

MA333 Introduction to Big Data Analysis

Course Introduction

Zhen Zhang

Southern University of Science and Technology

Outlines

Course Syllabus

What Is Data Science

Machine Learning

Mathematical Representation

Conclusion

Course Info

- Semester 2023-2024 Spring
- Instructor : ZHANG, Zhen (张振)
- Office : Room M5014, School of Science
- Phone : 88018753
- Email : zhangz@sustech.edu.cn
- Office hours : Wednesday afternoon, 15 :00-17 :00 ; or send email to make an appointment for other time.
- Lecture : 3 credits, 3 hours per week.
- Prerequisite : Calculus I&II, MA101b&MA102b, (or Mathematical Analysis I&II, MA101a&MA102a) ; Linear Algebra I, MA103b ; Probability Theory, MA215 (or Probability Theory and Mathematical Statistics).

Grading Policy

- Homework : Approximately 6 homework assignments (including programming assignments and written problems). The written homework could be handed in after class.
- In-class quizzes : typically once every two weeks, test how well you learned about the basic concepts, including fill-in-the-blank, single and multiple choices, and simple Q & A
- Programming projects : need to write codes and reports
- One closed-book final exam
- Grading policy : assignments (30%), quizzes (15%), programming projects (20%), and the final exam (35%).

Contents

- Intended for undergraduate students who are interested in pursuing industrial work and research in big data science.
- Concise and self-contained introduction to mathematical aspect of big data science, including **theoretical analysis, algorithms and programming with python**
- Major topics :
 - Introduction to python programming and data preprocessing
 - Three fundamental problems : classification, regression, clustering
 - Model selection, dimensionality reduction
 - Hot topics : text analysis, social network analysis, neural network and deep learning, distributed computing, and recommender systems if time permits

References

- 数据科学导引，欧高炎等著，高等教育出版社，2017.
- 机器学习，周志华著，清华大学出版社，2016.
- An Introduction to Statistical Learning with Applications in R, by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, Springer, 2013.
<http://web.stanford.edu/~hastie/pub.htm>
- Pattern Recognition and Machine Learning, by Christopher M. Bishop, Springer, 2006.
- The Elements of Statistical Machine Learning : Data mining, Inference and Prediction, 2nd Edition, by Trevor Hastie, Robert Tibshirani, and Jerome Friedman, Springer, 2009.
- Understanding Machine Learning, by Shai Shalev-Shwartz and Shai Ben-David, Cambridge University Press, 2018.
- Deep learning, by Ian Goodfellow, Yoshua Bengio, and Aaron Courville, MIT Press, 2016.

References



Outlines

Course Syllabus

What Is Data Science

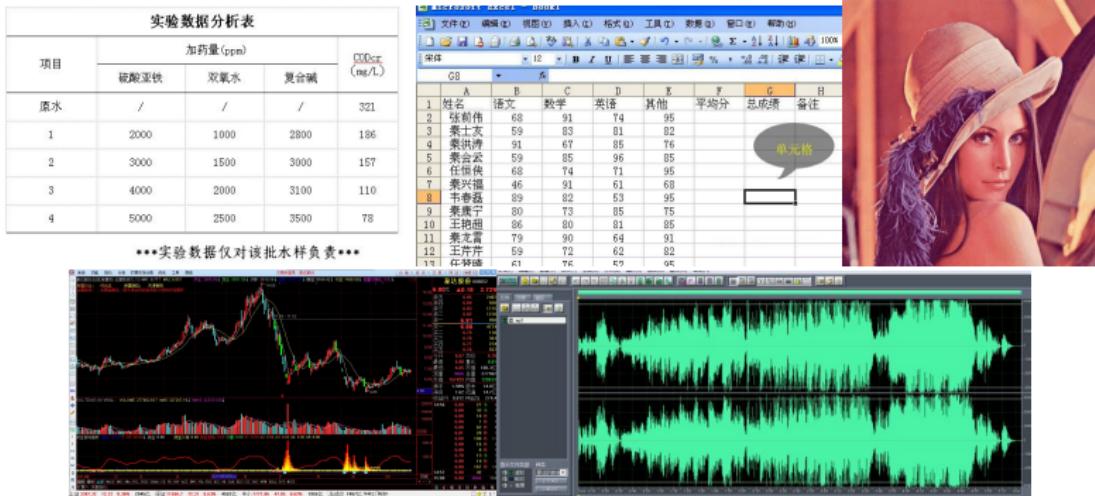
Machine Learning

Mathematical Representation

Conclusion

Some Examples of Data

Can you give some examples of data ?



Table, 1D signal (audio, stock price), 2D signal (image), 3D signal (video), etc.

Big Data : 4 Big “V”

- Volume : KB, MB, GB (10^9 bytes), TB, PB, EB (10^{18} bytes), ZB, YB
- Variety : different sources from business to industry, different types
- Value : redundant information contained in the data, need to retrieve useful information
- Velocity : fast speed for information transfer



What is data science

- Retrieve information from data with the help of computational power
- Transfer the information into knowledge
- Two perspectives of data sciences :
 - Study science with the help of data : bioinformatics, astrophysics, geosciences, etc.
 - Use scientific methods to exploit data : statistics, machine learning, data mining, pattern recognition, data base, etc.

Study Science with the Help of Data

A pioneering work of data science : Kepler's Laws



开普勒：分析数据产生价值



行星	周期 (年)	平均距离	周期 ² /距离 ³
水星	0.241	0.39	0.98
金星	0.615	0.72	1.01
地球	1.00	1.00	1.00
火星	1.88	1.52	1.01
木星	11.8	5.20	0.99
土星	29.5	9.54	1.00
天王星	84.0	19.18	1.00
海王星	165	30.06	1.00

Scientific Study of Data

- Grabbing data : business and industrial problem, professional areas
- Storing data : engineering problem, computer science, electronic engineering
- Analyzing data (**key problem**) : scientific problem, mathematics, statistics, computer science

Data Analysis

- Ordinary data types :
 - Table : classical data (could be treated as matrix)
 - Set of points : mathematical description
 - Time series : text, audio, stock prices, DNA sequences, etc.
 - Image : 2D signal (or matrix equivalently, e.g., pixels), MRI, CT, supersonic imaging
 - video : 2D in space and 1D in time (another kind of time series)
 - Webpage and newspaper : time series with spatial structure
 - Network : relational data, graph (nodes and edges)
- Basic assumption : the data are generated from an underlying model, which is unknown in practice
 - Set of points : probability distribution
 - Time series : stochastic processes, e.g., Hidden Markov Model (HMM)
 - Image : random fields, e.g., Gibbs random fields
 - Network : graphical models, Bayesian models

Difficulties

- Huge volume of data
- Extremely high dimensions
 - Curse of dimensionality : the model complexity and computational complexity become exponentially increasing with the growth of dimension
 - Solutions :
 - Make use of prior information
 - Restrict to simple models
 - Make use of special structures, e.g., sparsity, low rank, smoothness
 - Dimensionality reduction, e.g., PCA, LDA, etc.
- Complex variety of data
- Large noise : data are always contaminated with noises

Solution - Algorithms

- Algorithms are in the interdisciplinary part of computer science and mathematics : establish mathematical models, solve it numerically, implement it in the computer languages
- Reduce the algorithmic complexity, with the help of techniques from mathematics or computer science
- Distributional and parallel computing : divide-and-conquer, e.g., MapReduce, GPU
- IEEE 2006 top 10 algorithms in data mining : C4.5, K-Means, SVM, Apriori, EM, PageRank, NaiveBayes, K-Nearest Neighbors, AdaBoost, CART

Outlines

Course Syllabus

What Is Data Science

Machine Learning

Mathematical Representation

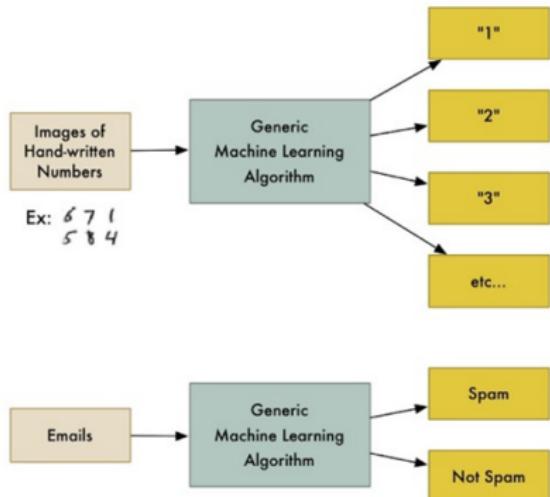
Conclusion

Definition

- Artificial Intelligence (AI) : learning from experiences (data), and improve the computer program adaptively
- Mathematics : Learning the underlying model from data, and generalize the model to adapt new data

We define *machine learning* as a set of methods that can automatically **detect patterns in data**, and then use the uncovered patterns to **predict future data**, or to **perform other kinds of decision making under uncertainty** (such as planning how to collect more data!).

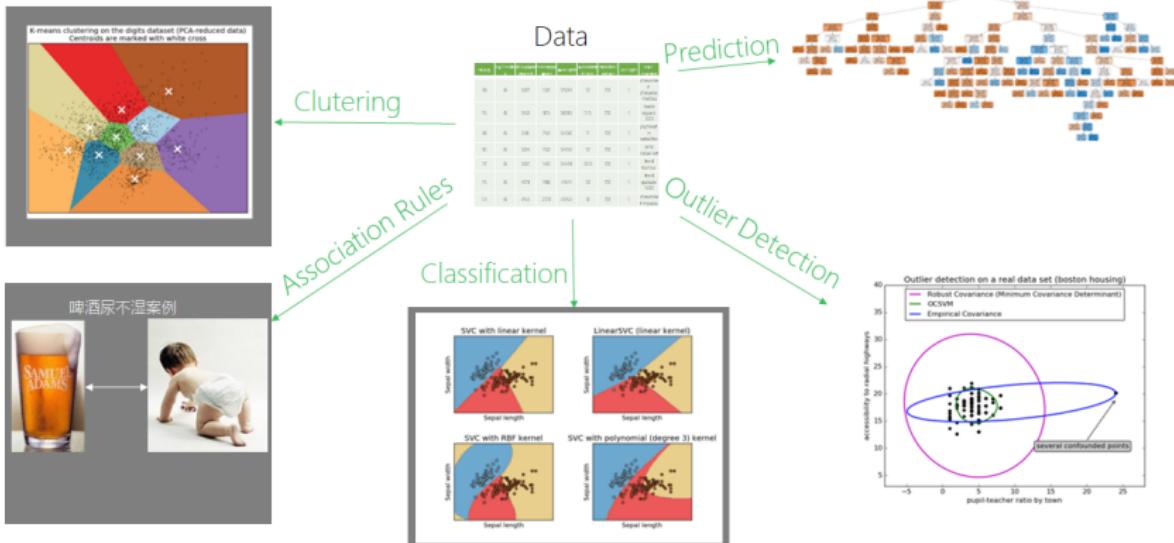
— 《Machine Learning: A probabilistic perspective》



Related Areas

- Control theory : optimize the cost with optimal control parameters
- Information theory : entropy, optimal coding with best information
- Psychology : reference for machine learning algorithms
- Neuroscience : artificial neural network
- Biology : genetic algorithms
- Theory of Computing : study the computational complexity
- Statistics : large-sample limiting behavior, statistical learning theory
- Artificial Intelligence : symbolic computing
- Bayesian theory : conditionally probabilistic network

Applications

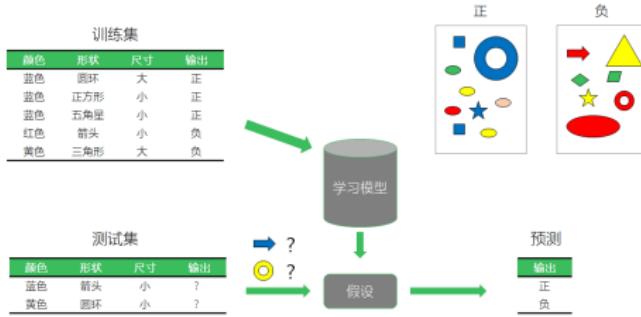


Supervised and Unsupervised Learning

- Supervised learning : classification, regression
- Unsupervised learning : density estimation, clustering, dimensionality reduction
- Semi-supervised learning : with missing data, e.g., EM ; self-supervised learning, learn the missing part of images, inpainting
- Reinforcement learning : play games, e.g., Go, StarCraft ; robotics ; auto-steering

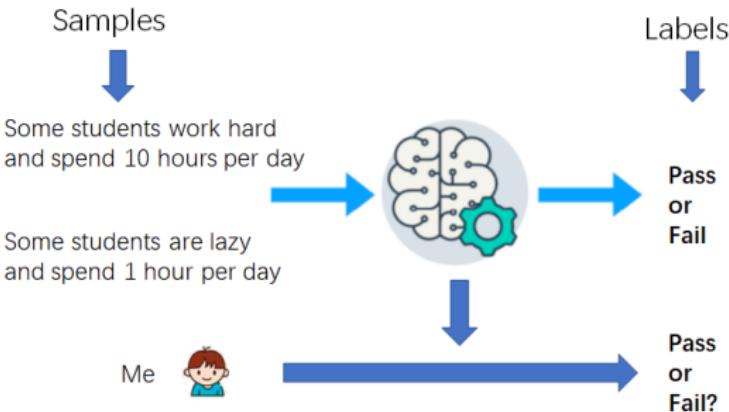
Supervised Learning

- Given labels of data : the labels could be symbols (spam or non-spam), integers (0 or 1), real numbers, etc.
- Training : find the optimal parameters (or model) to minimize the error between the prediction and target
- Classification : SVM, KNN, Desicion tree, etc.
- Regression : linear regression, CART, etc.



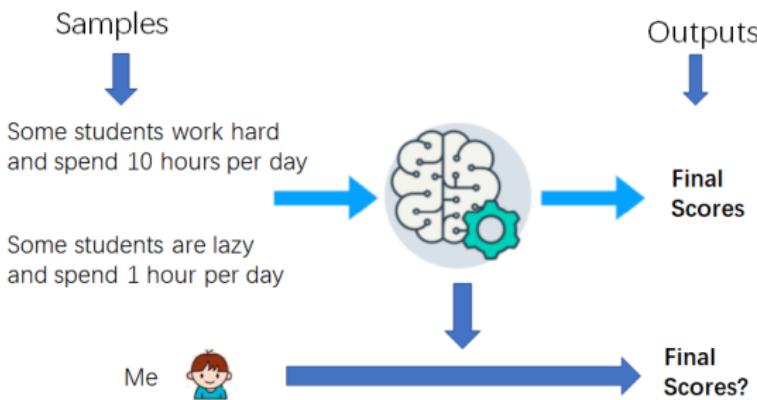
Classification

- Output is discrete
- Examples : given the study hours, in-class performance, and final grades (Pass or Fail) of past students, can you predict the final grades of the current students based on their study hours and in-class performance ?
- Applications : Credit risk evaluation, clinical prediction of tumor, classification of protein functions, etc.



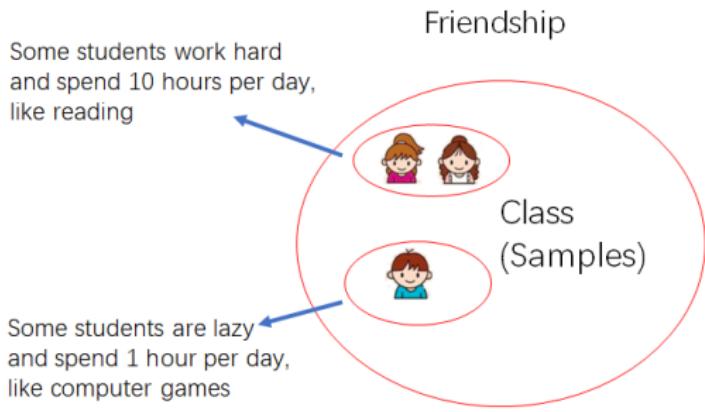
Regression

- Output is continuous
- Examples : given the study hours, in-class performance, and final scores of past students, can you predict the final scores of the current students based on their study hours and in-class performance ?
- Applications : epidemiology, finance, investment analysis, etc.



Unsupervised Learning

- No labels
- Optimize the parameters based on some natural rules, e.g., cohesion or divergence
- Clustering : K-Means, SOM

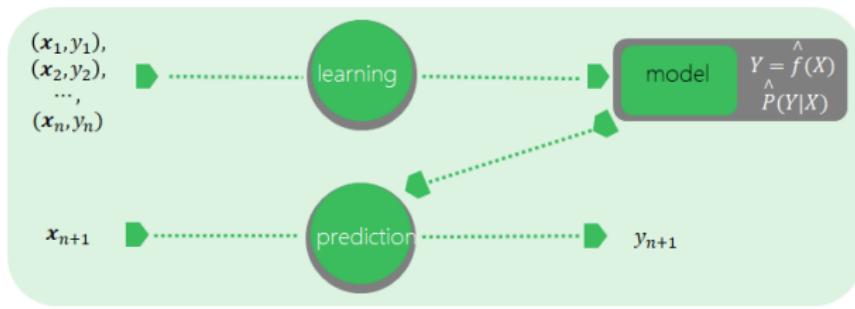


Representation of Data

- Input space $\mathcal{X} = \{\text{All possible samples}\}$; $\mathbf{x} \in \mathcal{X}$ is an input vector, also called feature, predictor, independent variable, etc.; typically multi-dimensional; e.g., $\mathbf{x} \in \mathbb{R}^P$ is a weight vector or coding vector
- Output space $\mathcal{Y} = \{\text{All possible results}\}$; $y \in \mathcal{Y}$ is an output vector, also called response, dependent variable, etc.; typically one-dimensional; e.g., $y = 0$ or 1 for classification problems, $y \in \mathbb{R}$ for regression problems.
- For supervised learning, assume that $(\mathbf{x}, y) \sim \mathcal{P}$, a joint distribution on the sample space $\mathcal{X} \times \mathcal{Y}$

Supervised Learning

- Goal : given \mathbf{x} , predict what is y ; in deterministic settings, find the dependence relation $y = f(\mathbf{x})$; in probabilistic settings, find the conditional distribution $P(y|\mathbf{x})$ of y given \mathbf{x}
- Training dataset : $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \stackrel{i.i.d.}{\sim} \mathcal{P}$, used to learn an approximation $\hat{f}(\mathbf{x})$ or $\hat{P}(y|\mathbf{x})$
- Test dataset : $\{(\mathbf{x}_j, y_j)\}_{j=n+1}^{n+m} \stackrel{i.i.d.}{\sim} \mathcal{P}$, used to make a prediction $\hat{y}_j = \hat{f}(\mathbf{x}_j)$ or $\hat{y}_j = \arg \max_{y_j} \hat{P}(y_j|\mathbf{x}_j)$, and verify how accurate the approximation is



Unsupervised Learning

- Goal : in probabilistic settings, find the distribution $P(\mathbf{x})$ of \mathbf{x} and approximate it ; there is no y
- Training dataset : $\{\mathbf{x}_i\}_{i=1}^n \stackrel{i.i.d.}{\sim} \mathcal{P}$, used to learn an approximation $\hat{P}(\mathbf{x})$; no test data in general



Learning Models

- Decision function (hypothesis) space :
 $\mathcal{F} = \{f_\theta | f_\theta = f_\theta(\mathbf{x}), \theta \in \Theta\}$ or $\mathcal{F} = \{P_\theta | P_\theta = P_\theta(y|\mathbf{x}), \theta \in \Theta\}$
- Loss function : a measure for the “goodness” of the prediction, $L(y, f(\mathbf{x}))$
 - 0-1 loss : $L(y, f(\mathbf{x})) = I_{y \neq f(\mathbf{x})} = 1 - I_{y=f(\mathbf{x})}$
 - Square loss : $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
 - Absolute loss : $L(y, f(\mathbf{x})) = |y - f(\mathbf{x})|$
 - Cross-entropy loss :
$$L(y, f(\mathbf{x})) = -y \log f(\mathbf{x}) - (1 - y) \log(1 - f(\mathbf{x}))$$
- Risk : in average sense,
$$R(f) = E_P[L(y, f(\mathbf{x}))] = \int_{\mathcal{X} \times \mathcal{Y}} L(y, f(\mathbf{x})) P(\mathbf{x}, y) d\mathbf{x} dy$$
- Target of learning : choose the best f^* to minimize $R_{exp}(f)$,
$$f^* = \min_f R_{exp}(f)$$

Risk Minimization Strategy

- Empirical risk minimization (ERM) : given training set

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n, R_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i))$$

- By law of large number, $\lim_{n \rightarrow \infty} R_{\text{emp}}(f) = R_{\text{exp}}(f)$
- Optimization problem : $\min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i))$

- Structural risk minimization (SRM) : given training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and a complexity functional $J = J(f)$,

$$R_{\text{srm}}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

- $J(f)$ measures how complex the model f is, typically the degree of complexity
- $\lambda \geq 0$ is a tradeoff between the empirical risk and model complexity
- Optimization problem : $\min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$

Algorithms

- Computational methods to solve the problem for f
- Numerical methods to solve the optimization problems
 - Gradient descent method, including coordinate descent, sequential minimal optimization (SMO), etc.
 - Newton's method and quasi-Newton's method
 - Combinatorial optimization
 - Genetic algorithms
 - Monte Carlo methods
 - ...

Model Assessment

Assume we have learned the model $y = \hat{f}(\mathbf{x})$, what is the error ?

- Training error : $R_{emp}(\hat{f}) = \frac{1}{n} \sum_{i=1}^n L(y_i, \hat{f}(\mathbf{x}_i))$, tells the difficulty of learning problem
- Test error : $e_{test}(\hat{f}) = \frac{1}{m} \sum_{j=n+1}^{n+m} L(y_j, \hat{f}(\mathbf{x}_j))$, tells the capability of prediction ; in particular, if 0-1 loss is used
 - Error rate : $e_{test}(\hat{f}) = \frac{1}{m} \sum_{j=n+1}^{n+m} I_{y_j \neq \hat{f}(\mathbf{x}_j)}$
 - Accuracy : $r_{test}(\hat{f}) = \frac{1}{m} \sum_{j=n+1}^{n+m} I_{y_j = \hat{f}(\mathbf{x}_j)}$
 - $e_{test} + r_{test} = 1$

Model Assessment (Cont')

- Generalization error :

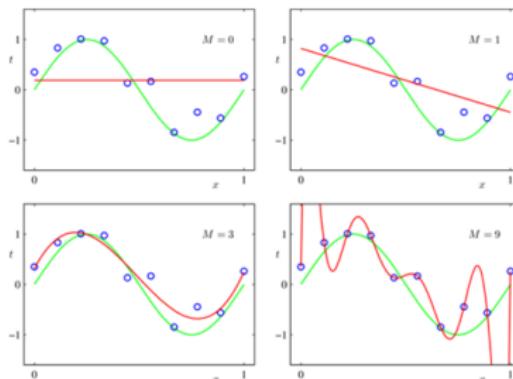
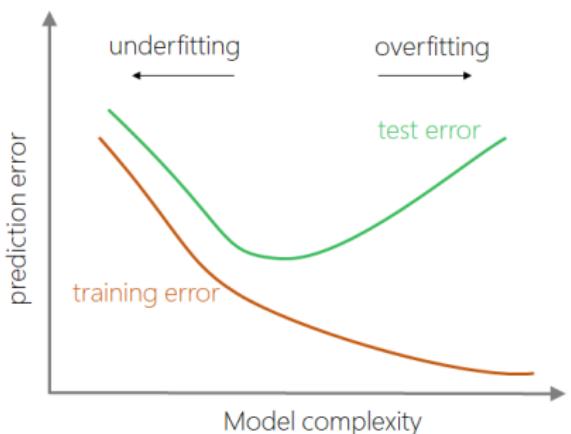
$$R_{\text{exp}}(\hat{f}) = E_P[L(y, \hat{f}(\mathbf{x}))] = \int_{\mathcal{X} \times \mathcal{Y}} L(y, \hat{f}(\mathbf{x})) P(\mathbf{x}, y) d\mathbf{x} dy,$$
 tells

the capability for predicting unknown data from the same distribution, its upper bound M defines the generalization ability

- As $n \rightarrow \infty$, $M \rightarrow 0$
- As F becomes larger, M increases

Overfitting

- Too many model parameters
- Better for training set, but worse for test set



fitting of degree M polynomial, green curve is the ground truth

Model Selection

- Regularization : $\min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) + \underbrace{\lambda J(f)}_{\text{penalty}}$, choose λ to minimize empirical risk and model complexity simultaneously
- Cross-validation (CV) : split the training set into training subset and validation subset, use training set to train different models repeatedly, use validation set to select the best model with the smallest (validation) error
 - Simple CV : randomly split the data into two subsets
 - K-fold CV : randomly split the data into K disjoint subsets with the same size, treat the union of $K - 1$ subsets as training set, the other one as validation set, do this repeatedly and select the best model with smallest mean (validation) error
 - Leave-one-out CV : $K = n$ in the previous case



Outlines

Course Syllabus

What Is Data Science

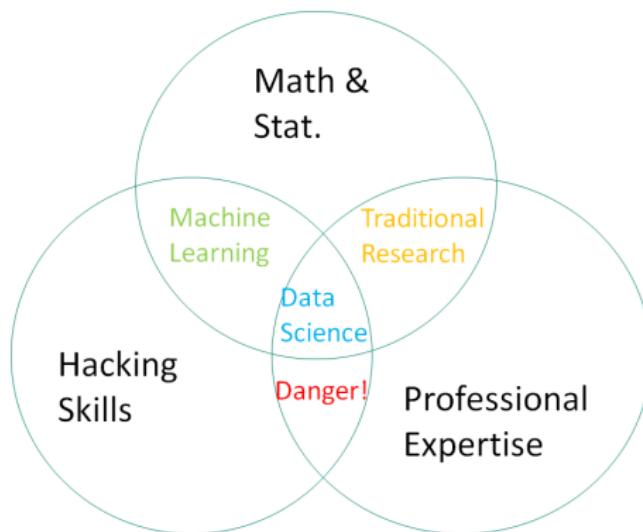
Machine Learning

Mathematical Representation

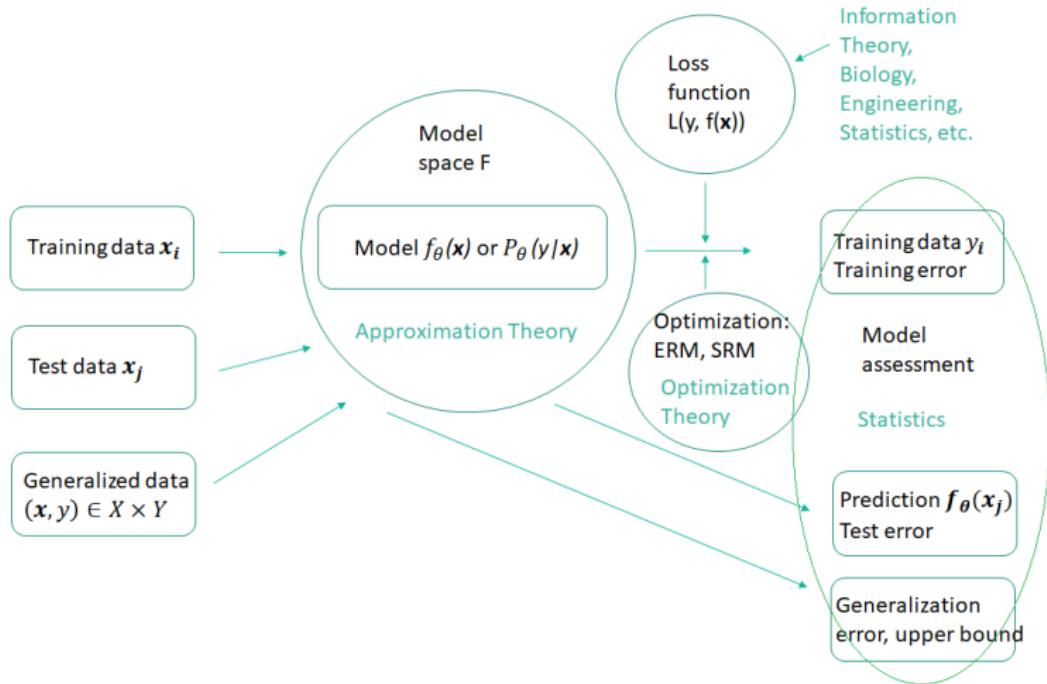
Conclusion

Data Science VS. Other Techniques

Data science is an interdisciplinary area using mathematics, statistics, computer science and engineering, and other profession techniques



Where Math and Statistics Emerge



MA333 Introduction to Big Data Analysis

Data Preprocessing

Zhen Zhang

Southern University of Science and Technology

Outlines

Basic Concepts

Data Preprocessing

Data Types

- Tabular data : matrices, vectors, objects, relations, etc.
 - Data objects : also called samples, examples, instances, data points, objects, tuples, vectors
 - Attributes : each row of a table, also called dimensions, features, variables
- Graphical data : networks, graphs, etc.
- Multi-media data : texts, images, videos, audios, etc.

行星	周期 (年)	平均距离	周期 ² /距离 ³
水星	0.241	0.39	0.98
金星	0.615	0.72	1.01
地球	1.00	1.00	1.00
火星	1.88	1.52	1.01
木星	11.8	5.20	0.99
土星	29.5	9.54	1.00
天王星	84.0	19.18	1.00
海王星	165	30.06	1.00

Types of Attributes

- Discrete : $x \in$ some countable sets, e.g., \mathbb{N}
 - Nominal : Countries={China, US, UK, France, Germany}, Universities={Peking U, Tsinghua U, SUSTech, Shenzhen U, HIT}, not comparable
 - Boolean : 0 or 1, male or female, spam or non-spam, etc.
 - Ordinal : Heights={tall, short}, Scores={A+, A, A-, B+, B, B-, C, C-, D, F}, can be compared, but cannot be operated arithmetically
- Continuous : $x \in$ some subset in \mathbb{R}^n
 - Numerical : Income, exact marks, weights, etc., can be operated arithmetically

Basic Statistics

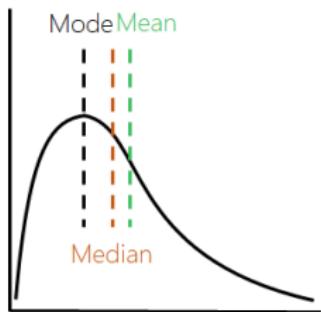
- Mean : $\text{E}X = \min_c \text{E}(X - c)^2 \approx \frac{1}{n} \sum_{i=1}^n x_i$
- Median :

$$\min_c \text{E}|X - c| = \begin{cases} x_{(\frac{n+1}{2})} & \text{if } n \text{ is odd} \\ (x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)})/2 & \text{otherwise} \end{cases}$$

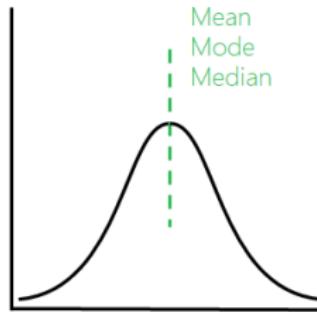
- Maximum : $\max_i x_i$; Minimum : $\min_i x_i$
- Quantile : a generalization of median, k -th q -quantile x_q :
 $P[X < x_q] \leq k/q$; interquartile range
 $(\text{IQR}) = Q_3(75\%) - Q_1(25\%)$
- Variance : $\text{Var}(X) = \text{E}[X - \text{E}X]^2 \approx \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$;
 Standard deviation : $\sqrt{\text{Var}(X)}$
- Mode : $\min_c \text{E}|X - c|^0$ = the most frequently occurring value
 (define $0^0 = 0$)

Central Tendency

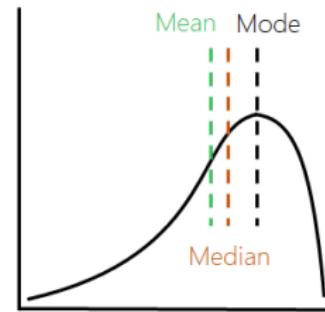
For one-peak skewed density distribution, empirical formula :
 $Mean - Mode = 3 \times (Mean - Median)$



Positive
skewed



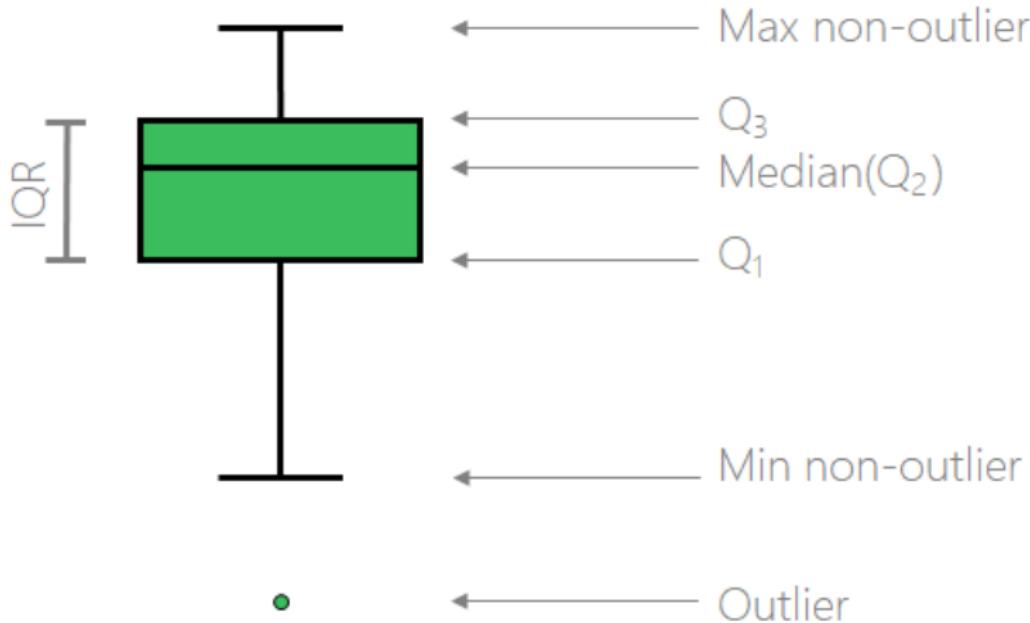
Symmetric



Negative
skewed

Box Plot

Measure the dispersion of data



Metrics

- Proximity :
 - Similarity : range is $[0, 1]$
 - Dissimilarity : range is $[0, \infty]$, sometimes distance
- For nominal data, $d(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_k I(x_{i,k} \neq x_{j,k})}{p}$; or one-hot encoding into Boolean data
- For Boolean data, symmetric distance $d(\mathbf{x}_i, \mathbf{x}_j) = \frac{r+s}{q+r+s+t}$ or Rand index $Sim_{Rand}(\mathbf{x}_i, \mathbf{x}_j) = \frac{q+t}{q+r+s+t}$; non-symmetric distance $d(\mathbf{x}_i, \mathbf{x}_j) = \frac{r+s}{q+r+s}$ or Jaccard index $Sim_{Jaccard}(\mathbf{x}_i, \mathbf{x}_j) = \frac{q}{q+r+s}$

		Sample <i>j</i>		sum
		1	0	
Sample <i>i</i>	1	<i>q</i>	<i>r</i>	<i>q+r</i>
	0	<i>s</i>	<i>t</i>	<i>s+t</i>
		sum	<i>q+s</i>	<i>r+t</i>
				<i>p</i>

Metrics : Distance

- Example : Let $H = F = 1$ and $L = S = 0$,
 $d(\text{LandRover}, \text{Jeep}) = \frac{1+0}{4+1+0} = 0.20$, $d(\text{LandRover}, \text{TOYOTA}) = \frac{3+1}{1+3+1} = 0.80$, $d(\text{Jeep}, \text{TOYOTA}) = \frac{3+2}{1+3+2} = 0.83$
- Minkowski distance : $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[h]{\sum_{k=1}^p |x_{ik} - x_{jk}|^h}$ is L_h -norm
 - Positive definiteness $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ and “=” if and only if $i = j$;
 - Symmetry $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$;
 - Triangle inequality $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k) + d(\mathbf{x}_k, \mathbf{x}_j)$

	Weight	Price	Acceleration	MPG	Quality	Sales Volume	Jeep	
Land Rover	H	H	F	H	H	L	1	0
Jeep	H	H	S	H	H	L	1	$q = 4$
TOYOTA	L	L	F	L	H	H	0	$s = 0$

Land Rover	1	$q = 4$	$r = 1$
	0	$s = 0$	$t = 1$

Metrics : Distance (Cont')

- Manhattan distance : $h = 1$,
and

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^p |x_{ik} - x_{jk}|$$

- Euclidean distance : $h = 2$,
and $d(\mathbf{x}_i, \mathbf{x}_j) =$

$$\sqrt{\sum_{k=1}^p |x_{ik} - x_{jk}|^2}$$

- Supremum distance :
 $h = \infty$, and

$$d(\mathbf{x}_i, \mathbf{x}_j) = \max_{k=1}^p |x_{ik} - x_{jk}|$$

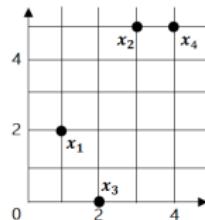
L_1	x_1	x_2	x_3	x_4
x_1	0			
x_2	5	0		
x_3	3	6	0	
x_4	6	1	7	0

(a) Manhattan

L_2	x_1	x_2	x_3	x_4
x_1	0			
x_2	3.61	0		
x_3	2.24	5.1	0	
x_4	4.24	1	5.39	0

(b) Euclidean

Point	Attr 1	Attr 2
x_1	1	2
x_2	3	5
x_3	2	0
x_4	4	5



L_∞	x_1	x_2	x_3	x_4
x_1	0			
x_2	3	0		
x_3	2	5	0	
x_4	3	1	5	0

(c) Supremum

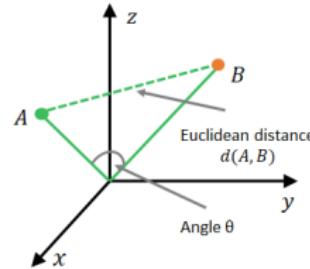
Metrics : Cosine Similarity

- Definition : $\cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^p x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^p x_{ik}^2} \sqrt{\sum_{k=1}^p x_{jk}^2}} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$
- Example : $\cos(\mathbf{x}_1, \mathbf{x}_2) = 0.94$

Euclidean vs. Cosine :

- Euclidean : measures the distance in absolute value, many applications
- Cosine : insensitive to absolute value, e.g., analyze users' interests based on movie ratings

Instance	Team	Coach	Hockey	Baseball	Soccer	penalty	Score	Win	Loss	Season
Instance 1	5	0	3	0	2	0	0	2	0	0
Instance 2	3	0	2	0	1	1	0	1	0	1



Metrics : Other Distances

- For ordinal data, mapping the data to numerical data :
 $X = \{x_{(1)}, x_{(2)}, \dots, x_{(n)}\}, x_{(i)} \mapsto \frac{i-1}{n-1} \in [0, 1]$
- For mixed type, use weighted distance with prescribed weights :

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{g=1}^G w_{ij}^{(g)} d_{ij}^{(g)}}{\sum_{g=1}^G w_{ij}^{(g)}}$$

Put the attributes of the same type into groups, for each data type g , use the corresponding distance $d_{ij}^{(g)}$

Outlines

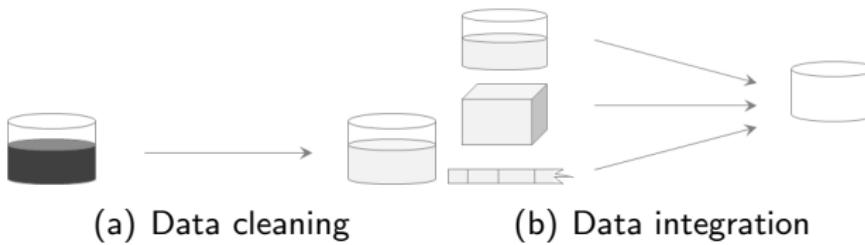
Basic Concepts

Data Preprocessing

Why Data Preprocessing?

- Missing values
- Noisy with outliers
- Inconsistent representations
- Redundancy
- Errors may come during data input, data gathering, and data transferring
- Errors occur in about 5% of the data

Four Types of Data Preprocessing



(a) Data cleaning

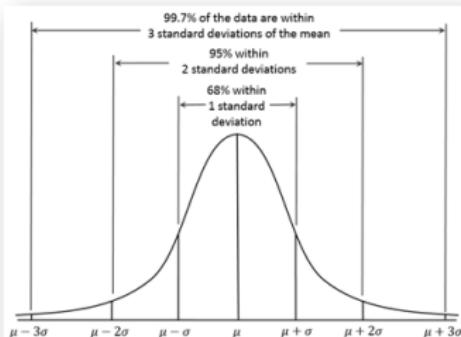
(b) Data integration

(c) Data conversion

(d) Data reduction

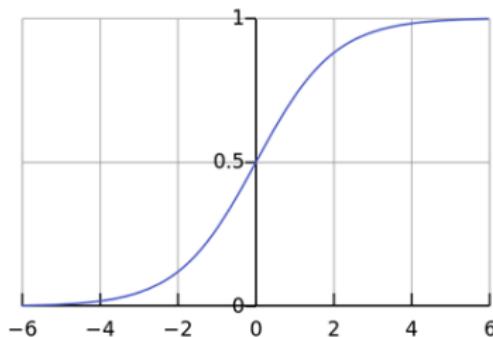
Data Scaling

- Why scaling :
 - For better performance : e.g., RBF in SVM and penalty in Lasso/ridge regression assume the zero mean and unit variance
 - Normalize different dimensions : many algorithms are sensitive to the variables with large variances, e.g., height (1.75m) and weight (70kg) in distance calculation
- Z-score scaling : $x_i^* = \frac{x_i - \hat{\mu}}{\hat{\sigma}}$, $\hat{\mu}$: sample mean, $\hat{\sigma}$: sample variance, applicable if max and min are unknown and the data distributes well



Data Scaling (Cont')

- 0-1 scaling : $x_i^* = \frac{x_i - \min_k x_k}{\max_k x_k - \min_k x_k} \in [0, 1]$, applicable for bounded data sets, need to recompute the max and min when new data arrive
- Decimal scaling : $x_i^* = \frac{x_i}{10^k}$, applicable for data varying across many magnitudes
- Logistic scaling : sigmoid transform $x_i^* = \frac{1}{1+e^{-x_i}}$, applicable for data concentrating nearby origin



Data Discretization

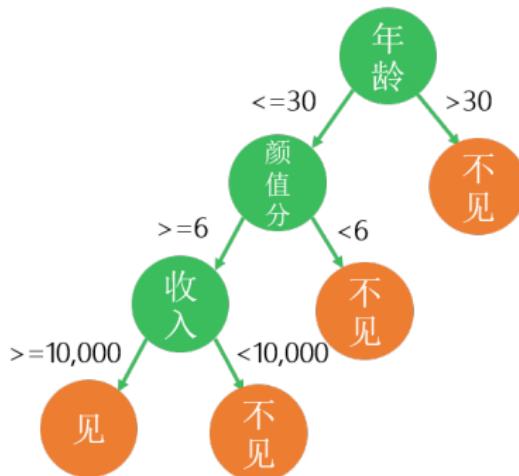
- Why discretization :
 - Improve the robustness : removing the outliers by putting them into certain intervals
 - For better interpretation
 - Reduce the storage and computational power
- Unsupervised discretization : equal-distance discretization, equal-frequency discretization, clustering-based discretization, 3σ -based discretization
- Supervised discretization : information gain based discretization, χ^2 -based discretization

Unsupervised Discretization

- Equal-distance discretization : split the range to n intervals (bins) with the same length, group the data into each bin, sensitive to outliers
- Equal-frequency discretization : group the data into n subset so that each subset has the same number of points, tend to separate samples with similar values and produce uniform distribution
- Clustering-based discretization : do hierarchical clustering and form a hierarchical structure (e.g., using K-Means), and put the samples in the same branch into the same interval (a natural example is family tree)
- 3σ -based discretization : put the samples into 8 intervals, need to take logarithm first

Supervised Discretization - Information Gain

- Top-down splitting, similar to create a decision tree
- Do a decision tree classification using information gain, find a proper splitting point for each continuous variable such that the information gain increases the most
- The final leaf nodes summarize the discrete intervals



Supervised Discretization - ChiMerge

- Bottom-up : similar to hierarchical clustering
- $\hat{\chi}^2$ statistics proposed by Karl Pearson, is used to test whether the observations dramatically deviate from theoretical distribution : $\hat{\chi}^2 = \sum_{i=1}^k \frac{(A_i - \mathbb{E}A_i)^2}{\mathbb{E}A_i} = \sum_{i=1}^k \frac{(A_i - np_i)^2}{np_i}$, where n_i is the number of samples in the i -th interval $A_i = [a_{i-1}, a_i]$ (frequency of observations), $\bigcup_{i=1}^k A_i$ covers the range of the variable, and $\mathbb{E}A_i = p_i$ is its expectation computed from the theoretical distribution ; it can be shown that $\hat{\chi}^2 \rightarrow \chi^2_{k-1}$
- ChiMerge : Given a threshold level t ,
 1. Treat each value of the continuous variable as an interval and sort them in increasing order ;
 2. For each pair of adjacent intervals, compute its $\hat{\chi}^2$ statistics, if $\hat{\chi}^2 < t$, merge them into a new interval ;
 3. Repeat the above steps until no adjacent intervals can be merged.
- Two shortcomings : t is hard to set appropriately ; too long loop for large sample set, computationally intensive

ChiMerge : Iris Data Example

- $\hat{\chi}^2 = \sum_{i=1}^m \sum_{j=1}^k \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$, where
 $m = 2$ (two adjacent intervals)
 k is the number of classes
 A_{ij} is the number of samples in i -th interval and in class k
 $R_i = \sum_{j=1}^k A_{ij}$ is the total number of samples in i -th interval
 $C_j = \sum_{i=1}^m A_{ij}$ is the total number of samples in class j
 $N = \sum_{i=1}^m \sum_{j=1}^k A_{ij}$ is the total number of samples
 $E_{ij} = R_i \cdot \frac{C_j}{N}$
- χ^2 of 4.3 and 4.4 : $C_1 = 4$,
 $C_2 = 0$, $C_3 = 0$, $N = 4$, $A_{11} = 1$,
 $A_{12} = A_{13} = 0$, $A_{21} = 3$,
 $A_{22} = A_{23} = 0$, $R_1 = 1$, $R_2 = 3$,
 $E_{11} = 1$, $E_{12} = E_{13} = 0$, $E_{21} = 3$,
 $E_{22} = E_{23} = 0$, $\hat{\chi}^2 = 0$.

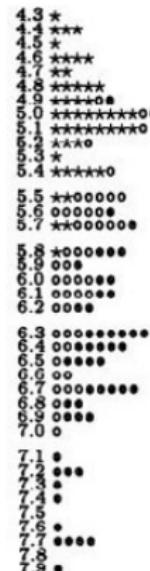


FIGURE: Sepal lengths of 3 types of iris

ChiMerge Results

Left : significance level is 0.5 and the threshold for χ^2 is 1.4 ;
 Right : significance level is 0.9 and the threshold for χ^2 is 4.6 ;
 The final results keep the intervals with χ^2 larger than the thresholds

Int	Class			χ^2
	frequency			
4.3	16	0	0	4.1
4.9	4	1	1	2.4
5.0	25	5	0	8.6
5.5	2	5	0	2.9
5.6	0	5	1	1.7
5.7	2	5	1	1.8
5.8	1	3	3	2.2
5.9	0	12	7	4.8
6.3	0	6	15	4.1
6.6	0	2	0	3.2
6.7	0	5	10	1.5
7.0	0	1	0	3.6
7.1	0	0	12	

Int	Class			χ^2
	frequency			
4.3	45	6	1	30.9
5.5	4	15	2	6.7
5.8	1	15	10	4.9
6.3	0	14	25	5.9
7.1	0	0	12	

Figure 2: ChiMerge discretizations for *sepal-length* at the .50 and .90 significance levels ($\chi^2 = 1.4$ and 4.6)

Data Redundancy

- When strong correlations exist among different attributes, then we say that some attributes can be derived from the others (Recall linear dependency for vectors)
- E.g., two attributes "Age" and "Birthday", then "Age" can be calculated from "Birthday"
- Determine the data redundancy by correlation analysis
- For continuous variables A and B , compute the correlation coefficient $\rho_{A,B} = \frac{\sum_{i=1}^k (a_i - \bar{A})(b_i - \bar{B})}{k\hat{\sigma}_A\hat{\sigma}_B} \in [-1, 1]$:
 - If $r > 0$, A and B are positively correlated ;
 - If $r < 0$, A and B are negatively correlated ;
 - If $r = 0$, A and B are uncorrelated.

Note that the correlation between A and B does not imply the causal inference.

- For discrete variables A and B , compute the χ^2 statistics : large $\hat{\chi}^2$ value implies small correlation

Missing Data

- Where missing data come from ?
 - Missing Completely At Random (MCAR) : the occurrence of missing data is a random event
 - Missing At Random (MAR) : depending on some control variables, e.g., the age > 20 is not acceptable in an investigation for teenager and thus is replaced by MAR
 - Missing Not At Random (MNAR) : missing data for bad performed employees after they are fired

Simple Methods

- Deleting samples : for small size of samples with missing values
- Deleting variables : for series missing values in variables

gradyear	gender	age	friends
2006	M	18.98	7
2006	F	18.801	0
2006	M	18.335	69
2006	F	18.875	0
2006	NA	18.995	10
2006	F		142
2006	F	18.93	72
2006	M	18.322	17
2006	F	19.055	52
2006	F	18.708	39
2006	F	18.543	8
2006	F	19.463	21
2006	F	18.097	87
2006	NA		0
2006	F	18.398	0
2006	NA		0
2006	NA		135
2006	F	18.987	26
2006	F	17.158	27
2006	F	18.497	123
2006	F	18.738	35

Filling Methods

- Filling with zero
- Filling with means for numerical type, and with modes for non-numerical type, applicable for MCAR ; drawback : concentrating in the mean and underestimating the variance ; solution : filling in different groups
- Filling with similar variables : auto-correlation is introduced
- Filling with past data
- Filling by K-Means : Compute the pairwise distances of the data using good variables (no missing values), then fill the missing values with the mean of the first K most similar good data, auto-correlation is introduced
- Filling with Expectation-Maximization (EM) : introduce hidden variables and use MLE to estimate the parameters (missing values)

Filling Methods (Cont')

- Random filling :
 - Bayesian Bootstrap : for discrete data with range $\{x_i\}_{i=1}^k$, randomly sample $k - 1$ numbers from $U(0, 1)$ as $\{a_{(i)}\}_{i=0}^k$ with $a_{(0)} = 0$ and $a_{(k)} = 1$; then randomly sample from $\{x_i\}_{i=1}^k$ with probability distribution $\{a_{(i)} - a_{(i-1)}\}_{i=1}^k$ accordingly to fill in the missing values
 - Approximate Bayesian Bootstrap: Sample with replacement from $\{x_i\}_{i=1}^k$ to form new data set $X^* = \{x_i^*\}_{i=1}^{k^*}$; then randomly sample n values from X^* to fill in the missing values, allowing for repeatedly filling missing values
- Model based methods : treat missing variable as y , other variables as x ; take the data without missing values as our training set to train a classification or regression model; take the data with missing values as our test set to predict the miss values

Filling by Interpolation

- For the data of numeric type, each attribute (column vector) can be viewed as the function values $z_i = f(x_i)$ at the points x_i , where x_i is a reference attribute (the reference attribute usually has no missing values, it can be chosen as the index)
- We can interpolate a function f using the existing values (x_i, z_i) , and then fill in the missing values z_k with $f(x_k)$
- Linear interpolation : treat $z = f(x)$ as linear function between the neighboring points x_{k-1} and x_{k+1} of x_k
- Lagrange interpolation : interpolate the $m + 1$ existing values $\{(x_{l_i}, z_{l_i})\}_{i=1}^{m+1}$ by a degree m polynomial $L_m(x)$

```
gen_data.interpolate()
```

	feature1	feature2	feature3
1	1.728534	-0.371519	1.451700
2	0.795975	-1.067026	-1.861944
3	-0.030449	-0.050409	1.299994
4	-0.856872	0.966208	0.987861

Missing value Missing value

Special Values and Dummy Variables

- In Python, “np.nan” means missing values (Not a Number, missing float value)
- “None” is a Python object, used to represent missing values of the object type
- Dummy variables : e.g., missing values in gender (“Male” or “Female”), then define a third value “unknown” for the missing values

```
import pandas as pd
import numpy as np

teenager_sns = pd.read_csv('teenager_sns.csv')

print teenager_sns['gender'].value_counts()

teenager_sns['gender'] = teenager_sns['gender'].replace(np.NaN, 'unknown')

print ""
print "哑变量方法处理后: \n"
print teenager_sns['gender'].value_counts()

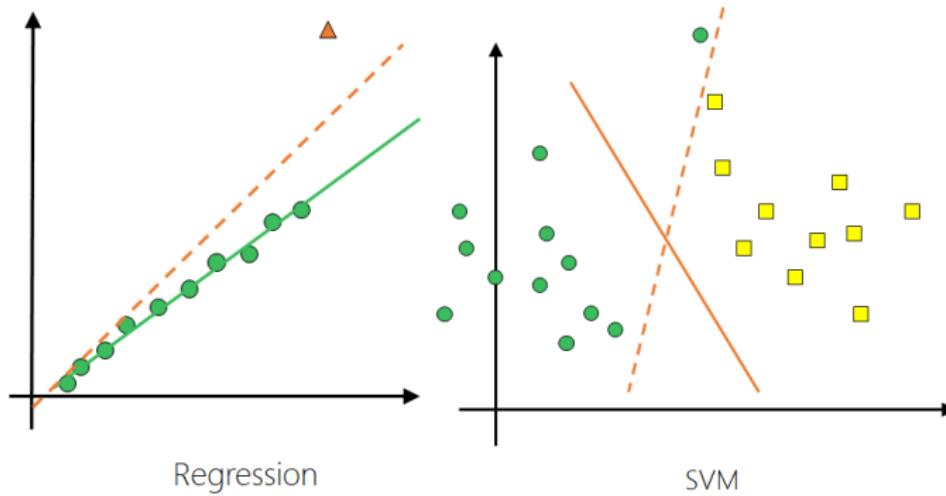
F      22054
M      5222
Name: gender, dtype: int64

哑变量方法处理后:

F      22054
M      5222
unknown    2724
Name: gender, dtype: int64
```

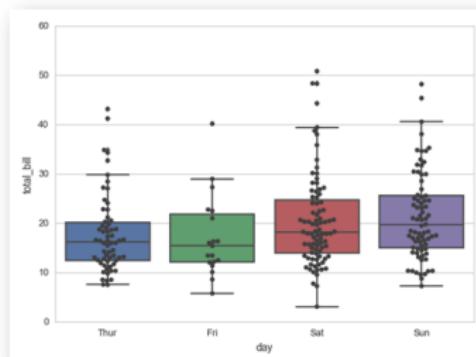
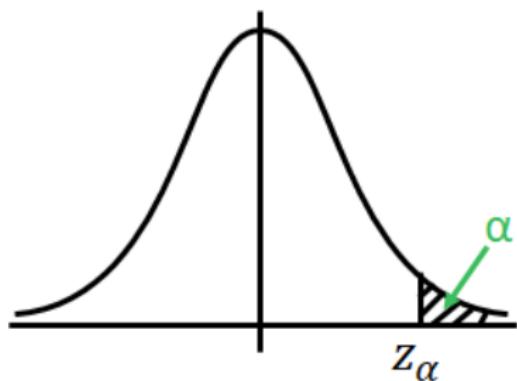
Outliers

- Outliers : the data points seem to come from different distribution, or noisy data
- Outlier detection : unsupervised, e.g., Credit cheating detection, medical analysis, and information security, etc.



Outliers Detection - Statistics Based Methods

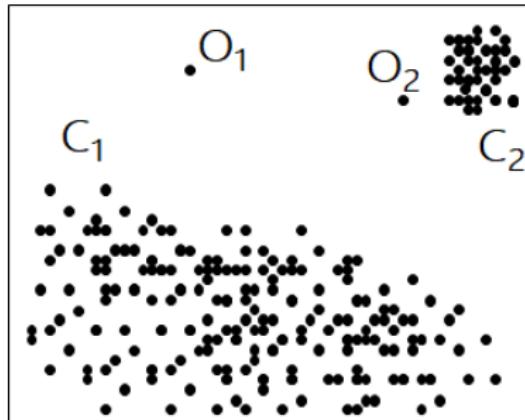
- The samples outside the upper and lower α -quantile for some small α (usually 1%)
- Observe from box plot



Outliers Detection - Local Outlier Factor

Local Outlier Factor (LOF) is a density based method :

1. We could compute the density at each position \mathbf{x} , e.g., $p(\mathbf{x})$ (how to define the density if we only have data samples);
2. We could compare the density of each point \mathbf{x} with the density of its neighbors, i.e., compare $p(\mathbf{x})$ with $p(\mathbf{x}_k)$ where \mathbf{x}_k is close to \mathbf{x} (in a neighborhood of \mathbf{x} , but how to define the neighborhood)



Computing Density by Distance

Some definitions :

- $d(A, B)$: distance between A and B;
- $d_k(A)$: k -distance of A, or the distance between A and the k -th nearest point from A
- $N_k(A)$: k -distance neighborhood of A, or the points within $d_k(A)$ from A;
- $rd_k(B, A)$: k -reach-distance from A to B, the repulsive distance from A to B as if A has a hard-core with radius $d_k(A)$,
 $rd_k(B, A) = \max\{d_k(A), d(A, B)\}$; note that $rd_k(A, B) \neq rd_k(B, A)$, which implies that k -reach-distance is not symmetric.

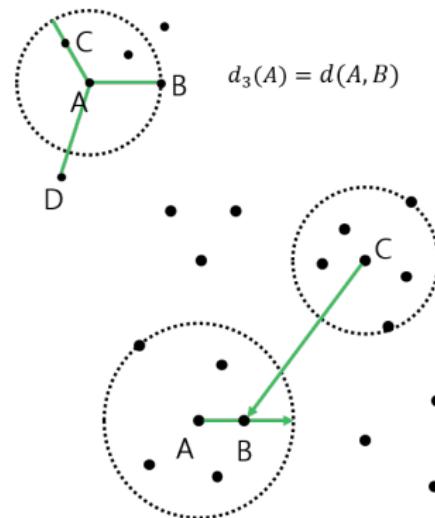


FIGURE: $rd_5(B, A) = d(A, B)$ and $rd_5(B, C) = d(B, C)$

Local Outlier Factor

Some definitions :

- $Ird_k(A)$: local reachability density is inversely proportional to the average distance,

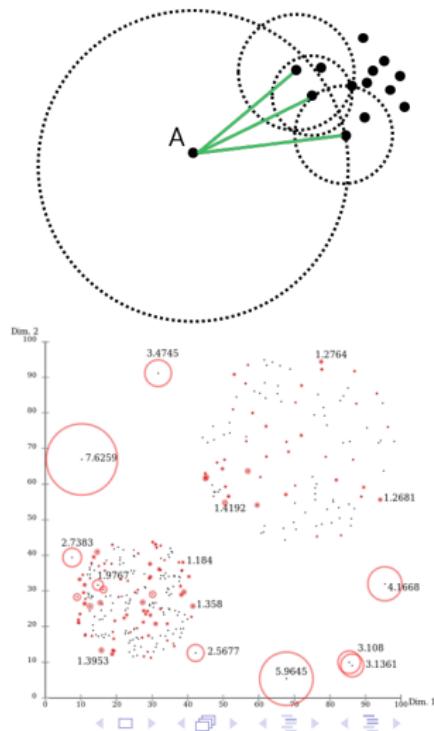
$$Ird_k(A) = \frac{1}{\left(\frac{\sum_{O \in N_k(A)} rd_k(A, O)}{|N_k(A)|} \right)};$$

intuitively, if for most $O \in N_k(A)$, more than k points are closer to O than A is, then the denominator is much larger than $d_k(A)$ and $Ird_k(A)$ is small ; e.g., $k = 3$ in the figure

- $LOF_k(A)$: local outlier factor,

$$LOF_k(A) = \frac{\sum_{O \in N_k(A)} \frac{Ird_k(O)}{Ird_k(A)}}{|N_k(A)|};$$

- $LOF_k(A) \ll 1$, the density of A is locally higher, dense point ;
 $LOF_k(A) \gg 1$, the density of A is locally lower, probably outlier



Further Study

- Other methods for outlier detection :
 - K-Means
 - K Nearest Neighbors
 - Isolation Forest
 - One-class support vector machine
 - Robust covariance
- Outlier processing :
 - Delete outliers (treat them as missing values)
 - Robust regression
 - Theil-Sen regression

Introduction to Big Data Analysis

Classification : Part 1

Zhen Zhang

Southern University of Science and Technology

Outlines

Introduction

k-Nearest Neighbor

Decision Trees

Naive Bayes

Model Assessment

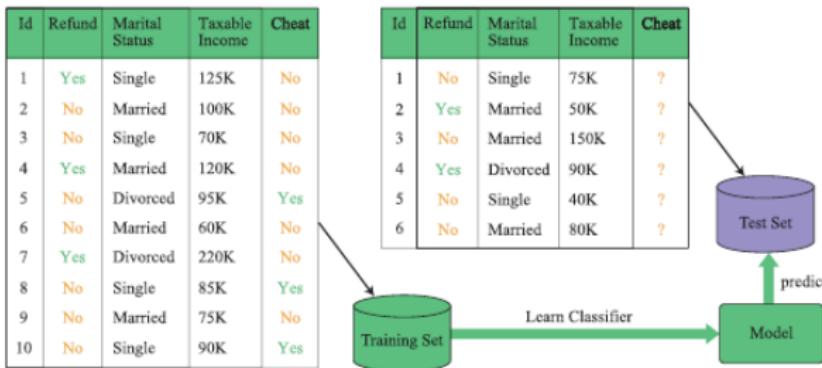
References

Why We Need Classification

- Knowing the classes of the data, we could easily manage the data and react to the possible outcomes
- Predict whether users would default in the future based on their basic information and historical transaction records
- Predict whether a tumor is benign or malignant based on their physical and geometrical features
- Predict the users' interests in the new products based on their historical purchasing records and behavioral preferences
- Separate spams and advertisements from emails

What is Classification

- Supervised learning : predict label y from features \mathbf{x}
- Training stage : Given a data set $D = \{(\mathbf{x}, y)\}$, including both features and labels, split $D = D_{train} \cup D_{test}$, find a classifier (function $y = f(\mathbf{x})$) that best relates y_{train} with \mathbf{x}_{train} , then evaluate how close $f(\mathbf{x}_{test})$ is to y_{test}
- Predicting stage : apply the predictor to the unlabeled data \mathbf{x}_{pred} (only features) to find the proper labels $y_{pred} = f(\mathbf{x}_{pred})$



Classification Methods

- Different assumptions on f lead to different models
- Basic classification models
 - k-nearest neighbor (kNN)
 - Decision trees
 - Naive Bayes
 - Support vector machines (SVM)
 - Logistic regression
 - Linear discriminant analysis (LDA)
 - Artificial neural network (ANN)
 - ...
- Ensemble learning : Random forest and Adaboost

Outlines

Introduction

k-Nearest Neighbor

Decision Trees

Naive Bayes

Model Assessment

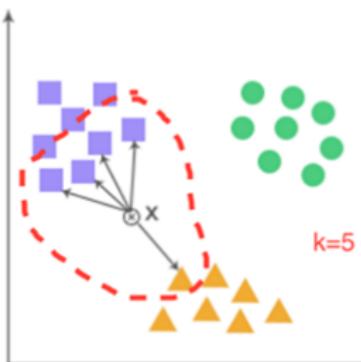
References

Introduction

- k-nearest neighbor (kNN) is the simplest supervised learning method, especially useful when prior knowledge on the data is very limited
- Do training and test simultaneously
- When classifying a test sample x , scan the training set and find the closest k samples $D_k = \{x_1, \dots, x_k\}$ to the test sample ; make vote based on the labels of the samples in D_k ; the majority vote is the label of the test sample
- Low bias, high variance
- Advantages : not sensitive to outliers, easy to implement and parallelize, good for large training set
- Drawbacks : need to tune k , take large storage, computationally intensive

Algorithm

- Input : training set $D_{train} = \{(x_1, y_1), \dots, (x_N, y_N)\}$, a test sample x without label y , k and distance metric $d(x, y)$
 - Output : predicted label y_{pred} for x
1. Compute $d(x, x_j)$ for each $(x_j, y_j) \in D_{train}$
 2. Sort the distances in an ascending order, choose the first k samples $(x_{(1)}, y_{(1)}), \dots, (x_{(k)}, y_{(k)})$
 3. Make majority vote $y_{pred} = \text{Mode}(y_{(1)}, \dots, y_{(k)})$



Distance Metrics

- Minkowski distance : $d_h(\mathbf{x}_1, \mathbf{x}_2) = \sqrt[h]{\sum_{i=1}^d (x_{1i} - x_{2i})^h}$; $h = 2$, Euclidean distance; $h = 1$, Manhattan distance
- Mahalanobis distance :
 $d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \hat{\Sigma}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)}$, where $\hat{\Sigma}$ is the covariance matrix of sample set; introduce correlations, could be applied to the non-scaling data
- Hamming distance : $Hamming(\mathbf{x}_1, \mathbf{x}_2) = d - \sum_{i=1}^d I(x_{1i} \neq x_{2i})$;
used to compare two strings, e.g.,
 $Hamming('toned', 'roses') = 3$,
 $Hamming('101110', '101101') = 2$

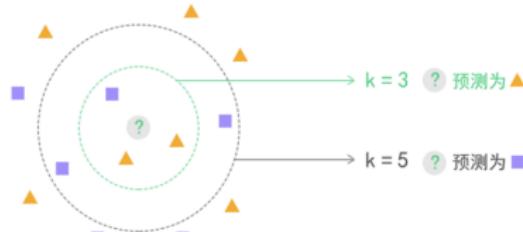
Distance Metrics - Similarity and Divergence

- Cosine similarity : $\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1^T \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|} = \frac{\sum_{i=1}^d x_{1i} x_{2i}}{\sqrt{\sum_{i=1}^d x_{1i}^2} \sqrt{\sum_{i=1}^d x_{2i}^2}}$; its range is $[-1, 1]$; the greater the cosine similarity, the more similar (closer) the two samples; insensitive to absolute value, popular in measuring user rankings; it is related to Pearson correlation coefficient
- Jaccard similarity for sets A and B : $Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$, used in comparing texts
- Kullback-Leibler (KL) divergence : $d_{KL}(P \| Q) = E_P \left[\log \frac{P(x)}{Q(x)} \right]$ measures the distance between two probability distributions P and Q ; in discrete case, $d_{KL}(p \| q) = \sum_{i=1}^m p_i \log \frac{p_i}{q_i}$

Tuning k

- Different values of $k = 3$ and $k = 5$ leads to different classification results
- M -fold Cross-validation (CV) to tune k : partition the dataset into M parts ($M = 5$ or 10), let κ : $\{1, \dots, N\} \rightarrow \{1, \dots, M\}$ be randomized partition index map, The CV estimate of prediction error is

$$CV(\hat{f}, k) = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{f}^{-\kappa(i)}(x_i, k))$$



1	2	3	4	5
Train	Train	Validation	Train	Train

Bayes Classifier (Oracle Classifier)

- Assume $Y \in \mathcal{Y} = \{1, 2, \dots, C\}$, the classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ is a piecewise constant function
- For 0-1 loss $L(y, f)$, the learning problem is to minimize

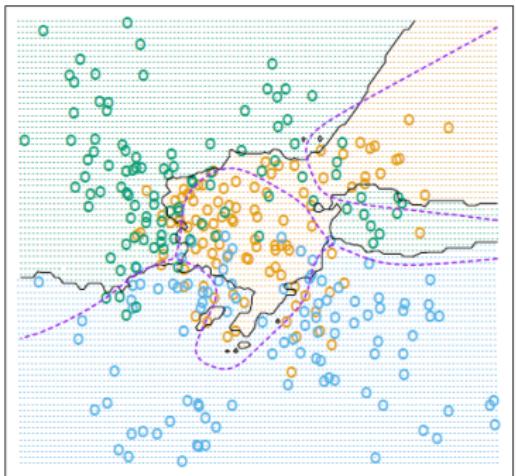
$$\begin{aligned}\mathcal{E}(f) &= \text{E}_{P(X, Y)} L(Y, f(X)) = 1 - P(Y = f(X)) \\ &= 1 - \int_{\mathcal{X}} P(Y = f(X)|X = x)p_X(x)dx\end{aligned}$$

- Bayes rule : $f^*(x) = \arg \max_c P(Y = c|X = x)$, “the most probable label under the conditional probability on x ”
- Bayes error rate : $\inf_f \mathcal{E}(f) = \mathcal{E}(f^*) = 1 - P(Y = f^*(X))$
- Bayes decision boundary : the boundary separating the K partition domains in \mathcal{X} on each of which $f^*(x) \in \mathcal{Y}$ is constant. For binary classification, it is the level set on which $P(Y = 1|X = x) = P(Y = 0|X = x) = 0.5$.

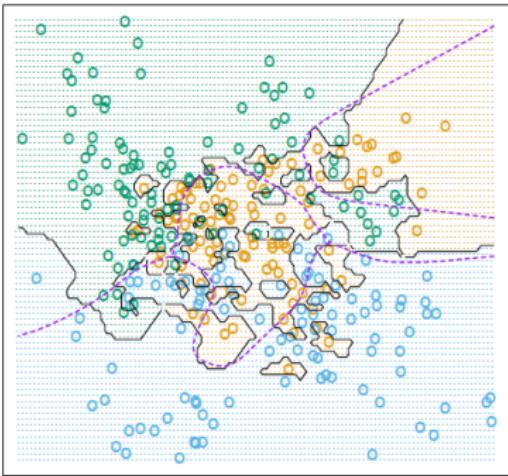
Decision Boundary

- The decision boundary of 15NN is smoother than that of 1NN

15-Nearest Neighbors



1-Nearest Neighbor



Analysis

- Time complexity : $O(mndK)$ where n is the number of training samples, m is the number of test samples, d is the dimension, and K is the number of nearest neighbors
- KD tree for indexing : K -dimensional binary search tree
- 1NN error rate is twice the Bayes error rate :
 - Bayes error = $1 - p_{c^*}(x)$ where $c^* = \arg \max_c p_c(x)$
 - Assume the samples are i.i.d., for any test sample x and small δ , there is always a training sample $z \in B(x, \delta)$ (the label of x is the same as that of z), then 1NN error is

$$\begin{aligned}\epsilon &= \sum_{c=1}^C p_c(x)(1 - p_c(z)) \xrightarrow{\delta \rightarrow 0} 1 - \sum_{c=1}^C p_c^2(x) \\ &\leq 1 - p_{c^*}^2(x) \\ &\leq 2(1 - p_{c^*}(x))\end{aligned}$$

(Remark : In fact, $\epsilon \leq 2(1 - p_{c^*}(x)) - \frac{C}{C-1}(1 - p_{c^*}(x))^2$)

Case Study

- Use kNN to diagnose breast cancer ([cookdata](#))
- Data scaling : 0-1 scaling or z-score scaling
- from sklearn.neighbors
import KNeighborsClassifier
- KNeighborsClassifier(n_neighbors = 10, metric = 'minkowski', p=2)
 - radius (半径)
 - texture (质地)
 - perimeter (周长)
 - area (面积)
 - smoothness (光滑度)
 - compactness (致密性= $perimeter^2 / area - 1.0$)
 - concavity (凹度)
 - concave points (凹点)
 - symmetry (对称性)
 - fractal dimension (分形维数)

Outlines

Introduction

k-Nearest Neighbor

Decision Trees

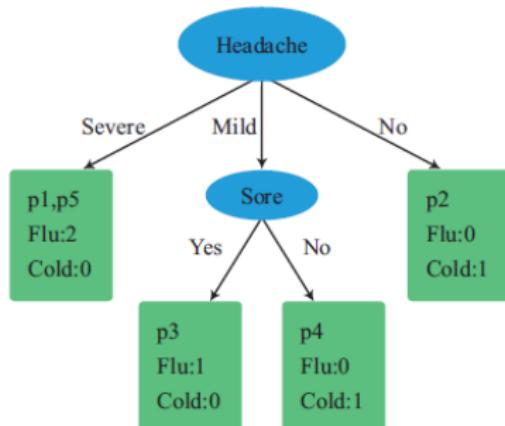
Naive Bayes

Model Assessment

References

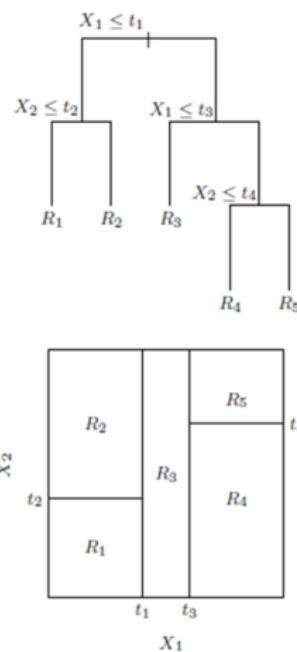
Decision Tree as Medical Diagnosis

- Diagnose whether it is flu or cold
- Rules :
 - If headache = severe, then flu
 - If headache = mild and sore = yes, then flu
 - If headache = mild and sore = no, then cold
 - If headache = no, cold



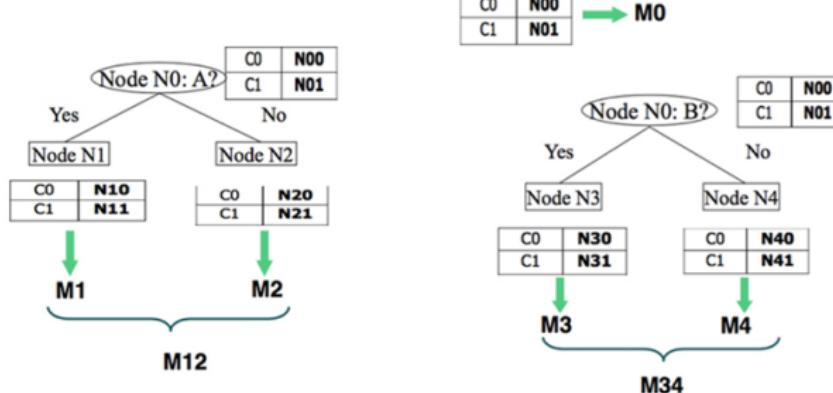
Decision Tree Algorithm

- Tree structure : internal nodes indicate features, while leaf nodes represent classes
- Start from root, choose a suitable feature x_i and its split point c_i at each internal node, split the node to two child nodes depending on whether $x_i \leq c_i$, until the child nodes are pure
- Equivalent to rectangular partition of the region



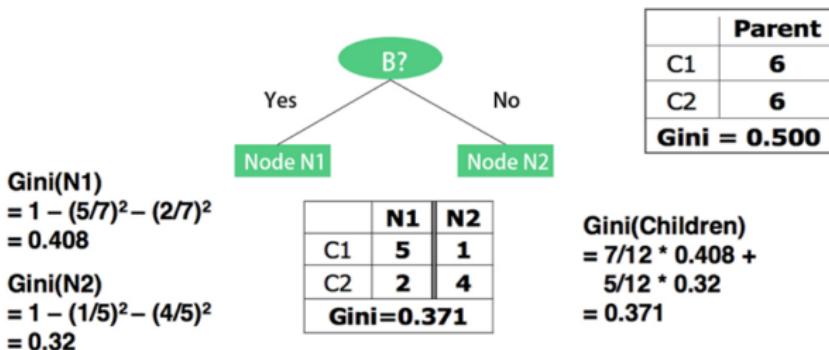
How to choose features and split points

- Impurity : choose the feature and split point so that after each slit the impurity should decrease the most
- $\text{Impurity}(M0) - \text{Impurity}(M12) > \text{Impurity}(M0) - \text{Impurity}(M34)$, choose A as split node ; otherwise choose B



Impurity Measures - GINI Index

- Gini index of node t : $Gini(t) = 1 - \sum_{c=1}^C (p(c|t))^2$ where $p(c|t)$ is the proportion of class- c data in node t
- Maximum at $1 - \frac{1}{C}$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some c
- Gini index of a split : $Gini_{split} = \sum_{k=1}^K \frac{n_k}{n} Gini(k)$ where n_k is the number of samples in the child node k , $n = \sum_{k=1}^K n_k$
- Choose the split so that $Gini(t) - Gini_{split}$ is maximized

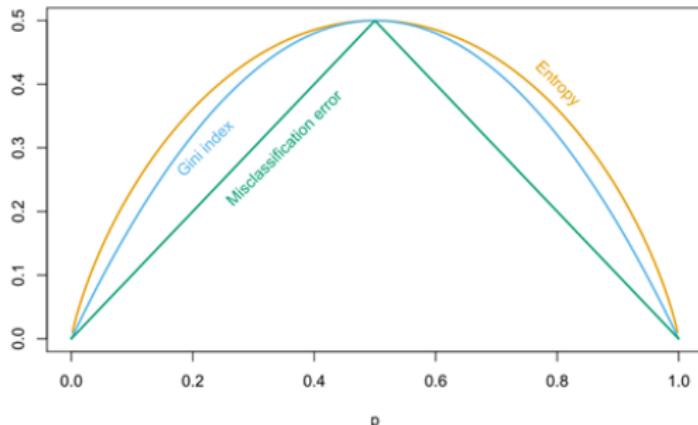


Impurity Measures - Information Gain

- Entropy at $t : H(t) = - \sum_{c=1}^C p(c|t) \log_2 p(c|t)$
- Maximum at $\log_2 C$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some c
- Information gain : $InfoGain_{split} = H(t) - \sum_{k=1}^K \frac{n_k}{n} H(k)$ where n_k is the number of samples in the child node k , $n = \sum_{k=1}^K n_k$
- Choose the split so that $InfoGain_{split}$ is maximized (ID3 algorithm)
- Drawback : easy to generate too many child nodes and overfit
- Introduce information gain ratio :
 $SplitINFO = - \sum_{k=1}^K \frac{n_k}{n} \log_2 \frac{n_k}{n}$, $InfoGainRatio = \frac{InfoGain_{split}}{SplitINFO}$
(C4.5 algorithm)

Impurity Measures - Misclassification Error

- Misclassification error at t : $Error(t) = 1 - \max_c p(c|t)$; use majority vote
- Maximum at $1 - \frac{1}{C}$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some c
- For two-class classification, $Gini(p) = 2p(1 - p)$,
 $H(p) = -\frac{1}{2}p \log_2 p - (1 - p) \log_2(1 - p)$ (up to a factor $\frac{1}{2}$),
 $Error(p) = 1 - \max(p, 1 - p)$



Comparing Three Impurity Measures

- Information gain and Gini index are more sensitive to changes in the node probabilities than the misclassification error
- Consider a two-class problem with 400 observations in each class, (400, 400); two possible splits, A : (300, 100) + (100, 300), and B : (200, 400) + (200, 0); B should be preferred
 - $Gini(A) = \frac{1}{2}Gini(A1) + \frac{1}{2}Gini(A2) = 2 \times \frac{1}{2}(2 \times \frac{3}{4} \times \frac{1}{4}) = \frac{3}{8}$,
 $Gini(B) = \frac{3}{4}Gini(A1) + \frac{1}{4}Gini(A2) = \frac{3}{4}(2 \times \frac{1}{3} \times \frac{2}{3}) = \frac{1}{3}$
 - $H(A) = 2 \times \frac{1}{2}(-\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4}) = 0.81$,
 $H(B) = \frac{3}{4}(-\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3}) = 0.69$
 - $Error(A) = 2 \times \frac{1}{2}(1 - \max(\frac{3}{4}, \frac{1}{4})) = \frac{1}{4}$,
 $Error(B) = \frac{3}{4}(1 - \max(\frac{1}{3}, \frac{2}{3})) = \frac{1}{4}$
- Gini index and information gain should be used when growing the tree
- In pruning, all three can be used (typically misclassification error)

Algorithms

- Iterative Dichotomiser 3 (ID3) : by Ross Quinlan (1986), based on Occam's Razor rule (be simple) ; information gain, choose feature values by enumeration
- C4.5 and C5.0 : by R. Quinlan (1993), use information gain ratio instead, choose split thresholds for continuous features
- Classification and Regression Tree (CART) : by Leo Breiman etc. (1984) ; for classification, use Gini index ; for regression, use mean square error ; binary split

算法	属性类型	不纯度度量	分割的子节点数量	目标属性类型
ID3	离散型	信息增益	$k \geq 2$	离散型
C4.5	离散型、连续型	信息增益率	$k \geq 2$	离散型
C5.0	离散型、连续型	信息增益率	$k \geq 2$	离散型
CART	离散型、连续型	GINI指数	$k = 2$	离散型、连续型

ID3 Algorithm

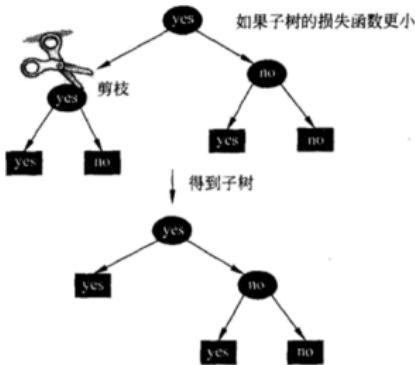
- Input : training set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$,
 $Y = \{y_1, \dots, y_n\}$, set of features $F = \{\text{column variables of } X = (\mathbf{x}_1 \dots \mathbf{x}_n)^T\}$
 - Output : decision tree T
1. Create a root node
 2. Check Y : if all are positive, then return a single node tree T with label “+” ; if all are negative, then return a single node tree T with label “-”
 3. Check F : if empty, then return a single node tree T with label as majority vote of Y
 4. For each feature in F , compute information gain, choose the feature $A \in F$ which maximizes information gain as root
 5. For $A = i$, let $D(i) = \{(\mathbf{x}_j, y_j) \in D | x_{jA} = i\}$:
 - 5.1 If $D(i) = \emptyset$, then create a leaf node and make majority vote of D as the label
 - 5.2 Else, let $D = D(i)$, go back to step 1 iteratively

Tree Pruning

- Too complex tree structure easily leads to overfitting
 - Prepruning : set threshold δ for impurity decrease in splitting a node ; if $\Delta Impurity_{split} > \delta$, do splitting, otherwise stop
 - Postpruning : based on cost function

$$Cost_{\alpha}(T) = \underbrace{\sum_{t=1}^{|T|} n_t Impurity(t)}_{\text{data fidelity}} + \underbrace{\alpha}_{\text{model complexity}} |T|$$

- Input : a complete tree T , α
 - Output : postpruning tree T_α
 1. Compute $\text{Impurity}(t)$ for $\forall t$
 2. Iteratively merge child nodes
bottom-up : T_A and T_B are the trees before and after merging, do merging if $\text{Cost}_\alpha(T_A) \geq \text{Cost}_\alpha(T_B)$



Pros and Cons

- Advantages

- Easy to interpret and visualize : widely used in finance, medical health, biology, etc.
- Easy to deal with missing values (treat as new data type)
- Could be extended to regression : decision tree is a rectangular partition of the domain, the predictor can be written as

$$f(\mathbf{x}) = \sum_{m=1}^M c_m I(\mathbf{x} \in R_m); \text{ for regression problems}$$

$$c_m = \bar{y}_m = \frac{1}{n_m} \sum_{i=1}^n y_i I(\mathbf{x}_i \in R_m) \text{ where } n_m = \sum_{i=1}^n I(\mathbf{x}_i \in R_m)$$

- Drawbacks :

- Easy to be trapped at local minimum because of greedy algorithm
- Simple decision boundary : parallel lines to the axes

Outlines

Introduction

k-Nearest Neighbor

Decision Trees

Naive Bayes

Model Assessment

References

Introduction

- Based on Bayes Theorem and conditional independency assumption on features
- Widely used in text analysis, spam filtering, recommender systems, and medical diagnosis
- Bayes Theorem : let X and Y be a pair of random variables having joint probability $P(X = x, Y = y)$; by definition, the condition probability of Y given X is $P(Y|X) = \frac{P(X,Y)}{P(X)}$; then by symmetry, $P(X|Y) = \frac{P(X,Y)}{P(Y)}$; upon eliminating $P(X, Y)$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- $P(Y)$ is prior prob. distribution, $P(X|Y)$ is likelihood function, $P(X)$ is evidence, $P(Y|X)$ is posterior prob. distribution

Naive Bayes

- The core problem of machine learning is to estimate $P(Y|X)$ (or its moments $E[Y|X] = \arg \min_f E[\|Y - f(X)\|^2]$)
- Let $X = \{X_1, \dots, X_d\}$, for fixed sample $X = x$, $P(X = x)$ is independent of Y , by Bayes Theorem

$$P(Y|X = x) \propto P(X = x|Y)P(Y)$$

- Assume conditional independency of X_1, \dots, X_d given $Y = c$:

$$P(X = x|Y = c) = \prod_{i=1}^d P(X_i = x_i|Y = c)$$

- Naive Bayes model :

$$\hat{y} = \arg \max_c P(Y = c) \prod_{i=1}^d P(X_i = x_i|Y = c)$$

Maximum Likelihood Estimate (MLE)

- Estimate $P(Y = c)$ and $P(X_i = x_i | Y = c)$ from the dataset $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$
- MLE for $P(Y = c) : P(Y = c) = \frac{\sum_{i=1}^n I(y_i=c)}{n}$
- When X_i is discrete variable with range $\{v_1, \dots, v_K\}$, MLE for $P(X_i = v_k | Y = c) = \frac{\sum_{i=1}^n I(x_i=v_k, y_i=c)}{\sum_{i=1}^n I(y_i=c)}$
- When X_i is continuous variable
 1. Do discretization, and go back to the above formula
 2. Assume X_i follows some distribution (e.g., $N(\mu, \sigma^2)$) :

$$P(X_i = x | Y = c) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Then use MLE to estimate μ and σ^2

Pros and Cons

- Where it is good
 - Spam filter : compute the posterior prob. distribution of frequently used words (convert to vector by word2vec)
 - Stable : for outliers and miss values
 - Robust : for uncorrelated features ; $P(X_i|Y)$ is independent of Y and thus has no effect on posterior probability
 - May outperform far more sophisticated alternatives even if conditional independency assumption is not satisfied
- Disadvantage
 - However, when conditional independency assumption is violated, performance of Naive Bayes can be poorer
 - Depends heavily on how well the parameter estimates are

Outlines

Introduction

k-Nearest Neighbor

Decision Trees

Naive Bayes

Model Assessment

References

Confusion Matrix

- For two-class classification :
 - True Positive (TP) : both true label and predicted label are positive
 - True Negative (TN) : both true label and predicted label are negative
 - False Positive (FP) : true label is negative, but predicted label is positive
 - False Negative (FN) : true label is positive, but predicted label is negative

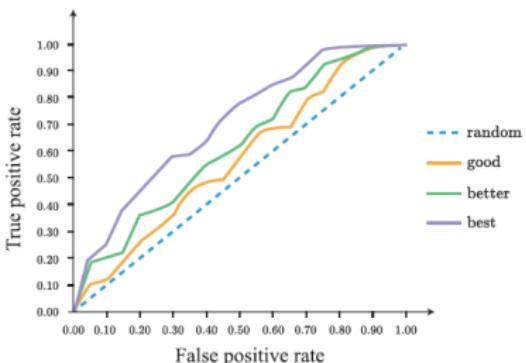
真实标签	预测结果	
	1 (正例)	0 (反例)
1 (正例)	TP (真正例)	FN (假反例)
0 (反例)	FP (假正例)	TN (真反例)

- $Accuracy = \frac{TP+TN}{TN+FN+FP+TP}$; not a good index when samples are imbalanced
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$; important in medical diagnosis (sensitivity)
- F score :

$$F_{\beta} = \frac{(1+\beta^2)Precision \times Recall}{\beta^2 \times Precision + Recall}$$
 ;
 $\beta = 1$, F_1 score
- $Specificity = \frac{TN}{TN+FP}$; recall for negative samples

Receiver Operating Characteristic (ROC) and AUC

- Aim to solve class distribution imbalance problem
- Set different threshold t for continuous predicted values (probability), e.g., if $P(Y = 1|X = x_i) > t$, then $\hat{y}_i = 1$
- Compute TPR ($= \frac{TP}{TP+FN}$, or recall) vs. FPR($= \frac{FP}{FP+TN}$) for different t and plot ROC curve
- The higher the ROC, the better the performance
- AUC : area under ROC, the larger the better, the more robust of the method for the change of t ; very good if > 0.75



Cohen's Kappa Coefficient

- $\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}$ measures the agreement between two raters
- p_o is the accuracy (or the relative observed agreement)
- p_e is the hypothetical probability of chance agreement,

$$p_e = \sum_{c=1}^C \frac{n_c^{pred}}{N} \frac{n_c^{true}}{N}$$
, where n_c^{pred} is the number of samples predicted in class c , n_c^{true} is the true number of samples in class c , N is the total number of samples
- Eg : $p_o = \frac{20+15}{50} = 0.7$, $p_e = \frac{25}{50} \times \frac{20}{50} + \frac{25}{50} \times \frac{30}{50} = 0.5$, $\kappa = 0.4$

		Predicted Label		
		1	0	Total
True Label	1	20 TP	10 FN	30 C
	0	5 FP	15 TN	20 D
	Total	25 A	25 B	50 N

The Values of Kappa Coefficient

- $\kappa \in [-1, 1]$
- $\kappa = 1$: perfect agreement between two raters
- $\kappa = -1$: completely disagreement
- $\kappa = 0$: no agreement among the raters other than what would be expected by chance
- $\kappa < 0$: worse than random
- $\kappa > 0$: the result is meaningful, agree more as κ gets larger
- $\kappa \geq 0.75$: good performance
- $\kappa < 0.4$: bad performance

Multiple Class Problem

- ROC and AUC are not well-defined
- Confusion matrix : $C \times C$, each entry means the number of samples in the intersection of the predicted class i and the true class j
- Positive sample is the sample belonging to the class i , negative sample is the sample not belonging to the class i , so every sample could be positive or negative
- Convert to multiple 0-1 classification problems
- Precision and recall are the averages of that in the each 0-1 classification problem
- $F1$ score is still defined as the harmonic average of precision and recall

Outlines

Introduction

k-Nearest Neighbor

Decision Trees

Naive Bayes

Model Assessment

References

References

- 数据分析导论, 博雅大数据学院
- 周志华, 机器学习, 2016
- T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning : Data mining, Inference, and Prediction*, 2nd Edition, 2009

Introduction to Big Data Analysis Regression

Zhen Zhang

Southern University of Science and Technology

Outlines

Introduction

Linear Regression

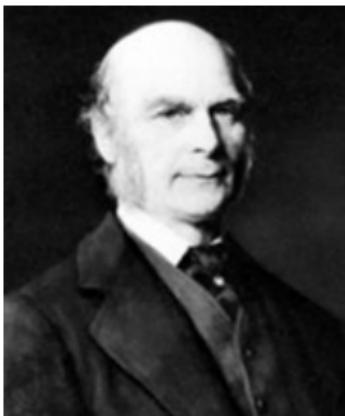
Regularizations

Model Assessment

References

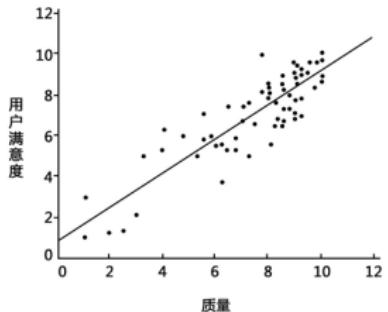
Regression

- Proposed by Francis Galton (left) and Karl Pearson (right), in the publication “Regression towards mediocrity in hereditary”
- The characteristics (e.g., height) in the offspring regress towards a mediocre point (mean) of that of their parents
- Generalization : predict the dependent variables y from the independent variables \mathbf{x} : $y = f(\mathbf{x})$ or $y = E[y|\mathbf{x}]$



Applications

- Predict medical expenses from the individual profiles of the patients
- Predict the scores on Douban from the quality of the movies
- Predict the tips from the total expenses



Outlines

Introduction

Linear Regression

Regularizations

Model Assessment

References

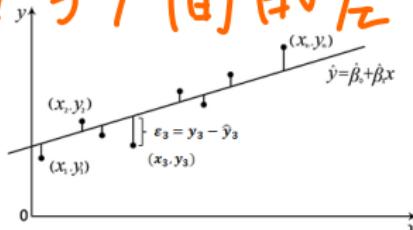
Univariate Linear Model

- Linear model : $y = w_0 + w_1x + \epsilon$, where w_0 and w_1 are regression coefficients, ϵ is the error or noise
- Assume $\epsilon \sim \mathcal{N}(0, \sigma^2)$, where σ^2 is a fixed but unknown variance; then $y|x \sim \mathcal{N}(w_0 + w_1x, \sigma^2)$
- Assume the samples $\{(x_i, y_i)\}_{i=1}^n$ are generated from this conditional distribution, i.e., $y_i|x_i \sim \mathcal{N}(w_0 + w_1x_i, \sigma^2)$
- Intuitively, find the best straight line (w_0 and w_1) such that the sample points fit it well, i.e., the residuals are minimized,

最小二乘
 $(\hat{w}_0, \hat{w}_1) = \arg \min_{w_0, w_1} \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$ 硬求导
 X与X之间、Y与Y之间的差：误差

X与Y之间：

残差

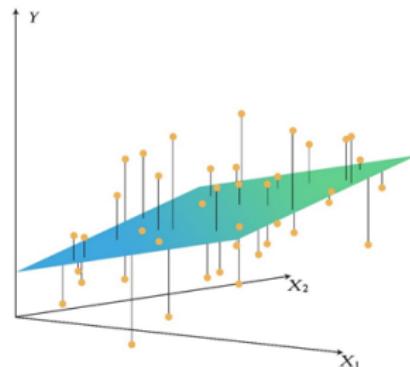


Multivariate Linear Model

- Linear model : $y = f(\mathbf{x}) + \epsilon = w_0 + w_1x_1 + \cdots + w_px_p + \epsilon$, where w_0, w_1, \dots, w_p are regression coefficients, $\mathbf{x} = (x_1, \dots, x_p)^T$ is the input vector whose components are independent variables or attribute values, $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the noise
- For the size n samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, let $\mathbf{y} = (y_1, \dots, y_n)^T$ be the response or dependent variables, $\mathbf{w} = (w_0, w_1, \dots, w_p)^T$, $\mathbf{X} = [\mathbf{1}_n, (\mathbf{x}_1, \dots, \mathbf{x}_n)^T] \in \mathbb{R}^{n \times (p+1)}$, and $\boldsymbol{\epsilon} = (\epsilon_1, \dots, \epsilon_n)^T \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$.

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}$$



Least Square (LS)

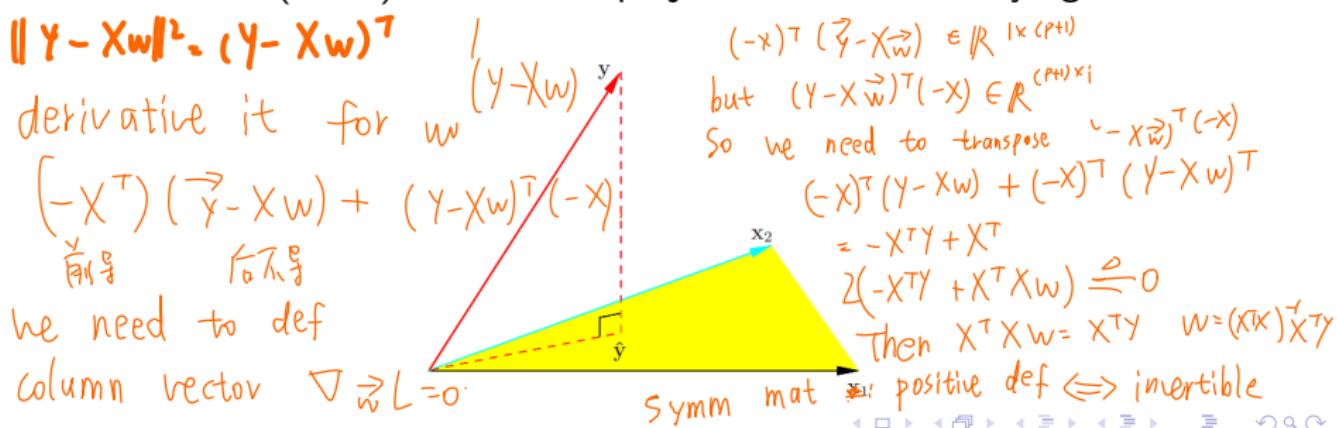
- Minimize the total residual sum-of-squares :

$$RSS(\mathbf{w}) = \sum_{i=1}^n (y_i - w_0 - w_1 x_1 - \cdots - w_p x_p)^2 = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

- When $\mathbf{X}^T \mathbf{X}$ is invertible, the minimizer $\hat{\mathbf{w}}$ satisfies

$$\nabla_{\mathbf{w}} RSS(\hat{\mathbf{w}}) = 0 \Rightarrow \hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- The prediction $\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{P}\mathbf{y}$ is a projection of \mathbf{y} onto the linear space spanned by the column vectors of \mathbf{X} ;
 $\mathbf{P} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is the projection matrix satisfying $\mathbf{P}^2 = \mathbf{P}$



Maximal Likelihood Estimate (MLE)

- A probabilistic viewpoint :

$$y|\mathbf{x} \sim \mathcal{N}(w_0 + w_1x_1 + \cdots + w_px_p, \sigma^2)$$

- Likelihood function :

$$L(\mathbf{w}; \mathbf{X}, \mathbf{y}) = P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^n P(y_i|\mathbf{x}_i, \mathbf{w}) \text{ with}$$

$$P(y_i|\mathbf{x}_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - w_0 - w_1x_{i1} - \cdots - w_px_{ip})^2}{2\sigma^2}}$$

- Maximal likelihood estimate : given the samples from some unknown parametric distribution, find the parameters such that the samples the most probably seem to be drawn from that distribution, i.e., $\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} L(\mathbf{w}; \mathbf{X}, \mathbf{y})$

- Equivalent to maximize the log-likelihood function

$$l(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \log L(\mathbf{w}; \mathbf{X}, \mathbf{y}) =$$

$$-n \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - w_0 - w_1x_{i1} - \cdots - w_px_{ip})^2$$

- The same minimizer as LS : $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

Projection by Orthogonalization

- Another useful formulation : let $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$, $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$, then OLS can be formulated by using the centralized data $\{\tilde{\mathbf{x}}_i, \tilde{y}_i\}_{i=1}^n = \{\mathbf{x}_i - \bar{\mathbf{x}}, y_i - \bar{y}\}_{i=1}^n$, $RSS(\tilde{\mathbf{w}}) = \sum_{i=1}^n (\tilde{y}_i - w_1 \tilde{x}_{i1} - \cdots - w_p \tilde{x}_{ip})^2 = \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}} \tilde{\mathbf{w}}\|_2^2$, with $\hat{w}_0 = \bar{y} - \tilde{\mathbf{w}}^T \bar{\mathbf{x}}$
- Ordinary least square (OLS) prediction $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$ is the projection of \mathbf{y} on the linear space spanned by the columns of \mathbf{X} , i.e., $\mathcal{X} = \text{Span}\{\mathbf{x}_{\cdot,0}, \mathbf{x}_{\cdot,1}, \dots, \mathbf{x}_{\cdot,p}\}$, recall that $\mathbf{x}_{\cdot,0} = \mathbf{1}_n$
- If $\{\mathbf{x}_{\cdot,0}, \mathbf{x}_{\cdot,1}, \dots, \mathbf{x}_{\cdot,p}\}$ forms a set of orthonormal basis, then $\hat{\mathbf{y}} = \sum_{i=0}^p \langle \mathbf{y}, \mathbf{x}_{\cdot,i} \rangle \mathbf{x}_{\cdot,i}$
- If not, we can first do orthogonalization by Gram-Schmidt procedure for the set $\{\mathbf{x}_{\cdot,0}, \mathbf{x}_{\cdot,1}, \dots, \mathbf{x}_{\cdot,p}\}$
- Similar orthogonalization procedures can be done by QR decomposition or SVD of the matrix $\mathbf{X}^T \mathbf{X}$ (classic topics in numerical linear algebra)

Regression by Successive Orthogonalization

- The expansion of \mathbf{y} on the standard orthonormal basis after Gram-Schmidt procedure can be summarised in the following algorithm :
 1. Initialize $\mathbf{z}_0 = \mathbf{x}_0 = \mathbf{1}_n$
 2. For $j = 1, \dots, p$:

Regress \mathbf{x}_j on $\{\mathbf{z}_0, \dots, \mathbf{z}_{j-1}\}$ to produce coefficients
 $\hat{\gamma}_{lj} = \langle \mathbf{z}_l, \mathbf{x}_j \rangle / \langle \mathbf{z}_l, \mathbf{z}_l \rangle$ with $l = 0, \dots, j-1$ and residual vectors
 $\mathbf{z}_j = \mathbf{x}_j - \sum_{k=0}^{j-1} \hat{\gamma}_{kj} \mathbf{z}_k$
 3. Regress \mathbf{y} on the residual \mathbf{z}_p to give the estimate \hat{w}_p
- If \mathbf{x}_p is highly correlated with some of the other \mathbf{x}_k 's, the residual vector \mathbf{z}_p will be close to zero ; in such situation, the coefficient \hat{w}_p with small Z-score $\frac{\hat{w}_p}{\hat{\sigma}_p}$ could be thrown out, where $\hat{\sigma}_p^2 = \frac{\hat{\sigma}^2}{\|\mathbf{z}_p\|_2^2}$ is an estimate of $\text{Var}(\hat{w}_p) = \frac{\sigma^2}{\|\mathbf{z}_p\|_2^2}$

Shortcomings of Fitting Nonlinear Data

- Evaluating the model by Coefficient of Determination R^2 :

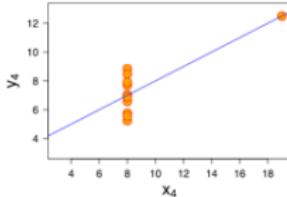
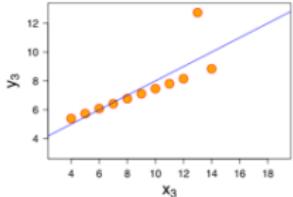
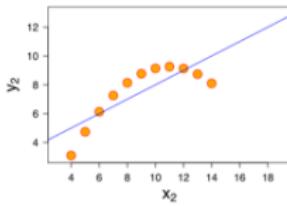
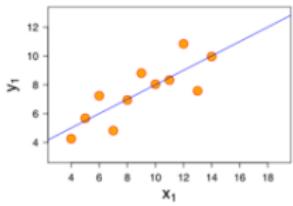
$R^2 := 1 - \frac{SS_{res}}{SS_{tot}}$ ($= \frac{SS_{reg}}{SS_{tot}}$ for linear regression), where

$SS_{tot} = \sum_{i=1}^n (y_i - \bar{y})^2$ is the total sum of squares,

$SS_{reg} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$ is the regression sum of squares, and

$SS_{res} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ is the residual sum of squares.

- The larger the R^2 , the better the model



Multicollinearity

- If the columns of \mathbf{X} are almost linearly dependent, i.e., multicollinearity, then $\det(\mathbf{X}^T \mathbf{X}) \approx 0$, the diagonal entries in $(\mathbf{X}^T \mathbf{X})^{-1}$ is quite large. This implies the variances of $\hat{\mathbf{w}}$ get large, and the estimate is not accurate
- Eg : 10 samples are drawn from the true model
 $y = 10 + 2x_1 + 3x_2 + \epsilon$; the LS estimator is $\hat{w}_0 = 11.292$, $\hat{w}_1 = 11.307$, $\hat{w}_2 = -6.591$, far from the true coefficients; correlation coefficient is $r_{12} = 0.986$
- Remedies : ridge regression, principal component regression, partial least squares regression, etc.

No.	1	2	3	4	5	6	7	8	9	10
x_1	1.1	1.4	1.7	1.7	1.8	1.8	1.9	2.0	2.3	2.4
x_2	1.1	1.5	1.8	1.7	1.9	1.8	1.8	2.1	2.4	2.5
ϵ_i	0.8	-0.5	0.4	-0.5	0.2	1.9	1.9	0.6	-1.5	-1.5
y_i	16.3	16.8	19.2	18.0	19.5	20.9	21.1	20.9	20.3	22.0

Overfitting

- Easily to be overfitted when introducing more variables, e.g., regress housing price with housing size
- The high degree model also fits the noises in the training data, so generalizes poorly to new data
- Remedy : regularization



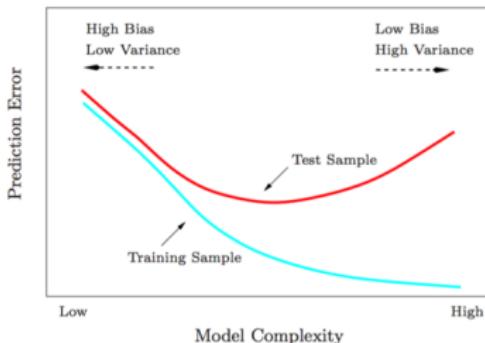
Bias-Variance Decomposition

- Bias-variance decomposition of generalization error in L^2 loss :

$$E_{train} R_{exp}(\hat{f}(\mathbf{x})) = E_{train} E_P[(y - \hat{f}(\mathbf{x}))^2 | \mathbf{x}] = \underbrace{\text{Var}(\hat{f}(\mathbf{x}))}_{\text{variance}} + \underbrace{\text{Bias}^2(\hat{f}(\mathbf{x}))}_{\text{bias}} + \underbrace{\sigma^2}_{\text{noise}}$$

where $P = P(y|\mathbf{x})$ is the conditional probability of y given \mathbf{x}

- Bias : $\text{Bias}(\hat{f}(\mathbf{x})) = E_{train} \hat{f}(\mathbf{x}) - f(\mathbf{x})$ is the average accuracy of prediction for the model (deviation from the truth)
- Variance : $\text{Var}(\hat{f}(\mathbf{x})) = E_{train} (\hat{f}(\mathbf{x}) - E_{train} \hat{f}(\mathbf{x}))^2$ is the variability of the model prediction due to different data set (stability)



Bias-Variance Decomposition (Derivation)

Model $y = f(\mathbf{x}) + \epsilon$, with $E(\epsilon) = 0$ and $\text{Var}(\epsilon) = \sigma^2$ (system error)

$$\begin{aligned} E_{train} R_{exp}(\hat{f}(\mathbf{x})) &= E_P[(y - f(\mathbf{x}))^2 | \mathbf{x}] + E_{train}[(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2] \\ &\quad + 2 \underbrace{E_{train} E_P[(y - f(\mathbf{x}))(f(\mathbf{x}) - \hat{f}(\mathbf{x})) | \mathbf{x}]}_{\text{vanishes since } E_P(y - f(\mathbf{x}) | \mathbf{x}) = 0} \\ &= \sigma^2 + E_{train}[(f(\mathbf{x}) - E_{train} \hat{f}(\mathbf{x}))^2] + E_{train}[(E_{train} \hat{f}(\mathbf{x}) - \hat{f}(\mathbf{x}))^2] \\ &\quad + 2 \underbrace{E_{train}[(f(\mathbf{x}) - E_{train} \hat{f}(\mathbf{x}))(E_{train} \hat{f}(\mathbf{x}) - \hat{f}(\mathbf{x}))]}_{\text{vanishes since } E_{train}[E_{train} \hat{f}(\mathbf{x}) - \hat{f}(\mathbf{x})] = 0} \\ &= \sigma^2 + \text{Bias}^2(\hat{f}(\mathbf{x})) + \text{Var}(\hat{f}(\mathbf{x})) \end{aligned}$$

The more complicated the model, the lower the bias, but the higher the variance.

Bias-Variance Decomposition : kNN Regression

- kNN can be used to do regression if the mode (majority vote) is replaced by mean : $\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_{(i)} \in N_k(\mathbf{x})} y_{(i)}$
- Generalization error of kNN regression is

$$\begin{aligned} E_{train} R_{exp}(\hat{f}(\mathbf{x})) &= \sigma^2 + (f(\mathbf{x}) - \frac{1}{k} \sum_{\mathbf{x}_{(i)} \in N_k(\mathbf{x})} f(\mathbf{x}_{(i)}))^2 \\ &\quad + E_{train} \underbrace{\left[\frac{1}{k} \sum_{\mathbf{x}_{(i)} \in N_k(\mathbf{x})} (y_{(i)} - f(\mathbf{x}_{(i)}))^2 \right]}_{\frac{1}{k} \sigma^2} \end{aligned}$$

where we have used the fact that $E_{train} y_i = f(\mathbf{x}_i)$ and $\text{Var}(y_i) = \sigma^2$.

- For small k , overfitting, bias ↘, variance ↗
- For large k , underfitting, bias ↗, variance ↘

Outlines

Introduction

Linear Regression

Regularizations

Model Assessment

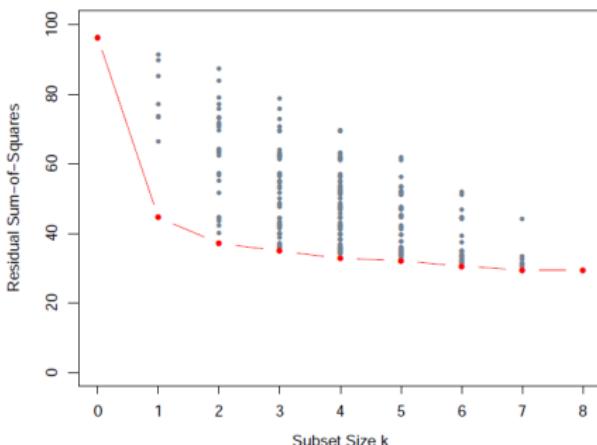
References

Regularization by Subset Selection

- In high dimensions, the more the input attributes, the larger the variance
- Shrinking some coefficients or setting them to zero can reduce the overfitting
- Using less input variables also help interpretation with the most important variables
- Subset selection: retaining only a subset of the variables, while eliminating the rest variables from the model
- Best-subset selection : find for each $k \in \{0, 1, \dots, p\}$ the subset $S_k \subset \{1, \dots, p\}$ of size k that gives the smallest $RSS(\mathbf{w}) = \sum_{i=1}^n (y_i - w_0 - \sum_{j \in S_k} w_j x_{ij})^2$

Best-Subset Selection

- The best subset of size $k + 1$ may not include the variables in the best subset of size k
- The RSS of the best subset of size k is not necessarily decreasing with k
- Choose k based on bias-variance tradeoff, usually by AIC and BIC, or practically by cross-validation

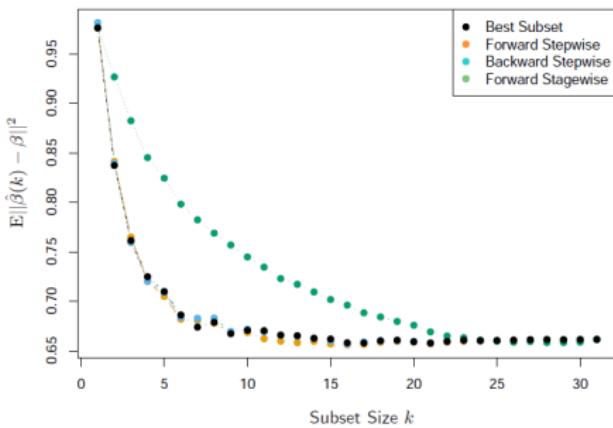


Forward (Backward) Stepwise Selection

- Forward-stepwise selection : start with the intercept \bar{y} , then sequentially add into the model the variables that improve the fit most (reduce RSS most)
- QR factorization helps search the candidate variables to add
- Greedy algorithm : the solution could be sub-optimal
- Computationally more efficient than best-subset selection ; statistically the constrained search enjoys lower variance than best-subset selection
- Backward-stepwise selection : start with the full model, then sequentially delete from the model the variables that has the least impact on the fit most (the candidate for dropping is the variable with the smallest Z-score) ; can only be used when $n > p$ in order to fit the full model by OLS

Forward-Stagewise (FS) Selection

- Starts with the intercept and centered variables with 0 coefficients
- At each step, identify the variables (among all variables) most correlated with the current residual, then regress the residual on this chosen variable and increment the current coefficient with the new regression coefficient
- Ends when no variables are correlated with the residual (arrive at the OLS fit when $n > p$)
- Slower than forward-stepwise : the other variables and their coefficients are not changed at each step except the chosen variable

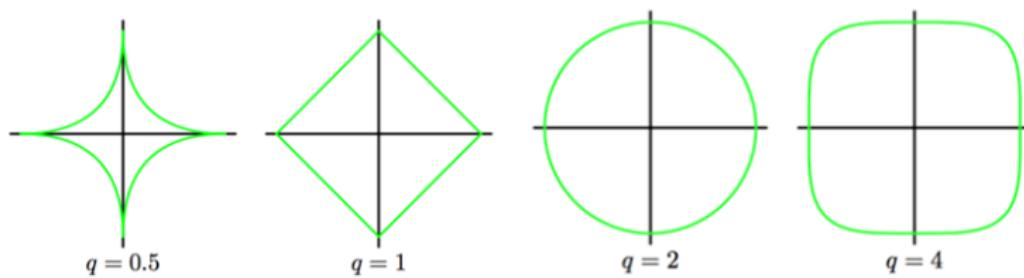


Regularization by Penalties

- Add a penalty term, in general l_q -norm

$$\begin{aligned} & \sum_{i=1}^n (y_i - w_0 - w_1x_1 - \cdots - w_px_p)^2 + \lambda \|\mathbf{w}\|_q^q \\ & = \|\mathbf{y} - \mathbf{Xw}\|_2^2 + \lambda \|\mathbf{w}\|_q^q \end{aligned}$$

- $q = 2$: ridge regression
- $q = 1$: LASSO regression



Ridge Regression

- The optimization problem turns to be

$$\begin{aligned}\hat{\mathbf{w}} &= \arg \min_{\mathbf{w}} \sum_{i=1}^n (y_i - w_0 - w_1 x_1 - \cdots - w_p x_p)^2 + \lambda \|\mathbf{w}\|_2^2 \\ &= \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{Xw}\|_2^2 + \lambda \|\mathbf{w}\|_2^2\end{aligned}$$

- $\lambda \geq 0$ is a fixed parameter which has to be tuned by cross-validation
- Equivalent to the constraint minimization problem :

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{Xw}\|_2^2, \quad \text{subject to} \quad \|\mathbf{w}\|_2 \leq \mu,$$

where $\mu \geq 0$ is a prescribed threshold (tuning parameter)

- The large λ corresponds to the small μ .

Solving Ridge Regression

- Easy to show that $\hat{\mathbf{w}}^{ridge} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_{p+1})^{-1} \mathbf{X}^T \mathbf{y}$
- The estimator is also a projection of \mathbf{y} :
$$\hat{\mathbf{y}}^{ridge} = \mathbf{X}(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_{p+1})^{-1} \mathbf{X}^T \mathbf{y}$$
- \mathbf{X} can be diagonalized by SVD : $\mathbf{X} = \mathbf{P} \mathbf{D} \mathbf{Q}$ with
 $\mathbf{D} = \text{diag}(\nu_1, \dots, \nu_{p+1})$, and $\mathbf{P} \in \mathbb{R}^{n \times (p+1)}$, $\mathbf{Q} \in \mathbb{R}^{(p+1) \times (p+1)}$
being orthogonal matrices ($\mathbf{P}^T \mathbf{P} = \mathbf{I}_{p+1}$)
- $\hat{\mathbf{y}}^{ridge} = \mathbf{P} \text{diag}\left(\frac{\nu_1^2}{\nu_1^2 + \lambda}, \dots, \frac{\nu_{p+1}^2}{\nu_{p+1}^2 + \lambda}\right) \mathbf{P}^T \mathbf{y}$, while $\hat{\mathbf{y}}^{OLS} = \mathbf{P} \mathbf{P}^T \mathbf{y}$
- In the spectral space, the ridge regression estimator is a shrinkage of the OLS estimator ($\lambda = 0$)

Bayesian Viewpoint of Ridge Regression

- Given \mathbf{X} and \mathbf{w} , the conditional distribution of \mathbf{y} is
 $P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}) \propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})\right)$
- In addition, assume \mathbf{w} has a prior distribution
 $P(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0) \propto \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_0)^T \boldsymbol{\Lambda}_0^{-1}(\mathbf{w} - \boldsymbol{\mu}_0)\right)$
- By Bayes theorem, the posterior distribution of \mathbf{w} given the data \mathbf{X} and \mathbf{y} is

$$\begin{aligned} P(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto P(\mathbf{y}|\mathbf{X}, \mathbf{w})P(\mathbf{w}) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{y}^T \mathbf{X}\mathbf{w})\right. \\ &\quad \left.- \frac{1}{2}(\mathbf{w}^T \boldsymbol{\Lambda}_0^{-1} \mathbf{w} - 2\boldsymbol{\mu}_0^T \boldsymbol{\Lambda}_0^{-1} \mathbf{w})\right) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_m)^T \boldsymbol{\Lambda}_m^{-1}(\mathbf{w} - \boldsymbol{\mu}_m)\right) \end{aligned}$$

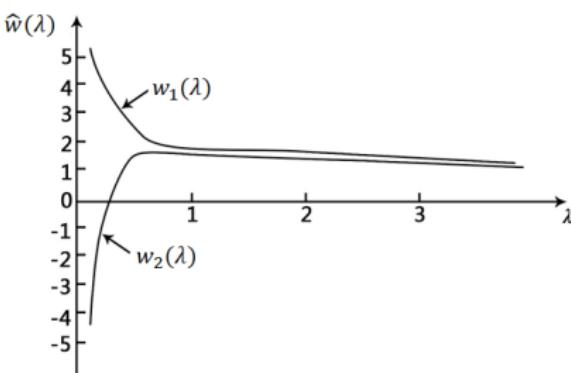
where $\boldsymbol{\Lambda}_m = (\frac{1}{\sigma^2} \mathbf{X}^T \mathbf{X} + \boldsymbol{\Lambda}_0^{-1})^{-1}$ and $\boldsymbol{\mu}_m = \boldsymbol{\Lambda}_m (\frac{1}{\sigma^2} \mathbf{X}^T \mathbf{y} + \boldsymbol{\Lambda}_0^{-1} \boldsymbol{\mu}_0)$

- If $\boldsymbol{\mu}_0 = 0$ and $\boldsymbol{\Lambda}_0 = \frac{\sigma^2}{\lambda} \mathbf{I}_{p+1}$, then $\hat{\mathbf{w}} = \boldsymbol{\mu}_m = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_{p+1})^{-1} \mathbf{X}^T \mathbf{y}$ maximizes the posterior probability $P(\mathbf{w}|\mathbf{X}, \mathbf{y})$

Ridge Trace

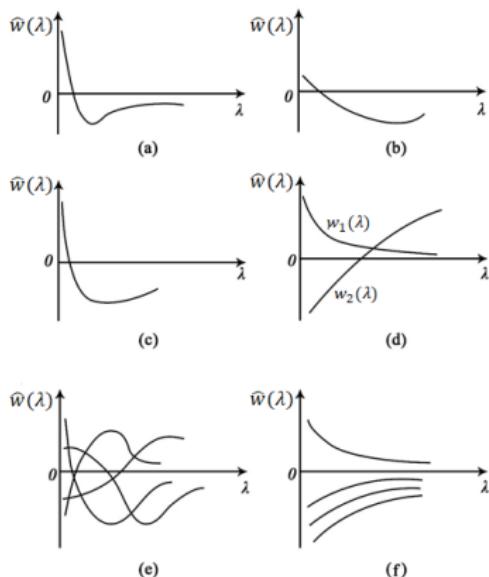
- The functional plot of $\hat{\mathbf{w}}^{ridge}(\lambda)$ with λ is called ridge trace
- The large variations in ridge trace indicate the multicollinearity in variables
- When $\lambda \in (0, 0.5)$, the ridge traces have large variations, it suggests to choose $\lambda = 1$

λ	0	0.1	0.15	0.2	0.3	0.4	0.5	1.0	1.5	2.0	3.0
$\hat{w}_1^{ridge}(\lambda)$	11.31	3.48	2.99	2.71	2.39	2.20	2.06	1.66	1.43	1.27	1.03
$\hat{w}_2^{ridge}(\lambda)$	-6.59	0.63	1.02	1.21	1.39	1.46	1.49	1.41	1.28	1.17	0.98



Reading from Ridge Trace

- Before plot ridge trace, do scaling for the variables
- The coefficients with stable trace and small absolute values should have little influence on y , as in (a)
- The coefficients with large stable absolute values should have great impact on y , as in (b) and (c)
- The ridge traces of the coefficients of two variables are not stable, but the sum of the coefficients is stable. This implies the multicollinearity as in (d)
- The stable ridge traces of all variables suggest good performance using OLS as in (f)



LASSO Regression

- Proposed by R. Tibshirani, short for “Least Absolute Shrinkage and Selection Operator”
- Can be used to estimate the coefficients and select the important variables simultaneously
- Reduce the model complexity, avoid overfitting, and improve the generalization ability
- Also improve the model interpretability

Regression Shrinkage and Selection via the Lasso - jstor

<https://www.jstor.org/stable/2346178> ▾ 翻译此页

作者: R Tibshirani - 1996 - 被引用次数: 27385 相关文章

Regression Shrinkage and Selection via the Lasso. By ROBERT TIBSHIRANI. University of Toronto, Canada. [Received January 1994. Revised January 1995].

LASSO Formulation

- The optimization problem

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} E(\mathbf{w})$$

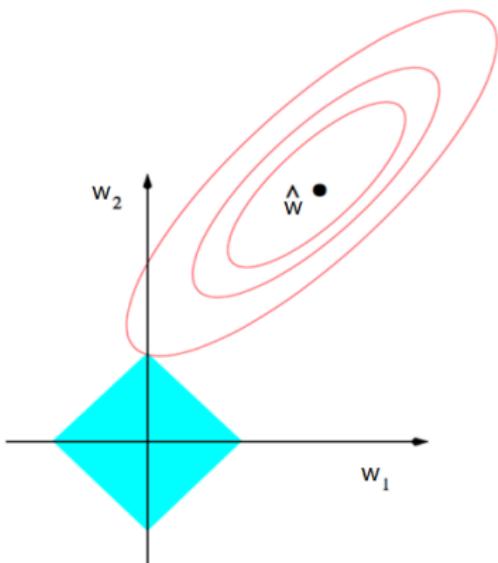
$$E(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1$$

- Equivalent to the constraint minimization problem :

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2,$$

subject to $\|\mathbf{w}\|_1 \leq \mu,$

- The large λ corresponds to the small μ .
- The optimal solution is sparse with $\hat{w}_2 = 0$

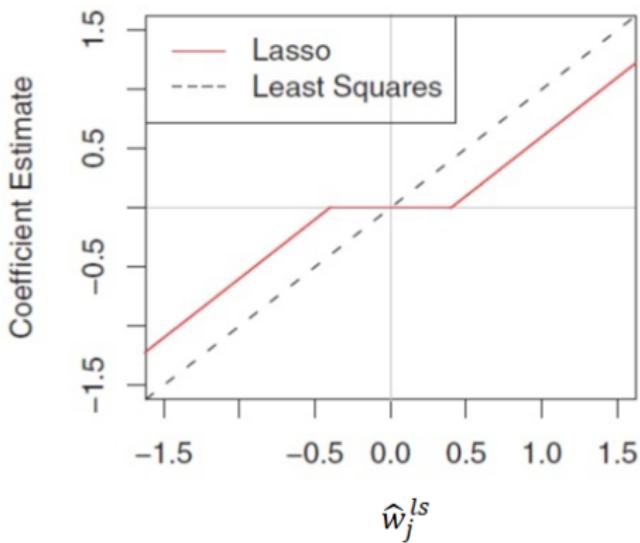


Solving LASSO Regression

- Assume $\mathbf{X}^T \mathbf{X} = \mathbf{I}_{p+1}$, then $\hat{\mathbf{w}}^{OLS} = \mathbf{X}^T \mathbf{y}$
- $\partial_{\mathbf{w}} E(\mathbf{w}) = \mathbf{w} - \mathbf{X}^T \mathbf{y} + \lambda(\partial|w_0| \times \cdots \times \partial|w_p|)$
- $\mathbf{0} \in \partial_{\mathbf{w}} E(\hat{\mathbf{w}}^{lasso})$ implies $0 \in \hat{w}_i^{lasso} - \hat{w}_i^{OLS} + \lambda \partial |\hat{w}_i^{lasso}|$
- If $\hat{w}_i^{lasso} > 0$, $\partial |\hat{w}_i^{lasso}| = \{1\}$, and $\hat{w}_i^{lasso} = \hat{w}_i^{OLS} - \lambda$ with $\hat{w}_i^{OLS} > \lambda$
- If $\hat{w}_i^{lasso} < 0$, $\partial |\hat{w}_i^{lasso}| = \{-1\}$, and $\hat{w}_i^{lasso} = \hat{w}_i^{OLS} + \lambda$ with $\hat{w}_i^{OLS} < -\lambda$
- If $\hat{w}_i^{lasso} = 0$, $\partial |\hat{w}_i^{lasso}| = [-1, 1]$, and $\hat{w}_i^{OLS} \in [-\lambda, \lambda]$
- In summary, $\hat{w}_i^{lasso} = (|\hat{w}_i^{OLS}| - \lambda)_+ \text{sign}(\hat{w}_i^{OLS})$

Shrinkage and Selection Property of LASSO

$\hat{w}_i^{Lasso} = (|\hat{w}_i^{OLS}| - \lambda)_+ \text{sign}(\hat{w}_i^{OLS})$ is called soft thresholding of \hat{w}_i^{OLS} , where $(a)_+ = \max(a, 0)$ is the positive part of a



Maximum A Posteriori (MAP) Estimation

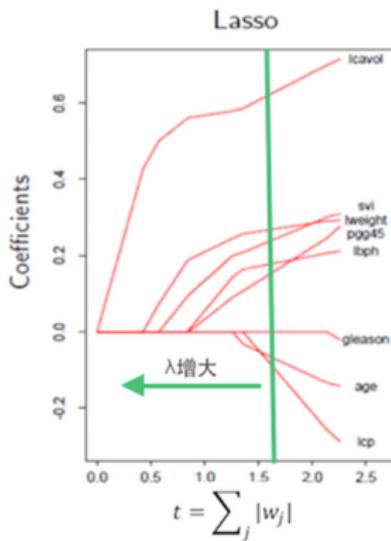
- Given θ , the conditional distribution of \mathbf{y} is $P(\mathbf{y}|\theta)$
- In addition, assume the parameter θ has a prior distribution $P(\theta)$
- The posterior distribution of θ given the data \mathbf{y} is $P(\theta|\mathbf{y}) \propto P(\mathbf{y}|\theta)P(\theta)$
- MAP choose the point of maximal posterior probability :

$$\hat{\theta}^{MAP} = \arg \max_{\theta} P(\theta|\mathbf{y}) = \arg \max_{\theta} (\log P(\mathbf{y}|\theta) + \log P(\theta))$$

- If $\theta = \mathbf{w}$, and we choose the log-prior proportional to $\lambda \|\mathbf{w}\|_2^2$ (i.e., the normal prior $\mathcal{N}(0, \frac{\sigma^2}{\lambda} \mathbf{I})$), we recover the ridge regression
- If the log-prior is proportional to $\lambda \|\mathbf{w}\|_1$, i.e., the prior is the tensor product of Laplace (or double exponential) distribution $\text{Laplace}(0, \frac{2\sigma^2}{\lambda})$
- Different log-prior lead to different penalties (regularization), but this is not the case in general : some penalties may not be the logarithms of probability distributions, some other penalties depend on the data (prior is independent of the data)

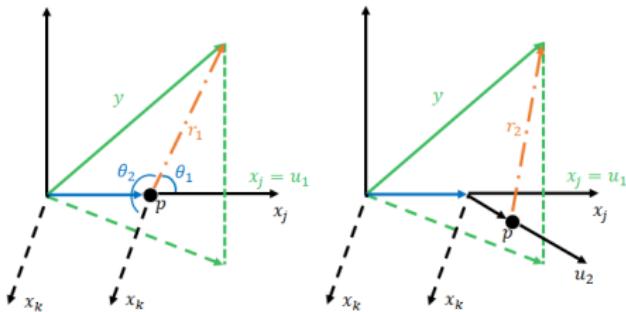
LASSO Path

- When λ varies, the values of the coefficients form paths (regularization paths)
- The paths are piecewise linear with the same change points, may cross the x-axis many times
- In practice, choose λ by cross-validation



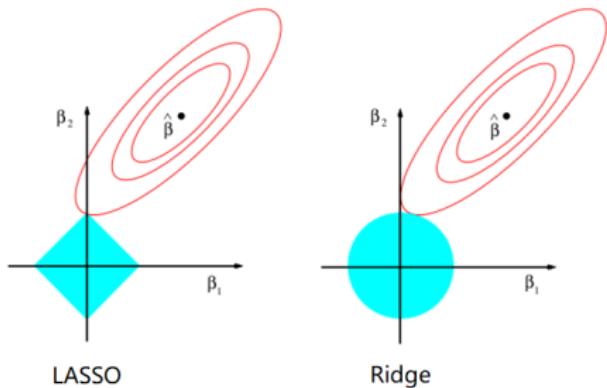
Solving LASSO by LARS (Hastie and Efron)

1. Start with all coefficients w_i equal to zero
2. Find the predictor x_i most correlated with y
3. Increase the coefficient w_i in the direction of the sign of its correlation with y . Take residuals $r = y - \hat{y}$ along the way. Stop when some other predictor x_k has as much correlation with r as x_i has
4. Increase (w_i, w_k) in their joint least squares direction, until some other predictor x_m has as much correlation with the residual r
5. Continue until all predictors are in the model



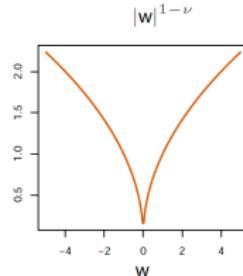
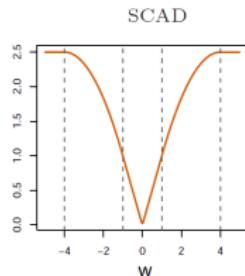
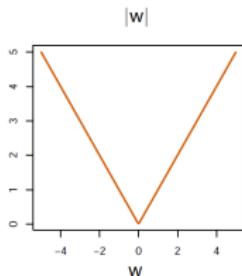
Other Solvers

- “glmnet” by Friedman, Hastie and Tibshirani, implemented by coordinate descent, can be used in linear regression, logistic regression, etc., with LASSO (ℓ_1), ridge (ℓ_2) and elastic net ($\ell_1 + \ell_2$) regularization terms
- Why LASSO seeks the sparse solution in comparison with ridge ?



Related Regularization Models

- Elastic net : $\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1$
- Group LASSO : $\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \sum_{g=1}^G \lambda_g \|\mathbf{w}_g\|_2$, where $\mathbf{w} = (\mathbf{w}_1, \dots, \mathbf{w}_G)$ is the group partition of \mathbf{w}
- Dantzig Selector : $\min_{\mathbf{w}} \|\mathbf{w}\|_1$, subject to $\|\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})\|_\infty \leq \mu$
- Smoothly clipped absolute deviation (SCAD) penalty by Fan and Li (2005) : replace the penalty $\lambda \sum_{i=0}^p |w_i|$ by $\sum_{i=0}^p J_a(w_i, \lambda)$, where $J_a(x, \lambda)$ satisfies (for $a \geq 2$) : $\frac{dJ_a}{dx} = \lambda \text{sign}(x) \left(I(|x| \leq \lambda) + \frac{(a\lambda - |x|)_+}{(a-1)\lambda} I(|x| > \lambda) \right)$
- Adaptive LASSO : weighted penalty $\sum_{i=0}^p \mu_i |w_i|$ where $\mu_i = \frac{1}{|\hat{w}_i^{OLS}|^\nu}$ with $\nu > 0$, as an approximation to $|w_i|^{1-\nu}$, non-convex penalty



Outlines

Introduction

Linear Regression

Regularizations

Model Assessment

References

Errors and R^2

- Mean absolute error (MAE) : $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$
- Mean square error (MSE) : $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
- Root mean square error (RMSE) :

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

- Coefficient of Determination R^2 : $R^2 := 1 - \frac{SS_{res}}{SS_{tot}}$, where
 $SS_{tot} = \sum_{i=1}^n (y_i - \bar{y})^2$ is the total sum of squares, and
 $SS_{res} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ is the residual sum of squares ;
 $R^2 \in [0, 1]$ (might be negative) ; the larger the R^2 , the smaller the ratio of SS_{res} to SS_{tot} , thus the better the model

Adjusted Coefficient of Determination

- Adjusted coefficient of determination : $R_{adj}^2 = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$
- n is the number of samples, p is the dimensionality (or the number of attributes)
- The larger the R_{adj}^2 value, the better performance the model
- When adding important variables into the model, R_{adj}^2 gets larger and SS_{res} is reduced
- When adding unimportant variables into the model, R_{adj}^2 may get smaller and SS_{res} may increase
- In fact, one can show that $1 - R_{adj}^2 = \frac{\hat{\sigma}^2}{S^2}$, where $\hat{\sigma}^2 = \frac{1}{n-p-1} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ and $S^2 = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2$ with $(n-p-1) \frac{\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-p-1}$ and $(n-1) \frac{S^2}{\sigma^2} \sim \chi^2_{n-1}$ if $\mathbf{w} = \mathbf{0}$.

Outlines

Introduction

Linear Regression

Regularizations

Model Assessment

References

References

- 数据分析导论，博雅大数据学院
- 周志华，机器学习，2016
- T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning : Data mining, Inference, and Prediction*, 2nd Edition, 2009

Introduction to Big Data Analysis Classification : Part 2

Zhen Zhang

Southern University of Science and Technology

Outlines

Logistic Regression

Linear Discriminant Analysis

Neural Network

Support Vector Machine

References

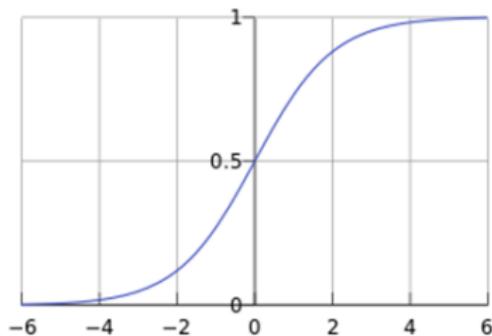
Logistic Regression

- Not regression, but a classification method
- Connection with linear regression :

$y = w_0 + w_1x + \epsilon$, y is binary (0 or 1); then

$$E(y|x) = P(y=1|x) = w_0 + w_1x$$

but $w_0 + w_1x$ may not be a probability
- Find a function to map it back to $[0, 1]$: Sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ with $z = w_0 + w_1x_1 + \dots + w_dx_d$



- Equivalently,

$$\log \frac{P(y=1|x)}{1-P(y=1|x)} = w_0 + w_1x_1 + \dots + w_dx_d$$
,
 logit transform

$$\text{logit}(z) = \log \frac{z}{1-z}$$

MLE for Logistic Regression

- The prob. distribution for two-class logistic regression model is

$$Pr(y = 1 | \mathbf{X} = \mathbf{x}) = \frac{\exp(\mathbf{w}^T \mathbf{x})}{1 + \exp(\mathbf{w}^T \mathbf{x})},$$

$$Pr(y = 0 | \mathbf{X} = \mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^T \mathbf{x})}.$$

- Let $P(y = k | \mathbf{X} = \mathbf{x}) = p_k(\mathbf{x}; \mathbf{w})$, $k = 0$ or 1 . The likelihood function is defined by $L(\mathbf{w}) = \prod_{i=1}^n p_{y_i}(\mathbf{x}_i; \mathbf{w})$
- MLE estimate of \mathbf{w} : $\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} L(\mathbf{w})$
- Solve $\nabla_{\mathbf{w}} \log L(\mathbf{w}) = 0$ by Newton-Raphson method

Outlines

Logistic Regression

Linear Discriminant Analysis

Neural Network

Support Vector Machine

References

Linear Discriminant Analysis (LDA)

- Bayes Classifier amounts to know the class posteriors $P(Y|\mathbf{X})$ for optimal classification : $k^* = \arg \max_k P(Y = k|\mathbf{X})$
- Let $\pi_k = P(Y = k)$ be the prior probability, $f_k(\mathbf{x}) = P(\mathbf{X} = \mathbf{x}|Y = k)$ be the density function of samples in each class $Y = k$
- By Bayes theorem, $P(Y|\mathbf{X} = \mathbf{x}) \propto f_k(\mathbf{x})\pi_k$ (Recall naive Bayes)
- Assume $f_k(x)$ is multivariate Gaussian :

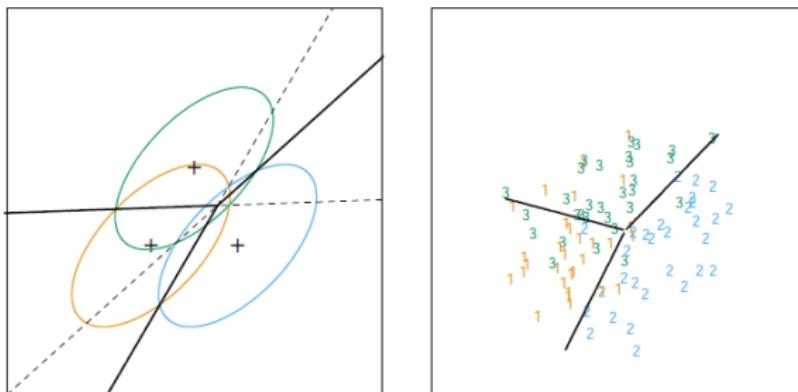
$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^p/2 |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_k^{-1} (\mathbf{x}-\mu_k)}$$
, with a common covariance matrix $\Sigma_k = \Sigma$, sufficient to look at the log-ratio

$$\begin{aligned} \log \frac{P(Y = k|\mathbf{X} = \mathbf{x})}{P(Y = l|\mathbf{X} = \mathbf{x})} &= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) \\ &\quad + \mathbf{x}^T \Sigma^{-1} (\mu_k - \mu_l) \end{aligned}$$

for the decision boundary between class k and l

Discriminant Rule

- Linear discriminant functions :
$$\delta_k(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \pi_k$$
- Then $\log \frac{P(Y=k|\mathbf{X}=\mathbf{x})}{P(Y=I|\mathbf{X}=\mathbf{x})} = \delta_k(\mathbf{x}) - \delta_I(\mathbf{x})$
- Decision rule : $k^* = \arg \max_k \delta_k(\mathbf{x})$
- Sample estimate of unknowns : $\hat{\pi}_k = N_k/N$, where
 $N = \sum_{k=1}^K N_k$, $\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{y_i=k} \mathbf{x}_i$,
 $\hat{\boldsymbol{\Sigma}} = \frac{1}{N-K} \sum_{k=1}^K \sum_{y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T$

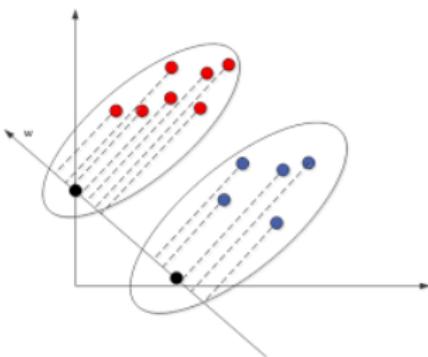


Two-class LDA

- LDA rule classifies to class 2 if

$$\left(\mathbf{x} - \frac{\hat{\mu}_1 + \hat{\mu}_2}{2}\right)^T \boldsymbol{\Sigma}^{-1} (\hat{\mu}_2 - \hat{\mu}_1) + \log \frac{\hat{\pi}_2}{\hat{\pi}_1} > 0$$

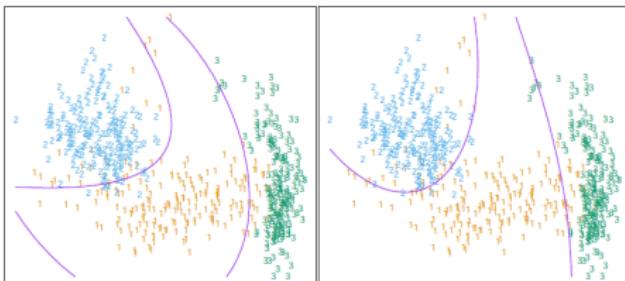
- Discriminant direction : $\beta = \boldsymbol{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$
- Bayes misclassification rate = $1 - \Phi(\beta^T(\mu_2 - \mu_1)/(\beta^T \boldsymbol{\Sigma} \beta)^{\frac{1}{2}})$, where $\Phi(x)$ is the Gaussian distribution function



Other Variants

- Quadratic discriminant analysis (QDA) :

$$\delta_k(\mathbf{x}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \log \pi_k$$
 - Regularized discriminant analysis : $\hat{\boldsymbol{\Sigma}}_k(\alpha) = \alpha \hat{\boldsymbol{\Sigma}}_k + (1 - \alpha) \hat{\boldsymbol{\Sigma}}$
 - Computations for LDA :
 1. Sphere the data with respect to $\hat{\boldsymbol{\Sigma}} = \mathbf{U}\mathbf{D}\mathbf{U}^T$: $\mathbf{X}^* = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{X}$.
 Then the common covariance estimate of \mathbf{X}^* is \mathbf{I}_p
 2. Classify to the closest class centroid in the transformed space, taking into account of the class prior probabilities π_k 's
 - Reduced-Rank LDA : see dimensionality reduction



Outlines

Logistic Regression

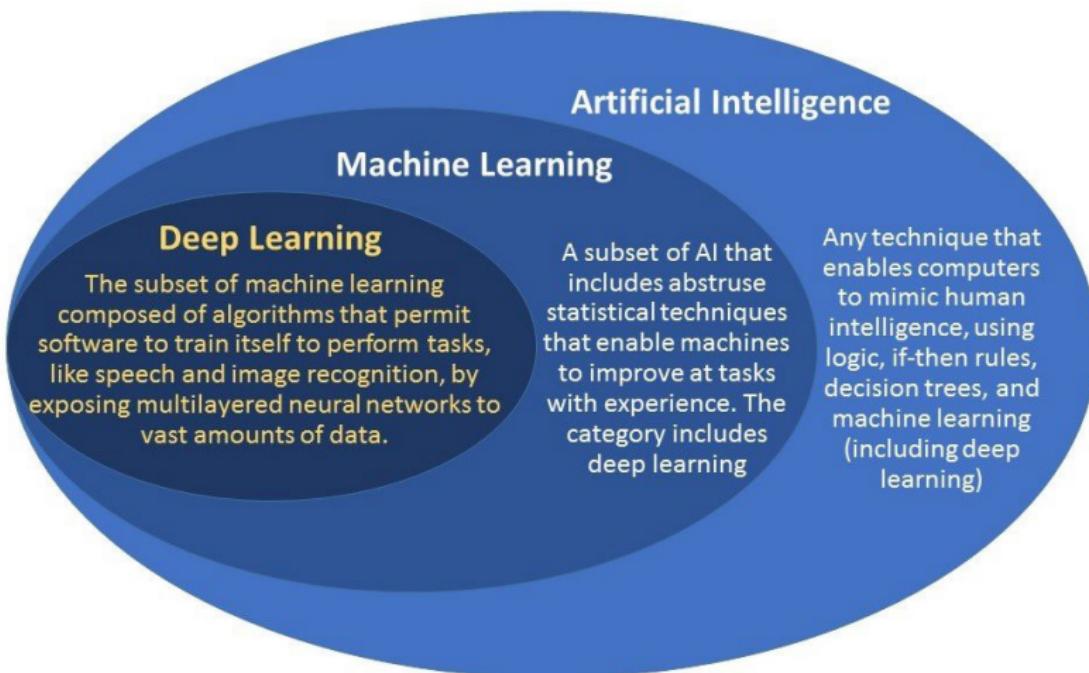
Linear Discriminant Analysis

Neural Network

Support Vector Machine

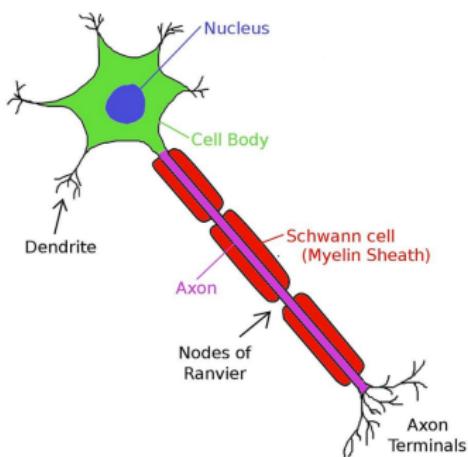
References

What is Deep Learning?



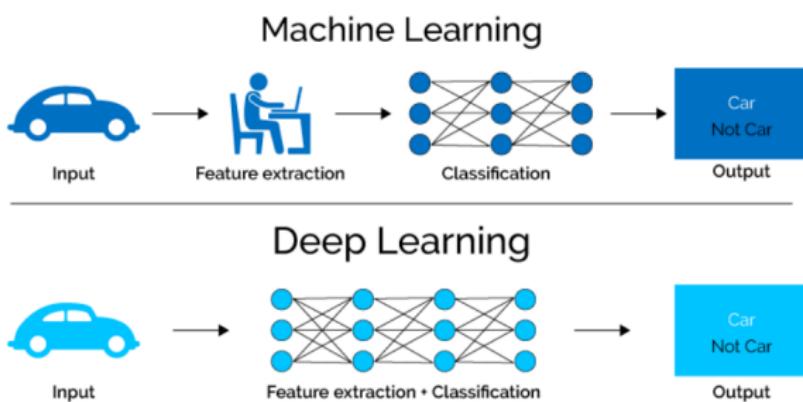
Deep Learning

- Deep learning is a sub field of Machine Learning that very closely tries to mimic human brain's working using neurons.
- These techniques focus on building Artificial Neural Networks (ANN) using several hidden layers.
- There are a variety of deep learning networks such as Multilayer Perceptron (MLP), Autoencoders (AE), Convolution Neural Network (CNN), Recurrent Neural Network (RNN).



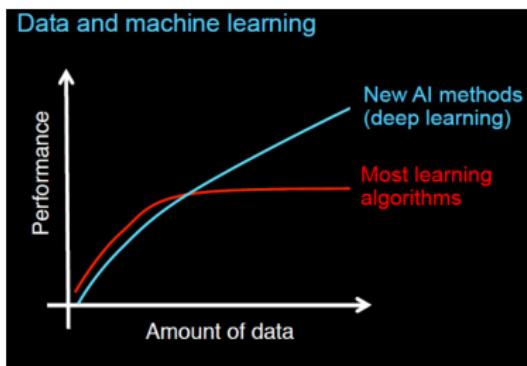
ML vs DL

- In Machine Learning, the features need to be identified by an domain expert.
- In Deep Learning, the features are learned by the neural network.



Why Deep Learning is Growing?

- Processing power needed for Deep learning is readily becoming available using GPUs, Distributed Computing and powerful CPUs.
- Moreover, as the data amount grows, Deep Learning models seem to outperform Machine Learning models.
- Explosion of features and datasets.
- Focus on customization and real time decisioning.



Why Now?

1952	Stochastic Gradient Descent
1958	Perceptron <ul style="list-style-type: none">• Learnable Weights
...	
1986	Backpropagation <ul style="list-style-type: none">• Multi-Layer Perceptron
1995	Deep Convolutional NN <ul style="list-style-type: none">• Digit Recognition
...	

Neural Networks date back decades, so why the resurgence?

1. Big Data

- Larger Datasets
- Easier Collection & Storage



2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable

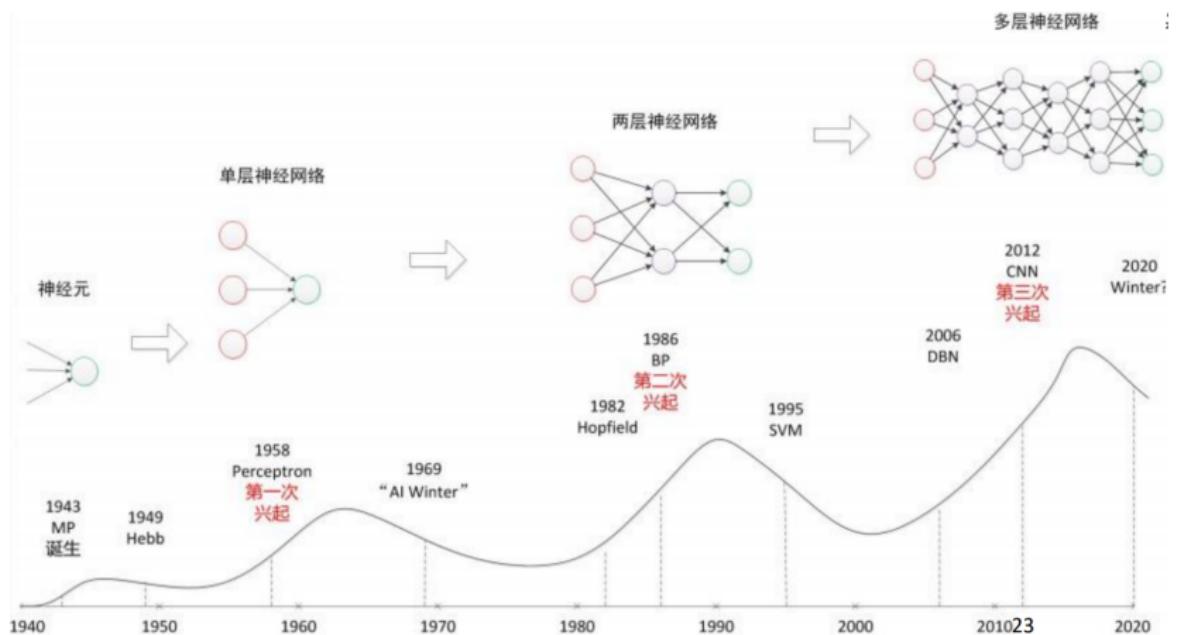


3. Software

- Improved Techniques
- New Models
- Toolboxes



History



Big Guys

COMPANIES & PEOPLE



Taken in NIPS2014, from left: Yann LeCun (Facebook, NYU), Geoffrey Hinton (Google, U of Toronto), Yoshua Bengio (U of Montreal), Andrew Ng (Baidu)



Outlines

Logistic Regression

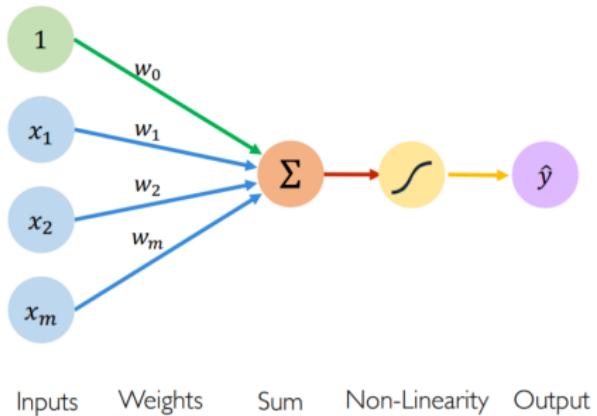
Linear Discriminant Analysis

Neural Network

Support Vector Machine

References

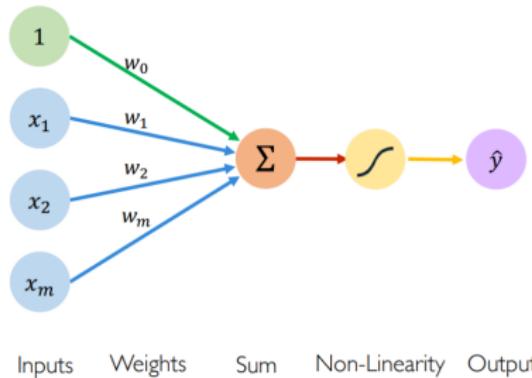
The Perceptron : Forward Propagation



$$\hat{y} = g \left(w_0 + \sum_{i=1}^m x_i w_i \right),$$

- \hat{y} is the Output,
- g is a Non-linear activation function,
- w_0 is the Bias.

The Perceptron : Forward Propagation

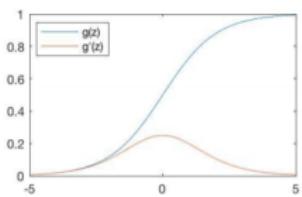


$$\hat{y} = g(w_0 + \mathbf{X}^T \mathbf{W}), \text{ where}$$

$$\mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}.$$

Common Activation Functions

Sigmoid Function

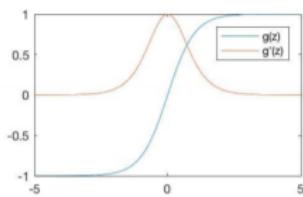


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.nn.sigmoid(z)`

Hyperbolic Tangent

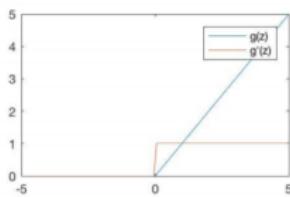


$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.nn.tanh(z)`

Rectified Linear Unit (ReLU)



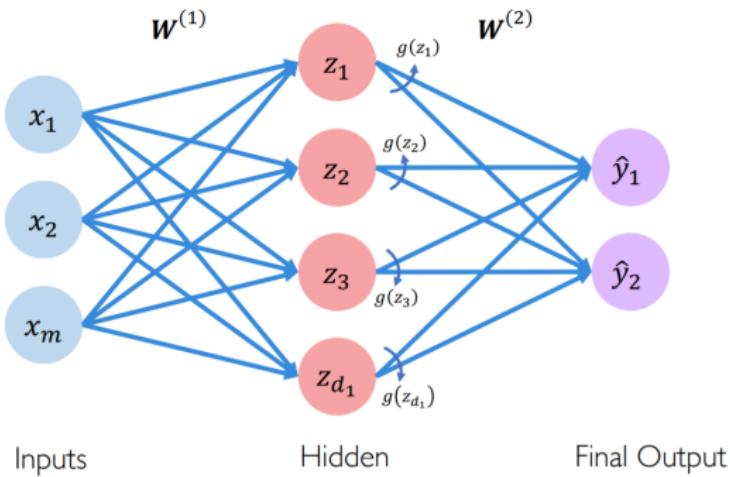
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

`tf.nn.relu(z)`

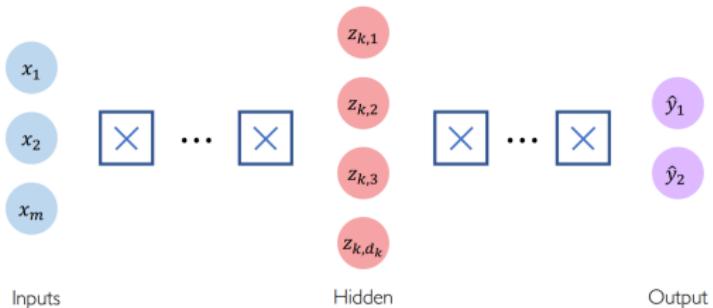
Note all activation functions are nonlinear.

Single Layer Neural Network



$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^m x_j w_{j,i}^{(1)}, \quad \hat{y}_i = g \left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j w_{j,i}^{(2)} \right).$$

Deep Neural Network



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j}) w_{j,i}^{(k)}.$$

Theorem (Universal approximation theorem (Cybenko 1980, 1989))

1. Any function can be approximated by a three-layer neural network within sufficiently high accuracy.
2. Any bounded continuous function can be approximated by a two-layer neural network within sufficiently high accuracy.

Loss Optimization

We want to find the network weights that achieve the lowest loss

$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}\left(f\left(x^{(i)}; \mathbf{W}\right), y^{(i)}\right),$$

where $\mathcal{L}\left(f\left(x^{(i)}; \mathbf{W}\right), y^{(i)}\right)$ is the loss function we defined according to the specific problem to measure the differences between output state $f\left(x^{(i)}; \mathbf{W}\right)$ and reference state $y^{(i)}$. It also can be written as

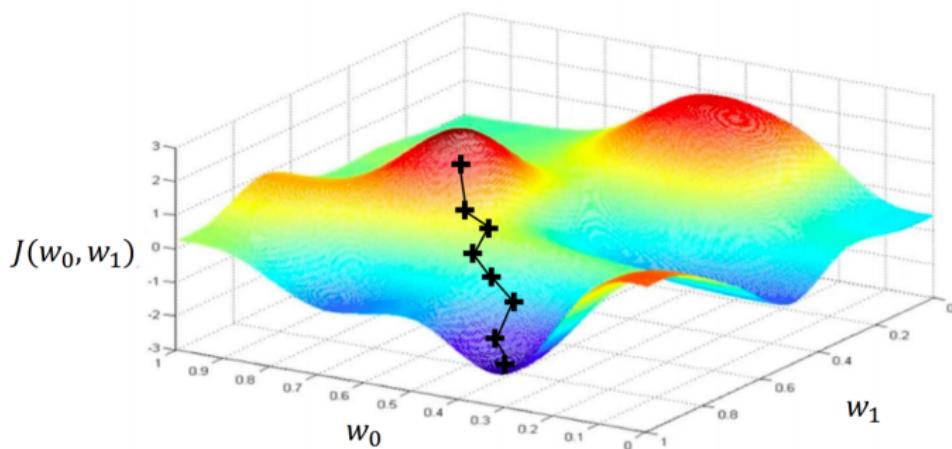
$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} C(\mathbf{W}).$$

Remember

$$\mathbf{W} = \left\{ \mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \dots \right\}.$$

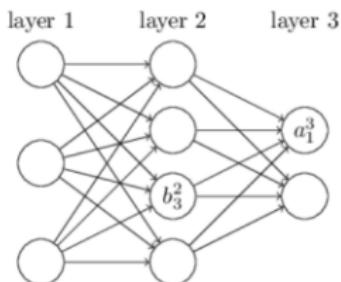
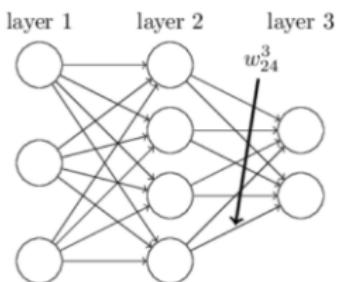
Gradient Decent

We can use Gradient Decent algorithm to find the optimal parameter \mathbf{W} .



Note that we should calculate $\frac{\partial C}{\partial \mathbf{W}}$ to update \mathbf{W} .

Notations



- w_{jk}^l is the weight for the connection from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer.
- for brevity $b_j^l = w_{j0}^l$ is the bias of the j^{th} neuron in the l^{th} layer.
- a_j^l for the activation of the j^{th} neuron in the l^{th} layer z_j^l .

$$a_j^l = g(z_j^l) = g \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right)$$

Four fundamental equations

We first define the error δ_j^l of neuron j in layer by

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l},$$

and we give the four fundamental equations of back propagation :

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad (BP1)$$

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma'(z^l) \quad (BP2)$$

$$\frac{\partial C}{\partial b_j^L} = \delta_j^l \quad (BP3)$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (BP4)$$

An equation for the error in the output layer (BP1)

The components of δ^L are given by

$$\delta^L = \nabla_a C \odot \sigma' (z^L) \quad (BP1)$$

Démonstration.

$$\begin{aligned}\delta_j^L &= \frac{\partial C}{\partial z_j^L} \\ &= \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial \sigma(z_j^L)}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)\end{aligned}$$



An equation for the error in the hidden layer (BP2)

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma' \left(z^l \right) \quad (BP2)$$

Démonstration.

$$\begin{cases} \delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ z_k^{l+1} = \left(\sum_i w_{ki}^{l+1} a_i^l \right) + b_k^{l+1} = \left(\sum_i w_{ki}^{l+1} \sigma(z_i^l) \right) + b_k^{l+1} \end{cases}$$

$$\Rightarrow \begin{cases} \delta_j^l = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ \frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l) \end{cases} \Rightarrow \delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l)$$



The change of the cost with respect to any bias (BP3)

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (BP3)$$

Démonstration.

$$\begin{cases} \frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} \\ z_j^l = \left(\sum_k w_{jk}^l d_k^{l-1} \right) + b_j^l \Rightarrow \frac{\partial z_k^l}{\partial b_j^l} = 1 \end{cases} \Rightarrow \frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \cdot 1 = \delta_j^l. \quad \square$$

The change of the cost with respect to any weight (BP4)

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (BP4)$$

Démonstration.

$$\begin{cases} \frac{\partial C}{\partial w_{jk}^l} = \sum_i \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\ z_j^l = \left(\sum_m w_{jm}^l a_m^{l-1} \right) + b_j^l \Rightarrow \frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \end{cases}$$
$$\Rightarrow \frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}$$



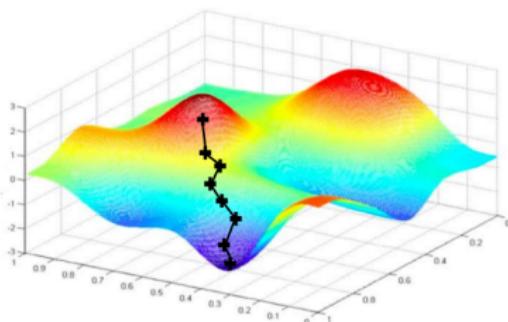
Back Propagation procedure

- 1. Input x : Set the corresponding activation a^1 for the input layer.
- 2. Feedforward : For each $l = 2, 3, \dots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.
- 3. Output error δ^L : Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.
- 4. Backpropagate the error : For each $l = L-1, L-2, \dots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
- 5. Output : The gradient of the cost function is given by $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ and $\frac{\partial C}{\partial b_j^l} = \delta_j^l$.

Gradient Descent

Algorithm

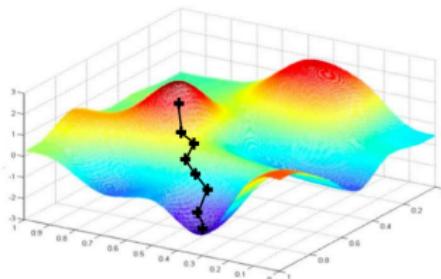
1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
4. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
5. Return weights



Stochastic Gradient Descent

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Pick single data point i
4. Compute gradient, $\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{W}}$
5. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
6. Return weights



- Mini-batches lead to fast training !
- Can parallelize computation + achieve significant speed increases on GPUs.

Outlines

Logistic Regression

Linear Discriminant Analysis

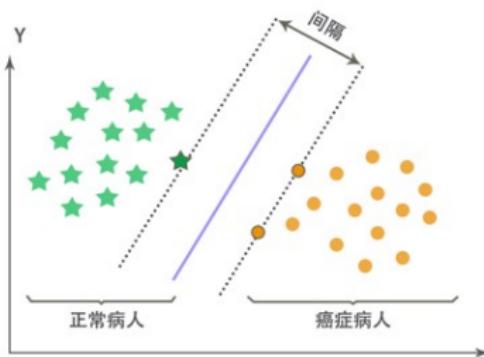
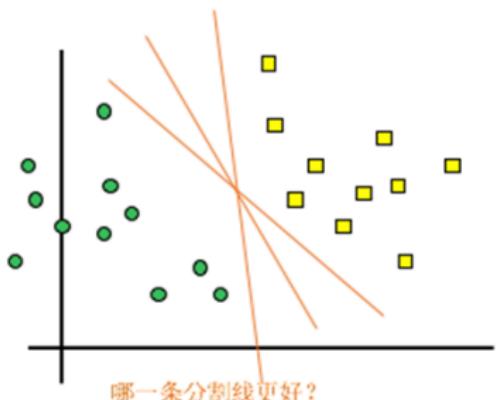
Neural Network

Support Vector Machine

References

Support Vector Machine (SVM)

- Use hyperplane to separate data : maximize margin
- Can deal with low-dimensional data that are not linearly separated by using kernel functions
- Decision boundary only depends on some samples (support vectors)



Linear SVM

- Training data : $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{-1, +1\}$
- Hyperplane : $S = \mathbf{w}^T \mathbf{x} + b$; decision function :
 $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$

$$\left. \begin{array}{l} f(\mathbf{x}_i) > 0 \Leftrightarrow y_i = 1 \\ f(\mathbf{x}_i) < 0 \Leftrightarrow y_i = -1 \end{array} \right\} \Rightarrow y_i f(\mathbf{x}_i) > 0$$

- Geometric margin between a point and hyperplane :
 $r_i = \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|_2}$
- Margin between dataset and hyperplane : $\min_i r_i$
- Maximize margin : $\max_{\mathbf{w}, b} \min_i \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|_2}$

Formulation as Constrained Optimization

- Without loss of generality, let $\min_i y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$ (multiply \mathbf{w} and b by the same proper constant)
- Maximize margin is equivalent to

$$\max_{\mathbf{w}, b} \frac{1}{\|\mathbf{w}\|_2}, \quad \text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, i = 1, \dots, n$$

- Further reduce to

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2, \quad \text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, i = 1, \dots, n$$

- This is primal problem : quadratical programming with linear constraints, computational complexity is $O(p^3)$ where p is dimension

Method of Lagrange Multipliers

- Introduce $\alpha_i \geq 0$ as Lagrange multiplier of constraint $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$
- Lagrange function :

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|_2^2 - \sum_{i=1}^n \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

- Since

$$\max_{\alpha} L(\mathbf{w}, b, \alpha) = \begin{cases} \frac{1}{2} \|\mathbf{w}\|_2^2, & y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, \forall i \\ +\infty, & y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 < 0, \exists i \end{cases}$$

- Primal problem is equivalent to the minimax problem

$$\min_{\mathbf{w}, b} \max_{\alpha} L(\mathbf{w}, b, \alpha)$$

Dual problem

- When slater condition is satisfied, $\min \max \Leftrightarrow \max \min$
- Dual problem : $\max_{\alpha} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$
- Solve for inner minimization problem :

$$\nabla_{\mathbf{w}} L = 0 \implies \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = 0 \implies \sum_i \alpha_i y_i = 0$$

- Plug into L : $L(\mathbf{w}^*, b^*, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$
- Dual optimization :

$$\min_{\alpha} \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) - \sum_i \alpha_i,$$

s.t. $\alpha_i \geq 0, i = 1, \dots, n, \sum_i \alpha_i y_i = 0$

KKT conditions

- Three more conditions from the equivalence of primal and minimax problems

$$\begin{cases} \alpha_i^* \geq 0, \\ y_i((\mathbf{w}^*)^T \mathbf{x}_i + b^*) - 1 \geq 0, \\ \alpha_i^*[y_i((\mathbf{w}^*)^T \mathbf{x}_i + b^*) - 1] = 0. \end{cases}$$

- These together with two zero derivative conditions form KKT conditions
- $\alpha_i > 0 \Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b^*) = 1$
- Index set of support vectors $S = \{i | \alpha_i > 0\}$
- $b = y_s - \mathbf{w}^T \mathbf{x}_s = y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s$
- More stable solution : $b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s \right)$

Sequential Minimal Optimization (SMO) Algorithm

- Invented by John C. Platt (1998)
- Coordinateally optimize dual problem, select two variables and fix others, then dual problem reduces to one variable quadratic programming with positivity constraint
 1. Initially, choose α_i and α_j
 2. Fix other variables, solve for α_i and α_j
 3. Update α_i and α_j , redo step 1 iteratively
 4. Stop until convergence
- How to choose α_i and α_j ? choose the pair far from KKT conditions the most
- Computational complexity $O(n^3)$
- Easy to generalize to high dimensional problem with kernel functions

Soft Margin

- When data are not linear separable, introduce slack variables (tolerance control of fault) $\xi_i \geq 0$
- Relax constraint to $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$
- Primal problem :

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \xi_i$$

s.t. $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \dots, n$

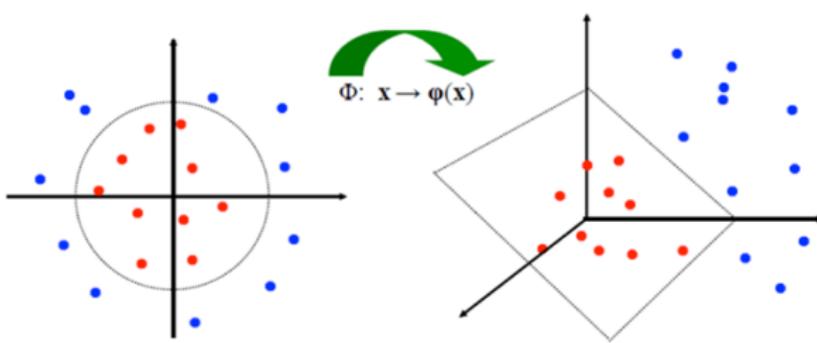
- Similar derivation to dual problem :

$$\min_{\alpha} \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) - \sum_i \alpha_i,$$

s.t. $0 \leq \alpha_i \leq C, i = 1, \dots, n, \sum_i \alpha_i y_i = 0$

Nonlinear SVM

- Nonlinear decision boundary could be mapped to linear boundary in high-dimensional space
- Modify objective function in dual problem :
$$\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)) - \sum_i \alpha_i$$
- Kernel function as inner product : $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$



Kernel Methods

- Reduce effect of curse of dimensionality
- Different kernels lead to different decision boundaries
- Popular kernels :

Kernel	Definition	Parameters
Polynomial	$(\mathbf{x}_1^T \mathbf{x}_2 + 1)^d$	d is positive integer
Gaussian	$e^{-\frac{\ \mathbf{x}_1 - \mathbf{x}_2\ ^2}{2\delta^2}}$	$\delta > 0$
Laplacian	$e^{-\frac{\ \mathbf{x}_1 - \mathbf{x}_2\ }{\delta^2}}$	$\delta > 0$
Fisher	$\tanh(\beta \mathbf{x}_1^T \mathbf{x}_2 + \theta)$	$\beta > 0, \theta < 0$

Pros and Cons

- Where it is good
 - Applications in pattern recognition : text classification, face recognition
 - Easy to deal with high-dimensional data with kernels
 - Robust (only depends on support vectors), and easy to generalize to new dataset
- Disadvantage
 - Poor for ultra high dimensional data
 - Low computational efficiency for nonlinear SVM when sample size is large
 - Poor interpretability without probability

Outlines

Logistic Regression

Linear Discriminant Analysis

Neural Network

Support Vector Machine

References

References

- 数据分析导论, 博雅大数据学院
- 周志华, 机器学习, 2016
- T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning : Data mining, Inference, and Prediction*, 2nd Edition, 2009

Introduction to Big Data Analysis Ensemble Methods

Zhen Zhang

Southern University of Science and Technology

Outlines

Introduction

Bagging and Random Forest

Boosting and AdaBoost

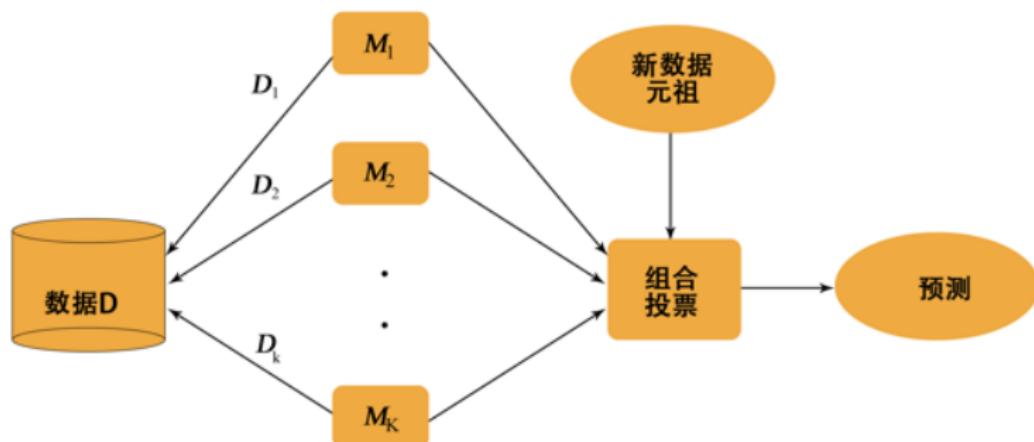
Gradient Boosting Decision Tree

XGBoost

Conclusion and Python Examples

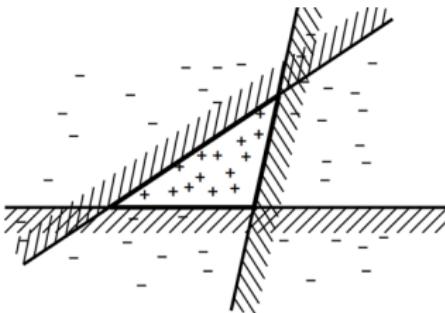
Ensemble Methods

- Wisdom of Crowds (“三个臭皮匠，顶个诸葛亮”)
- Multiple weak learners (base learners, may be heterogenous) can improve learning performance



Why it can improve the performance

- More expressive, can approximate larger functional space
 - Single linear classifier (perceptron) does not work
 - Try multiple classifiers



- Reduce misclassification rate
 - Misclassification rate of single classifier is p
 - Choose N classifiers, same but independent, voting
 - Error rate of majority vote = $\sum_{k>N/2} \binom{N}{k} p^k (1-p)^{N-k}$
 - When $N = 5, p = 0.1$, Error rate < 0.01

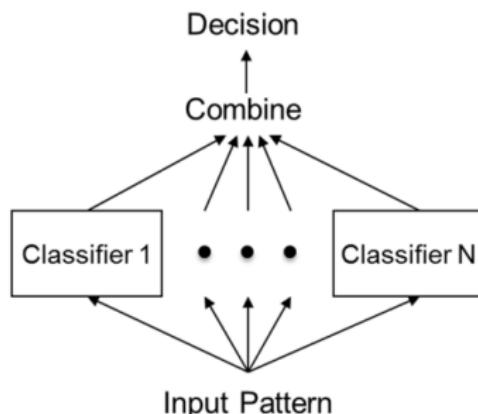
Two commonly used ensemble methods

- Bagging

- Random sampling : generating independent models, and averaging for regressions (making majority vote for classifications)
- Reducing variances
- Example : Random forests

- Boosting

- Sequential training : training the subsequent models based on the errors of previous models
- Reducing bias
- Examples : AdaBoost and GBDT



Outlines

Introduction

Bagging and Random Forest

Boosting and AdaBoost

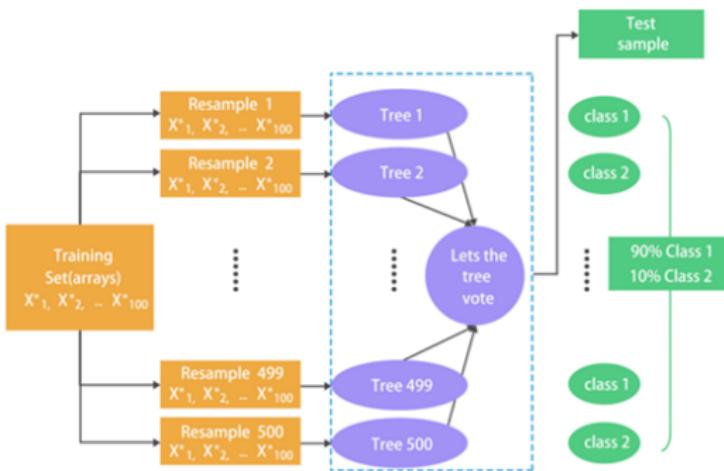
Gradient Boosting Decision Tree

XGBoost

Conclusion and Python Examples

Bagging

- Bagging is short for bootstrap aggregation
- Bagging generates a committee of predictors and combine them in a certain manner to the final model
- Single predictor suffers from instability, while bagging could improve the stability by majority vote (classification) or averaging (regression) over all single predictors



Sampling

- Given a dataset D of n samples, at the iteration $m = 1, \dots, M$, the training set D_m is obtained by sampling from D with replacement. Then D_m is used to construct classifier $\hat{f}_m(x)$.
- Sampling with replacement : some samples in D may be missing in D_m , while some other samples may occur more than once
- On average, 63.2% of the samples in D could be selected into D_m . In fact, for each sample, the probability that it is not selected in one round is $1 - \frac{1}{n}$. Then it is not selected in all n rounds with probability $\lim_{n \rightarrow \infty} (1 - \frac{1}{n})^n = 0.368$.

Algorithm

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
 - Output : additive model $\hat{f}_{bag}(x)$
1. For $m = 1$ to M :
 - 1.1 Sample from D with replacement to obtain D_m
 - 1.2 Train a model $\hat{f}_m(x)$ from the dataset D_m : for classification, $\hat{f}_m(x)$ returns a K-class 0-1 vector e_k ; for regression, it is just a value
 2. Compute bagging estimate $\hat{f}_{bag}(x) = \frac{1}{M} \sum_{m=1}^M \hat{f}_m(x)$: for classification, make majority vote $\hat{G}_{bag}(x) = \arg \max_k \hat{f}_k(x)$; for regression, just return the average value

Variance Reduction

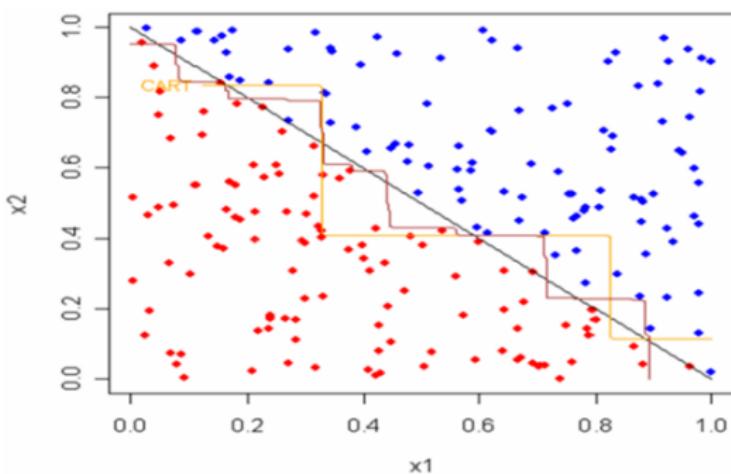
- In bagging, we use the same model to train different sample set in each iteration ; assume the models $\{\hat{f}_m(x)\}_{m=1}^M$ have the same variance $\sigma^2(x)$, while the correlation of each pair is $\rho(x)$
- Then the variance of the final model is :

$$\begin{aligned}\text{Var}(\hat{f}_{bag}(x)) &= \frac{1}{M^2} \left(\sum_{m=1}^M \text{Var}(\hat{f}_m(x)) + \sum_{t \neq m} \text{Cov}(\hat{f}_t(x) \hat{f}_m(x)) \right) \\ &= \rho(x) \sigma^2(x) + \frac{1 - \rho(x)}{M} \sigma^2(x)\end{aligned}$$

- As $M \rightarrow \infty$, $\text{Var}(\hat{f}_{bag}(x)) \rightarrow \rho(x) \sigma^2(x)$. This usually reduces the variance.
- If $\rho(x) = 0$, the variance could approach zero
- The random sampling in bagging is to reduce the correlation $\rho(x)$, i.e., make the sub-predictors as independent as possible

Limitations of Decision Tree

- Stuck at local optimum : The greedy algorithm makes it stop at the local optimum, as it seeks the maximal information gain in each tree split
- Decision boundary : Use one feature in each split, the decision boundary is parallel to the coordinate axes
- Bad representability and instability



Random Forest

- Random Forest further reduces the variance by adding independency to the committee of decision trees
- This is achieved by introducing more randomness.
- More randomness :
 - Sampling on the training data with replacement
 - Select features at random
- No pruning is needed.
- Example : RF consisting of 3 independent trees, each with an error rate of 40%. Then the probability that more than one tree misclassify the samples is
$$0.4^3 + 3 * 0.4^2 * (1 - 0.4) = 0.352$$

Random Forest Algorithm

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
 - Output : additive model $\hat{f}_{rf}(x)$
1. For $m = 1$ to M :
 - 1.1 Sample from D with replacement to obtain D_m
 - 1.2 Grow a random-forest tree T_m to the dataset D_m : by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached
 - 1.2.1 Select q features at random from the p features
 - 1.2.2 Pick the best feature/split-point among the q
 - 1.2.3 Split the node into two daughter nodes
 2. Output the ensemble of trees $\{T_m\}_{m=1}^M$: for regression,

$$\hat{f}_{rf}(x) = \frac{1}{M} \sum_{m=1}^M T_m(x)$$
 : for classification, make majority vote
- Small value of q increases the independency of trees ; empirically, $q = \log_2 p + 1$

Model Evaluation

- Margins : The difference between the percentage of decision trees that correctly classify the samples and the percentage of trees misclassifying the samples
- Out-of-bag (OOB) errors : The observation is called out-of-bag sample to some trees if it is not sampled for those trees. Denote the training set in the m -th sampling by D_m . OOB error is computed as :
 1. For each observation (x_i, y_i) , find the trees which treat it as OOB sample : $\{\hat{T}_m(\mathbf{x}) : (\mathbf{x}_i, y_i) \notin D_m\}$
 2. Use those trees to classify this observation and make majority vote as the label of this observation :
$$\hat{f}_{oob}(\mathbf{x}_i) = \arg \max_{y \in \mathcal{Y}} \sum_{m=1}^M I(\hat{f}_m(\mathbf{x}_i) = y) I(\mathbf{x}_i \notin D_m)$$
 3. Compute the number of misclassified samples, and take the ratio of this number to the total number of samples as OOB error : $Err_{oob} = \frac{1}{N} \sum_{i=1}^N I(\hat{f}_{oob}(\mathbf{x}_i) \neq y_i)$

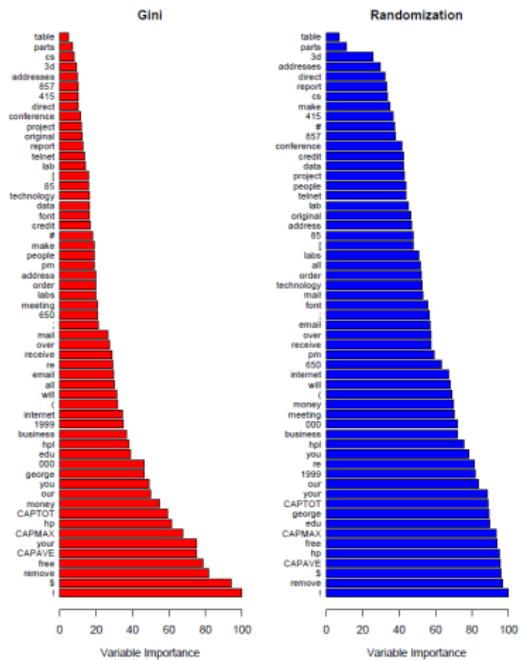
Feature Importance

- Using split criteria

- The improvement in the split-criterion as feature importance
- It is accumulated over all the trees for each variable

- Using OOB randomization

- Randomly permute the values of each feature in the OOB samples, and compute the prediction accuracy
- The decrease in accuracy as a result of this permutation is averaged over all trees as feature importance



Pros and Cons

- Where it is good
 - Bagging or random forest (RF) work for models with high variance but low bias
 - Better for nonlinear estimators
 - RF works for very high-dimensional data, and no need to do feature selection as RF gives the feature importance
 - Easy to do parallel computing
- Disadvantage
 - Overfitting when the samples are large-sized with great noise, or when the dimension of data is low
 - Slow computing performance comparing to single tree
 - Hard to interpret

Outlines

Introduction

Bagging and Random Forest

Boosting and AdaBoost

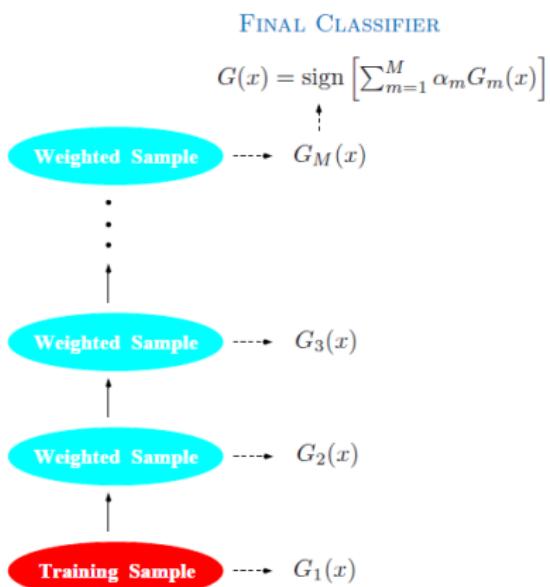
Gradient Boosting Decision Tree

XGBoost

Conclusion and Python Examples

Boosting

- Boosting : combines the outputs of many “weak” classifiers to produce a powerful “committee”
- Weak classifier : error rate < 0.5 (random guessing)
- **Sequentially** apply the weak classifiers to the repeatedly modified data, emphasizing the misclassified samples
- Combine weak classifiers through a weighted majority vote or averaging to produce the final prediction



Boosting Fits an Additive Model

- Additive model : $f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$
- Possible choices for basis function $b(x; \gamma)$:
 - Neural networks : $\sigma(\gamma_0 + \gamma_1^T x)$, where $\sigma(t) = 1/(1 + e^{-t})$
 - Wavelets
 - Cubic spline basis
 - Trees
 - Eigenfunctions in reproducing kernel Hilbert space (RKHS)
- Parameter fitting : $\min_{\{\beta_m, \gamma_m\}} \sum_{i=1}^N L(y_i, \sum_{m=1}^M \beta_m b(x_i; \gamma_m))$
- Loss function : squared error $L(y, f(x)) = (y - f(x))^2$ or likelihood-based loss

Forward Stagewise Additive Modeling

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
 - Output : additive model $f_M(x)$
1. Initialize $f_0(x) = 0$
 2. For $m = 1$ to M :
 - 2.1 Compute $(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
 - 2.2 Update $f_m(x) = f_{m-1}(x) + \beta_m b(x_i; \gamma_m)$
 - Squared error loss : in step 2.1,

$$L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)) = (\underbrace{y_i - f_{m-1}(x_i)}_{\text{residual}} - \beta b(x_i; \gamma))^2$$

Exponential Loss and AdaBoost

- Exponential loss : $L(y, f(x)) = \exp(-yf(x))$
- Classifier as basis function : $b(x; \gamma) = G(x) \in \{-1, 1\}$
- Let $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$, then step 2.1 turns to be :

$$\begin{aligned}
 (\beta_m, G_m) &= \arg \min_{\beta, G} \sum_{i=1}^n w_i^{(m)} \exp(-\beta y_i G(x_i)) \\
 &= \arg \min_{\beta, G} \left[\sum_{y_i \neq G(x_i)} w_i^{(m)} (e^\beta - e^{-\beta}) + e^{-\beta} \sum_{i=1}^n w_i^{(m)} \right]
 \end{aligned}$$

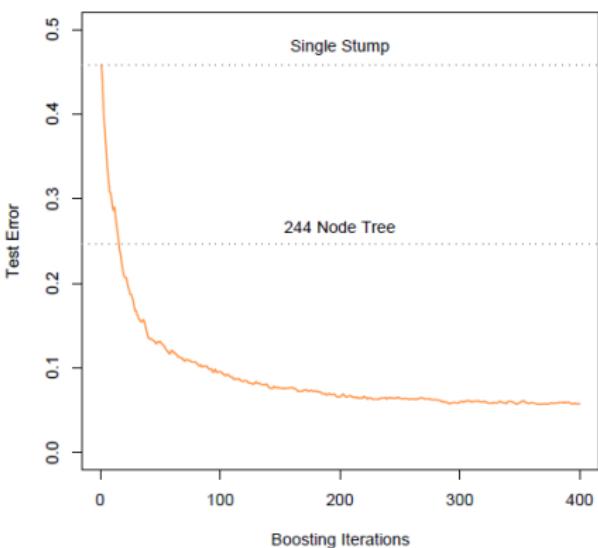
- $G_m = \arg \min_G \sum_{i=1}^n w_i^{(m)} I(y_i \neq G(x_i))$.
- $\beta_m = \arg \min_\beta \left[\epsilon_m (e^\beta - e^{-\beta}) + e^{-\beta} \right] = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$ where
 $\epsilon_m = \left(\sum_{i=1}^n w_i^{(m)} I(y_i \neq G(x_i)) \right) / \sum_{i=1}^n w_i^{(m)}$ is weighted error rate

AdaBoost Algorithm

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, loss function $L(y, f(x))$
 - Output : Weighted classifier $G(x)$
1. Initialize $w_i = 1/N, i = 1, \dots, N$
 2. For $m = 1$ to M :
 - 2.1 Fit a classifier $G_m(x)$ to the training data D with weight $\{w_i\}$
 - 2.2 Compute the error $\epsilon_m = (\sum_{i=1}^n w_i^{(m)} I(y_i \neq G_m(x_i))) / \sum_{i=1}^n w_i^{(m)}$
 - 2.3 Compute $\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}$ ($\alpha_m = 2\beta_m > 1$)
 - 2.4 Update the weight $w_i^{(m+1)} = w_i^{(m)} \exp(\alpha_m I(y_i \neq G_m(x_i))),$ for $i = 1, \dots, N$
 3. Output $G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$

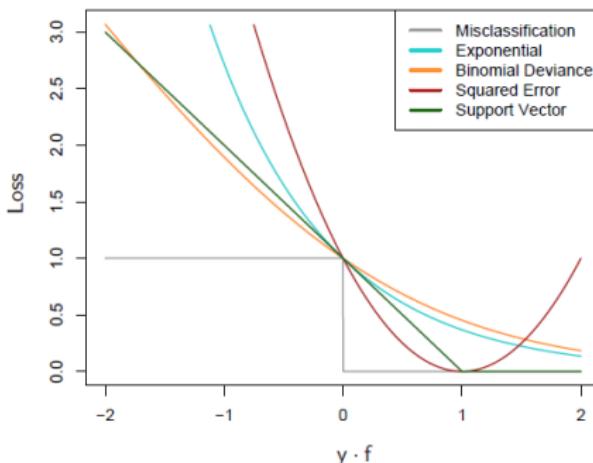
Illustration

- Weights of weak classifiers :
the better the classifier is,
the larger its weight is
- Weights of samples :
Re-weighting after each
step, increase the weights
for misclassified samples
- Simulation : 2-class
classification, 1000 training
samples from each class,
10,000 test samples ;
two-leaf classification tree
(stump) as base learner



Loss Functions

- For classification, exponential loss and binomial negative log-likelihood (deviance) loss $\log(1 + \exp(-2yf))$ share the same population minimizer ; thus it is equivalent to MLE rule
- For classification, squared error loss is not good (not monotonically decreasing) ; the exponential loss is good and binomial deviance is better (less penalty for large $-yf$)



Pros and Cons

- Where it is good
 - AdaBoost improve the classification performance comparing to weak classifiers
 - Many choices for weak classifiers : trees, SVMs, kNNs, etc.
 - Only one tuning parameter M : # of weak classifiers
 - prevent overfitting suffered by single weak classifiers (e.g. complex decision tree)
- Disadvantage
 - Weak interpretability
 - Overfitting when using very bad weak classifiers
 - Sensitive to outliers
 - Not easy for parallel computing

Outlines

Introduction

Bagging and Random Forest

Boosting and AdaBoost

Gradient Boosting Decision Tree

XGBoost

Conclusion and Python Examples

Boosting Tree

- Using classification trees or regression trees as base learners
- $f_M(x) = \sum_{m=1}^M T(x; \Theta_m)$ where $T(x; \Theta) = \sum_{j=1}^J \gamma_j I(x \in R_j)$
- Parameter set $\Theta = \{R_j, \gamma_j\}_{j=1}^J$
- Parameter finding : minimizing the empirical risk

$$\hat{\Theta} = \arg \min_{\Theta} \sum_{j=1}^J \sum_{x_i \in R_j} L(y_i, \gamma_j) \quad (\text{Combinatorial optimization})$$

- Approximate suboptimal solutions :
 1. Finding γ_j given R_j : $\gamma_j = \bar{y}_j = \frac{1}{|R_j|} \sum_{y_i \in R_j} y_i$ for L^2 loss ; and
 $\gamma_j = \text{modal class in } R_j$ for misclassification loss
 2. Finding R_j given γ_j : Difficult, need to estimate γ_j as well ;
greedy, top-down recursive partitioning algorithm

Boosting Tree as Forward Stagewise Algorithm

- $\hat{\Theta}_m = \arg \min_{\Theta_m} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$
 1. $\hat{\gamma}_{jm} = \arg \min_{\gamma_{jm}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm})$
 2. Finding R_{jm} is more difficult than for a single tree in general.
- Squared-error loss : fit a tree to the residual
 $L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)) = \underbrace{(y_i - f_{m-1}(x_i) - T(x_i; \Theta_m))^2}_{\text{residual}}$
- Two-class classification and exponential loss : AdaBoost for trees, $\hat{\Theta}_m = \arg \min_{\Theta_m} \sum_{i=1}^N w_i^{(m)} \exp[-y_i T(x_i; \Theta_m)]$
 1. $\hat{\gamma}_{jm} = \log \frac{\sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i=1)}{\sum_{x_i \in R_{jm}} w_i^{(m)} I(y_i=-1)}$
- Absolute error or the Huber loss : robust but slow

Gradient Descent for General Loss

- Supervised learning is equivalent to the optimization problem

$$\min_f L(f) = \min_f \sum_{i=1}^N L(y_i, f(x_i))$$

- Numerical optimization : $\hat{\mathbf{f}} = \arg \min_{\mathbf{f}} L(\mathbf{f})$ where $\mathbf{f} = \{f(x_1), f(x_2), \dots, f(x_N)\}$,
- Approximate $\hat{\mathbf{f}}$ by $\mathbf{f}_M = \sum_{m=0}^M \mathbf{h}_m$, where $\mathbf{f}_0 = \mathbf{h}_0$ is initial guess
- Gradient descent method : $\mathbf{f}_m = \mathbf{f}_{m-1} - \rho_m \mathbf{g}_m$, where $g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=f_{m-1}(x_i)}$, and $\mathbf{h}_m = -\rho_m \mathbf{g}_m$

Gradient Boosting Decision Tree (GBDT)

- Find a tree $T(x; \Theta_m)$ by minimization problem

$$\tilde{\Theta}_m = \arg \min_{\Theta_m} \sum_{i=1}^N (-g_{im} - T(x_i; \Theta_m))^2$$

In general $\tilde{R}_{jm} \neq R_{jm}$

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$ y_i - f(x_i) $	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $ y_i - f(x_i) \leq \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $ y_i - f(x_i) > \delta_m$ where $\delta_m = \alpha \text{th-quantile}\{ y_i - f(x_i) \}$
Classification	Deviance	k th component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

GBDT Algorithm

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, loss function $L(y, f(x))$
- Output : boosting tree $\hat{f}(x)$

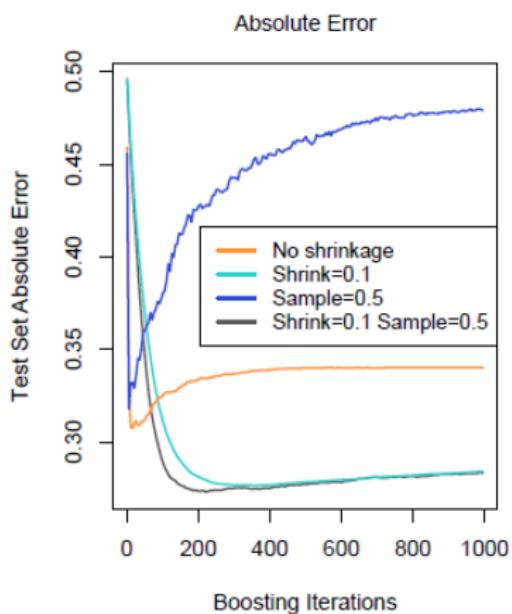
1. Initialize $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$
2. For $m = 1$ to M :
 - 2.1 For $i = 1, 2, \dots, N$ compute $r_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}$
 - 2.2 Fit a regression tree to the target (residual) r_{im} , giving terminal regions R_{jm} , $j = 1, \dots, J_m$
 - 2.3 For $j = 1, \dots, J_m$, compute

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$
 - 2.4 Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x_i \in R_{jm})$
3. $\hat{f}(x) = f_M(x)$

Regularization Techniques

- Shrinkage : the step 2.4 is modified as

$$f_m(x) = f_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x_i \in R_{jm})$$
- Subsampling : at each iteration, sample a fraction η of the training set and grow the next tree using the subsample
- Shrinkage + subsampling : best performance



Feature importance and Partial Dependence Plots

- Feature importance
 - When fitting a single tree T , at each node t , one feature $X_{v(t)}$ and one separate value $X_{v(t)} = c_{v(t)}$ are chosen to improve a certain quantity of criterion (e.g. GINI, entropy, squared error, etc.)
 - Sum all these improvements i_t brought by each feature X_k over all internal nodes : $I_k(T) = \sum_{t=1}^{J-1} i_t I(v(t) = k)$
 - Average the improvements of all trees \Rightarrow importance of that feature : $I_k = \frac{1}{M} \sum_{m=1}^M I_k(T_m)$
- Partial Dependence Plots
 - Partial dependence of $f(X)$ on X_S : $f_S(X_S) = \mathbb{E}_{X_C} f(X_S, X_C)$
 - Estimate by empirical mean : $\bar{f}_S(X_S) = \frac{1}{N} \sum_{i=1}^N f(X_S, x_{iC})$

Pros and Cons

- Where it is good
 - For all regression problems
 - Better for two-class classification, possible for multi-class problems (not suggested)
 - Various nonlinearity, strong representability
- Disadvantage
 - Sequential process, inconvenient for parallel computing
 - High computational complexity, not suitable for high-dimensional problems with sparse features

Outlines

Introduction

Bagging and Random Forest

Boosting and AdaBoost

Gradient Boosting Decision Tree

XGBoost

Conclusion and Python Examples

Introduction

- Developed by Tianqi Chen
(<http://homes.cs.washington.edu/~tqchen/>)
- Distributed gradient boosting : can be parallelized
- Highly efficient
- Good performance
- Out-of-Core Computing for big dataset
- Cache Optimization of data structures and algorithms

Cost Functions

- Cost function :

$$F(\Theta_m) = \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)) + R(\Theta_m), \text{ where}$$

$R(\Theta)$ is regularization term (L^0 , L^1 or L^2 penalties)

- Taylor expansion up to second order :

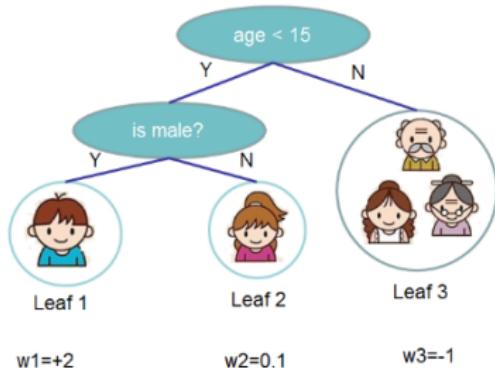
$$F(\Theta_m) \approx \sum_{i=1}^N \left[L(y_i, f_{m-1}(x_i)) + g_i^{(m)} T(x_i; \Theta_m) + \frac{1}{2} h_{ii}^{(m)} T(x_i; \Theta_m)^2 \right] + R(\Theta_m), \text{ where}$$

$g_i^{(m)} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=f_{m-1}(x_i)}$ is the gradient of loss

function, and $h_{ii}^{(m)} = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2} \right]_{f(x_i)=f_{m-1}(x_i)}$ is the diagonal of the Hessian of loss function (off-diagonals are zeros).

Penalties

- Take regression trees as examples : Let J_m be the number of leaf nodes (number of rectangles in the partition), γ_{jm} is the approximate constant (weight w) in the leaf node (region) R_{jm}
- The complexity of tree is the sum of L^0 and L^2 norm of $\{\gamma_{jm}\}$: $R(\Theta_m) = \frac{1}{2} \lambda \sum_{j=1}^{J_m} \gamma_{jm}^2 + \mu J_m$



$$R = \frac{1}{2} \lambda (4 + 0.01 + 1) + 3\mu$$

Optimal solutions

- Reformulation of approximated cost function :

$$\begin{aligned}
 F(\Theta_m) &\approx \sum_{i=1}^N L(y_i, f_{m-1}(x_i)) + \sum_{j=1}^{J_m} \left[\left(\sum_{x_i \in R_{jm}} g_i^{(m)} \right) \gamma_{jm} + \right. \\
 &\quad \left. \frac{1}{2} \left(\sum_{x_i \in R_{jm}} h_{ii}^{(m)} + \lambda \right) \gamma_{jm}^2 \right] + \mu J_m = \\
 &\quad \sum_{j=1}^{J_m} \left[G_j^{(m)} \gamma_{jm} + \frac{1}{2} (H_j^{(m)} + \lambda) \gamma_{jm}^2 \right] + \mu J_m + \text{constant, where}
 \end{aligned}$$

$$G_j^{(m)} = \sum_{x_i \in R_{jm}} g_i^{(m)} \text{ and } H_j^{(m)} = \sum_{x_i \in R_{jm}} h_{ii}^{(m)}$$

- By differentiation w.r.t. γ_{jm} , we have the optimal solution :

$$\hat{\gamma}_{jm} = -\frac{G_j^{(m)}}{H_j^{(m)} + \lambda}$$

- Simplified cost function :

$$F(\Theta_m) = -\frac{1}{2} \sum_{j=1}^{J_m} \frac{(G_j^{(m)})^2}{H_j^{(m)} + \lambda} + \mu J_m + \text{constant}$$

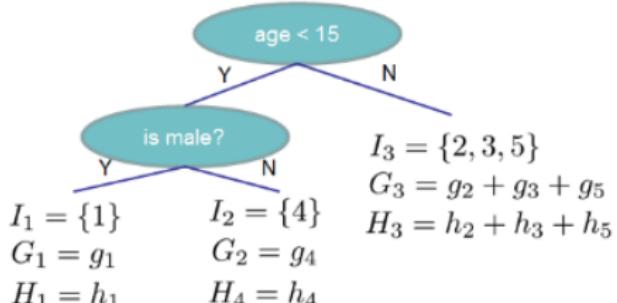
Structure Score

- Neglecting the constant term, we obtain the structure score :

$$SS = -\frac{1}{2} \sum_{j=1}^{J_m} \frac{(G_j^{(m)})^2}{H_j^{(m)} + \lambda} + \mu J_m$$

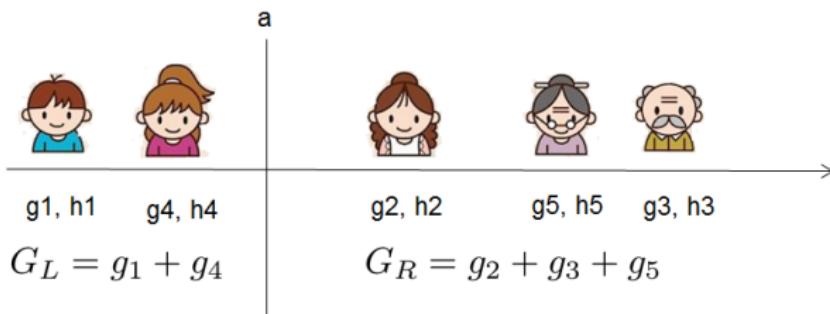
- It is similar to information gain : minimizing the structure score leads to the best tree

样本号	梯度数据
1	g1, h1
2	g2, h2
3	g3, h3
4	g4, h4
5	g5, h5



Node Splitting - Greedy Algorithm

- When splitting a node into left (L) and right (R) child nodes, we are maximizing $Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$
- Enumerate all possible splits at $x < a$ (e.g., age < 15) from left to right



Greedy Algorithm for split finding

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, loss function $L(y, f(x))$, the index set $I = \{i|x_i \in R_{jm}\}$ of current node R_{jm} , feature dimension d
 - Output : best split
1. Initialize $gain = 0$, $G = \sum_{i \in I} g_i$, $H = \sum_{i \in I} h_{ii}$
 2. For $k = 1$ to K :
 - 2.1 $G_L = 0$, $H_L = 0$
 - 2.2 For j in $\text{sorted}(I, \text{ by } x_{jk})$, do
 - 2.2.1 $G_L = G_L + g_j$, $H_L = H_L + h_{jj}$, $G_R = G - G_L$, $H_R = H - H_L$
 - 2.2.2 $score = \max(score, \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda})$
 3. Output split with max score

Loss Functions

- Square loss $L(y, f) = (y - f)^2 :$
 $g_i^{(m)} = 2(f_i - y_i) = 2 \times \text{residue}, h_{ii}^{(m)} = 2$
- Logistic loss $L(y, f) = y \ln(1 + e^{-f}) + (1 - y) \ln(1 + e^f) :$
 $g_i^{(m)} = -y_i \left(1 - \frac{1}{1+e^{-f_{m-1}(x_i)}} + (1 - y_i) \frac{1}{1+e^{-f_{m-1}(x_i)}} \right) =$
 $\text{Pred} - \text{Label}, h_{ii}^{(m)} = \frac{e^{-f_{m-1}(x_i)}}{(1+e^{-f_{m-1}(x_i)})^2} = \text{Pred} \times (1 - \text{Pred})$

Outlines

Introduction

Bagging and Random Forest

Boosting and AdaBoost

Gradient Boosting Decision Tree

XGBoost

Conclusion and Python Examples

Conclusions

- Ensemble methods have integrable abilities of single models, achieving better performance
- Easy to generalize to new data
- When there are strong noises, easy to overfit
- Computationally intensive

Python Examples

- Random forest :

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
# RandomForestClassifier(bootstrap=True, class_weight=None,
#   criterion='gini', max_depth=None, max_features='auto',
#   max_leaf_nodes=None, min_impurity_split=1e-07,
#   min_samples_leaf=1, min_samples_split=2,
#   min_weight_fraction_leaf=0.0, n_estimators=100,
#   n_jobs=1, oob_score=False, random_state=None,
#   verbose=0, warm_start=False)
# Feature importance in random forest
feature_imp = pd.Series(rf.feature_importances_)
rf.fit(X_train, Y_train)
Y_predict_rf = rf.predict(X_test)
oob_error = 1 - rf2.oob_score_
```

- AdaBoost :

```
from sklearn.ensemble import AdaBoostClassifier
adaboost = AdaBoostClassifier(n_estimators = 50)
adaboost.fit(X_train, Y_train)
adaboost.staged_predict(X_train)
Y_predict_ada = adaboost.predict(X_test)
```

References

- 数据分析导论，博雅大数据学院
- 周志华，机器学习，2016
- T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning : Data mining, Inference, and Prediction*, 2nd Edition, 2009

Introduction to Big Data Analysis Clustering Analysis

Zhen Zhang

Southern University of Science and Technology

Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

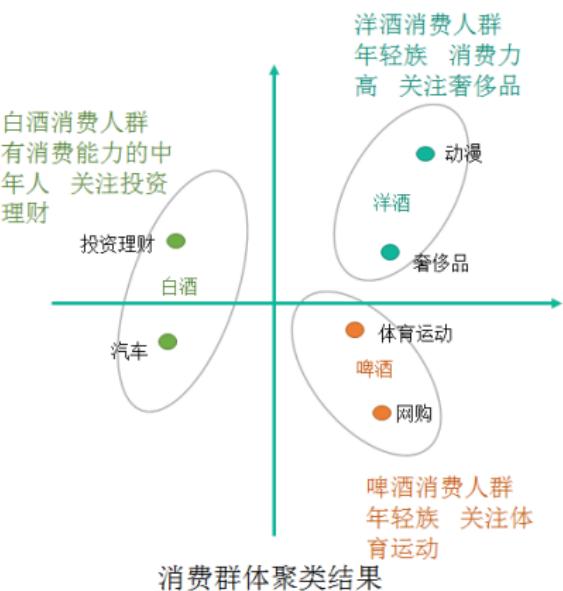
Spectral Clustering

Model Assessment

Case Study

Clustering

- Also called data segmentation, group a collection of objects into subsets or “clusters”
- Results : objects in each cluster are more similar to one another than objects in different clusters.
- Example : applications in consumption analysis
- Can be used in data preprocessing

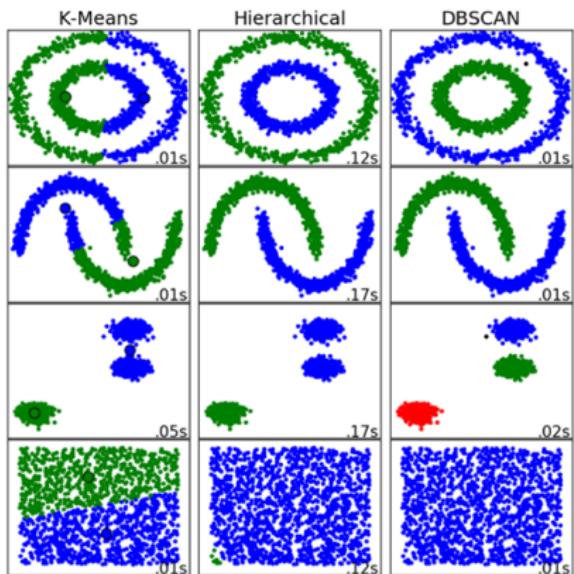


Concepts in Clustering

- Different from classification : it is unsupervised learning ; no outputs or labels
- Central goal : Optimize the similarity (or dissimilarity) between the individual objects being clustered :
 - Obtain great similarity of samples within cluster
 - Obtain small similarity of samples between clusters
- Cost functions : not related to the outputs, but related to the similarity
- Two kinds of input data :
 - $n \times n$ similarity (dissimilarity) matrix D : only depends on the distances between pairs of samples ; may lose some information on data
 - Original data with features $X \in \mathbb{R}^{n \times d}$

Clustering Methods

- Clustering process :
 - data preprocessing, especially standadization
 - Similarity matrix
 - Clustering Methods
 - Determine the best number of clusters
- Clustering methods :
 - Partitional clustering :
 - K-means
 - K-Medoids
 - Spectral clustering
 - DBSCAN
 - Hierarchical clustering



Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

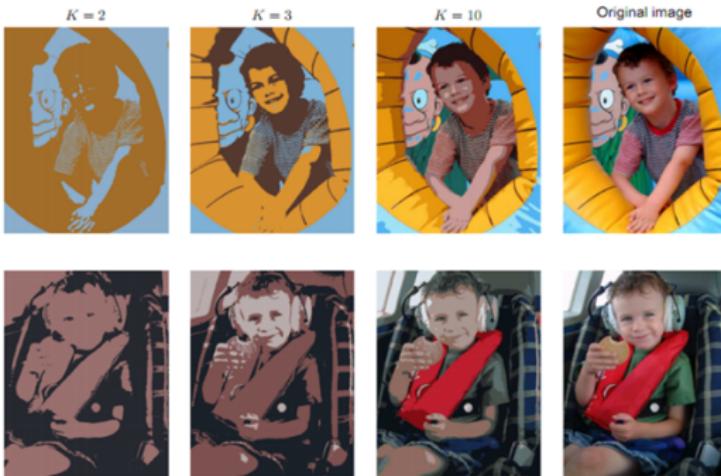
Spectral Clustering

Model Assessment

Case Study

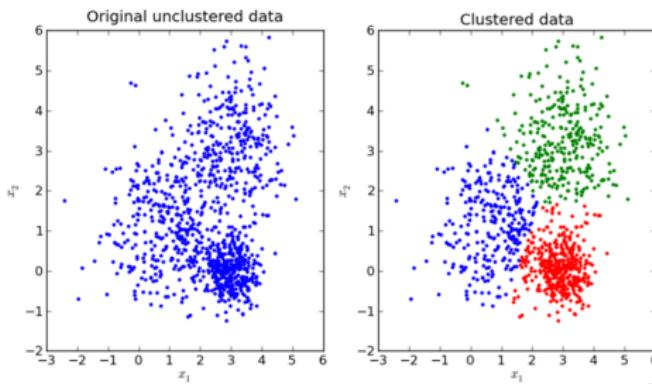
Introduction

- K-means clustering originates from signal processing, it is quite popular in image processing (segmentation)
- Group n samples to k clusters, making each sample belong to the nearest cluster
- In an image, each pixel is a sample



Idea

- Data set $\{\mathbf{x}_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$
- Representatives : Mass center of k th-cluster C_k is c_k ,
 $k = 1, \dots, K$
- Sample x_i belongs to cluster k if $d(x_i, c_k) < d(x_i, c_m)$ for
 $m \neq k$, where $d(x_i, x_j)$ is dissimilarity function
- Make the mass centers well-located so that the average
distance between each sample to its cluster center is as small
as possible



Optimization Problem

- Let $C : \{1, \dots, n\} \rightarrow \{1, \dots, k\}$ be the assignment from the data indices to the cluster indices. $C(i) = k$ means $x_i \in C_k$

- Total point scatter : $T = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n d(x_i, x_j) =$

$$\frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \left(\sum_{C(j)=k} d_{ij} + \sum_{C(j) \neq k} d_{ij} \right) = W(C) + B(C)$$

- Loss function : within-cluster point scatter

$$W(C) = \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(j)=k} d_{ij}; \text{ between-cluster point scatter}$$

$$B(C) = \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(j) \neq k} d_{ij}$$

- Minimize $W(C)$ is equivalent to maximize $B(C)$

Dissimilarities

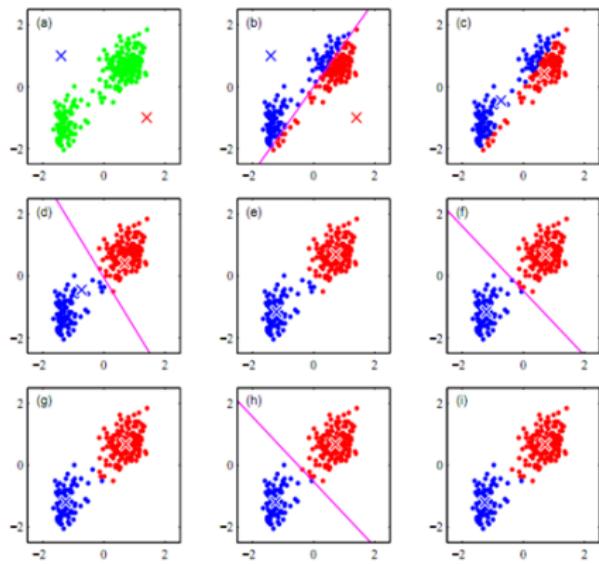
- Proximity matrices : $n \times n$ symmetric matrix D with nonnegative entries and zero diagonal elements provides information about dissimilarity between a pair of samples, this is not distance in general
- Dissimilarities based on attributes :
 - $d(x_i, x_j) = \sum_{k=1}^p d_k(x_{ik}, x_{jk})$; d_k can be squared distance
 - $d_k(x_{ik}, x_{jk}) = (x_{ik} - x_{jk})^2$, absolute distance
 - $d_k(x_{ik}, x_{jk}) = |x_{ik} - x_{jk}|$
- Weighted average : $d(x_i, x_j) = \sum_{k=1}^p w_k d_k(x_{ik}, x_{jk})$ where $\sum_{k=1}^p w_k = 1$; setting $w_k \sim 1/d_k$ with $\bar{d}_k = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n d_k(x_{ik}, x_{jk}) = \widehat{\text{Var}}(X_k)$ will assign equal influence to all features
- Dissimilarities based on correlation : $d(x_i, x_j) \propto 1 - \rho(x_i, x_j)$

K-Means (as Central Voronoi Tessellation)

- Minimizing $W(C)$ is in general infeasible since this is a greedy algorithm that only works for small data sets
- Taking squared dissimilarity, $W(C) = \sum_{k=1}^K n_k \sum_{C(i)=k} \|x_i - \bar{x}_k\|^2$,
 where $n_k = \sum_{i=1}^n I(C(i) = k)$ is the number of samples in cluster k , $\bar{x}_k = \frac{1}{n_k} \sum_{C(j)=k} x_j = \arg \min_{m_k} \sum_{C(j)=k} \|x_j - m_k\|^2$
- $\min_C W(C) \iff \min_{C, m_k} \sum_{k=1}^K n_k \sum_{C(i)=k} \|x_i - m_k\|^2$
- Alternating minimization :
 - Given C , solve for $m_k \implies m_k^* = \bar{x}_k$ (choose representatives)
 - Given m_k , solve for $C \implies C(i) = \arg \min_{1 \leq k \leq K} \|x_i - m_k\|^2$
 (partitioning, equivalent to Voronoi tessellation for given center m_k)

K-Means Iterations

- The alternating iterations can stop when the mass centers $\{\bar{x}_k\}_{k=1}^K$ do not change
 - Initial guess :
 - Random guess, try the best one with smallest $W(C)$
 - Base on other clustering methods (e.g., hierarchical clustering), choose the cluster centers as initial guess



How to choose K

- Minimizing Bayesian Information Criterion (BIC) :
 $BIC(\mathcal{M}|\mathbf{X}) = -2 \log \Pr(\mathbf{X}|\hat{\Theta}, \mathcal{M}) + p \log(n)$, where \mathcal{M} indicates the model, $\hat{\Theta}$ is the MLE of the model parameters in \mathcal{M} , $\Pr(\mathbf{X}|\mathcal{M})$ is the likelihood function, and p is the number of parameters in model \mathcal{M} ; trade-off between log-likelihood and model complexity
- Based on Minimum Description Length (MDL) : starting from large K , decreases K until the description length
 $-\log \Pr(\mathbf{X}|\hat{\Theta}, \mathcal{M}) - \log \Pr(\Theta|\mathcal{M})$ achieves its minimum (similar to MAP)
- Based on Gaussian distribution assumption : starting from $K = 1$, increases K until the points in every cluster follow Gaussian distribution

Pros and Cons

- Where it is good
 - Intuitive, easy to implement
 - Low computational complexity, $O(tnpK)$, where t is the number of iterations
- Disadvantage
 - Need to specify K first (K is tuning parameter)
 - Strong dependence on the initial guess of cluster center
 - Easy to stuck at local minimum
 - Naturally assume ball-shaped data, hard to deal with data which are not ball-shaped
 - Sensitive to outliers

Variant : Bisecting K-means

- Invented to deal with initial guess of center selection
- Idea : sequentially divide the poorest cluster into two sub-clusters
 1. Initially gather all data into one cluster
 2. Repeat :
 - 2.1 Select the cluster k that maximizes the within-cluster point scatter $\sum_{C(i)=k} \sum_{C(j)=k} \|x_i - x_j\|^2$
 - 2.2 Use 2-means to divide cluster k into two sub-clusters, with random initial guess of two centers
 - 2.3 Repeat step 2.2 p times, choose the best pair of clusters that minimizes the within-cluster point scatter
 3. Stop when there are K clusters (Or you can stop any time you like to have a satisfactory clustering result)

Variant : K-medoids

- Invented to overcome the influence of outliers
- Can deal with data of general type, assuming general dissimilarity $d(x_i, x_j)$
- Idea : centers for each cluster are restricted to be one of the observations assigned to that cluster
- Alternating minimization :
 1. Given C , solve for $m_k = x_{i_k^*}$ that minimizes the within-cluster point scatter : $i_k^* = \arg \min_{\{i: C(i)=k\}} \sum_{C(j)=k} d(x_i, x_j)$ (choose the real samples as representatives)
 2. Given m_k , solve for $C \implies C(i) = \arg \min_{1 \leq k \leq K} d(x_i, m_k)$
- More robust than K-means
- More computational effort when solving for the center in step 1 : $O(n_k^2)$ comparing to $O(n_k)$ in K-means

Other Variants

- K-medians : use Manhattan distance (L^1 -distance) instead in K-means ; then the centers are not means, but medians
- K-means++ : designed to select good initial centers that are far away from each other
- Rough-set-based K-means : each sample could be assigned to more than one cluster

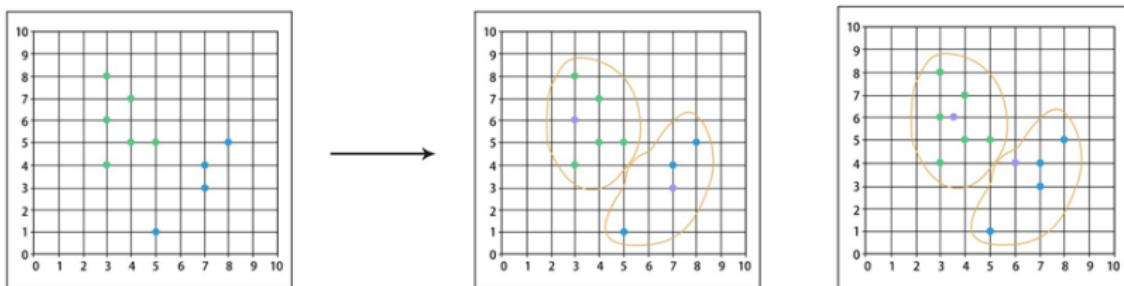


FIGURE: K-medoids

Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

Spectral Clustering

Model Assessment

Case Study

Hierarchical Clustering

- Clustering in different hierarchies, generating tree structure
- Two approaches :
 - Agglomerate clustering : bottom-up
 - Divisive clustering : top-down
- Limitation : once merged or divided, the operation cannot be modified

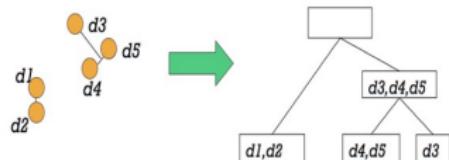


FIGURE: Agglomerate clustering



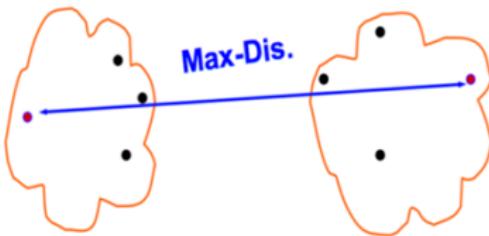
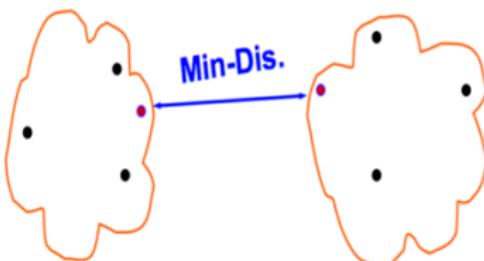
FIGURE: Divisive clustering

Agglomerate Clustering

- Given n samples and proximity matrix, do the following steps :
 1. Let every observation represent a singleton cluster
 2. Merge the two closest clusters into one single cluster
 3. Calculate the new proximity matrix (dissimilarity between two clusters)
 4. Repeat step 2 and 3, until all samples are merged into one cluster
- Three methods for computing intergroup dissimilarity :
 - Single linkage (SL)
 - Complete linkage (CL)
 - Average linkage (AL)

Intergroup Dissimilarity

- Single linkage : Greatest similarity or least dissimilarity $d_{SL}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$
- Complete linkage : Least similarity or greatest dissimilarity $d_{SL}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$
- Average linkage : Average similarity or dissimilarity $d_{AL}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$



Generalized Agglomerative Scheme

- Input : training set $D = \{(x_1), \dots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$
 - Output : A dendrogram containing $\{\mathcal{R}_t\}_{t=0}^{n-1}$, where \mathcal{R}_t is the clustering result at time t
1. Initialize the clustering result $\mathcal{R}_0 = \{\{x_1\}, \{x_2\}, \dots, \{x_n\}\}$, $t = 0$
 2. Do iterations :
 - 2.1 $t = t + 1$
 - 2.2 Choose (C_i, C_j) from \mathcal{R}_{t-1} so that $d(C_i, C_j) = \min_{(r,s)} d(C_r, C_s)$
 - 2.3 $C_q = C_i \cup C_j$
 - 2.4 $\mathcal{R}_t = (\mathcal{R}_{t-1} \setminus \{C_i, C_j\}) \cup \{C_q\}$
 3. Stop at $t = n - 1$ when $|\mathcal{R}_{n-1}| = 1$, return $\{\mathcal{R}_t\}_{t=0}^{n-1}$

Generalized Divisive Scheme

- Input : training set $D = \{(x_1), \dots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$
 - Output : A dendrogram containing $\{\mathcal{R}_t\}_{t=0}^{n-1}$, where $\mathcal{R}_t = \{C_{t,i}\}_{i=1}^{t+1}$ is the clustering result at time t
1. Initialize $\mathcal{R}_0 = \{X\}$, $t = 0$
 2. Do iterations :
 - 2.1 $t = t + 1$
 - 2.2 For $i = 1$ to t , do :
 - 2.2.1 Choose $(C_{t-1,i}^1, C_{t-1,i}^2)$ from $C_{t-1,i}$ so that

$$d(C_{t-1,i}^1, C_{t-1,i}^2) = \max_{G \cup H = C_{t-1,i}} d(G, H)$$
 - 2.2.2 Choose i_{t-1} so that $i_{t-1} = \arg \max_i d(C_{t-1,i}^1, C_{t-1,i}^2)$
 - 2.2.3 $\mathcal{R}_t = (\mathcal{R}_{t-1} \setminus \{C_{t-1,i_{t-1}}\}) \cup \{C_{t-1,i_{t-1}}^1, C_{t-1,i_{t-1}}^2\}$
 3. Stop at $t = n - 1$ when $|\mathcal{R}_{n-1}| = n$, return $\{\mathcal{R}_t\}_{t=0}^{n-1}$

Pros and Cons

- Where it is good
 - Hierarchical clustering computes tree structure of the whole clustering process in one stroke
 - SL and CL are sensitive to outliers, while AL gives a compromise
 - As $n \rightarrow \infty$, $d_{AL}(C_i, C_j) \rightarrow \int \int d(x, y)p_i(x)p_j(y)dxdy$, the expected dissimilarity w.r.t. the two densities $p_i(x)$ and $p_j(x)$
 - In contrast, $d_{SL}(C_i, C_j) \rightarrow 0$ and $d_{CL}(C_i, C_j) \rightarrow \infty$ independent of $p_i(x)$ and $p_j(x)$
- Disadvantage
 - Computationally intensive
 - Once a sample is incorrectly grouped into a branch, it will stay in the clusters corresponding to that branch no matter how you threshold the tree

Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

Spectral Clustering

Model Assessment

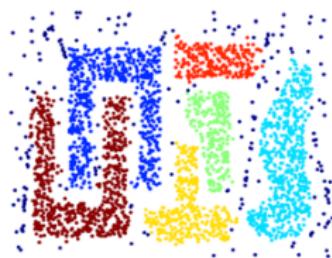
Case Study

Density-based Clustering

- Limitations of hierarchical clustering and K-means clustering : tend to discover convex clusters
- Density-based Clustering : looks for high-density regions separated by low-density regions, could discover clusters of any shape
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)



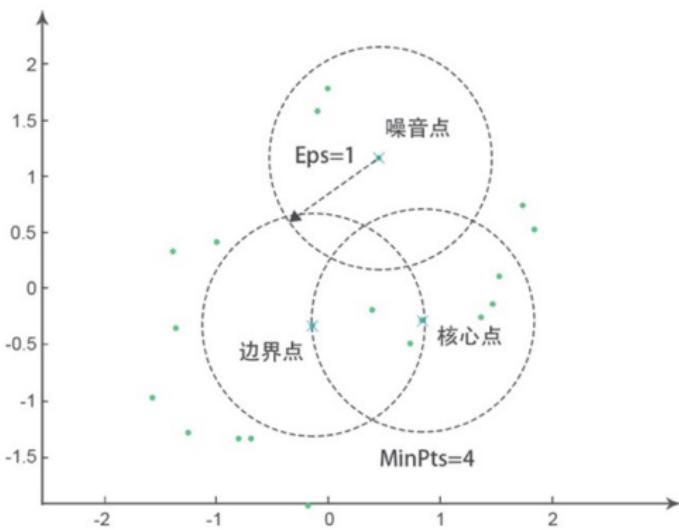
Original Points



Clusters

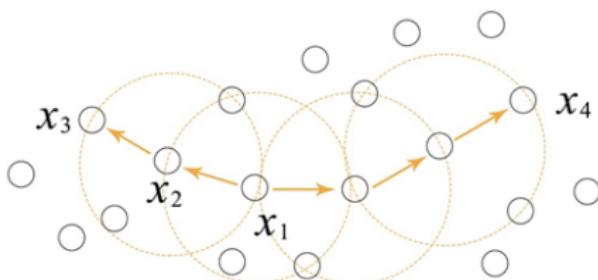
Concepts

- Three types of points :
 - Core point : # of samples in its ϵ -neighborhood $\geqslant \text{MinPts}$
 - Boundary point : it lies in the ϵ -neighborhood of some core point, # of samples in its ϵ -neighborhood $< \text{MinPts}$
 - Noise point : neither core point nor boundary point, it lies in the sparse region



Concepts

- ϵ -neighborhood : for each sample $x_i \in D$,
 $N_\epsilon(x_i) = \{x_j \in D | d(x_i, x_j) \leq \epsilon\}$
- Directly density-reachable : if the sample $x_j \in N_\epsilon(x_i)$, and x_i is core point, then x_j is directly density-reachable from x_i ;
- Density-reachable : for x_i and x_j , if there exist p_1, \dots, p_m , s.t. $p_1 = x_i$, $p_m = x_j$, and p_{k+1} is directly density-reachable from p_k , then x_j is density-reachable from x_i
- Density-connected : if there exists p , s.t. both x_i and x_j are density-reachable from p , then x_i and x_j are density-connected

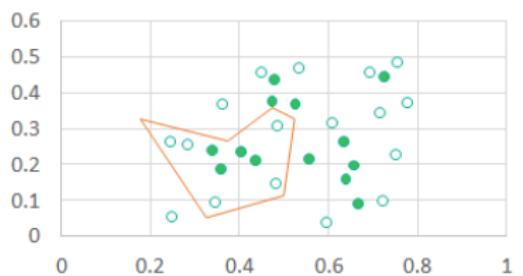


DBSCAN Algorithm

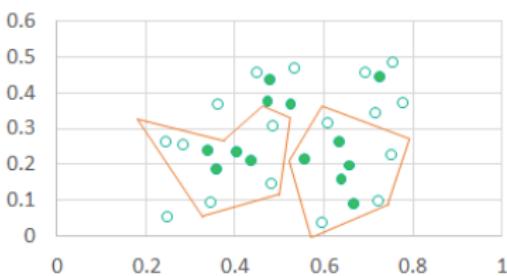
- Input : training set $D = \{(x_1), \dots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$, parameters $MinPts, \epsilon$
 - Output : a set of clusters $\{C_t\}$
1. Mark all samples in D as non-processed
 2. For each sample $p \in D$, do :
 - 2.1 If p has been grouped into some cluster or marked as noise point, go to check next sample
 - 2.2 Else, if $|N_\epsilon(p)| < MinPts$, then mark p as boundary point or noise point
 - 2.3 Else, mark p as core point, construct cluster $C = N_\epsilon(p)$. For each $q \in N_\epsilon(p)$, do :
 - 2.3.1 If $|N_\epsilon(q)| \geq MinPts$, then put all un-clustered points in $N_\epsilon(q)$ into C
 3. Stop when all samples in D have been clustered

Examples ($\epsilon = 0.11$, $MinPts = 5$)

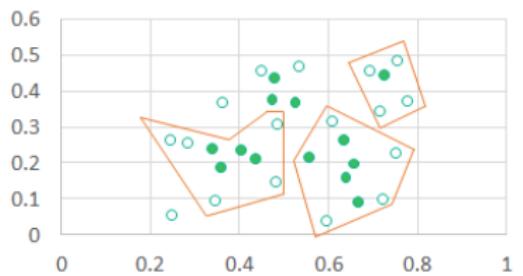
C1



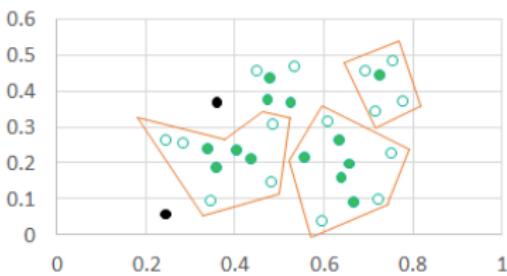
C2



C3



C4



DBSCAN vs. K-means

DBSCAN

- The clustering result is not a complete partition of original dataset (noise points are excluded)
- Could deal with clusters with any shape and size
- Could deal with noise points and outliers
- The definition of density must be meaningful
- Not efficient when dealing with high-dimensional data
- No implicit assumptions on the sample distribution

K-Means

- The clustering result is a complete partition of original dataset
- The clusters are nearly ball-shaped
- Sensitive to outliers
- The definition of cluster centers must be meaningful
- Efficient to deal with high-dimensional data
- The samples implicitly follow the Gaussian distribution assumption

Pros and Cons

- Computational complexity $O(n \times T)$, where t is the time for searching ϵ -neighborhood ; in the worst case, $O(n^2)$
- In low-dimensional space, could be improved as $O(n \log n)$ by KD-tree
- Where it is good
 - Fast for clustering
 - Better to deal with noise points
 - Effective for clusters of any shape
- Disadvantage
 - Need large memory
 - Bad performance when the density is not well-distributed and the between-cluster distances are large

Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

Spectral Clustering

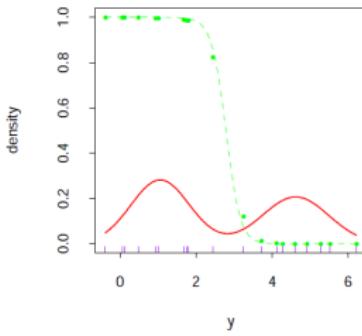
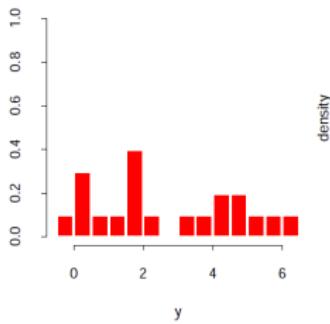
Model Assessment

Case Study

Gaussian Mixture Models

- We want to estimate the density of given data set. This is an unsupervised learning problem.
- Commonly used approach is the parametric estimation, such as maximum likelihood estimate (MLE).
- Consider the following set of data points :

-0.39	0.12	0.94	1.67	1.76	2.44	3.72	4.28	4.92	5.53
0.06	0.48	1.01	1.68	1.80	3.25	4.12	4.60	5.28	6.22



Latent Variables

- A single Gaussian family would not be appropriate. A mixture of two Gaussian distributions seems good.

$$Z_1 \sim N(\mu_1, \sigma_1^2), \quad Z_2 \sim N(\mu_2, \sigma_2^2), \quad Z = (1 - Y)Z_1 + YZ_2,$$

where $Y \in \{0, 1\}$ with $P(Y = 1) = c$.

- In general, for mixture of K Gaussian distributions, we assume there is a latent variable Y indicating which distribution the data \mathbf{x} is sampled from, i.e., $P(Y = y) = c_y$ with $y \in \{1, \dots, K\}$. Given $Y = y$, the random variable X follows the conditional distribution :

$$P(X = \mathbf{x} | Y = y) = \frac{1}{(2\pi)^{d/2}(\Sigma_y)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}_y^{-1} (\mathbf{x} - \boldsymbol{\mu}_y)\right).$$

- The density of X is then

$$\begin{aligned} P(X = \mathbf{x}) &= \sum_{y=1}^K P(Y = y) P(X = \mathbf{x} | Y = y) \\ &= \sum_{y=1}^K c_y \frac{1}{(2\pi)^{d/2}(\Sigma_y)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}_y^{-1} (\mathbf{x} - \boldsymbol{\mu}_y)\right). \end{aligned}$$

MLE of Gaussian Mixture

- Let $\theta = (c_y, \mu_y, \Sigma_y)_{y=1}^K$. Then the log-likelihood of the sample set $S = \{\mathbf{x}_i\}_{i=1}^n$ is

$$L(\theta) = \sum_{i=1}^n \log P_\theta(X = \mathbf{x}_i) = \sum_{i=1}^n \log \left(\sum_{y=1}^K P_\theta(X = \mathbf{x}_i, Y = y) \right).$$

- MLE : $\theta = \arg \max_{\theta} L(\theta)$ is hard due to the summation inside the log.
- Make a simple assumption : we have known the value of the latent variable for each sample \mathbf{x}_i , i.e., $Y_i \in \{1, \dots, K\}$ is known, then the choice from the Y_i -th Gaussian becomes deterministic, and the log-likelihood is replaced by

$$\begin{aligned}\tilde{L}(\theta) &= \sum_{i=1}^n \log \left(\sum_{y=1}^K I_{(Y_i=y)} P_\theta(X = \mathbf{x}_i, Y = y) \right) \\ &= \sum_{i=1}^n \sum_{y=1}^K I_{(Y_i=y)} \log P_\theta(X = \mathbf{x}_i, Y = y).\end{aligned}$$

Expectation-Maximization (EM) Algorithm (Dempster, Laird, and Rubin, 1977')

- But Y_i is indeed random, so that the event $Y_i = y$ happens with a probability $Q_{i,y}$ with $\sum_{y=1}^k Q_{i,y} = 1$.
- Consider the modified objective function defined over $Q = (Q_{i,y})_{i=1,\dots,n; y=1,\dots,K}$ and θ

$$F(Q, \theta) = \sum_{i=1}^n \sum_{y=1}^K Q_{i,y} \log \left(P_\theta(X = \mathbf{x}_i, Y = y) \right).$$

- The optimization problem $(Q, \theta) = \arg \max_{Q, \theta} F(Q, \theta)$ can be solved alternately :
 1. E-Step : Given $\theta^{(m)}$, solve for $Q^{(m+1)} = E_{\theta^{(m)}}(I_{(Y=y)} | X = \mathbf{x}_i) = P_{\theta^{(m)}}(Y = y | X = \mathbf{x}_i);$
 2. M-Step : Given $Q^{(m+1)}$, solve for $\theta^{(m+1)} = \arg \max_{\theta} F(Q^{(m+1)}, \theta)$ (assume this is tractable).
- Initial values of $Q^{(0)}$ and $\theta^{(0)}$ are chosen randomly.
- Terminate until satisfactory (not always converge to the maximum, but guaranteed convergent).

Illustrative Example (Algorithm)

EM Algorithm for two-component Gaussian Mixture :

1. Take initial guesses for the parameters $\hat{\theta}_1 = (\hat{\mu}_1, \hat{\sigma}_1^2)$, $\hat{\theta}_2 = (\hat{\mu}_2, \hat{\sigma}_2^2)$, \hat{c} ;
2. E-Step : $\hat{q}_i = P(Y_i = 1 | Z = z_i, \hat{\theta}_1, \hat{\theta}_2) = \frac{\hat{c}\phi_{\hat{\theta}_2}(z_i)}{(1-\hat{c})\phi_{\hat{\theta}_1}(z_i) + \hat{c}\phi_{\hat{\theta}_2}(z_i)}$, for $i = 1, 2, \dots, n$;
3. M-Step : Compute the weighted means and variances :

$$\begin{aligned}\hat{\mu}_1 &= \frac{\sum_{i=1}^n (1 - \hat{q}_i) z_i}{\sum_{i=1}^n (1 - \hat{q}_i)} & \hat{\sigma}_1^2 &= \frac{\sum_{i=1}^n (1 - \hat{q}_i)(z_i - \hat{\mu}_1)^2}{\sum_{i=1}^n (1 - \hat{q}_i)} \\ \hat{\mu}_2 &= \frac{\sum_{i=1}^n \hat{q}_i z_i}{\sum_{i=1}^n \hat{q}_i} & \hat{\sigma}_2^2 &= \frac{\sum_{i=1}^n \hat{q}_i(z_i - \hat{\mu}_2)^2}{\sum_{i=1}^n \hat{q}_i}\end{aligned}$$

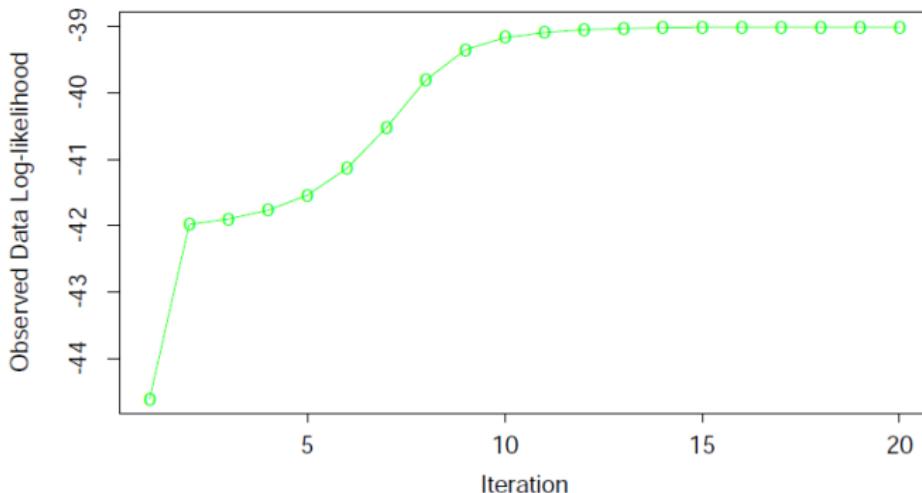
and the mixing probability $\hat{c} = \frac{1}{n} \sum_{i=1}^n \hat{q}_i$;

4. Iterate between E-Step and M-Step until convergence.

Illustrative Example (Result)

Iteration	1	5	10	15	20
\hat{c}	0.485	0.493	0.523	0.544	0.546

The final MLEs are $\hat{\mu}_1 = 4.62$, $\hat{\sigma}_1^2 = 0.87$, $\hat{\mu}_2 = 1.06$, $\hat{\sigma}_2^2 = 0.77$, $\hat{c} = 0.546$.



EM as Maximization-Maximization

- Introduce entropies as penalty to the modified objective function :

$$G(Q, \theta) = F(Q, \theta) - \sum_{i=1}^n \sum_{y=1}^K Q_{i,y} \log Q_{i,y},$$

where $Q \in \mathbb{Q} = \{Q \in [0, 1]^{n,K} : \sum_{y=1}^K Q_{i,y} = 1, \forall i\}$

- M-Step is equivalent to : $\theta^{(m+1)} = \arg \max_{\theta} G(Q^{(m+1)}, \theta)$
- E-Step is equivalent to : $Q^{(m+1)} = \arg \max_{Q \in \mathbb{Q}} G(Q, \theta^{(m)})$ (Exercise as conditional maximization) : by Jensen's inequality,

$$\begin{aligned} G(Q, \theta^{(m)}) &= \sum_{i=1}^n \left(\sum_{y=1}^K Q_{i,y} \log \frac{P_{\theta^{(m)}}(X = \mathbf{x}_i, Y = y)}{Q_{i,y}} \right) \\ &\leq \sum_{i=1}^n \log \left(\sum_{y=1}^K Q_{i,y} \frac{P_{\theta^{(m)}}(X = \mathbf{x}_i, Y = y)}{Q_{i,y}} \right) = L(\theta^{(m)}) \end{aligned}$$

where “=” iff $\frac{P_{\theta^{(m)}}(X = \mathbf{x}_i, Y = y)}{Q_{i,y}} = C, \forall i, y \Leftrightarrow Q_{i,y} = P_{\theta^{(m)}}(Y = y | X = \mathbf{x}_i)$.

- Monotonicity : $L(\theta^{(m+1)}) \geq L(\theta^{(m)})$

EM for Gaussian Mixture as Soft K-Means

- For simplicity, assume $\Sigma_y = I$ for any y .
- E-Step (Partition-Step in K-Means) : $P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i) = \frac{1}{Z_i} P_{\theta^{(m)}}(Y = y) P_{\theta^{(m)}}(X = \mathbf{x}_i|Y = y) = \frac{1}{Z_i} c_y^{(m)} \exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mu_y^{(m)}\|^2\right)$, where Z_i is a normalization factor.
- M-Step : $\max_{c_y, \mu_y} \sum_{i=1}^n \sum_{y=1}^K P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i) \left(\log c_y - \frac{1}{2}\|\mathbf{x}_i - \mu_y\|^2 \right)$
leads to

$$\mu_y = \sum_{i=1}^n P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i) \mathbf{x}_i \quad (\text{Mean-Step in K-Means})$$

$$c_y = \frac{\sum_{i=1}^n P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i)}{\sum_{y'=1}^K \sum_{i=1}^n P_{\theta^{(m)}}(Y = y'|X = \mathbf{x}_i)} \quad (\text{for partition in next step})$$

- “Soft” because the partition is done in probabilistic sense instead of deterministic sense and the average is weighted according to the probability.

Summary of EM Algorithm

- EM is unsupervised learning, an approach to perform MLE for mixture models with latent variables
- EM is an alternating optimization
- EM can be viewed as soft K-Means
- EM can deal with problems including missing data (treat missing data as latent variables and use Bayes formula, see Section 8.5.2 in the book "Elements of Statistical Learning" for general EM algorithm).
- EM can be used in the framework of Bayesian reasoning (MAP), e.g., Variational Bayesian EM algorithm
- EM is related to generative model, e.g., EM for Gaussian mixture is a population approach to learning the sample distributions, analogous to Gibbs sampling which is sampling approach to learning the distribution.
- EM is a general methodology, even used in natural language processing (e.g., latent dirichlet allocation), deep learning (e.g., restricted Boltzmann machine, deep belief network)

Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

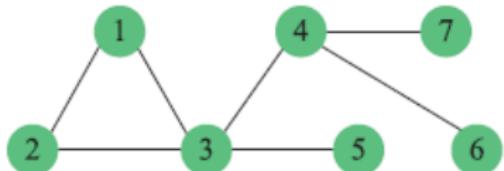
Spectral Clustering

Model Assessment

Case Study

Graphs

- A set of data points $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, similarity s_{ij} or distance d_{ij}
- Graph $G = (V, E)$, where $V = \{v_i\}_{i=1}^n$ with each v_i representing a sample \mathbf{x}_i
- v_i and v_j are connected ($w_{ij} > 0$) if $s_{ij} > \epsilon$ where $\epsilon \geq 0$ is a threshold ; then the edge is weighted by $w_{ij} = s_{ij}$
- Undirected graph $w_{ij} = w_{ji}$, adjacency matrix $W = \{w_{ij}\}$
- Degree of v_i : $d_i = \sum_{j=1}^n w_{ij}$; $D = \text{diag}(d_1, \dots, d_n)$



$$W = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Similarity Graphs

- ϵ -neighborhood graph : v_i and v_j are connected if $d(x_i, x_j) < \epsilon$; unweighted graph ; $\epsilon \sim (\log n / n)^p$; difficult to choose ϵ for data on different scales
- k-nearest neighbor graph : connect v_i to v_j if v_j is among the k-nearest neighbors of v_i , directed graph ; connect v_i and v_j if v_i and v_j are among the k-nearest neighbors of each other, mutual k-nearest neighbor graph, undirected ; $k \sim \log n$
- Fully connected graph : connect all points with positive similarity with each other ; model local neighborhood relationships ; Gaussian similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / (2\sigma^2))$, where σ controls the width of neighborhoods ; adjacency matrix is not sparse ; $\sigma \sim \epsilon$

Graph Laplacian

- Unnormalized graph Laplacian : $L = D - W$
 - Has $\mathbf{1}$ as an eigenvector corresponding to the eigenvalue 0
 - Symmetric and positive definite : $\mathbf{f}^T \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2$
 - Non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$
 - The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}$, where A_1, \dots, A_k are k connected components in the graph
- Normalized graph Laplacians :
 - Symmetric Laplacian : $L_{sym} = D^{-1/2} L D^{-1/2}$
 - Random walk Laplacian : $L_{rw} = D^{-1} L$
 - Both have similar properties as L

Spectral Clustering

- Graph cut : segment G into K clusters A_1, \dots, A_K , where $A_i \subset V$, this is equivalent to minimize the graph cut function

$$cut(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K W(A_k, \bar{A}_k)$$

where $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$. Trivial solution consists of a singleton and its complement

- RatioCut : $RatioCut(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K \frac{W(A_k, \bar{A}_k)}{|A_k|}$, where $|A|$ is the number of vertices in A
- Normalized cut : $Ncut(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K \frac{W(A_k, \bar{A}_k)}{vol(A_k)}$, where $vol(A) = \sum_{i \in A} d_i$; it is NP-hard

Relaxation of RatioCut to Eigenvalue Problems with $K = 2$

- $\min_{A \subset V} \text{RatioCut}(A, \bar{A})$
- Binary vector $f = (f_1, \dots, f_n)^T$ as indicator function :

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|}, & \text{if } v_i \in A \\ -\sqrt{|A|/\bar{A}|}, & \text{if } v_i \in \bar{A} \end{cases}$$
- $f^T L f = |V| \cdot \text{RatioCut}(A, \bar{A})$, $\sum_{i=1}^n f_i = 0$, and $\|f\|_2^2 = n$
- Relax f to be real-valued : $\min_{f \in \mathbb{R}^n} f^T L f$, subject to $f \perp \mathbf{1}$ and $\|f\|_2 = \sqrt{n}$
- By Rayleigh-Ritz theorem, the solution f is the eigenvector corresponding to the second smallest eigenvalue of L
- Cluster $\{f_i\}_{i=1}^n$ to two groups C and \bar{C} : $v_i \in A$ if $f_i \in C$, and else $v_i \in \bar{A}$

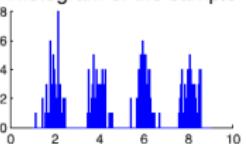
Relaxation of RatioCut and Ncut with general K

- RatioCut
 - Binary vector $h_j = (h_{1j}, \dots, h_{nj})^T$, $j = 1, \dots, K$, as indicator function :
$$h_{ij} = \begin{cases} 1/\sqrt{|A_j|}, & \text{if } v_i \in A_j \\ 0, & \text{otherwise} \end{cases}$$
 - $h_j^T L h_j = Cut(A_j, \bar{A}_j)/|A_j|$, $H = (h_1, \dots, h_K) \in \mathbb{R}^{n \times K}$,
 $RatioCut(A_1, \dots, A_K) = \text{Tr}(H^T L H)$, $H^T H = I$
 - Relax H : $\min_{H \in \mathbb{R}^{n \times K}} \text{Tr}(H^T L H)$, subject to $H^T H = I$
 - Solution : the first K eigenvectors of L as columns
 - Cluster the rows of H to K groups
- Ncut
 - Replacing $|A_j|$ by $vol(A_j)$, the same argument for the relaxation of Ncut : $\min_{H \in \mathbb{R}^{n \times K}} \text{Tr}(H^T L H)$, subject to $H^T D H = I$
 - Solution : the first K eigenvectors of L_{rw} as columns

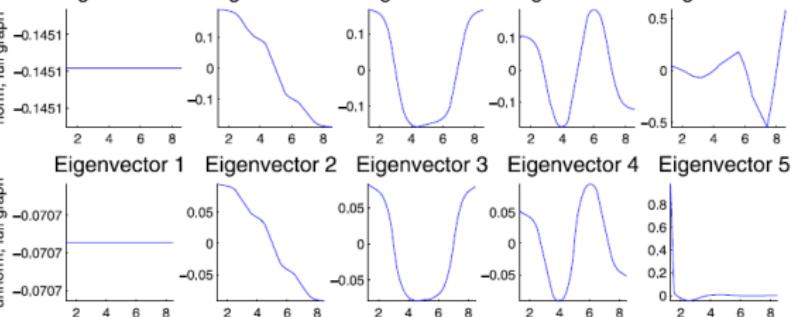
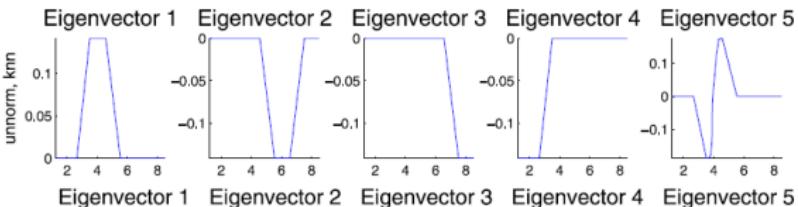
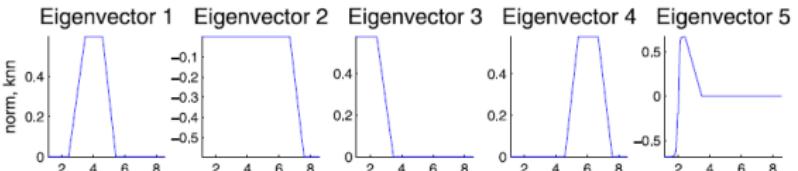
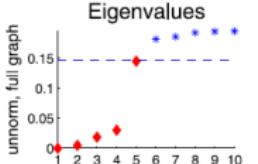
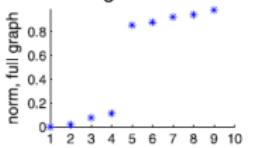
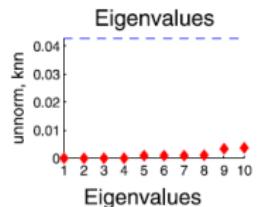
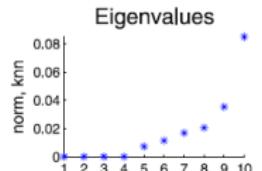
Spectral Clustering Algorithm

- Input : Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters
- Output : Clusters A_1, \dots, A_K of indices of vertices
- Algorithm :
 1. Construct a similarity graph $G = (V, E)$ with weighted adjacency matrix W
 2. Compute the unnormalized graph Laplacian L or normalized graph Laplacian L_{sym} or L_{rw}
 3. Compute the first K eigenvectors $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K] \in \mathbb{R}^{n \times K}$
 4. In the case of L_{sym} , normalize the rows of U to norm 1 ; for the other two cases, skip this step
 5. Let $\mathbf{y}_i \in \mathbb{R}^K$ be the i -th row of \mathbf{U} , use K-means to cluster the point set $\{\mathbf{y}_i\}_{i=1}^n$ into clusters C_1, \dots, C_K
 6. $A_k = \{i | y_i \in C_k\}$

Histogram of the sample

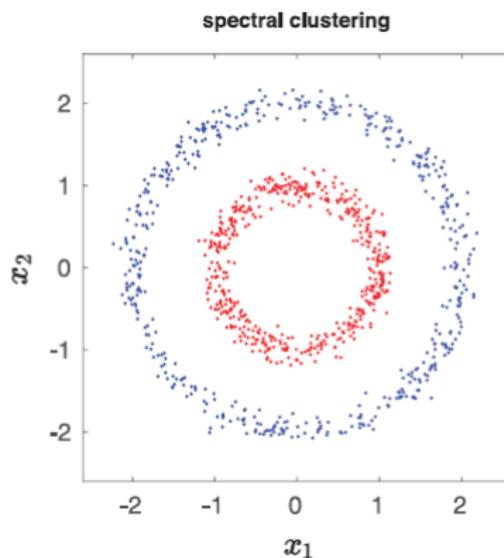
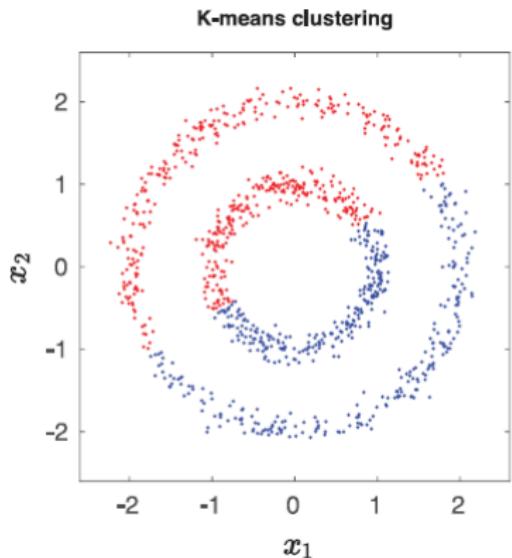


Mixture of 4 Gaussians on \mathbb{R} :



Interpretations

- Usually better than K-means



Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

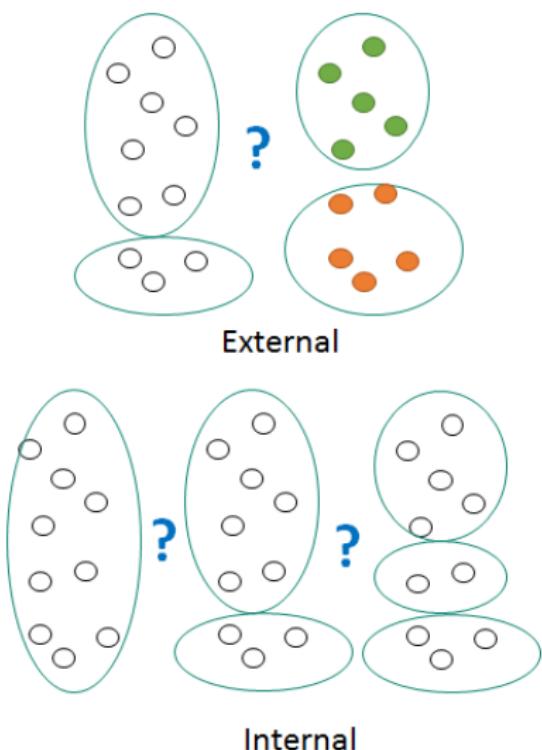
Spectral Clustering

Model Assessment

Case Study

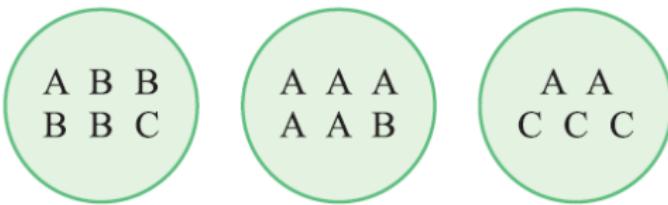
Two Types of Indices

- External indices : validate against ground truth (labels), or compare two clusters (how similar)
 - Purity
 - Jaccard coefficient and Rand index
 - Mutual information
- Internal indices : validate without external info, based on the within-cluster similarity and between-cluster distance
 - Davies-Bouldin index (DBI)
 - Silhouette coefficient (SI)



Purity

- Let n_{ij} be the number of samples that belong to label j but were assigned to cluster i
- Then $n_i = \sum_{j=1}^C$ is the total number of samples in cluster i
- $p_{ij} = n_{ij}/n_i$ is the probability distribution in cluster i
- Purity of cluster i : $p_i \triangleq \max_j p_{ij}$
- Total purity $\triangleq \sum_i \frac{n_i}{n} p_i$
- Example : purity = $\frac{6}{17} \cdot \frac{4}{6} + \frac{6}{17} \cdot \frac{5}{6} + \frac{5}{17} \cdot \frac{3}{5} = 0.71$
- Naive case : treating each sample as a cluster leads to purity 100%



Confusion Matrix

- SS (True Positive or TP) :
of pairs of samples belonging to the **same** cluster in **both** models
- DD (True Negative or TN) :
of pairs of samples belonging to **different** clusters in **both** models
- SD (False Positive or FP) :
of pairs of samples belonging to the **same** cluster in **clustering** model, but **different** clusters in **reference** model
- DS (False Negative or FN) :
of pairs of samples belonging to **different** clusters in **clustering** model, but the **same** cluster in **reference** model

Reference model

	Clustering model	
	Same Cluster	Different Cluster
Same class	SS	DS
Different class	SD	DD

Jaccard Coefficient and Rand Index

- Rand index (RI) : $RI = \frac{SS+DD}{SS+SD+DS+DD} \in [0, 1]$, similar to the accuracy in classification problems
- Jaccard coefficient (JC) : $JC = \frac{SS}{SS+SD+DS} \in [0, 1]$, compare the similarity and diversity of the samples
- Example : # of pairs in the same cluster in clustering model
 $= SS + SD = C_6^2 + C_6^2 + C_5^2 = 40$, and
 $SS = \underbrace{C_4^2}_{\text{cluster1}} + \underbrace{C_5^2}_{\text{cluster2}} + \underbrace{C_3^2 + C_2^2}_{\text{cluster3}} = 20$, so $SD = 20$; # of pairs in the same cluster in clustering model
 $= DS + DD = 6 \times 6 + 6 \times 5 + 6 \times 5 = 96$, and
 $DS = \underbrace{4 \times 1}_B + \underbrace{1 \times 5 + 1 \times 2 + 5 \times 2 + 1 \times 3}_A + \underbrace{5 \times 2 + 1 \times 3}_C = 24$, so $DD = 72$.

$$RI = \frac{20 + 72}{20 + 20 + 24 + 72} = 0.68, \quad JC = \frac{20}{20 + 20 + 24} = 0.31$$

Mutual Information (Wikipedia)

- Mutual information (MI) measures the uncertainty decrement of one random variable given another random variable
- Probability that a sample belongs to both cluster u_i and v_j :

$$p_{UV}(i,j) = \frac{|u_i \cap v_j|}{n}$$
- Its marginal probabilities are :

$$p_U(i) = \frac{u_i}{n}$$
 and $p_V(j) = \frac{v_j}{n}$
- Mutual information : $I(U, V) = \sum_{i=1}^R \sum_{j=1}^C p_{UV}(i,j) \log \frac{p_{UV}(i,j)}{p_U(i)p_V(j)}$
- MI attains its maximum $\min\{H(U), H(V)\}$ only when we have many small clusters
- Normalized MI :

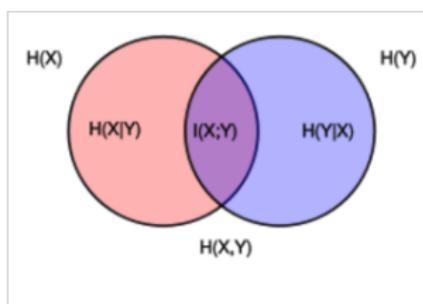
$$NMI(U, V) = \frac{I(U, V)}{(H(U)+H(V))/2}$$

- Entropy : $H(X) = - \sum_x p(x) \log p(x)$

- Conditional entropy :

$$\begin{aligned} H(X|Y) &= \sum_y p(y)H(X|Y=y) \\ &= \sum_y p(y)\left(-\sum_x p(x|y) \log p(x|y)\right) \end{aligned}$$

- MI : $I(X; Y) = H(X) - H(X|Y)$



Davies-Bouldin Index and Silhouette Coefficient

- Davies-Bouldin index (DBI) measures both the within-cluster divergence and between-clusters distance
- $DBI = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{div(c_i) + div(c_j)}{d(\mu_i, \mu_j)} \right)$ where $div(c_i)$ represents the average distance of samples within cluster c_i , μ_i is the center of cluster c_i
- Silhouette Coefficient (SC) : $SC = \frac{b_i - a_i}{\max(a_i, b_i)}$, where a_i is average distance between the i -th sample and every other sample in the same cluster, b_i is the minimal distance from the i -th sample to the other clusters ; range is $[-1, 1]$
- The smaller the DBI, or the larger the SC, the better the clustering results

Outlines

Introduction

K-Means Clustering

Hierarchical Clustering

DBSCAN

Expectation-Maximization Algorithm

Spectral Clustering

Model Assessment

Case Study

Case Study

- Use clustering to group the cars with similar performance based on parameters of the cars
- Dataset comes from “Auto” in the R package ISLR.

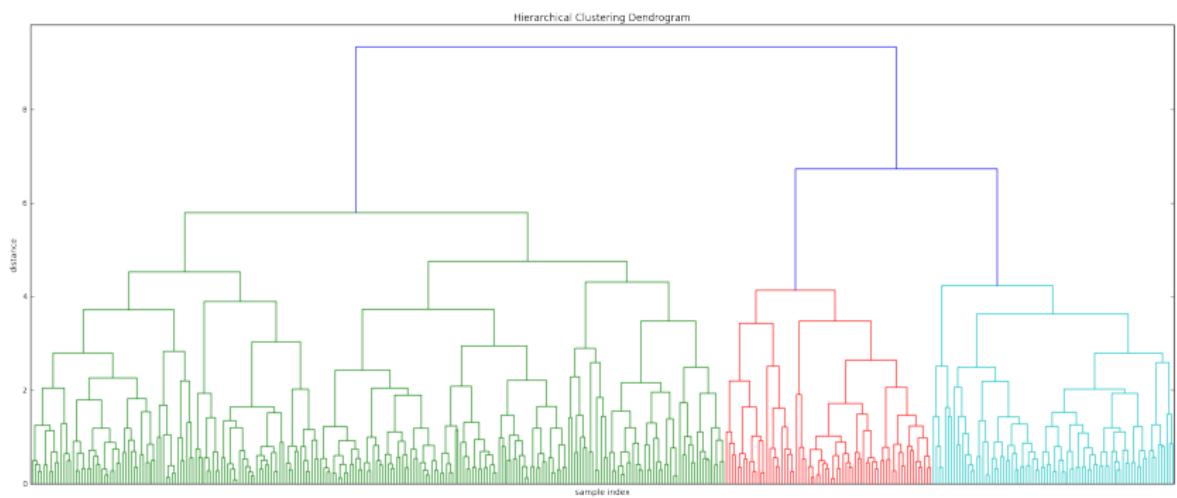
列名	类型	说明	示例
mpg	Float	一加仑汽油能支持的英里数	18
cylinders	Int	气缸数	8
displacement	Float	引擎排量	307
horsepower	Float	引擎马力	130
weight	Float	重量 / lbs	3504
acceleration	Float	从0加速到60mph所需时间 / s	12
year	Int	年份 / 模100	70
origin	Int	生产地, 1:美国2:欧洲 3:日本	1

Hierarchical Clustering

- Scaling of the feature values : Auto_Scaled
- from `scipy.cluster.hierarchy import dendrogram, linkage`
- Construct linkage matrix : $Z = \text{linkage}(\text{Auto_Scaled}, \text{method} = \text{'complete'}, \text{metric} = \text{'euclidean'})$, possible choice for metric could be 'euclidean', 'cityblock', 'minkowski', 'cosine', 'correlation', 'hamming', 'jaccard', etc.
- Data structure of linkage matrix : in the t-th iteration, clusters C_i with index "Z[i, 0]" and C_j with index "Z[i, 1]" are combined to form cluster C_q with index "n + i" ; "Z[i, 2]" is the distance between C_i and C_j ; "Z[i, 3]" is the number of samples in C_q

Dendrogram

dendrogram(Z, no_labels = True)



K-Means Clustering

- from sklearn.cluster import KMeans
- clf = KMeans(n_clusters=3, n_init=1, verbose=1)
- clf.fit(Auto_Scaled)
- Cluster 1 : (economy or compact vehicles) high mpg, low horsepower, low weight ; cluster 2 : (luxury vehicles) low mpg, high horsepower, high weight ; cluster 0 : intermediate performance

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
cluster								
0	19.630588	6.211765	231.423529	102.282353	3274.000000	16.475294	76.011765	1.035294
1	28.947418	4.061033	110.960094	79.779343	2338.370892	16.476995	77.075117	2.046948
2	14.429787	8.000000	350.042553	162.393617	4157.978723	12.576596	73.468085	1.000000

References

- 数据分析导论, 博雅大数据学院
- 周志华, 机器学习, 2016
- T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning : Data mining, Inference, and Prediction*, 2nd Edition, 2009
- Arthur, D., Vassilvitskii, S. “k-means++ : the advantages of careful seeding”. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027 – 1035, 2007
- Lingras P, West C, Interval Set Clustering of Web Users with Rough Kmeans, *Journal of Intelligent Information Systems* 23(1) :5 – 16, 2004

Introduction to Big Data Analysis Dimensionality Reduction

Zhen Zhang

Southern University of Science and Technology

Outlines

Introduction

Principal Component Analysis

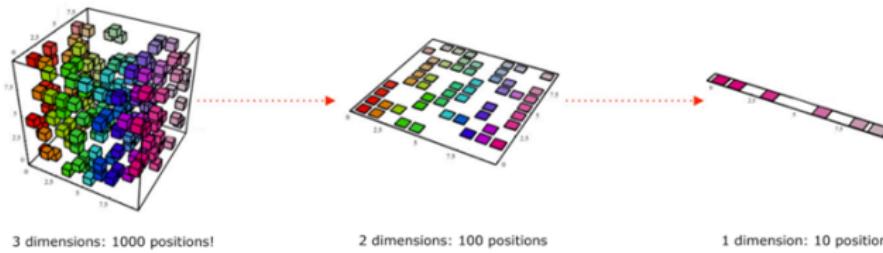
Linear Discriminant Analysis

Nonlinear Dimensionality Reduction

Feature Selection

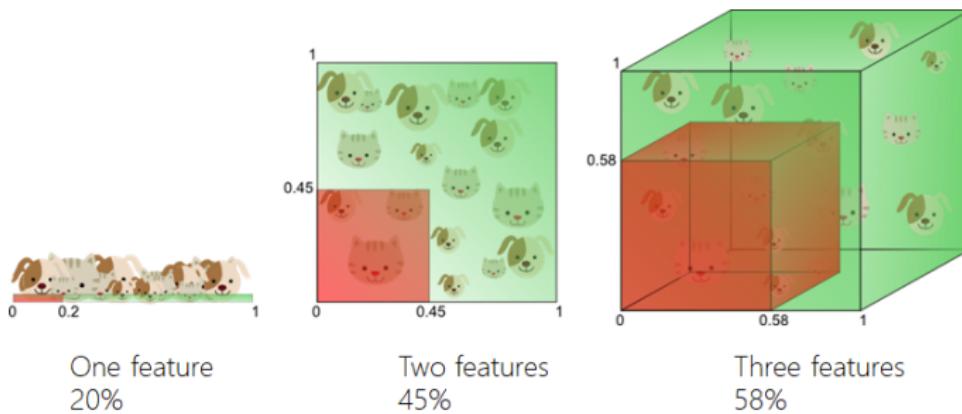
What is Dimensionality Reduction

- The process of reducing the number of random variables under consideration, via obtaining a set of “uncorrelated” principal variable
- By mapping from high-dimensional space to low-dimensional space
- Learning $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\dim \mathcal{X} = n$ and $\dim \mathcal{Y} = r$ with $n > r$.
- Including both unsupervised learning (mostly common) and supervised learning



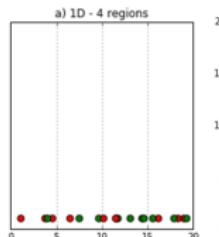
Why Need Dimensionality Reduction ?

- Curse of dimensionality
- Eg : classify cats and dogs using features, if we want to cover 20% of the feature space, how many data do we need ?
- However, the number of samples is limited in practice

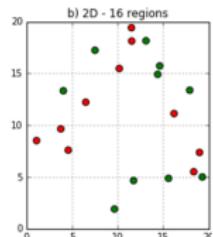


Why Need Dimensionality Reduction? (Cont')

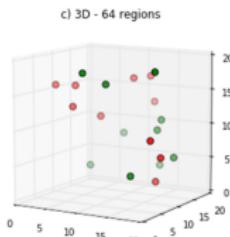
- Due to the sparsity of data in high dimensions, it is easy to overfit
- Hard to train a good model to classify the corner data (getting more in high dimensions)



Density: $20/4 = 5$



$20/16 = 1.25$

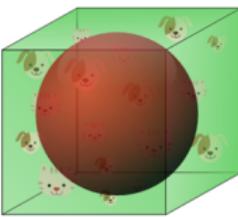
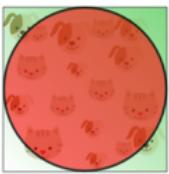


$20/64 \approx 0.31$

In 2D, # of corners:
 $2^2 = 4$

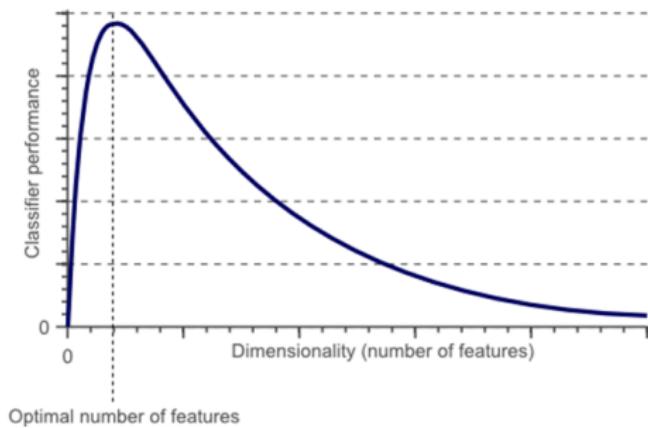
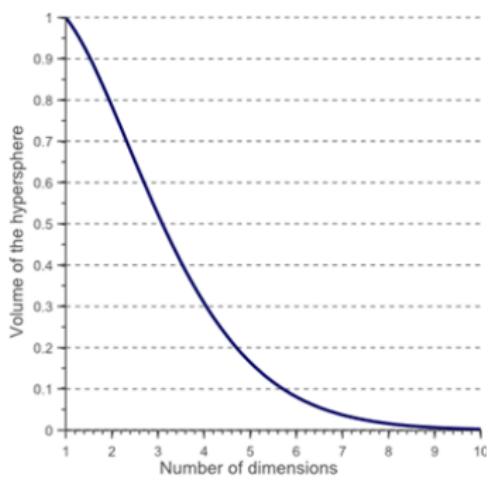
In 3D, # of corners:
 $2^3 = 8$

In 4D, # of corners:
 $2^8 = 256$



Curse of Dimensionality

- The volume of hypersphere decays to zero with the increase of dimension
- The performance gets worse with the increase of dimension



Roles of Dimensionality Reduction

- Data compression
- Denoising
- Feature extraction by mapping and feature selection (eg. Lasso)
- Reduce both spatial and time complexity, so that fewer parameters are needed and smaller computational power is required
- Data visualization

Methods in Dimensionality Reduction

- Linear dimensionality reduction :
 - Principal component analysis (PCA)
 - Linear discriminant analysis (LDA)
 - Independent component analysis (ICA)
- Nonlinear dimensionality reduction :
 - Kernel based methods (Kernel PCA)
 - Manifold learning (ISOMAP, Locally Linear Embedding (LLE), Multidimensional scaling (MDS), t-SNE)

Outlines

Introduction

Principal Component Analysis

Linear Discriminant Analysis

Nonlinear Dimensionality Reduction

Feature Selection

Variance and Covariance Matrix

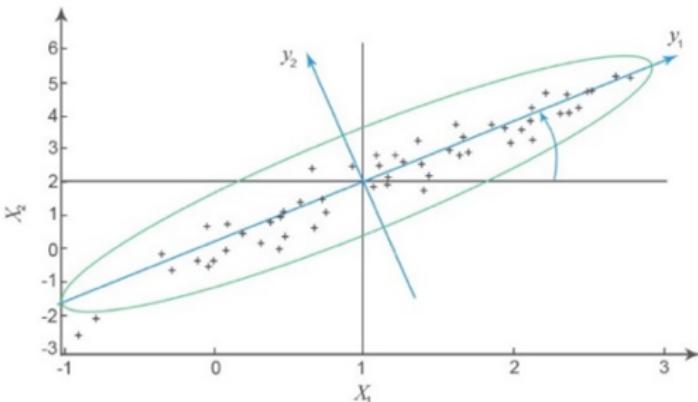
- Variance measures the variability or divergence of single variable : $\text{Var}(X) = E(X - EX)^2$, sample version $S^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$; standard deviation : $\text{Std}(X) = \sqrt{\text{Var}(X)}$
- For more variables, $\text{Cov}(X, Y) = E(X - EX)(Y - EY)$, sample version $C = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$
- If $X = (x_1, \dots, x_n)^T \in \mathbb{R}^{n \times p}$ is the sample matrix, then $C = \frac{1}{n-1} (X - \mathbf{1}_n \bar{x}^T)^T (X - \mathbf{1}_n \bar{x}^T) = \frac{1}{n-1} (X - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T X)^T (X - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T X) = \frac{1}{n-1} X^T J X$, where $J = I_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T$ is a projection matrix with rank $n - 1$.

Principal Component Analysis (PCA)

- PCA transforms a set of strongly correlated variables to another set (**typically much smaller**) of **weakly correlated** variables by using orthogonal transformation
- The new variables are called principal components
- The new set of variables are linear combinations of the original variables whose variance information is inherited as much as possible
- Unsupervised learning
- Proposed by Karl Pearson, successfully used in economics by Stone (1947) : keep 97.4% information, 17 variables about income and expenditure are finally reduced to 3 variables (F1 : total income, F2 : rate of change in total income, F3 : economic development or recession)

Geometric Interpretation

- Assume a set of 2D data follows Gaussian distribution (but not limited to Gaussian distribution !), the reduction to 1D is successfully achieved by taking a direction with larger variance (larger variability of data)
- The direction in the major axis contains more information than the other direction, since smaller variance indicates the variables are almost the same



Linear Algebra

- Let $\{\mathbf{e}_i\}_{i=1}^p$ be the canonical basis in Euclidean space, want to find another orthonormal basis $\{\tilde{\mathbf{e}}_i\}_{i=1}^p$ such that the random vector $\mathbf{v} = \sum_{i=1}^p x_i \mathbf{e}_i$ can be expressed in the new basis by $\mathbf{v} = \sum_{i=1}^p \tilde{x}_i \tilde{\mathbf{e}}_i$, where $\text{Var}(\tilde{x}_1) \geq \dots \geq \text{Var}(\tilde{x}_p)$ and $\text{Cov}(\tilde{x}_i, \tilde{x}_j) \approx 0$ for $i \neq j$
- By linear algebra, the coordinate transformation is given by the linear transformation : $(\tilde{\mathbf{e}}_1, \dots, \tilde{\mathbf{e}}_p) = (\mathbf{e}_1, \dots, \mathbf{e}_p)W$, where $W \in \mathbb{R}^{p \times p}$ is an invertible matrix
- The component coefficients is transformed accordingly :
$$\mathbf{x} = W\tilde{\mathbf{x}}$$

Eigendecomposition of Sample Covariance Matrix

- Assume we have n centralized samples $\{\mathbf{x}_i\}_{i=1}^n$ with $\frac{1}{n} \sum_{i=1}^n \mathbf{x}_i = \mathbf{0}_p$
- Then $X^T = (\mathbf{x}_1, \dots, \mathbf{x}_n) = W(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n) = W\tilde{X}^T$
- The sample covariance matrix of X is $\text{Cov}(X) = \frac{1}{n-1} X^T X$
- The sample covariance matrix of \tilde{X} is $\text{Cov}(\tilde{X}) = \frac{1}{n-1} \tilde{X}^T \tilde{X} = \frac{1}{n-1} W^T X^T X W = W^T \text{Cov}(X) W$
- Its diagonals are the sample versions of $\text{Var}(\tilde{x}_1), \dots, \text{Var}(\tilde{x}_p)$, while its off-diagonals are the covariances between \tilde{x}_i and \tilde{x}_j
- Need that $\text{Cov}(\tilde{X})$ is nearly diagonal with decreasing diagonal entries for some W .
- Equivalent to do eigendecomposition :
 $\text{Cov}(X) = O \text{diag}(\lambda_1, \dots, \lambda_p) O^T$ with some orthogonal matrix $O \in \mathbb{R}^{p \times p}$ and $\lambda_1 \geq \dots \geq \lambda_p \geq 0$, then let $W = O$ completes the job

Interpretations

- Variances in the transformed variables : $\text{Var}(\tilde{x}_i) = \lambda_i$, eigenvalues of $\text{Cov}(X)$
- The new basis consists of the columns of $W = O$, i.e., the eigenvectors of $\text{Cov}(X)$
- The percentage $\frac{\lambda_i}{\sum_{j=1}^p \lambda_j}$ explains the importance of the new variable \tilde{x}_i
- Given a threshold t , we can choose the number of variables r such that the total contribution to the variance of the new r variables $\sum_{i=1}^r \frac{\lambda_i}{\sum_{j=1}^p \lambda_j}$ exceeds the threshold t . Thus these r directions $\mathbf{w}_1, \dots, \mathbf{w}_r$ are enough to represent the original n variables
- For any random vector $\mathbf{x} \in \mathbb{R}^p$, the corresponding r principal components are thus $\mathbf{w}_1^T \mathbf{x}, \dots, \mathbf{w}_r^T \mathbf{x}$

Another Viewpoint - Best Reconstruction

- Note that the new basis $\{\tilde{\mathbf{e}}_j\}_{j=1}^p$ is given by $\tilde{\mathbf{e}}_j = \mathbf{w}_j$;
- After the projection (if we keep the first r components), the projected point of each sample \mathbf{x}_i is $\tilde{\mathbf{x}}_{i,1}\mathbf{w}_1 + \cdots + \tilde{\mathbf{x}}_{i,r}\mathbf{w}_r$, where the coordinate is given by $\tilde{\mathbf{x}}_{i,j} = \mathbf{w}_j^T \mathbf{x}_i$;
- The reconstruction error is the sum of all squared L^2 errors of all samples :

$$\begin{aligned}
 RE(W) &= \sum_{i=1}^n \left\| \sum_{j=1}^r \tilde{\mathbf{x}}_{i,j} \mathbf{w}_j - \mathbf{x}_i \right\|_2^2 = \sum_{i=1}^n \| (W_r W_r^T - I) \mathbf{x}_i \|_2^2 \\
 &= \sum_{i=1}^n \mathbf{x}_i^T (I - W_r W_r^T) \mathbf{x}_i = Tr \left(\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T (I - W_r W_r^T) \right) \\
 &= Tr(X^T X (I - W_r W_r^T)) = Tr(X^T X) - Tr(W_r^T X^T X W_r)
 \end{aligned}$$

- Resulting in an optimization problem :

$$\min_{W_r} -Tr(W_r^T X^T X W_r), \quad \text{subject to } W_r^T W_r = I$$

PCA Algorithm

- Given the data matrix $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T \in \mathbb{R}^{n \times p}$ and a threshold t (in some other cases, the number of principal components r) :
 1. Centralize the data by their mean $\bar{\mathbf{x}} = \frac{1}{n}\mathbf{1}_n^T X$, and compute the sample covariance matrix $C = \frac{1}{n-1}(X - \mathbf{1}_n\bar{\mathbf{x}}^T)^T(X - \mathbf{1}_n\bar{\mathbf{x}}^T)$
 2. Compute the eigenvalues $\{\lambda_i\}_{i=1}^p$ and the corresponding eigenvectors $\{\mathbf{w}_i\}_{i=1}^p$
 3. Order the eigenvalues as $\lambda_{(1)} \geq \dots \geq \lambda_{(p)}$, and compose an orthogonal matrix W by the eigenvectors columnwise in the same order : $W = (\mathbf{w}_1, \dots, \mathbf{w}_p)$
 4. Compute the variance contribution of the first r eigenvalues : $\sum_{i=1}^r \frac{\lambda_i}{\sum_{j=1}^p \lambda_j}$, find a suitable r such that this variance contribution is greater than the threshold t
 5. Pick the first r columns in W and form a matrix $W_r = (\mathbf{w}_1, \dots, \mathbf{w}_r) \in \mathbb{R}^{p \times r}$
 6. Output $\tilde{X}_r = XW_r \in \mathbb{R}^{n \times r}$ as the projected data matrix, whose rows consist of data points in r dimensional subspace

An Example

- The data : the monthly prices of three brands of vehicles (Jeep : x_1 , Toyota : x_2 , Benz : x_3)
- The covariance matrix is given by

$$C = \begin{pmatrix} 1 & \frac{2}{\sqrt{10}} & -\frac{2}{\sqrt{10}} \\ \frac{2}{\sqrt{10}} & 1 & -\frac{4}{5} \\ -\frac{2}{\sqrt{10}} & -\frac{4}{5} & 1 \end{pmatrix}$$

- Compute the characteristic polynomial :

$$\det(\lambda I - C) = \begin{vmatrix} \lambda - 1 & -\frac{2}{\sqrt{10}} & \frac{2}{\sqrt{10}} \\ -\frac{2}{\sqrt{10}} & \lambda - 1 & \frac{4}{5} \\ \frac{2}{\sqrt{10}} & \frac{4}{5} & \lambda - 1 \end{vmatrix}$$

- Solve for the eigenvalues : $\lambda_1 = 2.38$, $\lambda_2 = 0.42$, $\lambda_3 = 0.2$

An Example (Cont')

- Plug in each eigenvalues and solve for the corresponding eigenvectors, e.g., $(\lambda_1 I - C)\mathbf{w}_1 = \mathbf{0}$, or equivalently,

$$\begin{cases} 1.38w_{11} - \frac{2}{\sqrt{10}}w_{12} + \frac{2}{\sqrt{10}}w_{13} = 0, \\ -\frac{2}{\sqrt{10}}w_{11} + 1.38w_{12} + 0.8w_{13} = 0, \\ \frac{2}{\sqrt{10}}w_{11} + 0.8w_{12} + 1.38w_{13} = 0. \end{cases}$$

- One can find three eigenvectors as $\mathbf{w}_1 = (0.54, 0.59, -0.59)^T$, $\mathbf{w}_2 = (0.84, -0.39, 0.39)^T$, $\mathbf{w}_3 = (0, 0.71, 0.71)^T$
- The three components are

$$\tilde{x}_1 = \mathbf{w}_1^T \mathbf{x} = 0.54x_1 + 0.59x_2 - 0.59x_3,$$

$$\tilde{x}_2 = \mathbf{w}_2^T \mathbf{x} = 0.84x_1 - 0.39x_2 + 0.39x_3,$$

$$\tilde{x}_3 = \mathbf{w}_3^T \mathbf{x} = 0.71x_2 + 0.71x_3.$$

- As $\lambda_1 \gg \lambda_2, \lambda_3$, the first principal component \tilde{x}_1 reflects the change of prices in all three brands of vehicles

Outlines

Introduction

Principal Component Analysis

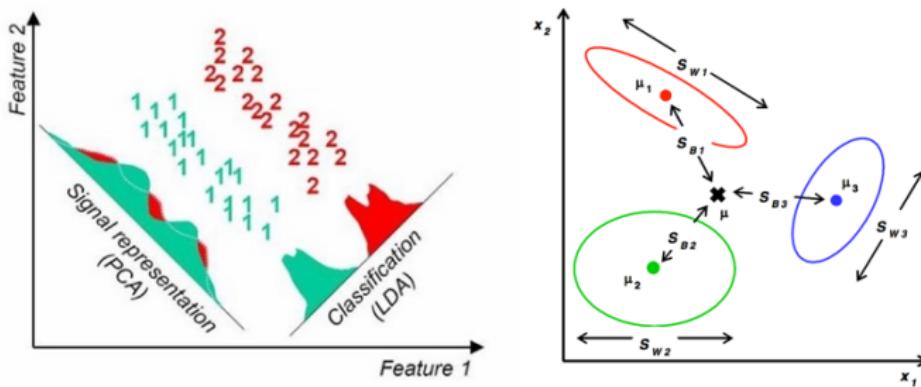
Linear Discriminant Analysis

Nonlinear Dimensionality Reduction

Feature Selection

Linear Discriminant Analysis (LDA)

- Supervised learning : based on the labels, do linear projection in order to maximize the between-class point scatter (variability) in low dimensions
 - Initially proposed by R. Fisher for two-class classification (1936)
 - Generalized by C. R. Rao (1948) to K classes $\{C_1, \dots, C_K\}$



Basic Concepts

- The number of samples in each class is $n_k = \sum_{i:\mathbf{x}_i \in C_k} 1$, whereas the total number of samples is $n = \sum_{k=1}^K n_k$
- The mean of samples in class k is $\mu_k = \frac{1}{n_k} \sum_{i:\mathbf{x}_i \in C_k} \mathbf{x}_i$, whereas the mean of all samples is $\mu = \sum_{k=1}^K \frac{n_k}{n} \mu_k$
- Before projection, the between-class point scatter is $S_b = \sum_{k=1}^K \frac{n_k}{n} (\mu_k - \mu)(\mu_k - \mu)^T$; after projection $W_r \in \mathbb{R}^{p \times r}$, the between-class point scatter is $\tilde{S}_b = W_r^T S_b W_r$
- Before projection, the within-class point scatter (variance) for each class C_k is $S_k = \frac{1}{n_k} \sum_{i:\mathbf{x}_i \in C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T$, thus the total within-class point scatter is $S_w = \sum_{k=1}^K \frac{n_k}{n} S_k$; after projection, the within-class point scatter for each class C_k is $\tilde{S}_k = W_r^T S_k W_r$, and the total within-class point scatter is $\tilde{S}_w = W_r^T S_w W_r$

Optimization Problem

- Need to find the optimal directions (columns of W_r) such that the between-class point scatter \tilde{S}_b is maximized and within-class point scatter \tilde{S}_w is minimized, i.e.,

$$\max_{\mathbf{w}} J(\mathbf{w}) = \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

- This is equivalent to solve

$$\max_{\mathbf{w}} J_b(\mathbf{w}) = \mathbf{w}^T S_b \mathbf{w}, \quad \text{subject to } \mathbf{w}^T S_w \mathbf{w} = 1$$

- By introducing a Lagrange multiplier λ , we define Lagrangian as $L(\mathbf{w}, \lambda) = \mathbf{w}^T S_b \mathbf{w} - \lambda(\mathbf{w}^T S_w \mathbf{w} - 1)$
- The optima is obtained as the solution to the equation

$$\nabla_{\mathbf{w}} L = 2S_b - 2\lambda S_w \mathbf{w} = \mathbf{0} \Rightarrow S_w^{-1} S_b \mathbf{w} = \lambda \mathbf{w}$$

- The optimal directions are the eigenvectors of $S_w^{-1} S_b$

An Example

- Given two sets of data : class 1 is $\{(4, 1)^T, (2, 4)^T, (2, 3)^T, (3, 6)^T, (4, 4)^T\}$, and class 2 is $\{(9, 10)^T, (6, 8)^T, (9, 3)^T, (8, 7)^T, (10, 8)^T\}$
- Class means : $\mu_1 = (3, 3.6)^T$, $\mu_2 = (8.4, 7.6)^T$, the point scatter metrics are

$$S_1 = \begin{pmatrix} 0.8 & -0.4 \\ -0.4 & 2.6 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 1.84 & -0.28 \\ -0.28 & 5.36 \end{pmatrix},$$

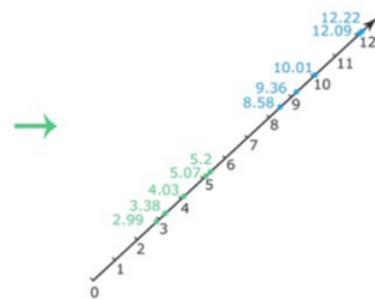
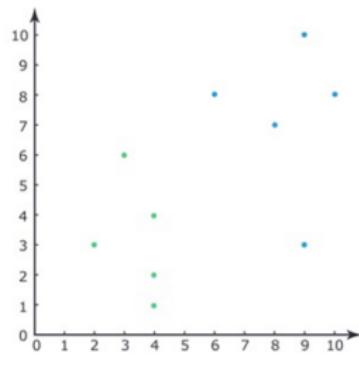
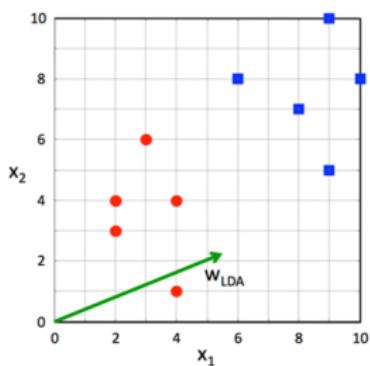
$$S_b = \begin{pmatrix} 7.29 & 4.86 \\ 4.86 & 3.24 \end{pmatrix}, \quad S_w = \begin{pmatrix} 1.32 & -0.34 \\ -0.34 & 4 \end{pmatrix}.$$

- The eigenvalue of $S_w^{-1}S_b$ is solved from

$$0 = \det(\lambda I - S_w^{-1}S_b) = \begin{vmatrix} \lambda - 5.97 & -3.98 \\ -1.72 & \lambda - 1.15 \end{vmatrix} \Rightarrow \lambda = 7.11$$

An Example (Cont')

- The optimal directions is $\mathbf{w}^* = (0.96, 0.28)^T$
- After projection, the data become 1D :
 - Class 1 : {4.12, 3.03, 2.75, 4.55, 4.95}
 - Class 2 : {11.42, 7.98, 9.48, 9.63, 11.83}



PCA vs. LDA

- PCA
 - Start from sample covariance matrix and find directions with maximal variances
 - Unsupervised learning, used as pre-training step, must be coupled with other learning methods
- LDA
 - Make use of labels and find projections after which the classification becomes more obvious
 - Supervised learning, can be used as classification or coupled with other learning methods

Outlines

Introduction

Principal Component Analysis

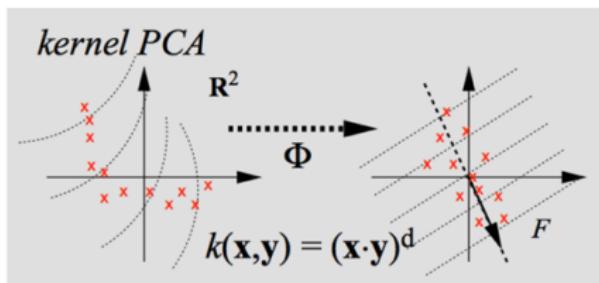
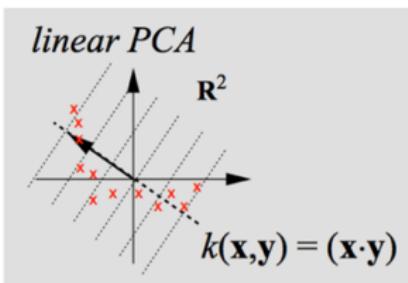
Linear Discriminant Analysis

Nonlinear Dimensionality Reduction

Feature Selection

Kernel PCA

- PCA works well for Gaussian distribution
- If the data do not follow Gaussian, we can find a map $\phi : \mathbb{R}^p \rightarrow \mathbb{R}^q$ so that $\phi(\mathbf{x})$ (almost) follows Gaussian
- We can do PCA for the transformed data $\{\phi(\mathbf{x}_i)\}_{i=1}^n$
- Similar to nonlinear SVM, kernel trick can be used to avoid explicit computation of ϕ



Covariance Matrix in Transformed Space

- Assume the transformed data are centralized :
 $\mu = \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{x}_i) = 0$
- Covariance Matrix $\tilde{C} = \frac{1}{n-1} \sum_{i=1}^n \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T$
- Do PCA for transformed data is equivalent to find the eigenvalues and eigenvectors of \tilde{C}
- Let λ be an eigenvalue of \tilde{C} and $\mathbf{v} \in \mathbb{R}^q$ be the corresponding eigenvector, i.e., $\tilde{C}\mathbf{v} = \lambda\mathbf{v}$.
- It can be shown that $\mathbf{v} = \sum_{i=1}^n \alpha_i \phi(\mathbf{x}_i)$ where
 $\alpha_i = \frac{1}{\lambda(n-1)} \phi(\mathbf{x}_i)^T \mathbf{v}$
- Furthermore, $\alpha_i = \frac{1}{\lambda(n-1)} \sum_{j=1}^n K(\mathbf{x}_i, \mathbf{x}_j) \alpha_j$, where
 $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is kernel function
- It is sufficient to solve the eigenvalue problem :
 $K\alpha = \lambda(n-1)\alpha$ where $K = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ is kernel matrix and
 $\alpha = (\alpha_i)$ is the coefficient vector of \mathbf{v}

Kernel PCA Algorithm

1. Choose a kernel function $K(x, y)$ satisfying the necessary properties
2. Compute the kernel matrix $K = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$
3. Compute the eigenvalues $\lambda_1 \geq \dots \geq \lambda_q$ and eigenvectors $\alpha^{(1)}, \dots, \alpha^{(q)}$ of K
4. For any new sample \mathbf{x} , the j component after projection is

$$z_j = \mathbf{v}_j^T \phi(\mathbf{x}) = \sum_{i=1}^n \alpha_i^{(j)} K(\mathbf{x}_i, \mathbf{x})$$

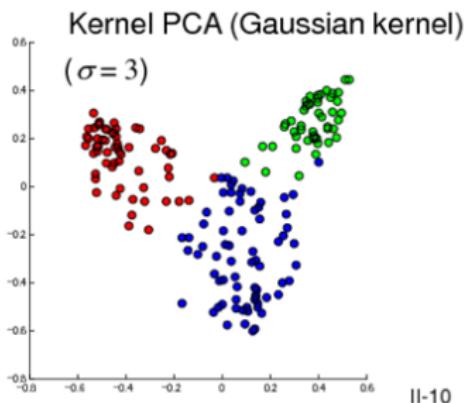
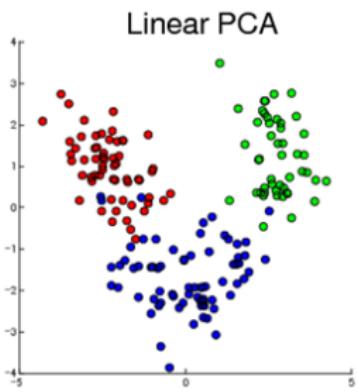
Kernel PCA : An Example

■ Wine data (from UCI repository)

13 dim. chemical measurements of for three types of wine. 178 data.

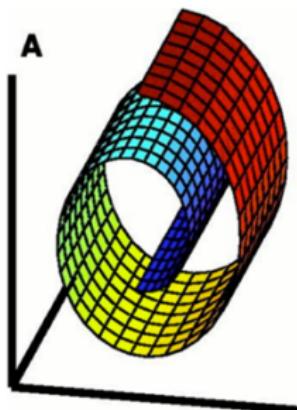
Class labels are NOT used in PCA, but shown in the figures.

First two principal components:



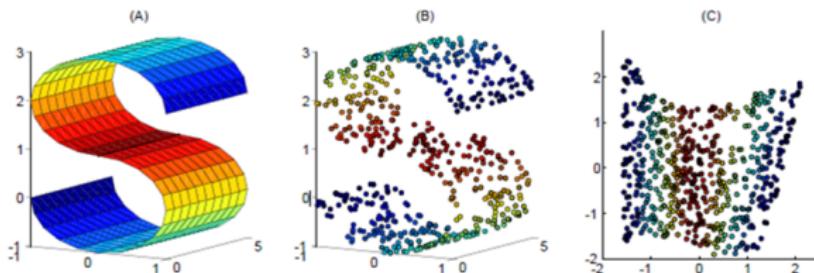
Manifolding Learning

- A manifold is a topological space that locally resembles Euclidean space near each point. It generalizes the concepts of curves and surfaces in Euclidean space.
- The dimension of a manifold is the minimal number of coordinates to represent a point on the manifold
- Some dimensionality reduction methods are based on the concept of manifold : ISOMAP, LLE, MDS, t-SNE



Locally Linear Embedding (LLE)

- Reduce the number of free coordinates while keeping the local geometric structure of the data, e.g., if \mathbf{x}_A and \mathbf{x}_B are neighbor in high dimension, after the dimension reduction (transformation), they must be close to each other in low dimension
- The clustering effect should also be inherited



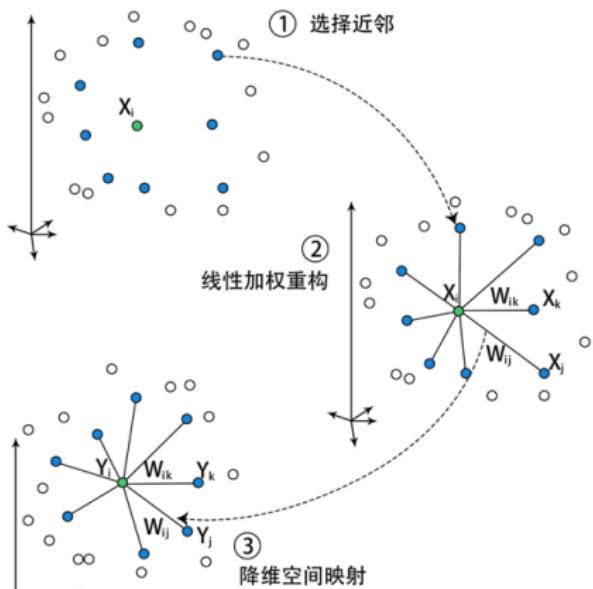
LLE Reconstruction

- Assume each data point is locally linearly dependent of its neighbors : it can be written as the linear combination of its K nearest neighbors $\{\mathbf{x}_{k_{ij}}\}_{j=1}^K$, with the KNN indices $\{k_{ij}\}_{j=1}^K$
 - The weight is determined by the optimization for each \mathbf{x}_i :

$$\min_{\mathbf{w}} \|\mathbf{x}_i - \sum_{j=1}^K w_{ik_{ij}} \mathbf{x}_{k_{ij}}\|_2^2$$

$$\text{subject to } \sum_{j=1}^K w_{ik_j} = 1, \quad w_{ij} \geq 0$$

where $w_{ij} = 0$ if $j \notin \{k_{ij}\}_{j=1}^K$



Low Dimensional Representation

- In r ($r < p$) dimensional space, find n points such that the local structure (e.g., clustering effect) is preserved

$$\min_{\mathbf{y}_1, \dots, \mathbf{y}_n} \sum_{i=1}^n \left\| \mathbf{y}_i - \sum_{j=1}^n w_{ij} \mathbf{y}_j \right\|_2^2$$

- This is equivalent to the matrix minimization problem

$$\min_{\mathbf{Y}} \text{Tr}(\mathbf{Y}^T \mathbf{M} \mathbf{Y}), \quad \text{s.t.} \quad \mathbf{Y} \mathbf{Y}^T = \mathbf{I},$$

where $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)^T \in \mathbb{R}^{n \times r}$ and $\mathbf{M} = (\mathbf{I} - \mathbf{W})^T (\mathbf{I} - \mathbf{W})$ with $\mathbf{W} = (w_{ij})_{i,j=1}^n$ being the weight matrix (not necessarily symmetric)

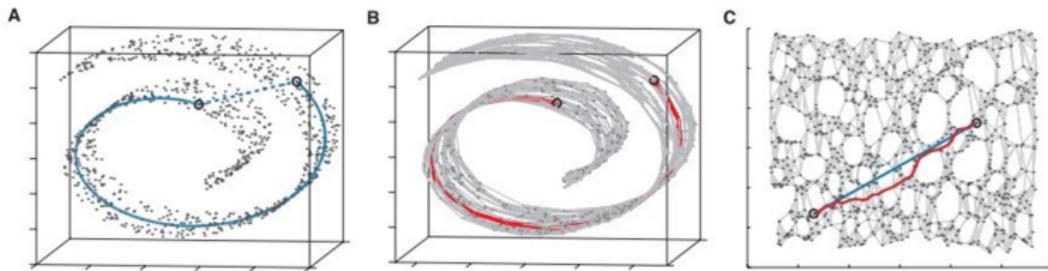
- This is solved by eigen-decomposition : The columns of \mathbf{Y} consist of the r eigenvectors corresponding to the r smallest eigenvalues of \mathbf{M}

Summary of LLE

- Only one tuning parameter K
- Linear algebra computation
- Only local information, no global information
- No explicit mapping as in PCA ($\tilde{X}_r = XW_r$)

The Motivation of ISOMAP

- The distance between two points may be different in different metrics (manifold metric vs. Euclidean metric)
- Geodesic distance could be a good metric instead of Euclidean distance
- Computation of geodesic distance, minimal path in graph



ISOMAP Algorithm

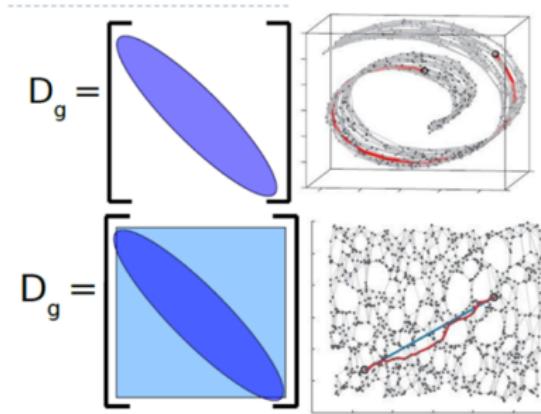
- Construct KNN graph

$$G = (V, E) :$$

- For each \mathbf{x}_i , find its K nearest neighbors $\{\mathbf{x}_j\}_{j \in N(i)}$
- The weight of the edge $< i, j >$ between \mathbf{x}_i and \mathbf{x}_j is the Euclidean distance for each $j \in N(i)$

- Use Floyd algorithm to compute the minimal path between each pair of vertices (i, j) as the geodesic distance $d_G(i, j)$
- Find the low dimensional representation (e.g. by MDS) :

$$\min_{\mathbf{y}_1, \dots, \mathbf{y}_n} \sum_{i \neq j} (d_G(i, j) - \|\mathbf{y}_i - \mathbf{y}_j\|)^2$$



Floyd Algorithm (Complexity $O(n^3)$)

1. Initialization :

$$d_G(i,j) = \begin{cases} d_x(i,j), & \text{if } < i,j > \in E \\ \infty, & \text{otherwise} \end{cases}$$

2. For each pair (i,j) , update the distance as follows : for each $k = 1, \dots, n$, $d_G(i,j) = \min\{d_G(i,j), d_G(i,k) + d_G(k,j)\}$
3. The final output $d_G(i,j)$ is the geodesic distance between i and j

Summary of ISOMAP

- Only one tuning parameter K
- High computational power
- Preserve the global information
- Sensitive to noise

Multidimensional Scaling (MDS)

- For data points in high dimensional space, $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^p$, find the distance or dissimilarity matrix $\{d_{ij}\}_{i,j}^n$, e.g., $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$
- Find $\{\mathbf{y}_i\}_{i=1}^n \subset \mathbb{R}^r$ ($r < p$), such that the distance information is preserved :

$$\min_{\mathbf{y}_1, \dots, \mathbf{y}_n} S_M(\mathbf{y}_1, \dots, \mathbf{y}_n)$$

where $S_M(\mathbf{y}_1, \dots, \mathbf{y}_n) = \sum_{i \neq j} (d_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2$ is the stress function. This is called least square or Kruskal-Shephard scaling

- Alternative objective function (Sammon mapping) :
 $S_{S_M}(\mathbf{y}_1, \dots, \mathbf{y}_n) = \sum_{i \neq j} \frac{(d_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2}{d_{ij}}$ takes care of small d_{ij}
- This is nonconvex minimization

t-distributed Stochastic Neighbor Embedding (t-SNE)

- Developed by Laurens van der Maaten and Geoffrey Hinton
- Effective for data visualization in 2D and 3D, applications in computer security research, music analysis, cancer research, especially for bioinformatic data
- Often display clusters in low dimensional space (may be false findings)
- With special parameter choices, approximates a simple form of spectral clustering

Similarity in High Dimensional Space

- For data points in high dimensional space, $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^p$, find the similarity of \mathbf{x}_i and \mathbf{x}_j in the form of probability p_{ij}
- The similarity of data point x_j to data point x_i is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i :

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$

- $p_{ij} = (p_{j|i} + p_{i|j})/2n$, $p_{ii} = 0$
- The bandwidth is adapted to the density of the data : smaller values of σ_i are used in denser parts of the data space

Similarity in Low Dimensional Space

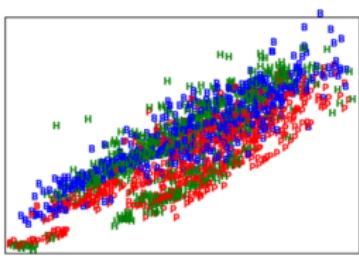
- t-SNE aims to learn a set of low dimensional data $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^r$ that reflects the similarity p_{ij} as well as possible
- The similarity between the data point \mathbf{y}_i and \mathbf{y}_j follows t-distribution : (assume $q_{ii} = 0$)

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_k - \mathbf{y}_i\|^2)^{-1}}$$

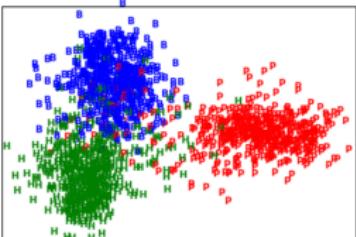
- t-distribution is heavy tailed so that large p_{ij} (dissimilar data pair) leads to even larger q_{ij} (falls apart)
- The closedness between the two similarity measures p_{ij} and q_{ij} is given by the Kullback-Leibler divergence :

$$D_{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

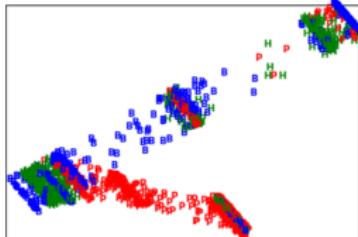
Comparison (Optical Character Recognition)



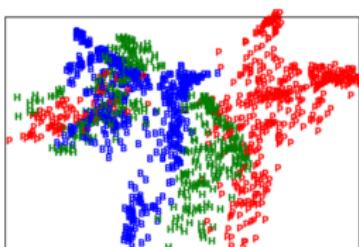
PCA



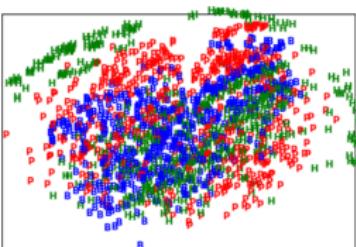
LDA



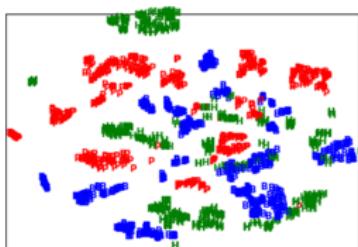
LLE



ISOMAP



MDS



t-SNE

Outlines

Introduction

Principal Component Analysis

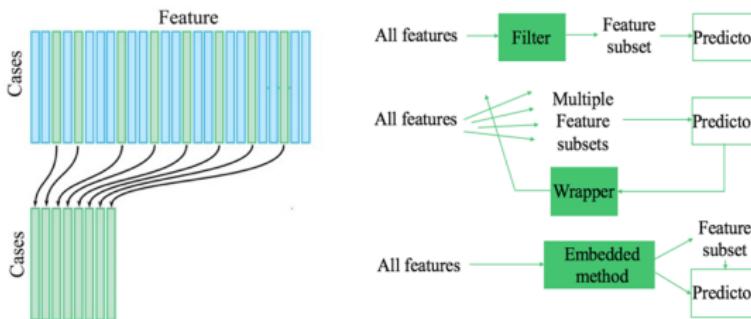
Linear Discriminant Analysis

Nonlinear Dimensionality Reduction

Feature Selection

What is Feature Selection

- Low computational cost, better accuracy (avoid overfitting), and better interpretation,
- Feature engineering : feature extraction and selection. Feature extraction is according to the knowledge of the professions, usually done by expertise in the professional areas
- Three types : Filter, Wrapper, and Embedded



Subset Selection

- Subset search :
 - Forward search (forward stepwise, forward stagewise) :
 $\emptyset \Rightarrow \{x_1\} \Rightarrow \{x_1, x_4\} \Rightarrow \dots$
 - Backward search (backward stepwise) :
 $\{x_1, x_2, \dots, x_p\} \Rightarrow \{x_1, x_2, \dots, x_p\} \setminus \{x_4\} \Rightarrow \dots$
 - Bidirectional search
- Evaluation metrics :
 - Distances : Euclidean, Manhattan, point scatter matrices, Kullback-Leibler divergence, etc.
 - Information : mutual information, information gain (IG), etc.
 - Correlations : Pearson correlation, Maximal information coefficients (MIC)
- Stopping rules : number of features, number of iterations, non-incremental metrics, attaining optimality, etc.
- Validation and comparison

Three Types of Feature Selection

- Filter : filter the features by their correlations (or MIC, IG) with response variables
- Wrapper : use accuracy, precision, recall, AUC, etc.
 - Akaike Information Criteria (AIC) : $AIC = -2 \ln(L) + 2k$
 - Bayes Information Criteria (BIC) : $BIC = -2 \ln(L) + k \ln(n)$
 - Minimize AIC or BIC, where L is likelihood function, k is the number of features (parameters), n is the number of samples
- Embedded :
 - Random forest : feature importance
 - Regularization : Ridge and LASSO
 - Recursive feature elimination (RFE) : select the best (worst) feature according to the coefficients (e.g. linear regression), then do this recursively to find the feature importance

References

- 数据分析导论, 博雅大数据学院
- 周志华, 机器学习, 2016
- T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning : Data mining, Inference, and Prediction*, 2nd Edition, 2009
- Arthur, D., Vassilvitskii, S. “k-means++ : the advantages of careful seeding”. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027 – 1035, 2007
- Lingras P, West C, Interval Set Clustering of Web Users with Rough Kmeans, *Journal of Intelligent Information Systems* 23(1) :5 – 16, 2004



MA333 Introduction to Big Data Science Deep Learning

Zhen Zhang

Southern University of Science and Technology

Outlines

Introduction

What is Deep Learning ?

Why Deep Learning is Growing ?

History

The structural building block of Deep Learning

Forward Propagation

Back Propagation

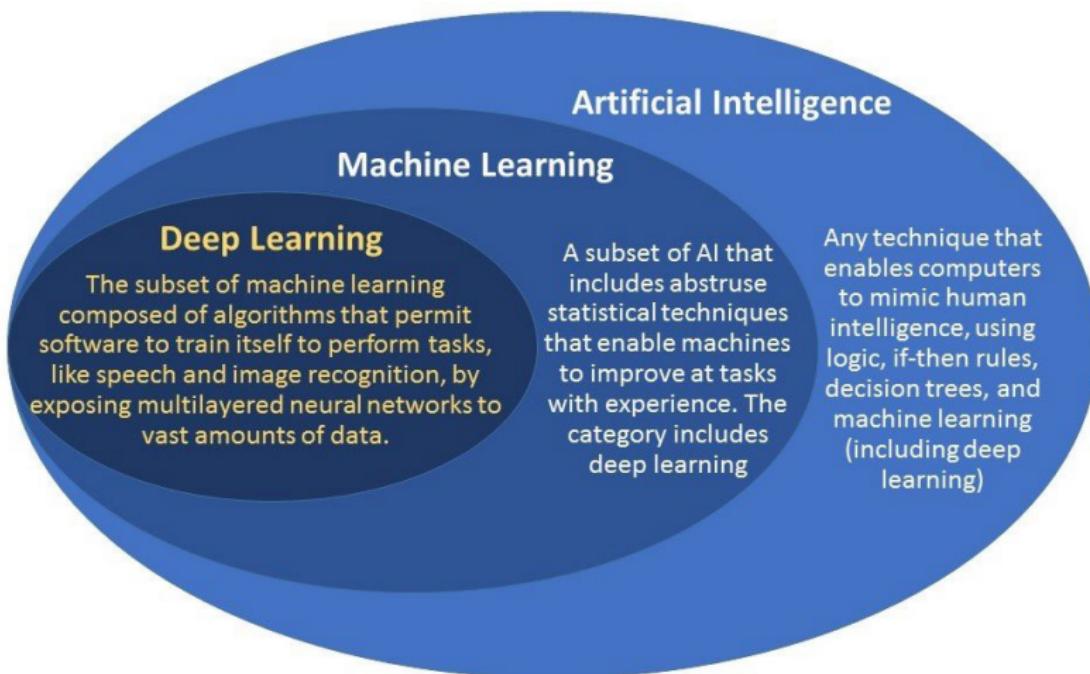
Architectures

Recurrent Neural Network (RNN)

Convolutional Neural Network (CNN)

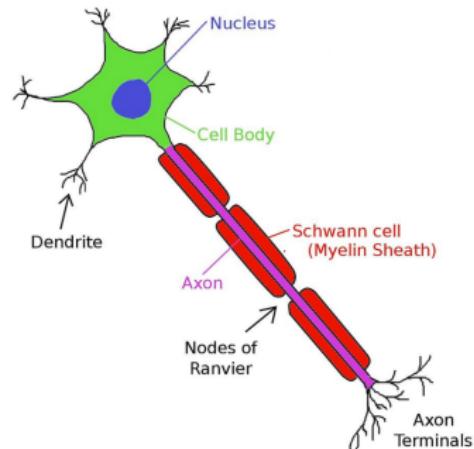
Generative model

What is Deep Learning ?



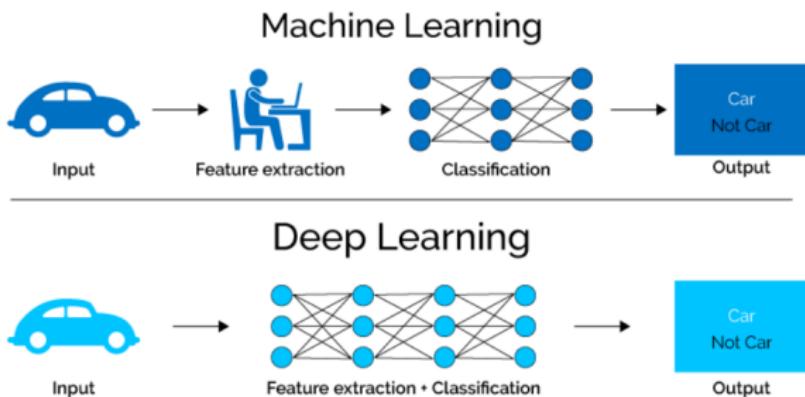
Deep Learning

- Deep learning is a sub field of Machine Learning that very closely tries to mimic human brain's working using neurons.
- These techniques focus on building Artificial Neural Networks (ANN) using several hidden layers.
- There are a variety of deep learning networks such as Multilayer Perceptron (MLP), Autoencoders (AE), Convolution Neural Network (CNN), Recurrent Neural Network (RNN).



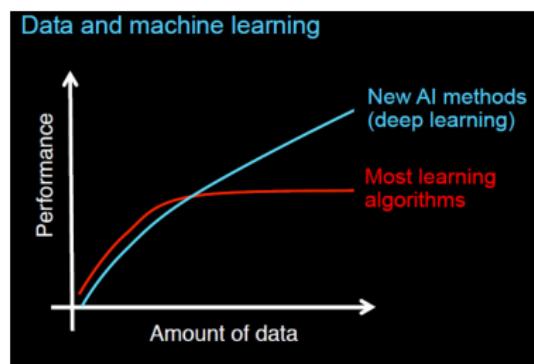
ML vs DL

- In Machine Learning, the features need to be identified by an domain expert.
- In Deep Learning, the features are learned by the neural network.



Why Deep Learning is Growing ?

- Processing power needed for Deep learning is readily becoming available using GPUs, Distributed Computing and powerful CPUs.
- Moreover, as the data amount grows, Deep Learning models seem to outperform Machine Learning models.
- Explosion of features and datasets.
- Focus on customization and real time decisioning.



Why Now?

	Stochastic Gradient Descent
1952	
1958	Perceptron
	<ul style="list-style-type: none"> Learnable Weights
1986	Backpropagation
	<ul style="list-style-type: none"> Multi-Layer Perceptron
1995	Deep Convolutional NN
	<ul style="list-style-type: none"> Digit Recognition

Neural Networks date back decades, so why the resurgence?

1. Big Data

- Larger Datasets
- Easier Collection & Storage



The Free Encyclopedia

2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable



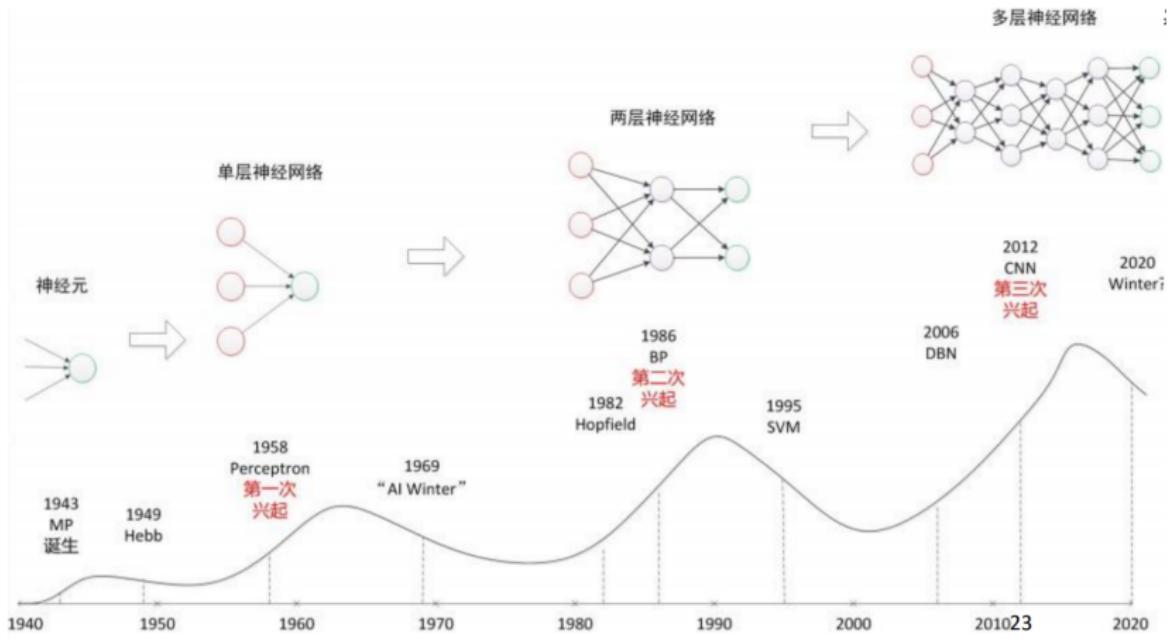
3. Software

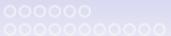
- Improved Techniques
- New Models
- Toolboxes





History





Big Guys

COMPANIES & PEOPLE



Taken in NIPS2014, from left: Yann LeCun (Facebook, NYU),
Geoffrey Hinton (Google, U of Toronto), Yoshua Bengio (U of
Montreal), Andrew Ng (Baidu)



北京大数据研究院



Outlines

Introduction

What is Deep Learning ?

Why Deep Learning is Growing ?

History

The structural building block of Deep Learning

Forward Propagation

Back Propagation

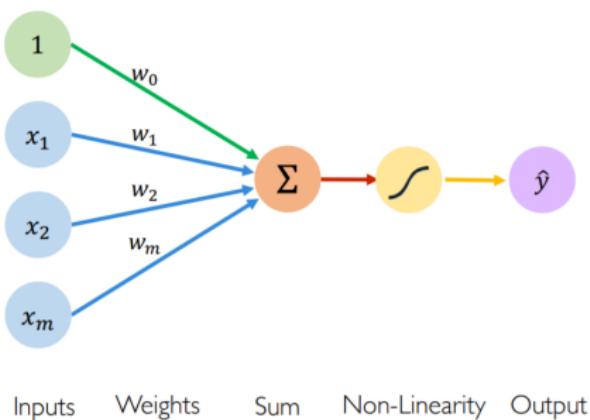
Architectures

Recurrent Neural Network (RNN)

Convolutional Neural Network (CNN)

Generative model

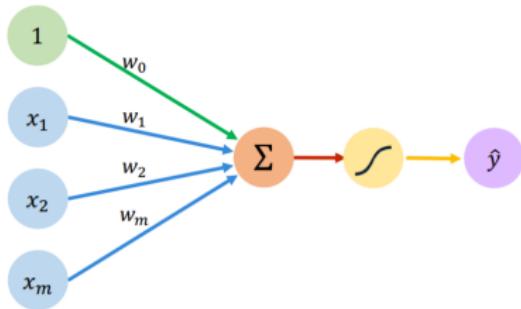
The Perceptron : Forward Propagation



$$\hat{y} = g \left(w_0 + \sum_{i=1}^m x_i w_i \right),$$

- \hat{y} is the Output,
- g is a Non-linear activation function,
- w_0 is the Bias.

The Perceptron : Forward Propagation



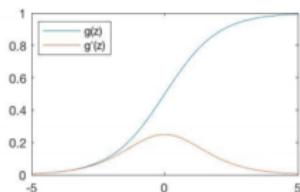
$$\hat{y} = g(w_0 + \mathbf{x}^T \mathbf{w}), \text{ where}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ and } \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}.$$



Common Activation Functions

Sigmoid Function

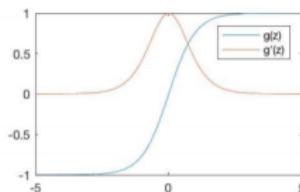


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.nn.sigmoid(z)`

Hyperbolic Tangent

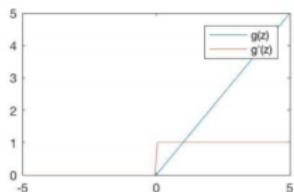


$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.nn.tanh(z)`

Rectified Linear Unit (ReLU)



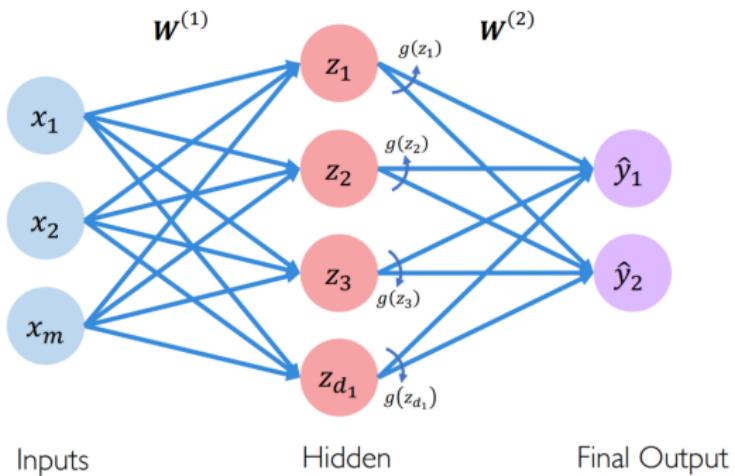
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

`tf.nn.relu(z)`

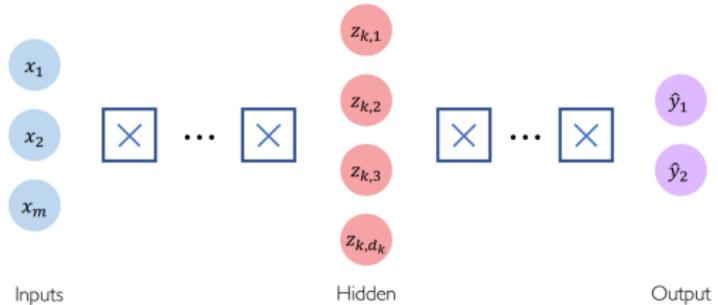
Note all activation functions are nonlinear.

Single Layer Neural Network



$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^m x_j w_{j,i}^{(1)}, \quad \hat{y}_i = g \left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j w_{j,i}^{(2)} \right).$$

Deep Neural Network



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j}) w_{j,i}^{(k)}.$$

Theorem (Universal approximation theorem (Cybenko 1980, 1989))

1. Any function can be approximated by a three-layer neural network within sufficiently high accuracy.
2. Any bounded continuous function can be approximated by a two-layer neural network within sufficiently high accuracy.



Loss Optimization

We want to find the network weights that achieve the lowest loss

$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}\left(f\left(x^{(i)}; \mathbf{W}\right), y^{(i)}\right),$$

where $\mathcal{L}\left(f\left(x^{(i)}; \mathbf{W}\right), y^{(i)}\right)$ is the loss function we defined according to the specific problem to measure the differences between output state $f\left(x^{(i)}; \mathbf{W}\right)$ and reference state $y^{(i)}$. It also can be written as

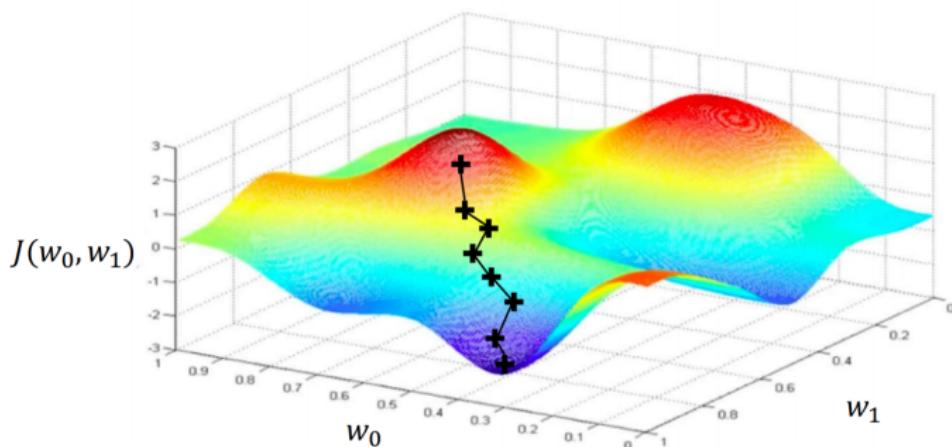
$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} C(\mathbf{W}).$$

Remember

$$\mathbf{W} = \left\{ \mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \dots \right\}.$$

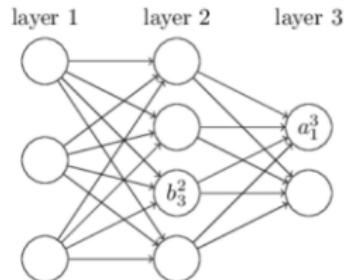
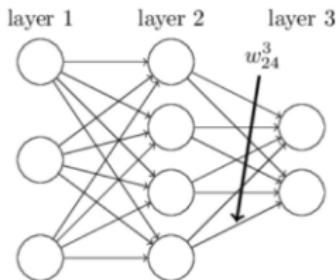
Gradient Decent

We can use Gradient Decent algorithm to find the optimal parameter \mathbf{W} .



Note that we should calculate $\frac{\partial C}{\partial \mathbf{W}}$ to update \mathbf{W} .

Notations



- w_{jk}^l is the weight for the connection from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer.
- for brevity $b_j^l = w_{j0}^l$ is the bias of the j^{th} neuron in the l^{th} layer.
- a_j^l for the activation of the j^{th} neuron in the l^{th} layer z_j^l .

$$a_j^l = g(z_j^l) = g \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right)$$

Four fundamental equations

We first define the error δ_j^l of neuron j in layer by

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l},$$

and we give the four fundamental equations of back propagation :

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad (BP1)$$

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma'(z^l) \quad (BP2)$$

$$\frac{\partial C}{\partial b_j^L} = \delta_j^l \quad (BP3)$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (BP4)$$

An equation for the error in the output layer (BP1)

The components of δ^L are given by

$$\delta^L = \nabla_a C \odot \sigma' (z^L) \quad (BP1)$$

Démonstration.

$$\begin{aligned}\delta_j^L &= \frac{\partial C}{\partial z_j^L} \\ &= \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial \sigma(z_j^L)}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)\end{aligned}$$



An equation for the error in the hidden layer (BP2)

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma' \left(z^l \right) \quad (BP2)$$

Démonstration.

$$\begin{cases} \delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ z_k^{l+1} = \left(\sum_i w_{ki}^{l+1} a_i^l \right) + b_k^{l+1} = \left(\sum_i w_{ki}^{l+1} \sigma(z_i^l) \right) + b_k^{l+1} \end{cases}$$

$$\Rightarrow \begin{cases} \delta_j^l = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ \frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l) \end{cases} \Rightarrow \delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l) \quad \square$$

The change of the cost with respect to any bias (BP3)

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (BP3)$$

Démonstration.

$$\begin{cases} \frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} \\ z_j^l = \left(\sum_k w_{jk}^l d_k^{l-1} \right) + b_j^l \Rightarrow \frac{\partial z_k^l}{\partial b_j^l} = 1 \end{cases} \Rightarrow \frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \cdot 1 = \delta_j^l. \quad \square$$

The change of the cost with respect to any weight (BP4)

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (BP4)$$

Démonstration.

$$\begin{cases} \frac{\partial C}{\partial w_{jk}^l} = \sum_i \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\ z_j^l = \left(\sum_m w_{jm}^l a_m^{l-1} \right) + b_j^l \Rightarrow \frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \end{cases}$$

$$\Rightarrow \frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}$$



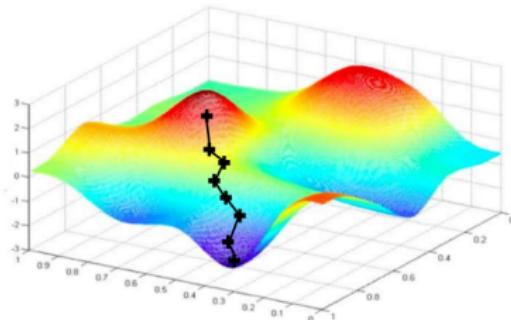
Back Propagation procedure

- 1. Input x : Set the corresponding activation a^1 for the input layer.
- 2. Feedforward : For each $l = 2, 3, \dots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.
- 3. Output error δ^L : Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.
- 4. Backpropagate the error : For each $l = L-1, L-2, \dots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
- 5. Output : The gradient of the cost function is given by $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ and $\frac{\partial C}{\partial b_j^l} = \delta_j^l$.

Gradient Descent

Algorithm

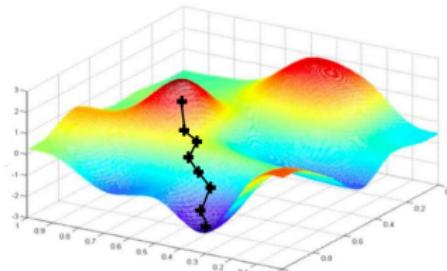
1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
4. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
5. Return weights



Stochastic Gradient Descent

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Pick single data point i
4. Compute gradient, $\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{W}}$
5. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
6. Return weights



- Mini-batches lead to fast training !
- Can parallelize computation + achieve significant speed increases on GPUs.

Outlines

Introduction

What is Deep Learning ?

Why Deep Learning is Growing ?

History

The structural building block of Deep Learning

Forward Propagation

Back Propagation

Architectures

Recurrent Neural Network (RNN)

Convolutional Neural Network (CNN)

Generative model

A sequence modeling problem : predict the next word

- **France** is where I grew up, but I now live in Boston. I speak fluent ???.
- The food was good, not bad at all.
vs. The food was bad, not good at all.

Sequence modeling : design criteria

To model sequences, we need to :

- 1. Handle variable-length sequences
- 2. Track long-term dependencies
- 3. Maintain information about order
- 4. Share parameters across the sequence

Today : Recurrent Neural Networks (RNNs) as an approach to sequence modeling problems

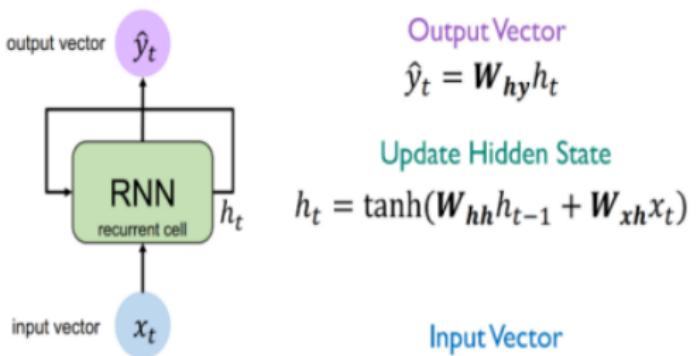
RNN state update and output

Apply a recurrence relation at every time step to process a sequence :

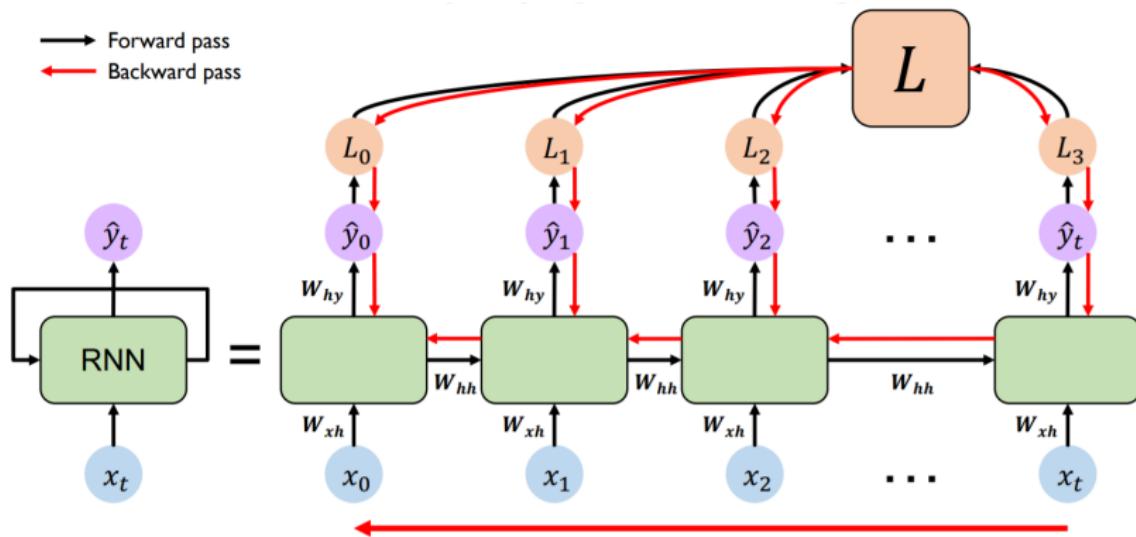
$$h_t = f_W(h_{t-1}, x_t)$$

new state function
parameterized
by W
old state
input vector at
time step t

Note : the same function and set of parameters are used at every time step

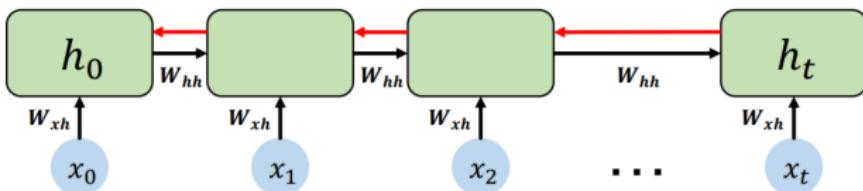


RNNs : backpropagation through time



Note that it reuse the same weight matrices at every time step

Standard RNN gradient flow : exploding gradients

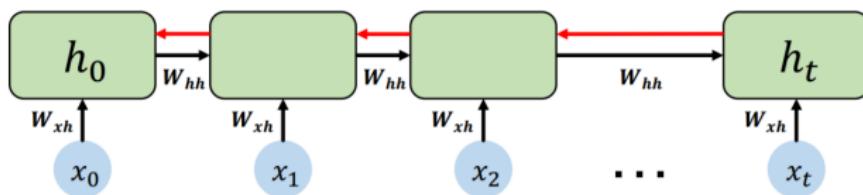


Computing the gradient wrt h_0 involves **many factors of W_{hh}** (and repeated f' !)

Many values > 1 :
exploding gradients

Gradient clipping to
scale big gradients

Standard RNN gradient flow : vanishing gradients



Computing the gradient wrt h_0 involves **many factors of W_{hh}** (and repeated f' !)

Largest singular value < 1 :
vanishing gradients

1. Activation function
 2. Weight initialization
 3. Network architecture

The problem of long-term dependencies

Why are vanishing gradients a problem?

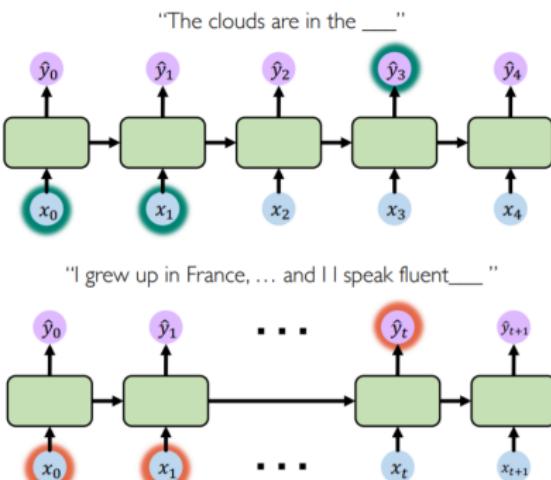
Multiply many **small numbers** together



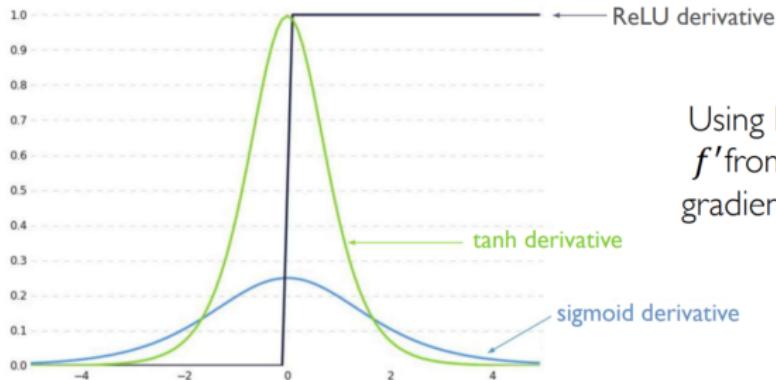
Errors due to further back time steps
have smaller and smaller gradients



Bias parameters to capture short-term dependencies



Trick 1 : activation functions



Using ReLU prevents
 f' from shrinking the
gradients when $x > 0$

Trick 2 : parameter initialization

- Initialize weights to identity matrix .
- Initialize biases to zero.

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

This helps prevent the weights from shrinking to zero.

Trick 3 : gated cells

Idea : use a more complex recurrent unit with gates to control what information is passed through

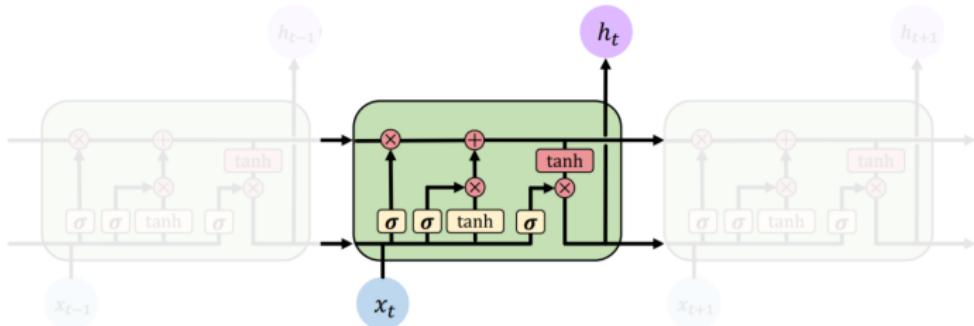
gated cell

LSTM, GRU, etc.

Long Short Term Memory (LSTMs) networks rely on a gated cell to track information throughout many time steps.

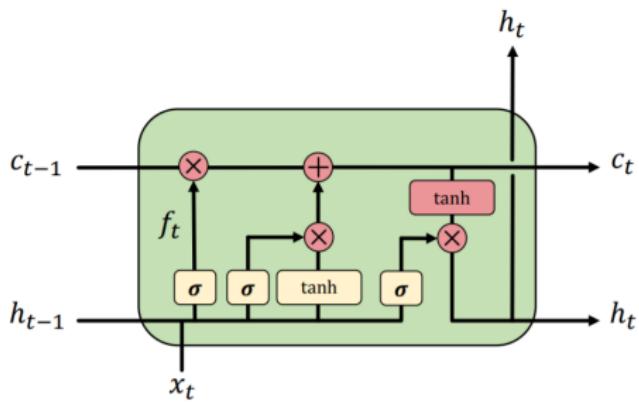
Long Short Term Memory (LSTMs)

LSTM repeating modules contain interacting layers that control information flow.



LSTM cells are able to track information throughout many time steps.

Long Short Term Memory (LSTMs)

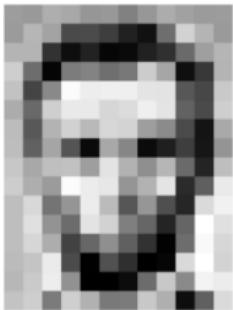


- What information to throw away ?
- What to store ?
- What to output ?

Recurrent neural networks (RNNs)

- 1. RNNs are well suited for sequence modeling tasks.
- 2. Model sequences via a recurrence relation.
- 3. Training RNNs with backpropagation through time.
- 4. Gated cells like LSTMs let us model long-term dependencies.
- 5. Models for music generation, classification, machine translation.

Tasks in Computer Vision



Input Image



187	183	174	168	160	162	159	151	172	141	186	166
185	182	162	74	76	62	38	17	112	210	180	154
180	180	89	14	24	6	10	33	48	196	189	181
206	199	6	122	131	171	120	204	164	16	66	160
156	68	137	231	237	236	239	228	227	87	71	231
173	195	97	217	230	230	234	239	239	234	97	74
188	88	179	209	185	210	211	198	130	75	20	169
188	97	165	84	10	160	134	11	31	62	32	148
159	148	191	118	106	227	178	142	182	196	36	150
205	174	195	200	236	231	149	178	220	43	96	234
160	216	116	149	236	187	86	180	79	36	218	241
186	224	147	114	227	210	127	102	36	101	295	224
180	214	179	56	103	143	96	80	3	109	249	215
187	196	230	76	1	84	47	0	8	217	280	211
181	202	237	145	0	0	13	108	205	136	243	236
186	206	123	207	177	121	133	200	175	13	96	218

Pixel Representation



Lincoln

0.8

Washington

0.1

Jefferson

0.05

Obama

0.8
0.1
0.05
0.05

- Regression : output variable takes continuous value.
- Classification : output variable takes class label. Can produce probability of belonging to a particular class.

Problems in Manual Feature Extraction

Domain knowledge

Define features

Detect features
to classify

Viewpoint variation



Scale variation



Deformation



Occlusion



Illumination conditions



Background clutter



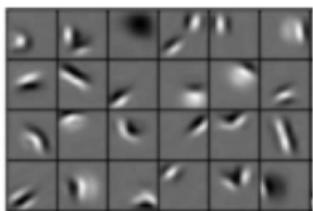
Intra-class variation



Learning Feature Representations

Can we learn a hierarchy of features directly from the data instead of hand engineering?

Low level features



Edges, dark spots

Mid level features



Eyes, ears, nose

High level features

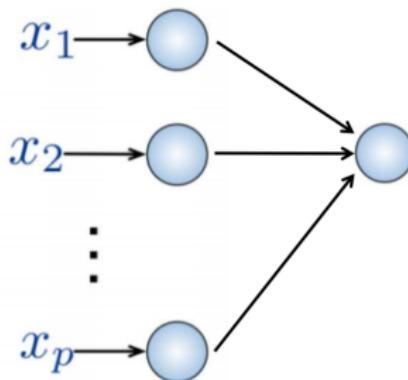


Facial structure

Fully Connected Neural Network

Input :

- 2D image.
- Vector of pixel values.



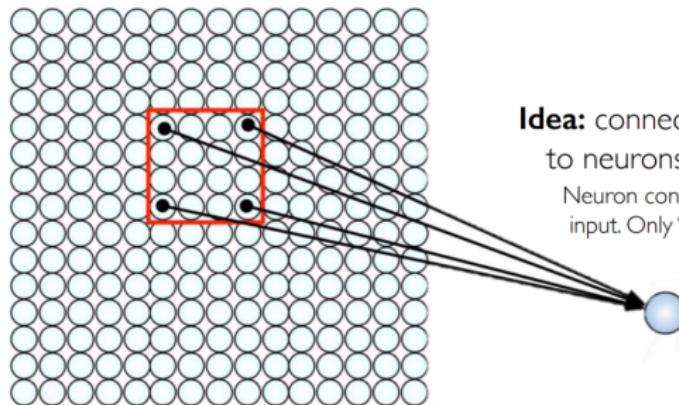
Fully Connected :

- Connect neuron in hidden layer to all neurons in input layer.
- No spatial information !
- And many, many parameters !

How can we use spatial structure in the input to inform the architecture of the network ?

Using Spatial Structure

Input: 2D image.
Array of pixel values



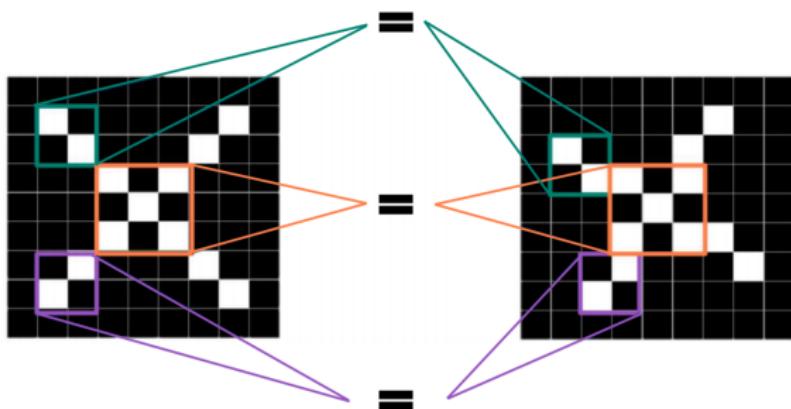
Idea: connect patches of input
to neurons in hidden layer.
Neuron connected to region of
input. Only "sees" these values.

- Connect patch in input layer to a single neuron in subsequent layer.
- Use a sliding window to define connections.
- How can we weight the patch to detect particular features ?

Applying Filters to Extract Features

- 1 Apply a set of weights (a filter) to extract local features.
- 2 Use multiple filters to extract different features.
- 3 Spatially share parameters of each filter. (features that matter in one part of the input should matter elsewhere.)

Features of X



- Image is represented as matrix of pixel values and computers are literal !
- We want to be able to classify an X as an X even if it is shifted, shrunk, rotated, deformed.

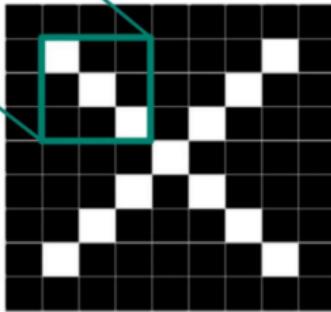
Filters to Detect X Features

filters

$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

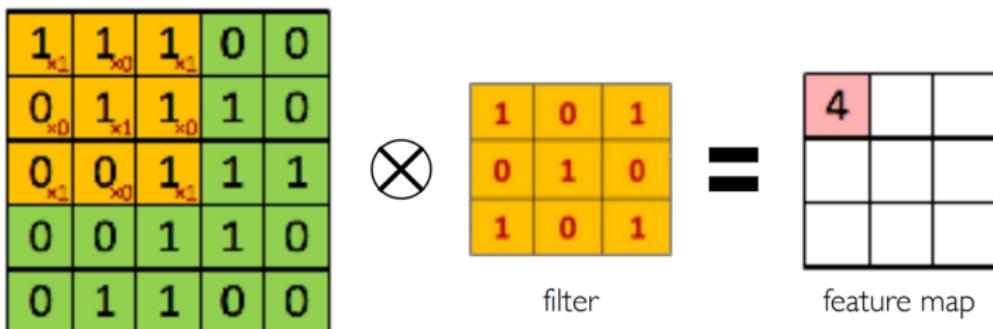
$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$



The Convolution Operation

- Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter.
- We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs :



The Convolution Operation

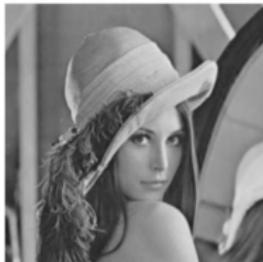
- Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter.
- We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs :

The diagram illustrates the convolution operation. On the left is a green input image (5x5) with values: 1, 1, 1, 0, 0; 0, 1, 1, 1, 0; 0, 0, 1_{x1}, 1_{x0}, 1_{x1}; 0, 0, 1_{x0}, 1_{x2}, 0_{x0}; 0, 1, 1_{x1}, 0_{x0}, 0_{x1}. In the center is a yellow filter (3x3) with values: 1, 0, 1; 0, 1, 0; 1, 0, 1. To the right is a pink feature map (3x3) with values: 4, 3, 4; 2, 4, 3; 2, 3, 4. Between the input and filter is a large black circle with a white cross inside, representing element-wise multiplication. To the right of the filter is a double equals sign (=), and below the feature map is the label "feature map".



Producing Feature Maps

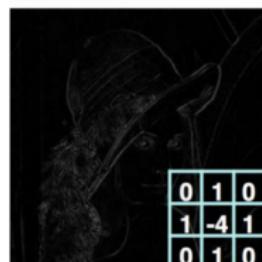
Different filters have different effects !



Original



Sharpen

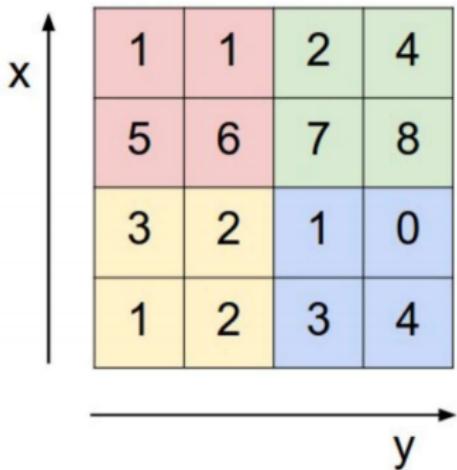


Edge Detect



"Strong" Edge
Detect

Pooling

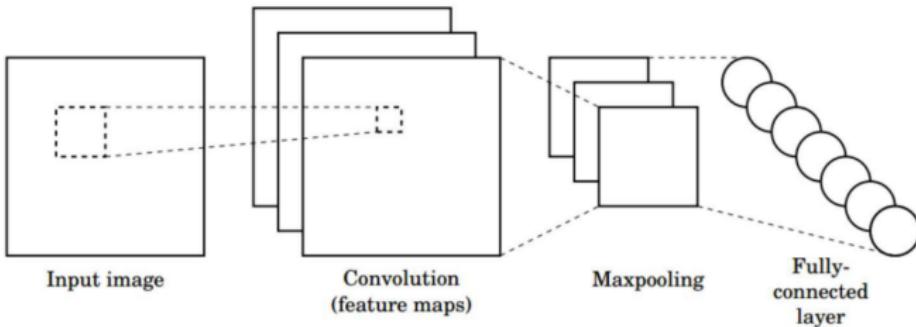


max pool with 2x2 filters
and stride 2

6	8
3	4

- 1) Reduced dimensionality
- 2) Spatial invariance

CNNs for Classification

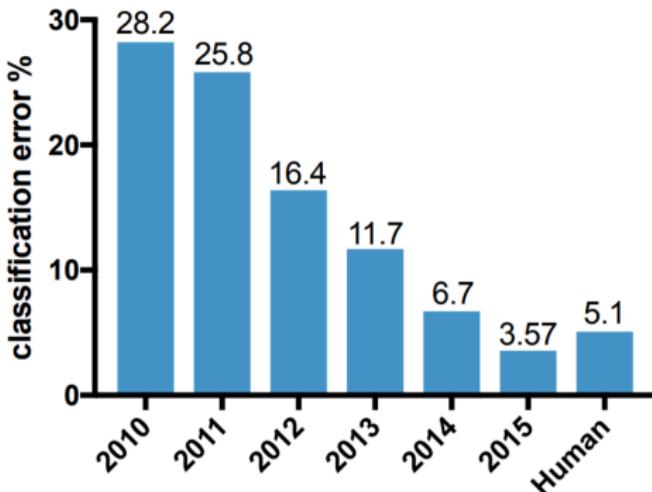


- 1. Convolution : Apply filters with learned weights to generate feature maps.
- 2. Non-linearity : Often ReLU.
- 3. Pooling : Downsampling operation on each feature map.

Train model with image data.

Learn weights of filters in convolutional layers.

ImageNet Challenge : Classification Task



2012: AlexNet. First CNN to win.

- 8 layers, 61 million parameters

2013: ZFNet

- 8 layers, more filters

2014: VGG

- 19 layers

2014: GoogLeNet

- "Inception" modules
- 22 layers, 5 million parameters

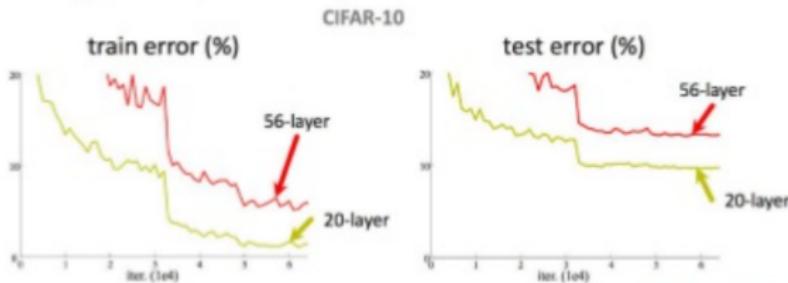
2015: ResNet

- 152 layers



The problem when go deeper

- Simply stack layers after layers

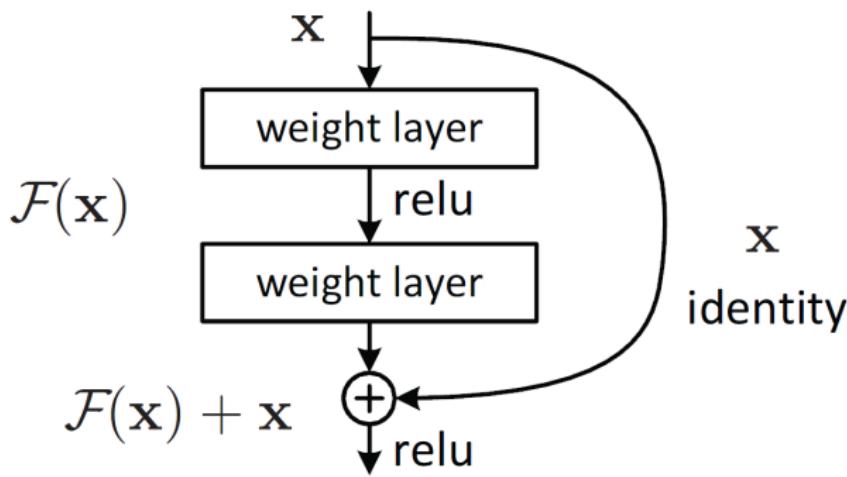


- Plain nets: stacking of 3x3 conv layers
- 56-layers got higher training and test error

Problem of going deeper?

- Vanishing Gradient
 - Most neurons will die during back propagating the weights.

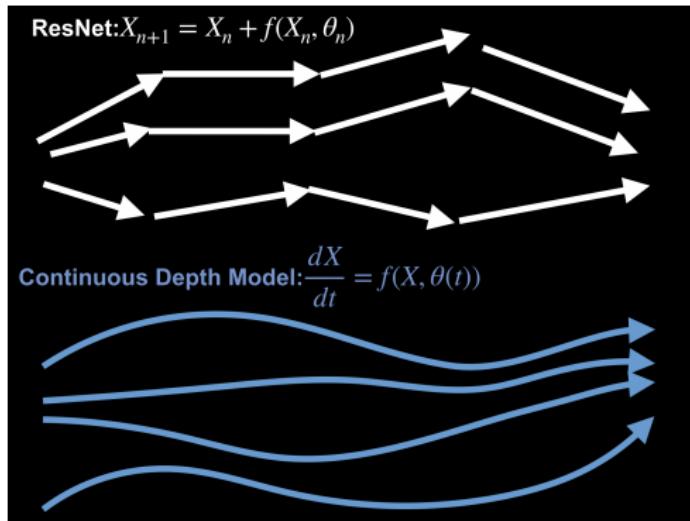
Better network structure for learning : ResNet



- Introduce shortcut connections (exists in prior literature in various forms).
- Key invention is to skip 2 layers. Skipping single layer do not give much improvement for some reason.

ResNet vs ODENet

- **ResNet :** $\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t),$
- **ODENet :** $\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta).$



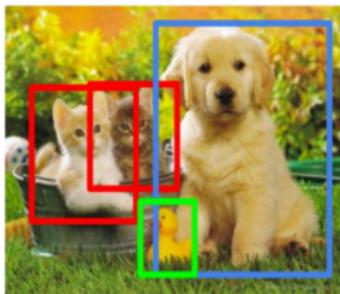
Beyond Classification

Semantic Segmentation



CAT

Object Detection



CAT, DOG, DUCK

Image Captioning



The cat is in the grass.

Supervised vs unsupervised learning

Supervised Learning

- Data : (x, y)
 x is data, y is label.
- Goal : Learn function to map $x \Rightarrow y$.
- Examples : Classification, regression, object detection, semantic segmentation, etc.

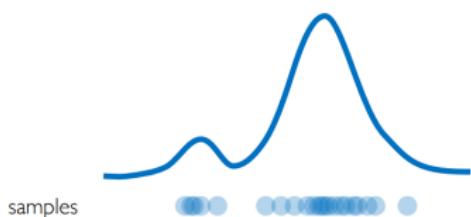
Unsupervised Learning

- Data : x
 x is data, no labels !
- Goal : Learn some hidden or underlying structure of the data.
- Examples : Clustering, feature or dimensionality reduction, etc.

Generative modeling

Goal : Take as input training samples from some distribution and learn a model that represents that distribution

Density Estimation



Sample Generation



Input samples

Training data $\sim P_{data}(x)$

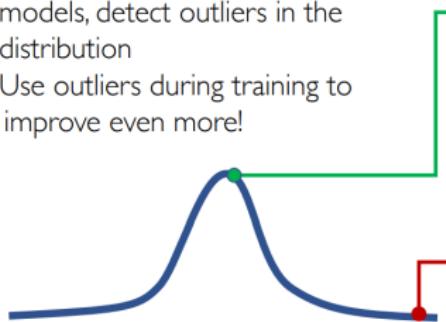
Generated samples

Generated $\sim P_{model}(x)$

How can we learn $p_{model}(x)$ similar to $p_{data}(x)$?

Why generative models? Outlier detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!



95% of Driving Data:

- (1) sunny, (2) highway, (3) straight road



Detect outliers to avoid unpredictable behavior when training



Edge Cases



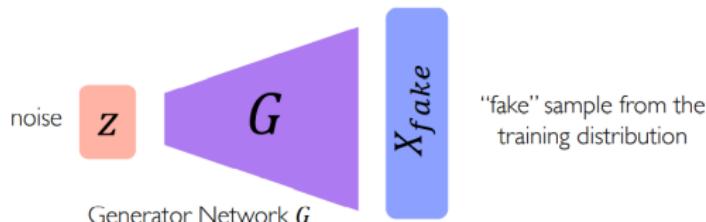
Harsh Weather



Pedestrians

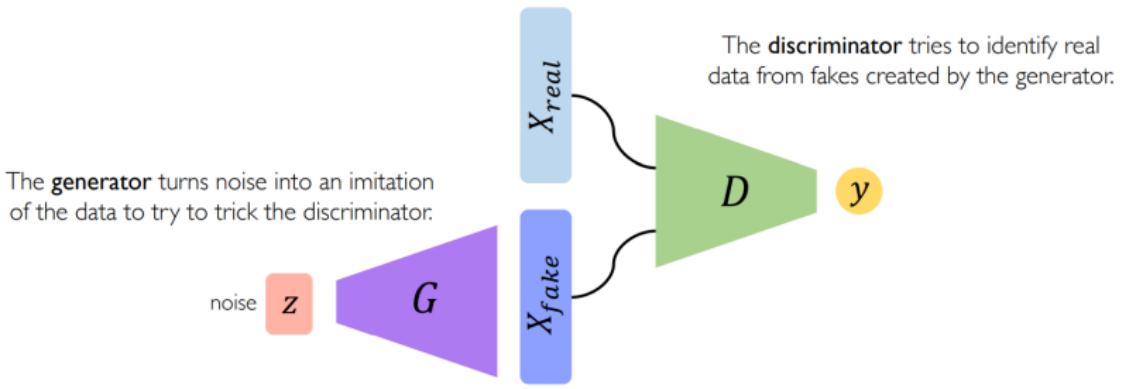
What if we just want to sample

- Idea : do not explicitly model density, and instead just sample to generate new instances.
- Problem : want to sample from complex distribution can not do this directly !
- Solution : sample from something simple (noise), learn a transformation to the training distribution.



Generative Adversarial Networks (GANs)

- Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.



Training GANs

- Neural Network Discriminator tries to identify real data from fakes created by the generator.
- Neural Network Generator tries to create imitations of data to trick the discriminator.

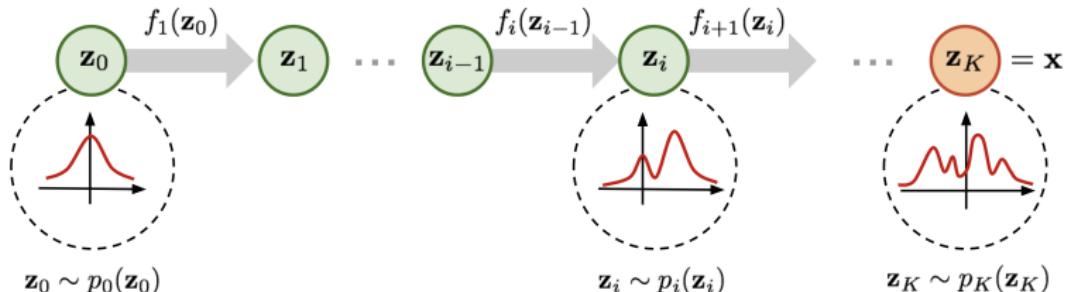


-

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

What if we want the model density

Flow based model



- f is bijective and differentiable functions.
- We have the explicitly model density $p_{\mathbf{z}_k}(\mathbf{z}_k)$.

CVF

Theorem

(Change of Variable Theorem) Given any $z_0 \sim p_{z_0}(z_0)$, if the transformation $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is differentiable and bijective, then the distribution of $x = f(z_0)$ is $p_x(x) = \frac{p_{z_0}(z_0)}{\left| \det \frac{\partial f}{\partial z_0} \right|}$.

- The density probability follows

$$\log p_{z_k}(z_k) = \log p_{z_0}(z_0) - \sum_{i=1}^k \left| \det \frac{\partial f_i}{\partial z_{i-1}} \right|.$$

- we want to find parameter λ of neural network f_i satisfies :

$$\arg \min_{\lambda} D_{KL}(p_{data} || p_{z_k}^{\lambda}) \approx \arg \max_{\lambda} \sum_{i=1}^m \log p_{z_k}^{\lambda}(x^{(i)}),$$

FUTURE : LONG WAY TO GO

- Theory : why deep learning works ? How ? Interpretability.
- Capacity : how deep is enough ? / Network structure selection.
- Manipulate network.
- Geometry of loss function.
- Deep learning with less data ?
- Interdisciplinary fields : statistics, physics, geometry, game theory...
- More exciting applications ! Healthcare, industrial process, finance, AI game play, animation...

MA233 Introduction to Big Data Science Mathematical Preliminary

Zhen Zhang

Southern University of Science and Technology

Outlines

Linear Algebra

Probability and Information Theory

Statistics and Machine Learning

References

Inner Product and Euclidean Norm

For $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, their inner product is defined as

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i = \mathbf{x}^T \mathbf{y}.$$

It satisfies

1. (Commutativity) $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$;
2. (Scalar Multiplication) $\langle \lambda \mathbf{x}, \mathbf{y} \rangle = \lambda \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, \lambda \mathbf{y} \rangle$;
3. (Bilinearity) $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$,
 $\langle \mathbf{x}, \mathbf{y} + \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle$;
4. (Positivity) $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$, and $\langle \mathbf{x}, \mathbf{x} \rangle = 0$ iff $\mathbf{x} = \mathbf{0}$.

The Euclidean norm (l_2 -norm) is $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

Linear Independency and Orthogonality

- Linear Independency :

A set of vectors $U = \{bx_1, \dots, x_k\}$ is linearly independent if for $\forall i$, x_i does not lie in the space spanned by $x_1, \dots, x_{i-1}, x_{i+1}, x_k$. We say U spans a subspace V if V is the span of the vectors in U . U is a basis of V if it is both independent and spans V . The dimension of V is the size of a basis of V (i.e., the number of linearly independent vectors in U).

- Orthogonality :

We say that U is an orthogonal set if for all $i \neq j$, $\langle x_i, x_j \rangle = 0$. We say that U is an orthonormal set if it is orthogonal and if for every i , $\|x_i\| = 1$.

Gram-Schmidt Orthogonalization

Given a set of linear independent vectors $V = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, we can apply Gram-Schmidt orthogonalization to obtain an orthonormal set $\{\mathbf{u}_1, \dots, \mathbf{u}_k\}$ which have the same span as $\text{span} V$. The procedure is as follows :

1. Let $\mathbf{u}_1 = \mathbf{v}_1 / \|\mathbf{v}_1\|$;
2. For $j = 2$ to k , project \mathbf{v}_j onto $\text{span}\{\mathbf{u}_1, \dots, \mathbf{u}_{j-1}\}$ and find the perpendicular part $\tilde{\mathbf{u}}_j = \mathbf{v}_j - \sum_{i=1}^{j-1} \langle \mathbf{u}_i, \mathbf{v}_j \rangle \mathbf{u}_i$, then normalize it to be $\mathbf{u}_j = \tilde{\mathbf{u}}_j / \|\tilde{\mathbf{u}}_j\|$;

This procedure is summarized in the matrix form : $Q = AP$, where $Q = (\mathbf{u}_1 \cdots \mathbf{u}_k) \in \mathbb{R}^{k \times k}$ is an orthogonal matrix whose columns are given by \mathbf{u}_i 's, $A = (\mathbf{v}_1 \cdots \mathbf{v}_k) \in \mathbb{R}^{k \times k}$ is a nonsingular matrix whose columns are given by \mathbf{v}_i 's, and $P \in \mathbb{R}^{k \times k}$ is an upper tridiagonal matrix whose upper tridiagonal (i, j) -entry is given by $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$. This is known as the QR factorization : $A = QR$ where $R = P^{-1}$.

Concepts in Matrix

- Kernel and Range :

Given a matrix $A \in \mathbb{R}^{n \times d}$, the range of A ($\text{Range}(A)$) is the span of its columns and the kernel of A ($\text{Ker}(A)$) is the subspace of all vectors that satisfy $Ax = \mathbf{0}$. The rank of A is the dimension of its range and is denoted by $\text{rank}(A)$ or $r(A)$ for short.

- Symmetric and Definite Matrix :

A is symmetric if $A = A^T$. A symmetric matrix $A \in \mathbb{R}^{d \times d}$ is positive definite if for all $x \in \mathbb{R}^d$, $\langle x, Ax \rangle \geq 0$, and equality holds if and only if ("iff") $x = \mathbf{0}$. This definition can be relaxed to give semidefiniteness : A symmetric matrix $A \in \mathbb{R}^{d \times d}$ is positive semidefinite if for all $x \in \mathbb{R}^d$, $x^T Ax \geq 0$. In particular, all the eigenvalues of a positive definite (resp. semidefinite) matrix are positive (resp. nonnegative). And $A = BB^T$ for some matrix B . (See next slides for eigen-decomposition)

Eigenvalues and Eigenvectors

Let $A \in \mathbb{R}^{d \times d}$ be a squared matrix. A nonzero vector $\mathbf{x} \in \mathbb{R}^d$ is an eigenvector of A with a corresponding eigenvalue λ if $A\mathbf{x} = \lambda\mathbf{x}$.

Theorem

(Eigen-decomposition or Spectral Decomposition) If $A \in \mathbb{R}^{d \times d}$ is a symmetric matrix of rank k , then there exists an orthogonal basis of \mathbb{R}^d , $\mathbf{x}_1, \dots, \mathbf{x}_d$, such that each \mathbf{x}_i is an eigenvector of A .

Furthermore, A can be written as $A = \sum_{i=1}^d \lambda_i \mathbf{x}_i \mathbf{x}_i^T$, where each λ_i is the eigenvalue corresponding to the eigenvector \mathbf{x}_i . In matrix form, this is $A = UDU^T$, where the columns of U are the vectors $\mathbf{x}_1, \dots, \mathbf{x}_d$, and $D = \text{diag}\{\lambda_1, \dots, \lambda_d\}$ is a diagonal matrix. Finally, $r(A)$ is the number of nonzero λ_i 's, and the corresponding eigenvectors span the range of A . The eigenvectors corresponding to the zero eigenvalues span the null space of A .

Singular Values Decomposition (SVD)

Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank r . Unit (nonzero) vector $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ are called right and left singular vectors of A with corresponding singular values σ if $A\mathbf{v} = \sigma\mathbf{v}$ and $\mathbf{u}^T A = \sigma\mathbf{u}^T$.

Theorem

(SVD) Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank r . Then there exist orthonormal sets of right and left singular vectors of A , say $\{\mathbf{v}_1, \dots, \mathbf{v}_r\}$ and $\{\mathbf{u}_1, \dots, \mathbf{u}_r\}$ respectively, and the corresponding singular values $\sigma_1, \dots, \sigma_r$, such that $A = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. In matrix form, this is $A = UDV^T$, where the columns of U are the vectors $\mathbf{u}_1, \dots, \mathbf{u}_r$, the columns of V are the vectors $\mathbf{v}_1, \dots, \mathbf{v}_r$, and $D = \text{diag}\{\sigma_1, \dots, \sigma_r\}$ is a diagonal matrix.

Corollary

The squared matrices $A^T A \in \mathbb{R}^{n \times n}$ and $AA^T \in \mathbb{R}^{m \times m}$ have (a subset of) the eigenvectors $\{\mathbf{v}_1, \dots, \mathbf{v}_r\}$ and $\{\mathbf{u}_1, \dots, \mathbf{u}_r\}$ respectively, corresponding to the same eigenvalues $\sigma_1^2, \dots, \sigma_r^2$.

Reyleigh Quotient

Theorem

Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank r . Define $\mathbf{v}_1 = \arg \max_{\mathbf{v} \in \mathbb{R}^n: \|\mathbf{v}\|=1} \|A\mathbf{v}\|$,

$\mathbf{v}_2 = \arg \max_{\substack{\mathbf{v} \in \mathbb{R}^n: \|\mathbf{v}\|=1 \\ \langle \mathbf{v}, \mathbf{v}_1 \rangle = 0}} \|A\mathbf{v}\|, \dots, \mathbf{v}_r = \arg \max_{\substack{\mathbf{v} \in \mathbb{R}^n: \|\mathbf{v}\|=1 \\ \forall i < r, \langle \mathbf{v}, \mathbf{v}_i \rangle = 0}} \|A\mathbf{v}\|.$ Then $\mathbf{v}_1, \dots, \mathbf{v}_r$

is an orthonormal set of right singular vectors of A .

Remark : (Reyleigh Quotient) If $A \in \mathbb{R}^{n \times n}$ is a squared matrix, then its eigenvalues can be found as the solution to the following optimization problems :

$$\lambda_1 = \max_{\mathbf{v} \in \mathbb{R}^n: \|\mathbf{v}\|=1} \mathbf{v}^T A \mathbf{v}, \quad \lambda_2 = \max_{\substack{\mathbf{v} \in \mathbb{R}^n: \|\mathbf{v}\|=1 \\ \langle \mathbf{v}, \mathbf{v}_1 \rangle = 0}} \mathbf{v}^T A \mathbf{v},$$

$$\dots, \quad \lambda_n = \max_{\substack{\mathbf{v} \in \mathbb{R}^n: \|\mathbf{v}\|=1 \\ \forall i < n, \langle \mathbf{v}, \mathbf{v}_i \rangle = 0}} \mathbf{v}^T A \mathbf{v}.$$

Power Method - Dominant Eigenvalue

Assume the eigenvalues of A can be sorted according to their magnitudes : $|\lambda_1| > |\lambda_2| \geqslant |\lambda_3| \geqslant \cdots \geqslant |\lambda_n| \geqslant 0$. If The corresponding eigenvectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ form a basis of \mathbb{R}^n , then any vector can be expressed as $\mathbf{x} = \sum_{i=1}^n \beta_i \mathbf{v}_i$. Multiplying \mathbf{x} by A on the left for n times, we have an idea

$$A^k \mathbf{x} = \sum_{i=1}^n \beta_i \lambda_i^k \mathbf{v}_i = \lambda_1^k \sum_{i=1}^n \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \mathbf{v}_i \sim \lambda_1^k \beta_1 \mathbf{v}_1, \quad k \rightarrow \infty$$

1. For any nonzero vector \mathbf{x} , let $\mathbf{y}^{(0)} = \mathbf{x}$;
2. For $k = 0, 1, \dots$: compute the smallest integer p_k such that satisfying $y_{p_k}^{(k)} = \|\mathbf{y}^{(k)}\|_\infty$, then compute $\mathbf{x}^{(k)} = \mathbf{y}^{(k)} / y_{p_k}^{(k)}$, $\mathbf{y}^{(k+1)} = A\mathbf{x}^{(k)}$, $\mu^{(k+1)} = y_{p_k}^{(k+1)}$.

It can be shown that $\lim_{k \rightarrow \infty} \mu^{(k)} = \lambda_1$ and $\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{v}_1 / \|\mathbf{v}_1\|_\infty$. Other methods : QR factorization, Householder transformations

Linear Systems

A system of m linear algebraic equations in n unknown variables can be written in the matrix form : $\mathbf{Ax} = \mathbf{b}$, where $A \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$. This system has solutions iff $r([A, \mathbf{b}]) = r(A)$.

Theorem (Solvability Condition)

- If $\text{Ker}(A) = \{0\}$, then $\mathbf{Ax} = \mathbf{b}$ either has a unique solution or has no solution. It has a solution iff $\mathbf{b} \perp \text{Ker}(A^T)$.*
- If $\text{Ker}(A) \neq \{0\}$, then $\mathbf{Ax} = \mathbf{b}$ either has infinitely many solutions or has no solution. It has a solution iff $\mathbf{b} \perp \text{Ker}(A^T)$.*

If $A \in \mathbb{R}^{n \times n}$ is a square matrix, we have a simple rule : the system has a unique solution iff $\det A \neq 0$. If the solution exists, we can solve it by $\mathbf{x} = A^{-1}\mathbf{b}$, where A^{-1} is the inverse of A satisfying $A^{-1}A = AA^{-1} = I$.

Moreover, we can find it by Cramer's rule : $x_i = \frac{\det A_i}{\det A}$, where A_i is the matrix obtained from A by replacing its i -th column with \mathbf{b} . The direct application of this formula requires $O(n!)$ arithmetic operations to find $\det A$, which is unacceptable for large n .

Gaussian Elimination

Gaussian Elimination is an algorithm that can reduce the computational complexity of solving linear systems to $O(n^3)$. It is equivalent to perform an elementary row transformation for A to obtain an upper or lower triangular matrix.

Another way to view Gaussian elimination is the LU decomposition : The k -th row transformation can be represented by a left multiplication by $M^{(k)}$, where $M^{(k)}$ is a lower triangular matrix with its diagonal entries being all 1's; after n operations, A is transformed to an upper triangular matrix U , i.e., $M^{(n)} \dots M^{(2)} M^{(1)} A = U$; since the inverse of a lower triangular matrix is also a lower triangular matrix, we have $A = LU$, where $L = (M^{(1)})^{-1} (M^{(2)})^{-1} \dots (M^{(n)})^{-1}$ with its diagonal entries being all 1's.

LU Decomposition

Theorem

An $n \times n$ nonsingular matrix A can be decomposed uniquely in the form $A = LU$, where

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n,1} & \cdots & l_{n,n-1} & 1 \end{pmatrix}, U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1,n} \\ 0 & u_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & 0 & u_{n,n} \end{pmatrix}.$$

The computational complexity for LU decomposition is $O(n^3)$. If A is symmetric, we have Cholesky decomposition $A = LL^T$, where L is a lower diagonal matrix.

Iterative Solver

We introduce two commonly used iterative methods : Jacobi iteration and Gauss-Seidel iteration. First we write $A = D - L - U$, where $D = \text{diag}\{a_{11}, \dots, a_{nn}\}$, $L = \{l_{ij}\}$ and $U = \{u_{ij}\}$ are the lower and upper diagonal parts of $-A$ respectively. That means $l_{ij} = -a_{ij}$ for $i > j$ and 0 for $i \leq j$, $u_{ij} = -a_{ij}$ for $i < j$ and 0 for $i \geq j$.

- Jacobi iteration : Rewrite the linear system as $D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b}$, if D^{-1} exists ($a_{ii} \neq 0$), then we can build the iteration $\mathbf{x}^{(k)} = D^{-1}(L + U)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}$, $k = 1, 2, \dots$
- Gauss-Seidel iteration : Rewrite the linear system as $(D - L)\mathbf{x} = U\mathbf{x} + \mathbf{b}$, if $(D - L)^{-1}$ exists ($a_{ii} \neq 0$), then we can build the iteration $\mathbf{x}^{(k)} = (D - L)^{-1}U\mathbf{x}^{(k-1)} + (D - L)^{-1}\mathbf{b}$, $k = 1, 2, \dots$

Both are easy to implement in component form and can be written as $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$, with $T = D^{-1}(L + U)$ for Jacobi iteration and $(D - L)^{-1}U$ for Gauss-Seidel iteration. (Fixed point iteration !)

Vector Norms

Vector Norm is a non-negative real-valued function on \mathbb{R}^n , usually denoted by $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$, with the following properties :

1. (Positivity) $\|\mathbf{x}\| \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$; $\|\mathbf{x}\| = 0$ iff $\mathbf{x} = \mathbf{0}$;
2. (Homogeneity) $\|\alpha \mathbf{x}\| = |\alpha| \|\mathbf{x}\|$ for $\forall \alpha \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$;
3. (Triangle Inequality) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Examples :

- l_2 -norm : $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}}$;
- l_1 -norm : $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$;
- l_∞ -norm : $\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$;

Theorem

Define l_p -norm as $\|\mathbf{x}\|_p = \left(\sum_{i=1}^n x_i^p \right)^{\frac{1}{p}}$, it is really a norm for $p \leq 1$.

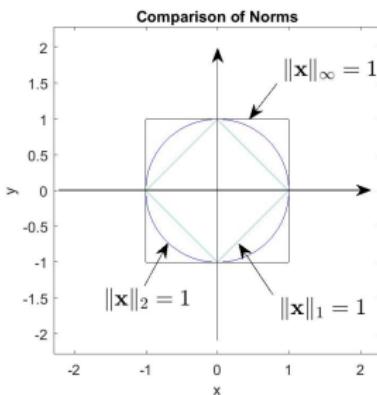
Vector Norms (Cont')

Remark : i) ℓ_p -norm is not a norm for $0 < p \leq 1$, since the triangular inequality is not satisfied. It is called semi-norm.
ii) Useful to define ℓ_0 -norm : $\|\mathbf{x}\|_0 = \#\{1 \leq i \leq n : x_i \neq 0\}$.
Induced Distances : $\text{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$, e.g., ℓ_2 -distance is

$$\|\mathbf{x} - \mathbf{y}\|_2 = \left(\sum_{i=1}^n (x_i - y_i)^2 \right)^{\frac{1}{2}}.$$

Theorem

$$\forall \mathbf{x} \in \mathbb{R}^n, \|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1.$$



Matrix Norms

Matrix Norm is a non-negative real-valued function on $\mathbb{R}^{n \times m}$, usually denoted by $\|\cdot\| : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}$, with the following properties :

1. (Positivity) $\|A\| \geq 0$ for all $A \in \mathbb{R}^{n \times m}$; $\|A\| = 0$ iff $A = 0$;
2. (Homogeneity) $\|\alpha A\| = |\alpha| \|A\|$ for $\forall \alpha \in \mathbb{R}$ and $A \in \mathbb{R}^{n \times m}$;
3. (Triangle Inequality) $\|A + B\| \leq \|A\| + \|B\|$ for $\forall A, B \in \mathbb{R}^{n \times m}$;
4. $\|AB\| \leq \|A\| \|B\|$ for $\forall A, B \in \mathbb{R}^{n \times m}$.

Theorem

If $\|\cdot\|$ is a vector norm on \mathbb{R}^n , then $\|A\| = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\| = \max_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|}$ is a matrix norm (called natural norm).

Corollary

$\|A\mathbf{x}\| \leq \|A\| \|\mathbf{x}\|$ for $\forall A \in \mathbb{R}^{n \times m}$ and $\mathbf{x} \in \mathbb{R}^n$.

Matrix Norm (Cont')

Examples :

- ℓ_1 -norm : $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$;
- ℓ_∞ -norm : $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$;

ℓ_2 -norm is not trivial. For a symmetric matrix A , define its spectral radius as $\rho(A) = \max_{1 \leq i \leq n} \lambda_i$, where $\lambda_i (i = 1, \dots, n)$ are the eigenvalues of A . Then

Theorem

1. $\|A\|_2 = \sqrt{\rho(A^T A)}$;
2. $\rho(A) \leq \|A\|$ for any natural norm $\|\cdot\|$.

Theorem (Convergence of Jacobi and Gauss-Seidel Iterations)

The Jacobi and Gauss-Seidel iterations converge to the unique solution of $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ iff $\rho(T) < 1$. Moreover, we have the error estimate $\|\mathbf{x} - \mathbf{x}^{(k)}\| \leq \frac{\|T\|^k}{1-\|T\|} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|$.

Matrix Calculus

By convention, the lowercase letter a denotes a scalar, the bold letter $\mathbf{x} = (x_1, \dots, x_n)^T$ denotes a column vector, and the uppercase letter $A = (a_{ij})$ denotes an $m \times n$ matrix. Assume \mathbf{x} (or x) is independent variables, \mathbf{a}, \mathbf{b} , etc. are constant vectors, A, B , etc. are constant matrices, $f(x)$, $g(x)$, $\mathbf{u}(\mathbf{x})$, and $\mathbf{v}(\mathbf{x})$ are (scalar or vector valued) functions of \mathbf{x} (or x)

- Vector-by-vector formula : (resulting in matrix $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \left(\frac{\partial y_i}{\partial x_j} \right)$)
 1. Linear vector-valued functions : $\frac{\partial \mathbf{a}}{\partial \mathbf{x}} = 0$, $\frac{\partial(A\mathbf{x})}{\partial \mathbf{x}} = A$,
 $\frac{\partial(\mathbf{x}^T A)}{\partial \mathbf{x}} = A^T$,
 2. Nonlinear vector-valued functions : $\frac{\partial \mathbf{u}}{\partial \mathbf{x}} = \left(\frac{\partial u_i}{\partial x_j} \right)$ is Jacobian,
 $\frac{\partial(a\mathbf{u}(\mathbf{x}) + b\mathbf{v}(\mathbf{x}))}{\partial \mathbf{x}} = a \frac{\partial(\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}} + b \frac{\partial(\mathbf{v}(\mathbf{x}))}{\partial \mathbf{x}}$,
 $\frac{\partial(f(\mathbf{x})\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}} = f(\mathbf{x}) \frac{\partial(\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}} + \mathbf{u} \frac{\partial(f(\mathbf{x}))}{\partial \mathbf{x}}$, $\frac{\partial(A\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}} = A \frac{\partial(\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}}$
 3. Chain rule : $\frac{\partial \mathbf{g}(\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}} = \frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}} \frac{\partial(\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}}$,

Matrix Calculus

- Scalar-by-vector : (resulting in row vector $\frac{\partial y}{\partial \mathbf{x}} = (\nabla_{\mathbf{x}} y)^T$)
Some of the formula can be obtained from the previous page by letting the numerator be of dimension one, the others are :

- Inner product : $\frac{\partial(\mathbf{a}^T \mathbf{x})}{\partial \mathbf{x}} = \frac{\partial(\mathbf{x}^T \mathbf{a})}{\partial \mathbf{x}} = \mathbf{a}^T$, $\frac{\partial \mathbf{a}^T \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{a}^T \frac{\partial(\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}}$,

$$\frac{\partial(\mathbf{u}(\mathbf{x})^T A \mathbf{v}(\mathbf{x}))}{\partial \mathbf{x}} = \mathbf{u}^T A \frac{\partial(\mathbf{v}(\mathbf{x}))}{\partial \mathbf{x}} + \mathbf{v}^T A^T \frac{\partial(\mathbf{u}(\mathbf{x}))}{\partial \mathbf{x}}$$
- Quadratic forms : $\frac{\partial(\mathbf{x}^T A \mathbf{x})}{\partial \mathbf{x}} = \mathbf{x}^T (A + A^T)$, $\frac{\partial(\mathbf{x}^T A \mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{x}^T A$ if A is symmetric,

$$\frac{\partial(\mathbf{a}^T \mathbf{x} \mathbf{x}^T \mathbf{b})}{\partial \mathbf{x}} = \mathbf{x}^T (\mathbf{a} \mathbf{b}^T + \mathbf{b} \mathbf{a}^T)$$

$$\frac{\partial(A\mathbf{x}+\mathbf{b})^T C(D\mathbf{x}+\mathbf{e})}{\partial \mathbf{x}} = (D\mathbf{x} + \mathbf{e})^T C^T A + (A\mathbf{x} + \mathbf{b})^T C D$$
- l_2 norm : $\frac{\partial \|\mathbf{x} - \mathbf{a}\|}{\partial \mathbf{x}} = \frac{(\mathbf{x} - \mathbf{a})^T}{\|\mathbf{x} - \mathbf{a}\|}$
- 2nd order derivative (resulting in a matrix) :

$$\frac{\partial^2(\mathbf{x}^T A \mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} = (A + A^T)$$
, $\frac{\partial^2(\mathbf{x}^T A \mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} = 2A$ if A is symmetric,

$$\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j} = H = \left(\frac{\partial f}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \right)$$
 is the Hessian matrix.

Trace and Frobenius inner product

Trace is defined as the sum of the diagonal entries in a matrix :

$$\text{tr}(A) = \sum_{i=1}^n a_{ii}$$

- $\text{tr}(A) = \text{tr}(A^T)$
- $\text{tr}(AB) = \text{tr}(BA)$, $\text{tr}(ABC) = \text{tr}(CAB) = \text{tr}(BCA)$
- $\frac{\partial \text{tr}(AB)}{\partial A} = B^T$, $\frac{\partial \text{tr}(ABA^T C)}{\partial A} = CAB + C^T AB^T$
- $a = \text{tr}(a)$ for scalar a , as a result,
 $\langle \mathbf{x}, \mathbf{y} \rangle = \text{tr}(\mathbf{x}^T \mathbf{y}) = \text{tr}(\mathbf{y} \mathbf{x}^T)$ (useful formula)

The Frobenius inner product is defined for matrices :

$$\langle A, B \rangle_F = \text{tr}(AB^T) = \sum_{i,j=1}^n a_{ij} b_{ij}. \text{ The induced norm is called}$$

$$\text{Frobenius norm : } \|A\|_F = \sqrt{\text{tr}(AA^T)} = \sqrt{\sum_{i,j=1}^n a_{ij}^2}.$$

$$\text{A last useful formula : } \frac{d}{dt} \log \det(A(t)) = \text{tr}(A(t)^{-1} A'(t))$$

Outlines

Linear Algebra

Probability and Information Theory

Statistics and Machine Learning

References

Basics

Three Sources of Uncertainty

1. Inherent randomness in the system being modeled.
(e.g. quantum mechanics, particles are probabilistic.)
2. Incomplete observability
(e.g. Monty Hall problem. 3 doors, one of which has a bonus behind it.)
3. Incomplete modeling
(e.g. linear regression.)

Random Variables (r.V.)

X : r.V., its values x

$X = x$ happens with a certain probability $p(X = x)$. Range of X may be discrete or continuous.

Discrete variables & Probability mass functions (PMF)

$P(X = x) = p(x)$ or $X \sim p(x)$, $0 \leq p(x) \leq 1$.

Joint probability distribution :

$p(X = x, Y = y) = p(x, y)$

Properties :

(1) $\text{Dom}(P) = \text{Range}(X) = \text{set of all possible states of } X$.

(2) $\forall x \in \text{Range}(X), 0 \leq p(x) \leq 1$.

(3) $\sum_{x \in \text{Range}(X)} p(x) = 1$ (Normalized)

e.g. uniform distribution : $P(X = x_i) = \frac{1}{k}, \quad i = 1, \dots, k$.

Continuous variables & Probability density function (p.d.f)

Properties :

- (1) $\text{Dom}(P) = \text{Range}(X) =$ set of all possible states of X .
- (2) $\forall x \in \text{Range}(X), p(x) \geq 0.$ (No need for $p(x) \leq 1$)
- (3) $\int p(x)dx = 1.$

$p(x)$ does not give the probability of a specific state x , but the prob of X landing inside the interval $[x, x + \delta x]$ with an infinitesimal δx . ($\text{prob} = p(x)\delta x$)

e.g. uniform distribution function on an interval $[a, b]$:

$$u(x; a, b) = \begin{cases} 0 & x \notin [a, b] \\ \frac{1}{b-a} & x \in [a, b] \end{cases} \quad x \sim U(a, b)$$

Marginal Probability & Conditional Probability

Marginal Probability :

prob over a subset of all variables.

discrete : $p(x) = P(X = x) = \sum_y P(X = x, Y = y) = \sum_y p(x, y)$

continuous : $p(x) = \int p(x, y) dy$

Conditional Probability :

prob of some event given some other events have happened.

$$P(Y = y | X = x) = \frac{P(X = x, Y = y)}{P(X = x)}$$

(well-defined if $P(X = x) > 0$)

Chain Rules :

$$P(X^{(1)}, \dots, X^{(n)}) = P(X^{(1)})P(X^{(2)}|X^{(1)})P(X^{(3)}|X^{(1)}, X^{(2)}) \dots P(X^{(n)}|X^{(1)}, \dots, X^{(n-1)})$$

Independence & Conditional Independence

X and Y are independent, if

$$\forall x, y, P(X = x, Y = y) = P(X = x)P(Y = y)$$

X, Y are conditionally independent given Z if $\forall x, y, z,$

$$P(X = x, Y = y|Z = z) = P(X = x|Z = z)P(Y = y|Z = z).$$

short-hand notation : $X \perp Y, X \perp Y|Z.$

Expectation Variance and Covariance

$$E_{x \sim p}[f(x)] = \begin{cases} \sum_x p(x)f(x) \\ \int p(x)f(x)dx \end{cases},$$

$$E[\alpha f(x) + \beta g(x)] = \alpha Ef(x) + \beta Eg(x)$$

$$Var[f(x)] = E[(f(x) - Ef(x))^2], \text{ std}[f(x)] = \sqrt{Var[f(x)]}$$

$$Cov[f(X), g(Y)] = E[(f(x) - Ef(x))(g(Y) - Eg(Y))]$$

measures how much two variable are linearly related to each other as well as their scales.

$$Cov[f(X), g(Y)] = \frac{Cov[f(X), g(Y)]}{std[f(X), g(Y)]} \in [-1, 1] \text{ normalized}$$

$$X \perp Y \Rightarrow Cov[X, Y] = 0 \text{ or } Corr[X, Y] = 0$$

" \Leftarrow " is only true when $X, Y \sim$ normal distribution

$$\text{Covariance Matrix : } Cov(\vec{X}) = (Cov(X_i, Y_j))_{1 \leq i, j \leq n}$$

$$\text{Diagonal } Cov(X_i, X_i) = Var(X_i)$$

Common Prob Distributions

- 1) Bernoulli : binary r.v. one parameter $\phi \in [0, 1]$

$$p(X = 1) = \phi, p(X = 0) = 1 - \phi \text{ or}$$

$$p(X = x) = \phi^x(1 - \phi)^{1-x} \quad x \in \{0, 1\},$$

$$E_X[X] = \phi \quad Var_X[X] = \phi(1 - \phi).$$

- 2) Multinoulli (Categorical)

$$Range(X) = \{0, 1, \dots, k\}$$

$$p(X = i) = p_i \quad \sum_{i=1}^k p_i = 1 \text{ only } k - 1 \text{ parameters.}$$

- 3) Gaussian (Normal)

$$N(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right)$$

$$E(X) = \mu, \quad Var[X] = \sigma^2 > 0$$

$\beta = \sigma^{-2}$ = precision.

- 4) ..., continue after the importance of Gaussian.

Why are Gaussian important

(1) Central Limit Theorem (many versions)

$\{X_i\}_{i=1}^n$ i.i.d. (independent and identically distributed),

$S_n = \frac{1}{n}(X_1 + \dots + X_n)$. (Law of Large numbers

$\Rightarrow S_n \rightarrow EX_1 = \mu$ as $n \rightarrow \infty$)

stronger result : $\sqrt{n}(S_n - \mu) \Rightarrow N(0, \sigma^2)$ as $n \rightarrow \infty$), in distribution $\sigma^2 = \text{Var}[X_1]$.

$\lim_{n \rightarrow \infty} P_r[\sqrt{n}(S_n - \mu) \leq z] = \Phi(\frac{z}{\sigma})$, where $\Phi(x)$ is c.d.f of $N(0, 1)$.

Why are Gaussian important

- (2) Information theory : Gaussian encodes the maximum amount of uncertainty

$$H[x] = E_{x \sim p}[-\log p(x)] = - \int p(x) \log p(x) dx.$$

$$\max_p E[X] \text{ s.t.}$$

$$\lambda_1 \quad \int p(x) dx = 1 \Rightarrow p^*(x) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(x - \mu)^2)$$

$$\lambda_2 \quad \int xp(x) dx = \mu$$

$$\lambda_3 \quad \int (x - \mu)^2 p(x) dx = \sigma^2,$$

$$p(x) = \exp\{\lambda_1 - 1 + \lambda_2(x - \mu) + \lambda_3(x - \mu)^3\}, \lambda_2 = 0,$$

λ_1 : normalization constant, $\lambda_3 = -1/(2\sigma^2)$.

Multivariate normal :

$$N(\vec{x}; \vec{\mu}, \vec{\Sigma}) = \sqrt{\frac{1}{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})\right)$$

Σ^{-1} = Ω precision matrix.

$$\text{Cov}(\vec{x}) = \Sigma, \quad E\vec{x} = \vec{\mu}$$

Common Prob Distributions (Continued)

4) Exponential and Laplace (Doubly exponential)

exponential : $p(x; \lambda) = \lambda \mathbb{1}_{x \geq 0} \exp(-\lambda x)$

Laplace ($x; \mu, r$) = $\frac{1}{2r} \exp(-\frac{|x-\mu|}{r})$.

5) Dirac and Empirical

Dirac : $p(x) = \delta(x - \mu)$ which puts prob $\frac{1}{m}$ on each of $x^{(i)}$.

(samples), i.e. $P(X = x^{(i)}) = \frac{1}{m}$, $i = 1, \dots, m$.

Common Prob Distributions (Continued)

6) Mixture :

$P(x) = \sum P(c = i)P(x|c = i)$, where $P(c)$ is multimoulli.

Empirical is mixture with Dirac.

Latent variable c , $P(X|c)$ relates the latent variable c to the visible variables X .

e.g. Gaussian Mixtures $P(x|c = i)$ is Gaussian for $\forall i$,
 $\sim N(x; \mu^{(i)}, \Sigma^{(i)})$.

$P(c = i)$ is prior prob. $P(c|x)$ is posterior prob.

Gaussian mixture model is universal approximator of density in the sense that any smooth density can be approximated with any specific non-zero amount of error by a Gaussian mixture model with enough components.

Bayes' Rule

$$P(x|y) = \frac{P(x)P(y|x)}{P(y)} \text{ i.e } P(x|y)P(y) = P(x, y) = P(x)P(y|x)$$

$$P(y) = \sum_x P(x)P(y|x)$$

$P(x)$: prior ; $P(x|y)$: posterior

Transformations

X, Y two r.v.s. $Y = g(X)$, g is invertible

p.d.f. of X is P_x , p.d.f. of Y is P_y , then $P_k(\vec{y}) = P_x(g^{-1}(\vec{y})) \left| \frac{\partial \vec{x}}{\partial \vec{y}} \right|$
or $P_k(\vec{x}) = P_y(g^{-1}(\vec{x})) \left| \frac{\partial \vec{y}}{\partial \vec{x}} \right| = P_y(g(\vec{x})) |\det J|.$

Let $J = (\frac{\partial y_i}{\partial x_j})_{i,j}$.

Information Theory

information measures the amount of uncertainty :

- (1) Likely events have low information
- (2) Less likely events have higher information
- (3) Information events have additive information. (toss a coin twice)

Self-information at event $X = x$, $I(x) \triangleq -\log P(x)$, other logarithms (base-2) is called bits or shannons.

Shannon entropy : $H(X) = E_{x \sim p}[I(X)] = -E_{x \sim p}[\log P(x)]$.

It gives a lower bound on the number of bits need on average to encode symbols drawn from P .

If X is continuous, $H(P)$ is differential entropy.

e.g. Discrete uniform distribution maximite discrete entropy within the distributions having the same number of states

$$\max_{\{p_i\}} \sum_{i=1}^k -p_i \log p_i = E \log p, \text{ s.t. } \sum_{i=1}^k p_i = 1. \Rightarrow p_i = \frac{1}{k}.$$

Examples of Entropy

If the sun rises in the East there is no information content, the sun always rises in the East.

If you toss an unbiased coin then there is information in whether it lands heads or tails up. If the coin is biased towards heads then there is more information if it lands tails.

Surprisal associated with an event is the negative of the logarithm of the probability of the event — $-\log_2(p)$.

People use different bases for the logarithm, but it doesn't make much difference, it only makes a scaling difference. But if you use base 2 then the units are the familiar bits. If the event is certain, so that $p = 1$, the information associated with it is zero. The lower the probability of an event the higher the surprise, becoming infinity when the event is impossible.

Examples of Entropy

But why logarithms? The logarithm function occurs naturally in information theory. Consider for example the tossing of four coins. There are 16 possible states for the coins, HHHH, HHHT, ..., TTTT. But only four bits of information are needed to describe the state. HTHH could be represented by 0100.

$$4 = \log_2(16) = -\log_2(1/16).$$

Going back to the biased coin, suppose that the probability of tossing heads is $3/4$ and $1/4$ of tossing tails. If I toss heads then that was almost expected, there's not that much information. Technically it's $-\log_2(0.75) = 0.415$ bits. But if I toss tails then it is $-\log_2(0.25) = 2$ bits.

Examples of Entropy

This leads naturally to looking at the average information, this is our entropy :

$$-\sum p \log_2(p),$$

where the sum is taken over all possible outcomes.

(Note that when there are only two possible outcomes the formula for entropy must be the same when p is replaced by $1 - p$. And this is true here.)

Cross Entropy

Suppose you have a *model* for the probability of discrete events, call this p_k^M where the index k just means one of K possibilities. The sum of these probabilities must obviously be one.

And suppose that you have some empirical data for the probabilities of those events, p_k^E . With the sum again being one.

The cross entropy is defined as

$$-\sum_k p_k^E \ln(p_k^M).$$

It is a measure of how far apart the two distributions are.

Example of Cross Entropy

Suppose that you have a machine-learning algorithm that is meant to tell you whether a fruit is a passion fruit, orange or guava.

As a test you input the features for an orange. And the algorithm is going to output three numbers, perhaps thanks to the softmax function, which can be interpreted as the probabilities of the fruit in question (the orange) being one of P , O or G . Will it correctly identify it as an orange?

The model probabilities come out of the algorithm as

$$p_P^M = 0.13, \quad p_O^M = 0.69, \quad \text{and} \quad p_G^M = 0.18.$$

Ok, it's done quite well. It thinks the fruit is most likely to be an orange. But it wasn't 100% sure.

Example of Cross Entropy

Empirically we know that

$$p_P^E = 0, \quad p_O^E = 1, \quad \text{and} \quad p_G^E = 0,$$

because it definitely is an orange. The cross entropy is thus

$$-(0 \times \ln(0.13) + 1 \times \ln(0.69) + 0 \times \ln(0.18)) = 0.371.$$

The cross entropy is minimized when the model probabilities are the same as the empirical probabilities.

Example of Cross Entropy

To see this we can use Lagrange multipliers. Write

$$L = - \sum_k p_k^E \ln(p_k^M) - \lambda \left(\sum_k p_k^M - 1 \right).$$

The second term on the right is needed because the sum of the model probabilities is constrained to be one.

Now differentiate with respect to each model probability, and set the results to zero :

$$\frac{\partial L}{\partial p_k^M} = -\frac{p_k^E}{p_k^M} - \lambda = 0.$$

But since the sums of the two probabilities must be one we find that $\lambda = -1$ and $p_k^M = p_k^E$.

Example of Cross Entropy

Because the cross entropy is minimized when the model probabilities are the same as the empirical probabilities we can see that cross entropy is a candidate for a useful cost function when you have a classification problem.

If you take another look at the sections on MLE and on cost functions, and compare with the above on entropy you'll find a great deal of overlap and similarities in the mathematics. The same ideas keep coming back in different guises and with different justifications and uses.

Kullback-Leibler (KL) divergence

Two distributions $P(x), Q(x)$

$$H(P, Q) = -E_{x \sim p} \log Q(x) \text{ cross-entropy}$$

$$D_{KL}(P||Q) = E_{x \sim p} [\log \frac{P(X)}{Q(X)}] = -H(P) + H(P, Q) \text{ Extra information "distance".}$$

Properties : $D_{KL}(P||Q) = 0$ iff $P = Q$ a.e. $D_{KL}(P||Q) \geq 0$
But $D_{KL}(P||Q) \neq D_{KL}(Q||P)$, $0 \log 0 = \lim_{x \rightarrow 0} x \log x = 0$.

Outlines

Linear Algebra

Probability and Information Theory

Statistics and Machine Learning

References

Tasks of Machine Learning

- (1) Classification : find $f : \mathbb{R}^n \rightarrow \{1, \dots, k\}$ s.t. $y \approx f(x)$.
- (2) Regression : find $f : \mathbb{R}^n \rightarrow \mathbb{R}$, s.t. $y \approx f(x)$.
- (3) Anomaly Detection : find $P(x)$ for normal samples, predict abnormal samples \hat{x} with small prob $P(\hat{x}) < \varepsilon$.
- (4) Imputation of missing values : predict $P(x)$ for $x = (x_1, \dots, x_n)$ then insert the value of x_i for a new sample \vec{x} with x_i missing.
- (5) Denoising : Given a corrupted example $\tilde{x} \in \mathbb{R}^n$, find a clean example $\hat{x} \in \mathbb{R}^n$ s.t. $x \approx \hat{x}$ with some noise, i.e. find $P(x|\tilde{x})$.

Tasks of Machine Learning

(6) Density Estimation or PMF estimation :

find $P_{model} : \mathbb{R}^n \rightarrow \mathbb{R}$, $P_{model}(x)$ for given samples \vec{x} . All previous problems, and clustering dimensionality reduction etc, could fall into this category.

This is rather difficult, computationally intractable.

supertrained learning : $P(y|x) = \frac{P(x,y)}{\sum_{y'} P(x,y')}$.

Learning conditional statistics (e.g. expectation). given a measure of deviation (loss function). $L(y, f)$ want to find the best f^* that minimize it i.e. $\min_f E_{x,y \sim P_{data}} L(y, f(x))$.

If $L = \|y - f\|_2^2$, then $f^*(x) = E_{y \sim P_{data}(y|x)}[y]$.

If $L = \|y - f\|_1$, then $f^*(x) = \text{conditional median}$.

Linear Regression

$$P_{data}(x, y) = \frac{1}{m} \sum_{i=1}^m \delta(x - x^{(i)}, y - y^{(i)})$$

$$f(x) = w^T x,$$

$$\min_{f=w^T x} E_{x,y \sim P_{data}(x,y)} \|y - f(x)\|_2^2 \approx \min_w \frac{1}{m} \sum_{i=1}^m \|y^{(i)} - w^T x^{(i)}\|_2^2$$

$$P_{data}(x, y) \approx P_{train}(x, y)$$

$$\nabla_w MSE_{train} = 0 \Leftrightarrow w = (X^{(train)T} X^{(train)})^{-1} X^{(train)T} y^{(train)}$$

Goals :

- 1) Make MSE_{train} small (under fitting if this is not achieved)
- 2) Make $MSE_{train} - MSE_{test}$ small (over fitting if this is not achieved)

Linear Regression

e.g. $y = b + \sum_{i=1}^m w_i x^i$

$m = 9$, overfitting

$m = 1$, underfitting

$m = 3$, optimal

Linear Regression

$$\text{Performance : } MSE_{test} = \frac{1}{m^{(test)}} \|X^{(test)}w - y^{(test)}\|_2^2.$$

If $(X^{(train)}, y^{(train)})$ and $(X^{(test)}, y^{(test)}) \sim P_{data}(x, y)$, then
 $MSE_{train} = MSE_{test}$ for \hat{w} computed from 1).

However, in general, $MSE_{train} \leq MSE_{test}$, since \hat{w} is computed from 2).

Regularizaition

Instead of minimizing MSE_{train} , we take into account the model complexity, in the case of linear regression, $\lambda \|w\|_2^2$,
 $J(w) = MSE_{train} + \lambda \|w\|_e^2$, $\lambda \rightarrow 0$ over fitting.

$\lambda \rightarrow +\infty$, underfitting, optimal $\lambda \in (0, +\infty)$.

Statistical Estimators

Point Estimation : estimate Q in $f(x; Q)$ by \hat{Q} .

$$\begin{aligned}\hat{Q} &= \arg \min_Q E_{x,y \sim P_{\text{data}}(x,y)} L(y, f(x, Q)) \\ &= \arg \min_Q \frac{1}{m} \sum_{i=1}^m L(y^{(i)}, f(x^{(i)}; Q)) \\ \Rightarrow \hat{Q}_m &= g((x^{(i)}, y^{(n)}) \sim (x^{(m)}, y^{(m)}); Q)\end{aligned}$$

Function Estimation : nonparameterized $f(x)$ in some functional space \mathfrak{H} , then the least square rule gives \hat{f} as the best approximation of f among \mathfrak{H} -point estimation in \mathfrak{H} .

$$\text{bais}(\hat{Q}_m) = E[\hat{Q}_m] - Q$$

\hat{Q} is unbiased if $\text{bais}(\hat{Q}_m) = 0$.

asymptotically unbiased if $\lim_{m \rightarrow \infty} \text{bais}(\hat{Q}_m) = 0$.

Statistical Estimators

e.g. Bernoulli, $\{x^{(1)}, \dots, x^{(m)}\}$, $P(x^{(i)}; Q) = Q^{x^{(i)}}(1 - Q)^{1-x^{(i)}}$.

Let $\hat{Q}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$, $E[\hat{Q}_m] = \frac{1}{m} \sum_{i=1}^m E[x^{(i)}] = 0$, unbiased.

e.g. Gaussian, $\{x^{(1)}, \dots, x^{(m)}\}$,

$$P(x^{(i)}) = N(x^{(i)}; \mu, \sigma^2) = \frac{1}{\sqrt{2\sigma}} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)$$

$\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$, $E[\hat{\mu}_m] = \frac{1}{m} \sum_{i=1}^m E[x^{(i)}] = \mu$. unbiased

$\hat{\sigma}_m^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \hat{\mu}_m)^2$ sample variances

$E[\hat{\sigma}_m^2] = \frac{m-1}{m} \sigma^2 \neq \sigma^2$ biased

but $\hat{\sigma}_m^2 = \frac{1}{m-1} \sum_{i=1}^m (x^{(i)} - \hat{\mu}_m)^2 = \frac{m-1}{m} \hat{\sigma}_m^2$ is unbiased.

Variance and Standard Error

\hat{Q}_m :estimator. $\sqrt{\text{Var}(\hat{Q}_m)}$ =standard error = $SE(\hat{Q}_m)$

$$\text{e.g. } SE(\hat{\mu}_m) = \sqrt{\text{Var}\left[\frac{1}{m} \sum_{i=1}^m x^{(i)}\right]} = \frac{\sigma}{\sqrt{m}}$$

CLT $\Rightarrow \hat{\mu}_m \sim N(\mu, SE^2(\hat{\mu}_m))$

Confidence interval : $(\hat{\mu}_m - 1.96SE(\hat{\mu}_m), \hat{\mu}_m + 1.96SE(\hat{\mu}_m)).$

$$\text{e.g. Bernoulli : } \text{Var}(\hat{Q}_m) = \frac{1}{m^2} \sum_{i=1}^m \text{Var}(x^{(i)}) = \frac{1}{m} Q(1 - Q),$$

$$SE(\hat{Q}_m) = \frac{\sqrt{Q(1-Q)}}{\sqrt{m}} \text{ decreasing function of } m.$$

Bias-variance Tradeoff

$$\begin{aligned} MSE &= E[(\hat{Q}_m - Q)^2] = \text{Bais}(\hat{Q}_m)^2 + \text{Var}(\hat{Q}_m) \\ &= E[(\hat{Q}_m - E\hat{Q}_m)^2 + 2(\hat{Q}_m - E\hat{Q}_m)(E\hat{Q}_m - Q) + (E\hat{Q}_m - Q)^2] \end{aligned}$$

Variance and Standard Error

Underfitting : High bias, low variance

Overfitting : Low bias, high variance

Consistency :

\hat{Q}_m is a consistent estimator if $\hat{Q}_m \xrightarrow{P} Q$ ($m \rightarrow \infty$).

i.e. $P(\hat{Q}_m - Q|_{>\varepsilon}) \rightarrow 0$ ($m \rightarrow \infty$) for $\forall \varepsilon > 0$.

Consistency \Rightarrow Asymptotic unbiasedness.

A counter-example for the reverse statement :

$x^{(i)} \sim N(x; \mu, \sigma^2)$, $\hat{\mu} = x^{(1)}$, then $E[\hat{\mu}] = E[x^{(1)}] = \mu$.

But $\hat{\mu}$ does not trends to μ as $m \rightarrow \infty$.

Maximum Likelihood Estimation (MLE)

Usually used in parametric density estimation.

$P_{data}(x) \approx P_{model}(x; Q)$, $\{x^{(i)}\}_{i=1}^m$ drawn i.i.d from $P_{data}(x)$.

Assume $\exists Q$, as if $x^{(i)} \sim P_{model}(x; Q)$.

MLE for Q is

$$Q_{ML} = \arg \max_Q P_{model}(X; Q) = \arg \max_Q \prod_{i=1}^m P_{model}(x^{(i)}; Q)$$

or

$$\begin{aligned} Q_{ML} &= \arg \max_Q \prod_{i=1}^m \log P_{model}(x^{(i)}; Q) \\ &= \arg \max_Q E_{X \sim \hat{P}_{data}} \log P_{model}(X; Q) \end{aligned}$$

Property : Q_{ML} minimizes KL divergence (dissimilarity) between \hat{P}_{data} and P_{model}

$$D_{KL}(\hat{P}_{data} || P_{model}) = E_{X \sim \hat{P}_{data}} [\log \hat{P}_{data}(X) - \log P_{model}(X; Q)]$$

$$\min_Q D_{KL} \Leftrightarrow \min_Q E_{X \sim \hat{P}_{data}} [-\log P_{model}(X; Q)]$$

Examples of MLE

Maximum Likelihood Estimation (MLE) is a common method for estimating parameters in a statistical/probabilistic model.

In words, you simply find the parameter (or parameters) that maximizes the likelihood of observing what actually happened.

Let's see this in a few classical examples.

Examples of MLE

Example : Taxi numbers

You arrive at the train station in a city you've never been to before. You go to the taxi rank so as to get to your final destination. There is one taxi, you take it. While discussing European politics with the taxi driver you notice that the cab's number is 1234. How many taxis are in that city ?

To answer this we need some assumptions. Taxi numbers are positive integers, starting at 1, no gaps and no repeats. We'll need to assume that we are equally likely to get into any cab. And then we introduce the parameter N as the number of taxis.

What is the MLE for N ?

Examples of MLE

Example : Taxi numbers

What is the MLE for N ?

Well, what is the probability of getting into taxi number 1234 when there are N taxis ?

It is $\frac{1}{N}$ for $N \geq 1234$ and zero otherwise.

What value of N maximizes this expression ? Obviously it is $N = 1234$. That is the MLE for the parameter N . It looks a bit disturbing because it seems a coincidence that you happened to get into the cab with the highest number. But then the probability of getting into any cab is equally likely. It is also disturbing that if there are N taxis then the average cab number is $(N + 1)/2$, and we somehow feel that should play a role.

Examples of MLE

Example : Coin tossing

Suppose you toss a coin n times and get h heads. What is the probability, p , of tossing a head next time?

The probability of getting h heads from n tosses is, assuming that the tosses are independent,

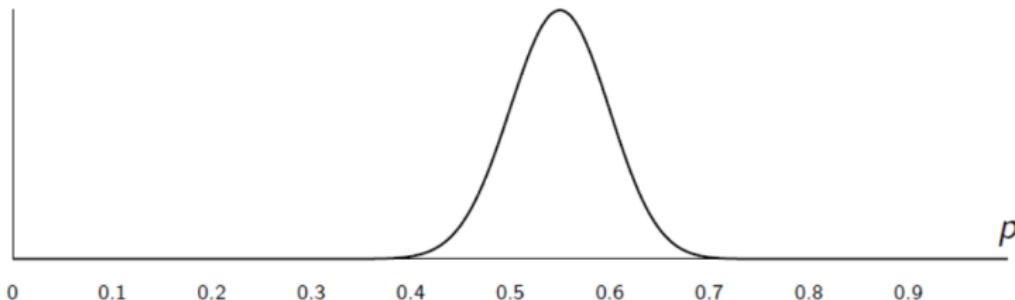
$$\frac{n!}{h!(n-h)!} p^h (1-p)^{n-h} = \binom{n}{h} p^h (1-p)^{n-h}.$$

Applying MLE is the same as maximizing this expression with respect to p .

Examples of MLE

Example : Coin tossing

This likelihood function (without the coefficient in the front that is independent of p) is shown below for $n = 100$ and $h = 55$. There is a very obvious maximum.



Examples of MLE

Example : Coin tossing

Often with MLE when multiplying probabilities, as here, you will take the logarithm of the likelihood and maximize that.

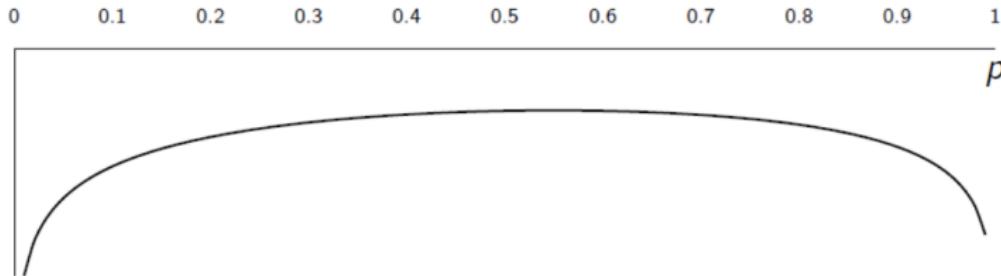
This doesn't change the maximizing value but it does stop you from having to multiply many small numbers, which is going to be problematic with finite precision.

(Look at the scale of the numbers on the vertical axis in the figure.)

Examples of MLE

Example : Coin tossing

Since the first part of this expression is independent of p we maximize $h \ln p + (n - h) \ln(1 - p)$ with respect to p . See below.



This just means differentiating with respect to p and setting the derivative equal to zero. This results in $p = \frac{h}{n}$, which seems eminently reasonable.

Conditional Log-likelihood

MLE for Q in $P(Y|X; Q)$

$$Q_{ML} = \arg \max_Q P(Y|X; Q) = \arg \max_Q \prod_{i=1}^m P(y^{(i)}|x^{(i)}; Q)$$

e.g. linear Regression : $y = w^T x + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$.

Then $P(y|x; w) \sim N(y; w^T x, \sigma^2)$, $\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y-w^T x)^2}{2\sigma^2}\right)$.

Conditional Log-likelihood

$$\begin{aligned} MLE &\Leftrightarrow \max_w \sum_{i=1}^m \log P(y^{(i)}|x^{(i)}; w) \\ &= \max_w \sum_{i=1}^m \left(-\log \sigma - \frac{1}{2} \log(2\pi) - \frac{(y^{(i)} - w^T x^{(i)})^2}{2\sigma^2} \right) \\ &= -m \log \sigma - \frac{m}{2} \log(2\pi) - \min_w \sum_{i=1}^m (y^{(i)} - w^T x^{(i)})^2 / \sigma^2 \\ &\Leftrightarrow \min_w \frac{1}{m} \sum_{i=1}^m (y^{(i)} - w^T x^{(i)})^2 = \min_w MSE_{train} \end{aligned}$$

Properties of MLE

Under certain conditions (given below). MLE is a consistent estimator of the truth.

- (1) P_{data} lies in $\{P_{\text{model}}(\cdot; Q); Q\}$; otherwise, no estimator can recover P_{data} .
- (2) $\exists Q$, s.t. $P_{data} = P_{\text{model}}(\cdot; Q)$; otherwise, MLE can recover P_{data} , but not be able to determine Q .

$E_{X \sim P_{data}} [\hat{Q}_{ML} - Q]^2 \searrow$ Cramer-Rao bound as $m \rightarrow \infty$.

But MLE is not always unbiased, e.g. $\hat{\sigma}_{ML}^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \hat{\mu})^2$

Bayesian Statistics

Previously, Q is fixed but unknown. \hat{Q} is a random variable as a function of data $\{x^{(i)}\}_{i=1}^m$ ($x^{(i)}$ is random).

Bayesian : $\{x^{(i)}\}_{i=1}^m$ is observed and non-random. Q is unknown and uncertain (random).

Prior prob distribution $P(Q)$, before observing the data. Given Q , $\{x^{(i)}\}_{i=1}^m$ is generated from $P(x^{(1)}, \dots, x^{(m)}|Q)$.

$$\text{Bayes' rule} \Rightarrow P(Q|x^{(1)}, \dots, x^{(m)}) = \frac{P(x^{(1)}, \dots, x^{(m)}|Q)P(Q)}{P(x^{(1)}, \dots, x^{(m)})}.$$

$P(Q)$ is usually given e.g. uniform or Gaussian with high entropy, observation of data causes the posterior to loose entropy and concentrate around a few highly likely values of parameters. Bayesian estimates the distribution of Q instead of point estimate.

$$P \in X^{(m+1)}|x^{(1)}, \dots, x^{(m)}) = \int P(x^{(m+1)}|Q)P(Q|x^{(1)}, \dots, x^{(m)})dQ$$

As more observations are given, knowledge about Q becomes different (more).

Bayesian Statistics

	Point Estimate	Bayesian
uncertainty	variance of estimator through random sampling of data	distribution, integral over it
prior info	No	Yes with human knowledge
performance	good as sample size increases	generalize better for limited training data
computational cost	low	high

Bayesian Statistics

e.g. Bayesian Linear Regression $\hat{y} = w^T x$.

Given $(X^{(train)}, y^{(train)})$, $\hat{y}^{(train)} = X^{(train)}w$

$$P(y^{(train)} | X^{(train)}, w) = N(y^{(train)}; X^{(train)}w, I) \exp(-\frac{1}{2}(y^{(train)} - X^{(train)}w)^T (y^{(train)} - X^{(train)}w))$$

Prior of w : $P(w) = N(w; \mu_0, \Lambda_0) \exp(-\frac{1}{2}(w - \mu_0)^T \Lambda_0^{-1} (w - \mu_0))$

Posterior : $P(w|X^{(train)}, y^{(train)}) P(y^{(*)}|X^{(train)}, w) P(w)$

$$\exp(-\frac{1}{2}(y - Xw)^T (y - Xw)) \exp(-\frac{1}{2}(w - \mu_0)^T \Lambda_0^{-1} (w - \mu_0))$$

$$\exp(-\frac{1}{2}(-2y^T Xw + w^T X^T Xw + w^T \Lambda_0^{-1} w - 2\mu_0^T \Lambda_0^{-1} w))$$

$$\exp(-\frac{1}{2}(w - \mu_m)^T \Lambda_m^{-1} (w - \mu_m) + \frac{1}{2}\mu_m^T \Lambda_m^{-1} \mu_m)$$

$$\exp(-\frac{1}{2}(w - \mu_m)^T \Lambda_m^{-1} (w - \mu_m))$$

terms without w are normalizing constant.

If $\mu_0 = 0$, $\Lambda_0 = \frac{1}{\alpha}I$, $\mu_m = (X^T X + \alpha I)^{-1} X^T y$ is the ridge regression estimator of w .

also gives the variance $\Lambda_m = (X^T X + \alpha I)^{-1}$

Maximum A Posterior (MAP) similar to MLE

e.g. μ_m is MAP in the previous example.

MAP choose the point of maximum posterior prob.

$$Q_{MAP} = \arg \max_Q P(Q|X) = \arg \max_Q \{\log P(X|Q) + \log P(Q)\}$$

Bayesian linear regression, $\log -prior \|w\|_2^2$ ridge penalties.

Additional information in prior helps to reduce the variance in the MAP, but does not increase bias.

Different regularizations (penalties) corresponds to different log-prior. (but not all, some penalty may not be a logarithm of a prob distribution, some others depend on data)

Lasso Penalty \rightarrow Laplace distribution $\text{Laplace}(x; 0, \lambda^{-1})$.

Outlines

Linear Algebra

Probability and Information Theory

Statistics and Machine Learning

References

References

- Numerical Analysis, 9th Edition, by Richard L. Burden, J. Douglas Faires, Brooks/Cole, 2011.
- Machine learning : an applied mathematics introduction, by Wilmott, Paul. Panda Ohana Publishing, 2019.
- Deep learning, by Ian Goodfellow, Yoshua Bengio, and Aaron Courville, MIT Press, 2016.