18.7 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.

18.8 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])

18.9 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.

18.10 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

18.11 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] J. Baker, Li Deng, Jim Glass, S. Khudanpur, C.-H. Lee, N. Morgan, and D. O'Shaughnessy (2009). “Research Developments and Directions in Speech Recognition and Understanding, Part 1,” *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 75-80, 2009.

[22] Y. Bengio (1991). "Artificial Neural Networks and their Application to Speech/Sequence Recognition," Ph.D. thesis, McGill University, Canada.

[23] L. Deng, K. Hassanein, M. Elmasry. (1994) "Analysis of correlation structure for a neural predictive model with applications to speech recognition," *Neural Networks*, vol. 7, No. 2., pp. 331-339.

[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/

18.12 Text and image sources, contributors, and licenses

18.12.1 Text

- **Machine learning** *Source:* https://en.wikipedia.org/wiki/Machine_learning?oldid=670568665 *Contributors:* Arvindn, ChangChienFu, Michael Hardy, Kku, Delirium, Ahoerstemeier, Ronz, BenKovitz, Mxn, Hike395, Silvonen, Furrykef, Buridan, Jmartinezot, Phoebe, Shizhao, Topbanana, Robbot, Plehn, KellyCoinGuy, Fabiform, Centrx, Giftlite, Seabhcan, Levin, Dratman, Jason Quinn, Khalid hassani, Utcursch, APH, Gene s, Clemwang, Nowozin, Bender235, ZeroOne, Superbacana, Aaronbrick, Jojit fb, Nk, Rajah, Tritium6, Haham hanuka, Mdd, HasharBot~enwiki, Vilapi, Arcenciel, Denoir, Wjbean, Stephen Turner, Rrenaud, Leondz, Soultaco, Ruud Koot, BlaiseFEgan, JimmyShelter~enwiki, Essjay, Joerg Kurt Wegner, Adiel, BD2412, Qwertyus, Rjwilmsi, Emrysk, VKokielov, Eubot, Celendin, Intgr, Predictor, Kri, BMF81, Chobot, Bobdc, Adoniscik, YurikBot, Misterwindupbird, Trondtr, Nesbit, Grafen, Gareth Jones, Srinivasasha, Crasshopper, DaveWF, Masatran, CWenger, Fram, KnightRider~enwiki, SmackBot, Mneser, InverseHypercube, CommodiCast, Jyoshimi, Mcld, KYN, Ohnoitsjamie, Chris the speller, FidesLT, Cfallin, Moorejh, JonHarder, Baguasquirrel, Krexer, Shadow1, Philpraxis~enwiki, Sina2, ChaoticLogic, NongBot~enwiki, RexSurvey, Beetstra, WMod-NS, Julthep, Dsilver~enwiki, Dicklyon, Vsweiner, Ctacmo, MTSbot~enwiki, Ralf Klinkenberg, Dave Runger, Doceddi, Scigrex14, Pgr94, Innohead, Bumbulski, Peterdjones, Dancter, Msnicki, Quintopia, Thijs!bot, Mereda, Djbwiki, GordonRoss, Kinimod~enwiki, Damienfrancois, Natalie Erin, Seaphoto, AnAj, Ninjakannon, Kimptoc, Penguinbroker, The Transhumanist, Jrennie, Hut 8.5, Kyhui, Magioladitis, Ryszard Michalski, Jwojt, Transcendence, Tedickey, Pebkac, Robotman1974, Jroudh, Businessman332211, Pmbhagat, Calltech, STBot, Glrx, Nickvence, Salih, AntiSpamBot, Gombang, Chriblo, Dana2020, DavidCBryant, Bonadea, WinterSpw, RJASE1, Funandtrvl, James Kidd, LokiClock, Redgecko, Markcsg, Jrljrl, Like.liberation, A4bot, Daniel347x, Joel181, Wikidemon, Lordvolton, Defza, Chrisoneall, Spiral5800, Cvdwalt, Why Not A Duck, Sebastjanmm, LittleBenW, Gal chechik, Biochaos, Cmbishop, Jbmurray, IradBG, Smsarmad, Scorpion451, Kumioko (renamed), CharlesGillingham, StaticGull, CultureDrone, Anchor Link Bot, ImageRemovalBot, ClueBot, GorillaWarfare, Ahyeek, Sonu mangla, Ggia, Debejyo, D.scain.farenzena, He7d3r, Magdon~enwiki, WilliamSewell, Jim15936, Vanished user uih38riiw4hjlsd, Evansad, PseudoOne, André P Ricardo, Darnelr, MystBot, Dsimic, YrPolishUncle, MTJM, Addbot, Mortense, Fyrael, Aceituno, MrOllie, LaaknorBot, Jarble, Movado73, Luckas-bot, QuickUkie, Yobot, NotARusski, Genius002, Examtester, AnomieBOT, Piano non troppo, Materialscientist, Clickey, Devantheryv, Vivohobson, ArthurBot, Quebec99, Xqbot, Happyrabbit, Gtfjbl, Kithira, J04n, Addingrefs, Webidiap, Shirik, Joehms22, Aaron Kauppi, Velblod, Prari, FrescoBot, Jdizzle123, WhatWasDone, Siculars, Proffviktor, Boxplot, Swordsmankirby, Wikinacious, Skyerise, Mostafa mahdieh, Lars Washington, TobeBot, AXRL, Иъ Лю Ха, BertSeghers, Edouard.darchimbaud, Winnerdy, Zosoin, Helwr, EmausBot, Dzkd, Wht43, Chire, GZ-Bot, Jcautilli, AManWithNoPlan, Pintaio, L Kensington, Ataulf, Yoshua.Bengio, Casia wyq, Ego White Tray, Blaz.zupan, Shinosin, Marius.andreiana, Lovok Sovok, Graytay, Liuyipei, ClueBot NG, Tillander, Keefaas, Lawrence87, Aiwing, Pranjic973, Candace Gillhoolley, Robiminer, Leonardo61, Wrdieter, Arrandale, O.Koslowski, WikiMSL, Helpful Pixie Bot, RobertPollak, BG19bot, Smorsy, Mohamed CJ, Lisasolomonsalford, Anubhab91, Chafe66, Ishq2011, Autologin, DasAllFolks, Billhodak, Debora.riu, Ohandyya, Davidmetcalfe, Mdann52, JoshuSasori, Ulugen, IjonTichyIjonTichy, Keshav.dhandhania, Mogism, Djfrost711, Bkuhlman80, Frosty, Jamesx12345, Shubhi choudhary, Jochen Burghardt, Joeinwiki, Brettrmurphy, Ppilotte, Delafé, InnocuousPilcrow, Kittensareawesome, Statpumpkin, Neo Poz, Dustin V. S., TJLaher123, Ankit.ufl, Francisbach, Aleks-ger, MarinMersenne, LokeshRavindranathan, Tonyszedlak, Proneat123, GrowthRate, Sami Abu-El-Haija, Mpgoldhirsh, Work Shop Corpse, Superploro, Dawolakamp, Justincahoon, Jorge Guerra Pires, Hm1235, Velvel2, Vidhul sikka, Erik Itter, Annaelison, Tgriffin9, Chazdywaters, Rmashrmash, Komselvam, Robbybluedogs, EricVSiegel, KenTancwell, Justinqnabel, Rusky.ai, Datapablo, Aetilley, JenniferTheEmpress0, Dsysko and Anonymous: 365
- **Scikit-learn** *Source:* https://en.wikipedia.org/wiki/Scikit-learn?oldid=665329845 *Contributors:* Nealmcb, Qwertyus, Ogrisel, Janto, Cydebot, Gaijin42, Gioto, Andreas Mueller, Beeblebrox, Addbot, Tadej janez, RjwilmsiBot, Zephyrus Tavvier, BG19bot, Mark Arsten, NelleV, Mark viking, Impsswoon, Wdv4758h, T3kcit, Blink1073 and Anonymous: 10
- **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, Albertttt, 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
- **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, Citruscoconut, GargantuanDan, TejDham, Valcust, Jamesmcmahon0, Waynechew87, Nullhypothesistester, Penitence, Anrnusna, Kfudala, Tertius51, Monkbot, BethNaught, TedPSS, 1980na, Libarian, SolidPhase, Anon124, Zjillanz and Anonymous: 324

- **Linear regression** *Source:* https://en.wikipedia.org/wiki/Linear_regression?oldid=671138540 *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, 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
- **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

- **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
- **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
- **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=670528911 *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, Nameofmyveryown, 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, 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
- **Decision tree** *Source:* https://en.wikipedia.org/wiki/Decision_tree?oldid=662147818 *Contributors:* Waveguy, Michael Hardy, Rp, Ronz, TheEternalVortex, Nerd~enwiki, Hike395, Mydogategodshat, Charles Matthews, Dcoetzee, Dysprosia, Daniel Quinlan, Raul654, Robbot, Sbisolo, Altenmann, AaronS, Tobias Bergemann, Giftlite, Bfinn, Peruvianllama, Neilc, Pgan002, Raand, MacGyverMagic, Karl-Henner, Fintor, Andreas Kaufmann, Rich Farmbrough, Aaronbrick, Bobo192, Pearle, Mdd, Frodet, Wtmitchell, Jheald, Oleg Alexandrov, StradivariusTV, Jeff3000, Damicatz, Rjwilmsi, AlexHorovitz, Salix alba, Michal.burda, Evandrojr, DouglasGreen~enwiki, Dmccreary, Fijal, RexNL, BMF81, DVdm, The Rambling Man, YurikBot, Ogai, Dotancohen, C777, Yrithinnd, Retardo, SmackBot, Alexlinsker@gmail.com, InverseHypercube, Verne Equinox, Object01, Chris the speller, DHN-bot~enwiki, Chendy, Can't sleep, clown will

18.12.2 Images

18.12.3 Content license