Supervised Learning

wikipedia book

Contents

Chapter 1

Supervised learning

See also: Unsupervised learning

Supervised learning is the machine learning task of inferring a function from labeled training data.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias).

The parallel task in human and animal psychology is often referred to as concept learning.

1.1 Overview

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.

2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.

4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.

5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.

6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

A wide range of supervised learning algorithms is available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems (see the No free lunch theorem).

There are four major issues to consider in supervised learning:

1.1.1 Bias-variance tradeoff

Main article: Bias-variance dilemma

A first issue is the tradeoff between *bias* and *variance*.[2] Imagine that we have available several different, but equally good, training data sets. A learning algorithm is biased for a particular input x if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for x . A learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets. The prediction error of a learned classifier is related to the sum of the bias and the variance of the learning algorithm.[3] Generally, there is a tradeoff between bias and variance. A learning algorithm with low bias must be “flexible” so that it can fit the data well. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance. A key aspect of many supervised learning methods is that they are able to adjust this tradeoff between bias and variance (either automatically or by providing a bias/variance parameter that the user can adjust).

1.1.2 Function complexity and amount of training data

The second issue is the amount of training data available relative to the complexity of the “true” function (classifier or regression function). If the true function is simple, then an “inflexible” learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex (e.g., because it involves complex interactions among many different input features and behaves differently in different parts of the input space), then the function will only be learnable from a very large amount of training data and using a “flexible” learning algorithm with low bias and high variance. Good learning algorithms therefore automatically adjust the bias/variance tradeoff based on the amount of data available and the apparent complexity of the function to be learned.

1.1.3 Dimensionality of the input space

A third issue is the dimensionality of the input space. If the input feature vectors have very high dimension, the learning problem can be difficult even if the true function only depends on a small number of those features. This is because the many “extra” dimensions can confuse the learning algorithm and cause it to have high variance. Hence, high input dimensionality typically requires tuning the classifier to have low variance and high bias. In practice, if the engineer can manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, there are many algorithms for feature selection that seek to identify the relevant features and discard the irrelevant ones. This is an instance of the more general strategy of dimensionality reduction, which seeks to map the input data into a lower-dimensional space prior to running the supervised learning algorithm.

1.1.4 Noise in the output values

A fourth issue is the degree of noise in the desired output values (the supervisory target variables). If the desired output values are often incorrect (because of human error or sensor errors), then the learning algorithm should not attempt to find a function that exactly matches the training examples. Attempting to fit the data too carefully leads to overfitting. You can overfit even when there are no measurement errors (stochastic noise) if the function you are trying to learn is too complex for your learning model. In such a situation that part of the target function that cannot be modeled “corrupts” your training data - this phenomenon has been called deterministic noise. When either type of noise is present, it is better to go with a higher bias, lower variance estimator.

In practice, there are several approaches to alleviate noise in the output values such as early stopping to prevent overfitting as well as detecting and removing the noisy training examples prior to training the supervised learning algorithm. There are several algorithms that identify noisy training examples and removing the suspected noisy training examples prior to training has decreased generalization error with statistical significance.[4][5]

1.1.5 Other factors to consider

Other factors to consider when choosing and applying a learning algorithm include the following:

1. Heterogeneity of the data. If the feature vectors include features of many different kinds (discrete, discrete ordered, counts, continuous values), some algorithms are easier to apply than others. Many algorithms, including Support Vector Machines, linear regression, logistic regression, neural networks, and nearest neighbor methods, require that the input features be numerical and scaled to similar ranges (e.g., to the $[-1,1]$ interval). Methods that employ a distance function, such as nearest neighbor methods and support vector machines with Gaussian kernels, are particularly sensitive to this. An advantage of decision trees is that they easily handle heterogeneous data.

2. Redundancy in the data. If the input features contain redundant information (e.g., highly correlated features), some learning algorithms (e.g., linear regression, logistic regression, and distance based methods) will perform poorly because of numerical instabilities. These problems can often be solved by imposing some form of regularization.

3. Presence of interactions and non-linearities. If each of the features makes an independent contribution to the output, then algorithms based on linear functions (e.g., linear regression, logistic regression, Support Vector Machines, naive Bayes) and distance functions (e.g., nearest neighbor methods, support vector machines with Gaussian kernels) generally perform well. However, if there are complex interactions among features, then algorithms such as decision trees and neural networks work better, because they are specifically designed to discover these interactions. Linear methods can also be applied, but the engineer must manually specify the interactions when using them.

When considering a new application, the engineer can compare multiple learning algorithms and experimentally determine which one works best on the problem at hand (see cross validation). Tuning the performance of a learning algorithm can be very time-consuming. Given fixed resources, it is often better to spend more time collecting additional training data and more informative features than it is to spend extra time tuning the learning algorithms.

The most widely used learning algorithms are Support Vector Machines, linear regression, logistic regression, naive Bayes, linear discriminant analysis, decision trees, k-nearest neighbor algorithm, and Neural Networks (Multilayer perceptron).

1.2 How supervised learning algorithms work

Given a set of N training examples of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$ such that x_i is the feature vector of the i-th example and y_i is its label (i.e., class), a learning algorithm seeks a function $g : X \to Y$, where X is the input space and Y is the output space. The function g is an element of some space of possible functions G , usually called the *hypothesis space*. It is sometimes convenient to represent g using a scoring function $f : X \times Y \to \mathbb{R}$ such that g is defined as returning the y value that gives the highest score: $g(x) = \arg\max_y\ f(x, y)$. Let F denote the space of scoring functions.

Although G and F can be any space of functions, many learning algorithms are probabilistic models where g takes the form of a conditional probability model $g(x) = P(y|x)$, or f takes the form of a joint probability model $f(x, y) = P(x, y)$. For example, naive Bayes and linear discriminant analysis are joint probability models, whereas logistic regression is a conditional probability model.

There are two basic approaches to choosing f or g : empirical risk minimization and structural risk minimization.[6] Empirical risk minimization seeks the function that best fits the training data. Structural risk minimize includes a *penalty function* that controls the bias/variance tradeoff.

In both cases, it is assumed that the training set consists of a sample of independent and identically distributed pairs, $(x_i,\ y_i)$. In order to measure how well a function fits the training data, a loss function $L : Y \times Y \to \mathbb{R}^{\geq 0}$ is defined. For training example $(x_i,\ y_i)$, the loss of predicting the value $\hat{y}$ is $L(y_i, \hat{y})$.

The *risk* $R(g)$ of function g is defined as the expected loss of g . This can be estimated from the training data as

$$R_{emp}(g) = \frac{1}{N} \sum_i L(y_i, g(x_i))$$

1.2.1 Empirical risk minimization

Main article: Empirical risk minimization

In empirical risk minimization, the supervised learning algorithm seeks the function g that minimizes $R(g)$. Hence, a supervised learning algorithm can be constructed by applying an optimization algorithm to find g .

When g is a conditional probability distribution $P(y|x)$ and the loss function is the negative log likelihood: $L(y, \hat{y}) = -\log P(y|x)$, then empirical risk minimization is equivalent to maximum likelihood estimation.

When G contains many candidate functions or the training set is not sufficiently large, empirical risk minimization leads to high variance and poor generalization. The learning algorithm is able to memorize the training examples without generalizing well. This is called overfitting.

1.2.2 Structural risk minimization

Structural risk minimization seeks to prevent overfitting by incorporating a regularization penalty into the optimization. The regularization penalty can be viewed as implementing a form of Occam's razor that prefers simpler functions over more complex ones.

A wide variety of penalties have been employed that correspond to different definitions of complexity. For example, consider the case where the function g is a linear function of the form

$$g(x) = \sum_{j=1}^{d} \beta_j x_j$$

A popular regularization penalty is $\sum_j \beta_j^2$, which is the squared Euclidean norm of the weights, also known as the L_2 norm. Other norms include the L_1 norm, $\sum_j |\beta_j|$, and the L_0 norm, which is the number of non-zero β_j s. The penalty will be denoted by $C(g)$.

The supervised learning optimization problem is to find the function g that minimizes

$$J(g) = R_{emp}(g) + \lambda C(g).$$

The parameter λ controls the bias-variance tradeoff. When $\lambda = 0$, this gives empirical risk minimization with low bias and high variance. When λ is large, the learning algorithm will have high bias and low variance. The value of λ can be chosen empirically via cross validation.

The complexity penalty has a Bayesian interpretation as the negative log prior probability of g , $-\log P(g)$, in which case $J(g)$ is the posterior probabability of g .

1.3 Generative training

The training methods described above are *discriminative training* methods, because they seek to find a function g that discriminates well between the different output values (see discriminative model). For the special case where $f(x,y)=P(x,y)$ is a joint probability distribution and the loss function is the negative log likelihood $-\sum_i \log P(x_i,y_i)$, a risk minimization algorithm is said to perform *generative training*, because f can be regarded as a generative model that explains how the data were generated. Generative training algorithms are often simpler and more computationally efficient than discriminative training algorithms. In some cases, the solution can be computed in closed form as in naive Bayes and linear discriminant analysis.

1.4 Generalizations of supervised learning

There are several ways in which the standard supervised learning problem can be generalized:

1. Semi-supervised learning: In this setting, the desired output values are provided only for a subset of the training data. The remaining data is unlabeled.
2. Active learning: Instead of assuming that all of the training examples are given at the start, active learning algorithms interactively collect new examples, typically by making queries to a human user. Often, the queries are based on unlabeled data, which is a scenario that combines semi-supervised learning with active learning.
3. Structured prediction: When the desired output value is a complex object, such as a parse tree or a labeled graph, then standard methods must be extended.
4. Learning to rank: When the input is a set of objects and the desired output is a ranking of those objects, then again the standard methods must be extended.

1.5 Approaches and algorithms

- Analytical learning
- Artificial neural network
- Backpropagation
- Boosting (meta-algorithm)
- Bayesian statistics
- Case-based reasoning
- Decision tree learning
- Inductive logic programming
- Gaussian process regression
- Group method of data handling
- Kernel estimators
- Learning Automata
- Minimum message length (decision trees, decision graphs, etc.)
- Multilinear subspace learning
- Naive bayes classifier
- Nearest Neighbor Algorithm
- Probably approximately correct learning (PAC) learning
- Ripple down rules, a knowledge acquisition methodology
- Symbolic machine learning algorithms
- Subsymbolic machine learning algorithms
- Support vector machines
- Random Forests
- Ensembles of Classifiers
- Ordinal classification
- Data Pre-processing
- Handling imbalanced datasets
- Statistical relational learning
- Proaftn, a multicriteria classification algorithm

1.6 Applications

- Bioinformatics
- Cheminformatics
  - Quantitative structure–activity relationship
- Database marketing
- Handwriting recognition
- Information retrieval
  - Learning to rank
- Object recognition in computer vision
- Optical character recognition
- Spam detection
- Pattern recognition
- Speech recognition

1.7 General issues

- Computational learning theory
- Inductive bias
- Overfitting (machine learning)
- (Uncalibrated) Class membership probabilities
- Version spaces

1.8 References

[1] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) *Foundations of Machine Learning*, The MIT Press ISBN 9780262018258.

[2] S. Geman, E. Bienenstock, and R. Doursat (1992). Neural networks and the bias/variance dilemma. Neural Computation 4, 1–58.

[3] G. James (2003) Variance and Bias for General Loss Functions, Machine Learning 51, 115-135. (http://www-bcf.usc.edu/~{}gareth/research/bv.pdf)

[4] C.E. Brodely and M.A. Friedl (1999). Identifying and Eliminating Mislabeled Training Instances, Journal of Artificial Intelligence Research 11, 131-167. (http://jair.org/media/606/live-606-1803-jair.pdf)

[5] M.R. Smith and T. Martinez (2011). "Improving Classification Accuracy by Identifying and Removing Instances that Should Be Misclassified". *Proceedings of International Joint Conference on Neural Networks (IJCNN 2011)*. pp. 2690–2697.

[6] Vapnik, V. N. The Nature of Statistical Learning Theory (2nd Ed.), Springer Verlag, 2000.

1.9 External links

- mloss.org: a directory of open source machine learning software.

Chapter 2

Statistical classification

For the unsupervised learning approach, see Cluster analysis.

In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).

In the terminology of machine learning,[1] classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

Often, the individual observations are analyzed into a set of quantifiable properties, known variously explanatory variables, *features*, etc. These properties may variously be categorical (e.g. "A", "B", "AB" or "O", for blood type), ordinal (e.g. "large", "medium" or "small"), integer-valued (e.g. the number of occurrences of a part word in an email) or real-valued (e.g. a measurement of blood pressure). Other classifiers work by comparing observations to previous observations by means of a similarity or distance function.

An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

Terminology across fields is quite varied. In statistics, where classification is often done with logistic regression or a similar procedure, the properties of observations are termed explanatory variables (or independent variables, regressors, etc.), and the categories to be predicted are known as outcomes, which are considered to be possible values of the dependent variable. In machine learning, the observations are often known as *instances*, the explanatory variables are termed *features* (grouped into a feature vector), and the possible categories to be predicted are *classes*. There is also some argument over whether classification methods that do not involve a statistical model can be considered "statistical". Other fields may use different terminology: e.g. in community ecology, the term "classification" normally refers to cluster analysis, i.e. a type of unsupervised learning, rather than the supervised learning described in this article.

2.1 Relation to other problems

Classification and clustering are examples of the more general problem of pattern recognition, which is the assignment of some sort of output value to a given input value. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence; etc.

A common subclass of classification is probabilistic classification. Algorithms of this nature use statistical inference to find the best class for a given instance. Unlike other algorithms, which simply output a "best" class, probabilistic algorithms output a probability of the instance being a member of each of the possible classes. The best class is normally then selected as the one with the highest probability. However, such an algorithm has numerous advantages over non-probabilistic classifiers:

- It can output a confidence value associated with its choice (in general, a classifier that can do this is known as a *confidence-weighted classifier*).
- Correspondingly, it can *abstain* when its confidence of choosing any particular output is too low.
- Because of the probabilities which are generated, probabilistic classifiers can be more effectively incorporated into larger machine-learning tasks, in a

way that partially or completely avoids the problem of *error propagation*.

2.2 Frequentist procedures

Early work on statistical classification was undertaken by Fisher,[2][3] in the context of two-group problems, leading to Fisher's linear discriminant function as the rule for assigning a group to a new observation.[4] This early work assumed that data-values within each of the two groups had a multivariate normal distribution. The extension of this same context to more than two-groups has also been considered with a restriction imposed that the classification rule should be linear.[4][5] Later work for the multivariate normal distribution allowed the classifier to be nonlinear:[6] several classification rules can be derived based on slight different adjustments of the Mahalanobis distance, with a new observation being assigned to the group whose centre has the lowest adjusted distance from the observation.

2.3 Bayesian procedures

Unlike frequentist procedures, Bayesian classification procedures provide a natural way of taking into account any available information about the relative sizes of the sub-populations associated with the different groups within the overall population.[7] Bayesian procedures tend to be computationally expensive and, in the days before Markov chain Monte Carlo computations were developed, approximations for Bayesian clustering rules were devised.[8]

Some Bayesian procedures involve the calculation of group membership probabilities: these can be viewed as providing a more informative outcome of a data analysis than a simple attribution of a single group-label to each new observation.

2.4 Binary and multiclass classification

Classification can be thought of as two separate problems – binary classification and multiclass classification. In binary classification, a better understood task, only two classes are involved, whereas multiclass classification involves assigning an object to one of several classes.[9] Since many classification methods have been developed specifically for binary classification, multiclass classification often requires the combined use of multiple binary classifiers.

2.5 Feature vectors

Most algorithms describe an individual instance whose category is to be predicted using a feature vector of individual, measurable properties of the instance. Each property is termed a feature, also known in statistics as an explanatory variable (or independent variable, although in general different features may or may not be statistically independent). Features may variously be binary ("male" or "female"); categorical (e.g. "A", "B", "AB" or "O", for blood type); ordinal (e.g. "large", "medium" or "small"); integer-valued (e.g. the number of occurrences of a particular word in an email); or real-valued (e.g. a measurement of blood pressure). If the instance is an image, the feature values might correspond to the pixels of an image; if the instance is a piece of text, the feature values might be occurrence frequencies of different words. Some algorithms work only in terms of discrete data and require that real-valued or integer-valued data be *discretized* into groups (e.g. less than 5, between 5 and 10, or greater than 10).

The vector space associated with these vectors is often called the *feature space*. In order to reduce the dimensionality of the feature space, a number of dimensionality reduction techniques can be employed.

2.6 Linear classifiers

A large number of algorithms for classification can be phrased in terms of a linear function that assigns a score to each possible category k by combining the feature vector of an instance with a vector of weights, using a dot product. The predicted category is the one with the highest score. This type of score function is known as a linear predictor function and has the following general form:

$$\operatorname{score}(\mathbf{X}_i, k) = \boldsymbol{\beta}_k \cdot \mathbf{X}_i,$$

where $\mathbf{X}i$ is the feature vector for instance i, $\boldsymbol{\beta}k$ is the vector of weights corresponding to category k, and score($\mathbf{X}i$, k) is the score associated with assigning instance i to category k. In discrete choice theory, where instances represent people and categories represent choices, the score is considered the utility associated with person i choosing category k.

Algorithms with this basic setup are known as linear classifiers. What distinguishes them is the procedure for determining (training) the optimal weights/coefficients and the way that the score is interpreted.

Examples of such algorithms are

- Logistic regression and Multinomial logistic regression
- Probit regression

- The perceptron algorithm
- Support vector machines
- Linear discriminant analysis.

2.7 Algorithms

Examples of classification algorithms include:

- Linear classifiers
  - Fisher's linear discriminant
  - Logistic regression
  - Naive Bayes classifier
  - Perceptron
- Support vector machines
  - Least squares support vector machines
- Quadratic classifiers
- Kernel estimation
  - k-nearest neighbor
- Boosting (meta-algorithm)
- Decision trees
  - Random forests
- Neural networks
- Learning vector quantization

2.8 Evaluation

Classifier performance depends greatly on the characteristics of the data to be classified. There is no single classifier that works best on all given problems (a phenomenon that may be explained by the no-free-lunch theorem). Various empirical tests have been performed to compare classifier performance and to find the characteristics of data that determine classifier performance. Determining a suitable classifier for a given problem is however still more an art than a science.

The measures precision and recall are popular metrics used to evaluate the quality of a classification system. More recently, receiver operating characteristic (ROC) curves have been used to evaluate the tradeoff between true- and false-positive rates of classification algorithms.

As a performance metric, the uncertainty coefficient has the advantage over simple accuracy in that it is not affected by the relative sizes of the different classes. [10] Further, it will not penalize an algorithm for simply *rearranging* the classes.

2.9 Application domains

See also: Cluster analysis § Applications

Classification has many applications. In some of these it is employed as a data mining procedure, while in others more detailed statistical modeling is undertaken.

- Computer vision
  - Medical imaging and medical image analysis
  - Optical character recognition
  - Video tracking
- Drug discovery and development
  - Toxicogenomics
  - Quantitative structure-activity relationship
- Geostatistics
- Speech recognition
- Handwriting recognition
- Biometric identification
- Biological classification
- Statistical natural language processing
- Document classification
- Internet search engines
- Credit scoring
- Pattern recognition
- Micro-array classification

2.10 See also

- Class membership probabilities
- Classification rule
- Binary classification
- Compound term processing
- Data mining
- Fuzzy logic
- Data warehouse
- Information retrieval
- Artificial intelligence
- Machine learning
- Recommender system

2.11 References

[1] Alpaydin, Ethem (2010). *Introduction to Machine Learning*. MIT Press. p. 9. ISBN 978-0-262-01243-0.

[2] Fisher R.A. (1936) " The use of multiple measurements in taxonomic problems", *Annals of Eugenics*, 7, 179–188

[3] Fisher R.A. (1938) " The statistical utilization of multiple measurements", *Annals of Eugenics*, 8, 376–386

[4] Gnanadesikan, R. (1977) *Methods for Statistical Data Analysis of Multivariate Observations*, Wiley. ISBN 0-471-30845-5 (p. 83–86)

[5] Rao, C.R. (1952) *Advanced Statistical Methods in Multivariate Analysis*, Wiley. (Section 9c)

[6] Anderson,T.W. (1958) *An Introduction to Multivariate Statistical Analysis*, Wiley.

[7] Binder, D.A. (1978) "Bayesian cluster analysis", *Biometrika*, 65, 31–38.

[8] Binder, D.A. (1981) "Approximations to Bayesian clustering rules", *Biometrika*, 68, 275–285.

[9] Har-Peled, S., Roth, D., Zimak, D. (2003) "Constraint Classification for Multiclass Classification and Ranking." In: Becker, B., Thrun, S., Obermayer, K. (Eds) *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, MIT Press. ISBN 0-262-02550-7

[10] Peter Mills (2011). "Efficient statistical classification of satellite measurements". *International Journal of Remote Sensing*. doi:10.1080/01431161.2010.507795.

2.12 External links

- Classifier showdown A practical comparison of classification algorithms.
- Statistical Pattern Recognition Toolbox for Matlab.
- TOOLDIAG Pattern recognition toolbox.
- Statistical classification software based on adaptive kernel density estimation.
- PAL Classification suite written in Java.
- kNN and Potential energy (Applet), University of Leicester
- scikit-learn a widely used package in python
- Weka A java based package with an extensive variety of algorithms.

Chapter 3

Regression analysis

In statistics, **regression analysis** is a statistical process for estimating the relationships among variables. It includes many techniques for modeling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed. Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables – that is, the average value of the dependent variable when the independent variables are fixed. Less commonly, the focus is on a quantile, or other location parameter of the conditional distribution of the dependent variable given the independent variables. In all cases, the estimation target is a function of the independent variables called the **regression function**. In regression analysis, it is also of interest to characterize the variation of the dependent variable around the regression function which can be described by a probability distribution.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable;[1] for example, correlation does not imply causation.

Many techniques for carrying out regression analysis have been developed. Familiar methods such as linear regression and ordinary least squares regression are parametric, in that the regression function is defined in terms of a finite number of unknown parameters that are estimated from the data. Nonparametric regression refers to techniques that allow the regression function to lie in a specified set of functions, which may be infinite-dimensional.

The performance of regression analysis methods in practice depends on the form of the data generating process, and how it relates to the regression approach being used. Since the true form of the data-generating process is generally not known, regression analysis often depends to some extent on making assumptions about this process. These assumptions are sometimes testable if a sufficient quantity of data is available. Regression models for prediction are often useful even when the assumptions are moderately violated, although they may not perform optimally. However, in many applications, especially with small effects or questions of causality based on observational data, regression methods can give misleading results.[2][3]

3.1 History

The earliest form of regression was the method of least squares, which was published by Legendre in 1805,[4] and by Gauss in 1809.[5] Legendre and Gauss both applied the method to the problem of determining, from astronomical observations, the orbits of bodies about the Sun (mostly comets, but also later the then newly discovered minor planets). Gauss published a further development of the theory of least squares in 1821,[6] including a version of the Gauss–Markov theorem.

The term “regression” was coined by Francis Galton in the nineteenth century to describe a biological phenomenon. The phenomenon was that the heights of descendants of tall ancestors tend to regress down towards a normal average (a phenomenon also known as regression toward the mean).[7][8] For Galton, regression had only this biological meaning,[9][10] but his work was later extended by Udny Yule and Karl Pearson to a more general statistical context.[11][12] In the work of Yule and Pearson, the joint distribution of the response and explanatory variables is assumed to be Gaussian. This assumption was weakened by R.A. Fisher in his works of 1922 and 1925.[13][14][15] Fisher assumed that the conditional distribution of the response variable is Gaussian, but the joint distribution need not be. In this respect, Fisher’s assumption is closer to Gauss’s formulation of 1821.

In the 1950s and 1960s, economists used electromechanical desk calculators to calculate regressions. Before 1970, it sometimes took up to 24 hours to receive the result from one regression.[16]

Regression methods continue to be an area of active research. In recent decades, new methods have been developed for robust regression, regression involving correlated responses such as time series and growth curves, regression in which the predictor or response variables are curves, images, graphs, or other complex data objects, regression methods accommodating various types of missing data, nonparametric regression, Bayesian methods for regression, regression in which the predictor variables are measured with error, regression with more predictor variables than observations, and causal inference with regression.

3.2 Regression models

Regression models involve the following variables:

- The **unknown parameters**, denoted as $\boldsymbol{\beta}$, which may represent a scalar or a vector.
- The **independent variables**, $\mathbf{X}$.
- The **dependent variable**, Y.

In various fields of application, different terminologies are used in place of dependent and independent variables.

A regression model relates Y to a function of $\mathbf{X}$ and $\boldsymbol{\beta}$.

$$Y \approx f(\mathbf{X}, \boldsymbol{\beta})$$

The approximation is usually formalized as $E(Y \mid \mathbf{X}) = f(\mathbf{X}, \boldsymbol{\beta})$. To carry out regression analysis, the form of the function f must be specified. Sometimes the form of this function is based on knowledge about the relationship between Y and $\mathbf{X}$ that does not rely on the data. If no such knowledge is available, a flexible or convenient form for f is chosen.

Assume now that the vector of unknown parameters $\boldsymbol{\beta}$ is of length k. In order to perform a regression analysis the user must provide information about the dependent variable Y:

- If N data points of the form $(Y, \mathbf{X})$ are observed, where $N < k$, most classical approaches to regression analysis cannot be performed: since the system of equations defining the regression model is underdetermined, there are not enough data to recover $\boldsymbol{\beta}$.
- If exactly $N = k$ data points are observed, and the function f is linear, the equations $Y = f(\mathbf{X}, \boldsymbol{\beta})$ can be solved exactly rather than approximately. This reduces to solving a set of N equations with N unknowns (the elements of $\boldsymbol{\beta}$), which has a unique solution as long as the $\mathbf{X}$ are linearly independent. If f is nonlinear, a solution may not exist, or many solutions may exist.
- The most common situation is where $N > k$ data points are observed. In this case, there is enough information in the data to estimate a unique value for $\boldsymbol{\beta}$ that best fits the data in some sense, and the regression model when applied to the data can be viewed as an overdetermined system in $\boldsymbol{\beta}$.

In the last case, the regression analysis provides the tools for:

1. Finding a solution for unknown parameters $\boldsymbol{\beta}$ that will, for example, minimize the distance between the measured and predicted values of the dependent variable Y (also known as method of least squares).
2. Under certain statistical assumptions, the regression analysis uses the surplus of information to provide statistical information about the unknown parameters $\boldsymbol{\beta}$ and predicted values of the dependent variable Y.

3.2.1 Necessary number of independent measurements

Consider a regression model which has three unknown parameters, β_0, β_1, and β_2. Suppose an experimenter performs 10 measurements all at exactly the same value of independent variable vector $\mathbf{X}$ (which contains the independent variables X_1, X_2, and X_3). In this case, regression analysis fails to give a unique set of estimated values for the three unknown parameters; the experimenter did not provide enough information. The best one can do is to estimate the average value and the standard deviation of the dependent variable Y. Similarly, measuring at two different values of $\mathbf{X}$ would give enough data for a regression with two unknowns, but not for three or more unknowns.

If the experimenter had performed measurements at three different values of the independent variable vector $\mathbf{X}$, then regression analysis would provide a unique set of estimates for the three unknown parameters in $\boldsymbol{\beta}$.

In the case of general linear regression, the above statement is equivalent to the requirement that the matrix $\mathbf{X}^T\mathbf{X}$ is invertible.

3.2.2 Statistical assumptions

When the number of measurements, N, is larger than the number of unknown parameters, k, and the measurement errors ε_i are normally distributed then *the excess of information* contained in $(N - k)$ measurements is used to make statistical predictions about the unknown parameters. This excess of information is referred to as the degrees of freedom of the regression.

3.3 Underlying assumptions

Classical assumptions for regression analysis include:

- The sample is representative of the population for the inference prediction.
- The error is a random variable with a mean of zero conditional on the explanatory variables.
- The independent variables are measured with no error. (Note: If this is not so, modeling may be done instead using errors-in-variables model techniques).
- The predictors are linearly independent, i.e. it is not possible to express any predictor as a linear combination of the others.
- The errors are uncorrelated, that is, the variance–covariance matrix of the errors is diagonal and each non-zero element is the variance of the error.
- The variance of the error is constant across observations (homoscedasticity). If not, weighted least squares or other methods might instead be used.

These are sufficient conditions for the least-squares estimator to possess desirable properties; in particular, these assumptions imply that the parameter estimates will be unbiased, consistent, and efficient in the class of linear unbiased estimators. It is important to note that actual data rarely satisfies the assumptions. That is, the method is used even though the assumptions are not true. Variation from the assumptions can sometimes be used as a measure of how far the model is from being useful. Many of these assumptions may be relaxed in more advanced treatments. Reports of statistical analyses usually include analyses of tests on the sample data and methodology for the fit and usefulness of the model.

Assumptions include the geometrical support of the variables.[17] Independent and dependent variables often refer to values measured at point locations. There may be spatial trends and spatial autocorrelation in the variables that violate statistical assumptions of regression. Geographic weighted regression is one technique to deal with such data.[18] Also, variables may include values aggregated by areas. With aggregated data the modifiable areal unit problem can cause extreme variation in regression parameters.[19] When analyzing data aggregated by political boundaries, postal codes or census areas results may be very distinct with a different choice of units.

3.4 Linear regression

Main article: Linear regression
See simple linear regression for a derivation of these formulas and a numerical example

In linear regression, the model specification is that the dependent variable, y_i is a linear combination of the *parameters* (but need not be linear in the *independent variables*). For example, in simple linear regression for modeling n data points there is one independent variable: x_i , and two parameters, β_0 and β_1 :

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = 1, \dots, n.$$

In multiple linear regression, there are several independent variables or functions of independent variables.

Adding a term in *xi*^{2} to the preceding regression gives:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i,\ i = 1, \dots, n.$$

This is still linear regression; although the expression on the right hand side is quadratic in the independent variable x_i , it is linear in the parameters β_0 , β_1 and β_2.

In both cases, ε_i is an error term and the subscript i indexes a particular observation.

Given a random sample from the population, we estimate the population parameters and obtain the sample linear regression model:

$$\widehat{y_i} = \widehat{\beta}_0 + \widehat{\beta}_1 x_i.$$

The residual, $e_i = y_i - \widehat{y}_i$, is the difference between the value of the dependent variable predicted by the model, $\widehat{y}_i$, and the true value of the dependent variable, y_i . One method of estimation is ordinary least squares. This method obtains parameter estimates that minimize the sum of squared residuals, SSE,[20][21] also sometimes denoted RSS:

$$SSE = \sum_{i=1}^{n} e_i^2.$$

Minimization of this function results in a set of normal equations, a set of simultaneous linear equations in the parameters, which are solved to yield the parameter estimators, $\widehat{\beta}_0, \widehat{\beta}_1$.

In the case of simple regression, the formulas for the least squares estimates are

$$\widehat{\beta_1} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \text{ and } \hat{\beta}_0 = \bar{y} - \widehat{\beta_1}\bar{x}$$

where $\bar{x}$ is the mean (average) of the x values and $\bar{y}$ is the mean of the y values.

Under the assumption that the population error term has a constant variance, the estimate of that variance is given by:

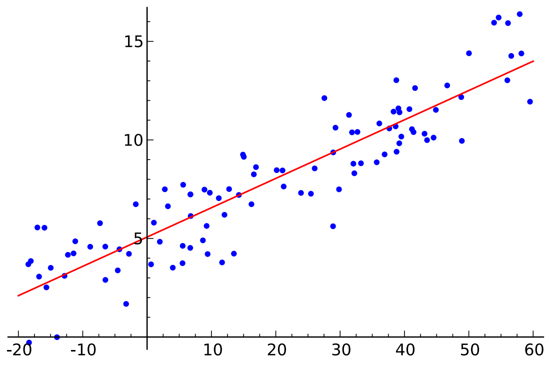


Illustration of linear regression on a data set.

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

This is called the mean square error (MSE) of the regression. The denominator is the sample size reduced by the number of model parameters estimated from the same data, (n-p) for p regressors or (n-p−1) if an intercept is used.[22] In this case, p=1 so the denominator is n−2.

The standard errors of the parameter estimates are given by

$$\hat{\sigma}_{\beta_0} = \hat{\sigma}_\varepsilon \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum(x_i - \bar{x})^2}}$$

$$\hat{\sigma}_{\beta_1} = \hat{\sigma}_\varepsilon \sqrt{\frac{1}{\sum(x_i - \bar{x})^2}}.$$

Under the further assumption that the population error term is normally distributed, the researcher can use these estimated standard errors to create confidence intervals and conduct hypothesis tests about the population parameters.

3.4.1 General linear model

For a derivation, see linear least squares
For a numerical example, see linear regression

In the more general multiple regression model, there are p independent variables:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i,$$

where *xij* is the *i*th observation on the *j*th independent variable, and where the first independent variable takes the value 1 for all *i* (so β_1 is the regression intercept).

The least squares parameter estimates are obtained from p normal equations. The residual can be written as

$$\varepsilon_i = y_i - \hat{\beta}_1 x_{i1} - \cdots - \hat{\beta}_p x_{ip}.$$

The **normal equations** are

$$\sum_{i=1}^{n} \sum_{k=1}^{p} X_{ij} X_{ik} \hat{\beta}_k = \sum_{i=1}^{n} X_{ij} y_i, \ j = 1, \ldots, p.$$

In matrix notation, the normal equations are written as

$$(\mathbf{X}^\top \mathbf{X}) \hat{\boldsymbol{\beta}} = \mathbf{X}^\top \mathbf{Y},$$

where the *ij* element of X is *xij*, the *i* element of the column vector Y is *yi*, and the *j* element of $\hat{\beta}$ is $\hat{\beta}_j$. Thus X is n×p, Y is n×1, and $\hat{\beta}$ is p×1. The solution is

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

3.4.2 Diagnostics

See also: Category:Regression diagnostics.

Once a regression model has been constructed, it may be important to confirm the goodness of fit of the model and the statistical significance of the estimated parameters. Commonly used checks of goodness of fit include the R-squared, analyses of the pattern of residuals and hypothesis testing. Statistical significance can be checked by an F-test of the overall fit, followed by t-tests of individual parameters.

Interpretations of these diagnostic tests rest heavily on the model assumptions. Although examination of the residuals can be used to invalidate a model, the results of a t-test or F-test are sometimes more difficult to interpret if the model's assumptions are violated. For example, if the error term does not have a normal distribution, in small samples the estimated parameters will not follow normal distributions and complicate inference. With relatively large samples, however, a central limit theorem can be invoked such that hypothesis testing may proceed using asymptotic approximations.

3.4.3 "Limited dependent" variables

The phrase "limited dependent" is used in econometric statistics for categorical and constrained variables.

The response variable may be non-continuous ("limited" to lie on some subset of the real line). For binary (zero or one) variables, if analysis proceeds with least-squares linear regression, the model is called the linear probability model. Nonlinear models for binary dependent variables include the probit and logit model. The multivariate probit model is a standard method of estimating a joint relationship between several binary dependent variables and some independent variables. For categorical variables with more than two values there is the multinomial logit. For ordinal variables with more than two values, there are the ordered logit and ordered probit models. Censored regression models may be used when the dependent variable is only sometimes observed, and Heckman correction type models may be used when the sample is not randomly selected from the population of interest. An alternative to such procedures is linear regression based on polychoric correlation (or polyserial correlations) between the categorical variables. Such procedures differ in the assumptions made about the distribution of the variables in the population. If the variable is positive with low values and represents the repetition of the occurrence of an event, then count models like the Poisson regression or the negative binomial model may be used instead.

3.5 Interpolation and extrapolation

Regression models predict a value of the *Y* variable given known values of the *X* variables. Prediction *within* the range of values in the dataset used for model-fitting is known informally as interpolation. Prediction *outside* this range of the data is known as extrapolation. Performing extrapolation relies strongly on the regression assumptions. The further the extrapolation goes outside the data, the more room there is for the model to fail due to differences between the assumptions and the sample data or the true values.

It is generally advised that when performing extrapolation, one should accompany the estimated value of the dependent variable with a prediction interval that represents the uncertainty. Such intervals tend to expand rapidly as the values of the independent variable(s) moved outside the range covered by the observed data.

For such reasons and others, some tend to say that it might be unwise to undertake extrapolation.[23]

However, this does not cover the full set of modelling errors that may be being made: in particular, the assumption of a particular form for the relation between *Y* and *X*. A properly conducted regression analysis will include an assessment of how well the assumed form is matched by the observed data, but it can only do so within the range of values of the independent variables actually available. This means that any extrapolation is particularly reliant on the assumptions being made about the structural form of the regression relationship. Best-practice advice here is that a linear-in-variables and linear-in-parameters relationship should not be chosen simply for computational convenience, but that all available knowledge should be deployed in constructing a regression model. If this knowledge includes the fact that the dependent variable cannot go outside a certain range of values, this can be made use of in selecting the model – even if the observed dataset has no values particularly near such bounds. The implications of this step of choosing an appropriate functional form for the regression can be great when extrapolation is considered. At a minimum, it can ensure that any extrapolation arising from a fitted model is "realistic" (or in accord with what is known).

3.6 Nonlinear regression

Main article: Nonlinear regression

When the model function is not linear in the parameters, the sum of squares must be minimized by an iterative procedure. This introduces many complications which are summarized in Differences between linear and non-linear least squares

3.7 Power and sample size calculations

There are no generally agreed methods for relating the number of observations versus the number of independent variables in the model. One rule of thumb suggested by Good and Hardin is $N = m^n$, where N is the sample size, n is the number of independent variables and m is the number of observations needed to reach the desired precision if the model had only one independent variable.[24] For example, a researcher is building a linear regression model using a dataset that contains 1000 patients (N). If the researcher decides that five observations are needed to precisely define a straight line (m), then the maximum number of independent variables the model can support is 4, because

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

3.8 Other methods

Although the parameters of a regression model are usually estimated using the method of least squares, other methods which have been used include:

- Bayesian methods, e.g. Bayesian linear regression
- Percentage regression, for situations where reducing *percentage* errors is deemed more appropriate.[25]
- Least absolute deviations, which is more robust in the presence of outliers, leading to quantile regression

- Nonparametric regression, requires a large number of observations and is computationally intensive
- Distance metric learning, which is learned by the search of a meaningful distance metric in a given input space.[26]

3.9 Software

Main article: List of statistical packages

All major statistical software packages perform least squares regression analysis and inference. Simple linear regression and multiple regression using least squares can be done in some spreadsheet applications and on some calculators. While many statistical software packages can perform various types of nonparametric and robust reon, these methods are less standardized; different software packages implement different methods, and a method with a given name may be implemented differently in different packages. Specialized regression software has been developed for use in fields such as survey analysis and neuroimaging.

3.10 See also

- Curve fitting
- Forecasting
- Fraction of variance unexplained
- Kriging (a linear least squares estimation algorithm)
- Local regression
- Modifiable areal unit problem
- Multivariate adaptive regression splines
- Multivariate normal distribution
- Pearson product-moment correlation coefficient
- Prediction interval
- Robust regression
- Segmented regression
- Stepwise regression
- Trend estimation

3.11 References

[1] Armstrong, J. Scott (2012). "Illusions in Regression Analysis". *International Journal of Forecasting (forthcoming)* **28** (3): 689. doi:10.1016/j.ijforecast.2012.02.001.

[2] David A. Freedman, *Statistical Models: Theory and Practice*, Cambridge University Press (2005)

[3] R. Dennis Cook; Sanford Weisberg Criticism and Influence Analysis in Regression, *Sociological Methodology*, Vol. 13. (1982), pp. 313–361

[4] A.M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*, Firmin Didot, Paris, 1805. "Sur la Méthode des moindres quarrés" appears as an appendix.

[5] C.F. Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum*. (1809)

[6] C.F. Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae*. (1821/1823)

[7] Mogull, Robert G. (2004). *Second-Semester Applied Statistics*. Kendall/Hunt Publishing Company. p. 59. ISBN 0-7575-1181-3.

[8] Galton, Francis (1989). "Kinship and Correlation (reprinted 1989)". *Statistical Science* (Institute of Mathematical Statistics) **4** (2): 80–86. doi:10.1214/ss/1177012581. JSTOR 2245330.

[9] Francis Galton. "Typical laws of heredity", Nature 15 (1877), 492–495, 512–514, 532–533. *(Galton uses the term "reversion" in this paper, which discusses the size of peas.)*

[10] Francis Galton. Presidential address, Section H, Anthropology. (1885) *(Galton uses the term "regression" in this paper, which discusses the height of humans.)*

[11] Yule, G. Udny (1897). "On the Theory of Correlation". *Journal of the Royal Statistical Society* (Blackwell Publishing) **60** (4): 812–54. doi:10.2307/2979746. JSTOR 2979746.

[12] Pearson, Karl; Yule, G.U.; Blanchard, Norman; Lee,Alice (1903). "The Law of Ancestral Heredity". *Biometrika* (Biometrika Trust) **2** (2): 211–236. doi:10.1093/biomet/2.2.211. JSTOR 2331683.

[13] Fisher, R.A. (1922). "The goodness of fit of regression formulae, and the distribution of regression coefficients". *Journal of the Royal Statistical Society* (Blackwell Publishing) **85** (4): 597–612. doi:10.2307/2341124. JSTOR 2341124.

[14] Ronald A. Fisher (1954). *Statistical Methods for Research Workers* (Twelfth ed.). Edinburgh: Oliver and Boyd. ISBN 0-05-002170-2.

[15] Aldrich, John (2005). "Fisher and Regression". *Statistical Science* **20** (4): 401–417. doi:10.1214/088342305000000331. JSTOR 20061201.

[16] Rodney Ramcharan. Regressions: Why Are Economists Obessessed with Them? March 2006. Accessed 2011-12-03.

[17] N. Cressie (1996) Change of Support and the Modiable Areal Unit Problem. Geographical Systems 3:159–180.

[18] Fotheringham, A. Stewart; Brunsdon, Chris; Charlton, Martin (2002). *Geographically weighted regression: the analysis of spatially varying relationships* (Reprint ed.). Chichester, England: John Wiley. ISBN 978-0-471-49616-8.

[19] Fotheringham, AS; Wong, DWS (1 January 1991). "The modifiable areal unit problem in multivariate statistical analysis". *Environment and Planning A* **23** (7): 1025–1044. doi:10.1068/a231025.

[20] M. H. Kutner, C. J. Nachtsheim, and J. Neter (2004), "Applied Linear Regression Models", 4th ed., McGraw-Hill/Irwin, Boston (p. 25)

[21] N. Ravishankar and D. K. Dey (2002), "A First Course in Linear Model Theory", Chapman and Hall/CRC, Boca Raton (p. 101)

[22] Steel, R.G.D, and Torrie, J. H., *Principles and Procedures of Statistics with Special Reference to the Biological Sciences.*, McGraw Hill, 1960, page 288.

[23] Chiang, C.L, (2003) *Statistical methods of analysis*, World Scientific. ISBN 981-238-310-7 - page 274 section 9.7.4 "interpolation vs extrapolation"

[24] Good, P. I.; Hardin, J. W. (2009). *Common Errors in Statistics (And How to Avoid Them)* (3rd ed.). Hoboken, New Jersey: Wiley. p. 211. ISBN 978-0-470-45798-6.

[25] Tofallis, C. (2009). "Least Squares Percentage Regression". *Journal of Modern Applied Statistical Methods* **7**: 526–534. doi:10.2139/ssrn.1406472.

[26] YangJing Long (2009). "Human age estimation by metric learning for regression problems" (PDF). *Proc. International Conference on Computer Analysis of Images and Patterns*: 74–82.

3.12 Further reading

- William H. Kruskal and Judith M. Tanur, ed. (1978), "Linear Hypotheses," *International Encyclopedia of Statistics*. Free Press, v. 1,

  Evan J. Williams, "I. Regression," pp. 523–41.

  Julian C. Stanley, "II. Analysis of Variance," pp. 541–554.

- Lindley, D.V. (1987). "Regression and correlation analysis," New Palgrave: A Dictionary of Economics, v. 4, pp. 120–23.
- Birkes, David and Dodge, Y., *Alternative Methods of Regression*. ISBN 0-471-56881-3
- Chatfield, C. (1993) "Calculating Interval Forecasts," *Journal of Business and Economic Statistics,* **11**. pp. 121–135.
- Draper, N.R.; Smith, H. (1998). *Applied Regression Analysis* (3rd ed.). John Wiley. ISBN 0-471-17082-8.
- Fox, J. (1997). *Applied Regression Analysis, Linear Models and Related Methods.* Sage
- Hardle, W., *Applied Nonparametric Regression* (1990), ISBN 0-521-42950-1
- Meade, N. and T. Islam (1995) "Prediction Intervals for Growth Curve Forecasts" *Journal of Forecasting,* **14**, pp. 413–430.
- A. Sen, M. Srivastava, *Regression Analysis — Theory, Methods, and Applications*, Springer-Verlag, Berlin, 2011 (4th printing).
- T. Strutz: *Data Fitting and Uncertainty (A practical introduction to weighted least squares and beyond).* Vieweg+Teubner, ISBN 978-3-8348-1022-9.
- Malakooti, B. (2013). Operations and Production Systems with Multiple Objectives. John Wiley & Sons.

3.13 External links

- Hazewinkel, Michiel, ed. (2001), "Regression analysis", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Earliest Uses: Regression – basic history and references
- Regression of Weakly Correlated Data – how linear regression mistakes can appear when Y-range is much smaller than X-range

Chapter 4

Perceptron

"Perceptrons" redirects here. For the book of that title, see Perceptrons (book).

In machine learning, the **perceptron** is an algorithm for supervised learning of binary classifiers: functions that can decide whether an input (represented by a vector of numbers) belong to one class or another.[1] It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time.

The perceptron algorithm dates back to the late 1950s; its first implementation, in custom hardware, was one of the first artificial neural networks to be produced.

4.1 History

See also: History of artificial intelligence, AI winter

The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt,[2] funded by the United States Office of Naval Research.[3] The perceptron was intended to be a machine, rather than a program, and while its first implementation was in software for the IBM 704, it was subsequently implemented in custom-built hardware as the "Mark 1 perceptron". This machine was designed for image recognition: it had an array of 400 photocells, randomly connected to the "neurons". Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors.[4]:193

In a 1958 press conference organized by the US Navy, Rosenblatt made statements about the perceptron that caused a heated controversy among the fledgling AI community; based on Rosenblatt's statements, *The New York Times* reported the perceptron to be "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."[3]

Although the perceptron initially seemed promising, it was quickly proved that perceptrons could not be trained to recognise many classes of patterns. This led to the field of neural network research stagnating for many years, before it was recognised that a feedforward neural network with two or more layers (also called a multilayer perceptron) had far greater processing power than perceptrons with one layer (also called a single layer perceptron). Single layer perceptrons are only capable of learning linearly separable patterns; in 1969 a famous book entitled *Perceptrons* by Marvin Minsky and Seymour Papert showed that it was impossible for these classes of network to learn an XOR function. It is often believed that they also conjectured (incorrectly) that a similar result would hold for a multi-layer perceptron network. However, this is not true, as both Minsky and Papert already knew that multi-layer perceptrons were capable of producing an XOR function. (See the page on *Perceptrons (book)* for more information.) Three years later Stephen Grossberg published a series of papers introducing networks capable of modelling differential, contrast-enhancing and XOR functions. (The papers were published in 1972 and 1973, see e.g.:Grossberg (1973). "Contour enhancement, short-term memory, and constancies in reverberating neural networks" (PDF). *Studies in Applied Mathematics* **52**: 213–257.). Nevertheless the often-miscited Minsky/Papert text caused a significant decline in interest and funding of neural network research. It took ten more years until neural network research experienced a resurgence in the 1980s. This text was reprinted in 1987 as "Perceptrons - Expanded Edition" where some errors in the original text are shown and corrected.

The kernel perceptron algorithm was already introduced in 1964 by Aizerman et al.[5] Margin bounds guarantees were given for the Perceptron algorithm in the general non-separable case first by Freund and Schapire (1998),[1] and more recently by Mohri and Rostamizadeh (2013) who extend previous results and give new L1 bounds.[6]

4.2 Definition

In the modern sense, the perceptron is an algorithm for learning a binary classifier: a function that maps its input

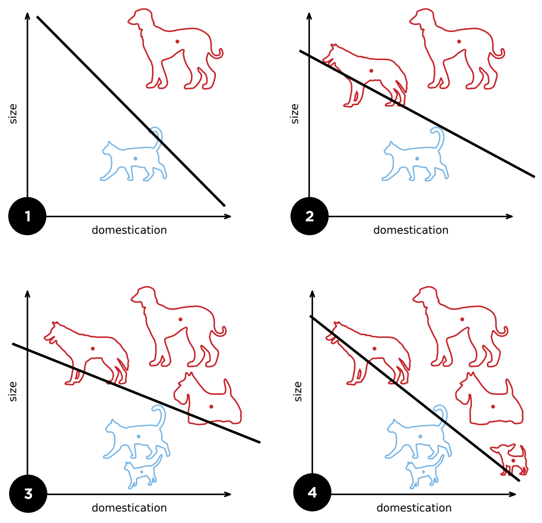
x (a real-valued vector) to an output value $f(x)$ (a single binary value):

$$f(x) = \begin{cases} 1 & \text{if} w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

where w is a vector of real-valued weights, $w \cdot x$ is the dot product $\sum_i w_i x_i$, and b is the *bias*, a term that shifts the decision boundary away from the origin and does not depend on any input value.

The value of $f(x)$ (0 or 1) is used to classify x as either a positive or a negative instance, in the case of a binary classification problem. If b is negative, then the weighted combination of inputs must produce a positive value greater than $|b|$ in order to push the classifier neuron over the 0 threshold. Spatially, the bias alters the position (though not the orientation) of the decision boundary. The perceptron learning algorithm does not terminate if the learning set is not linearly separable. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly. The most famous example of the perceptron's inability to solve problems with linearly nonseparable vectors is the Boolean exclusive-or problem. The solution spaces of decision boundaries for all binary functions and learning behaviors are studied in the reference.[7]

In the context of neural networks, a perceptron is an artificial neuron using the Heaviside step function as the activation function. The perceptron algorithm is also termed the **single-layer perceptron**, to distinguish it from a multilayer perceptron, which is a misnomer for a more complicated neural network. As a linear classifier, the single-layer perceptron is the simplest feedforward neural network.



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

4.3 Learning algorithm

Below is an example of a learning algorithm for a (single-layer) perceptron. For multilayer perceptrons, where a hidden layer exists, more sophisticated algorithms such as backpropagation must be used. Alternatively, methods such as the delta rule can be used if the function is non-linear and differentiable, although the one below will work as well.

When multiple perceptrons are combined in an artificial neural network, each output neuron operates independently of all the others; thus, learning each output can be considered in isolation.

4.3.1 Definitions

We first define some variables:

- $y = f(\mathbf{z})$ denotes the *output* from the perceptron for an input vector $\mathbf{z}$.
- b is the *bias* term, which in the example below we take to be 0.
- $D = \{(\mathbf{x}_1, d_1), \dots, (\mathbf{x}_s, d_s)\}$ is the *training set* of s samples, where:
  - $\mathbf{x}_j$ is the n -dimensional input vector.
  - d_j is the desired output value of the perceptron for that input.

We show the values of the features as follows:

- $x_{j,i}$ is the value of the i th feature of the j th training *input vector*.
- $x_{j,0} = 1$.

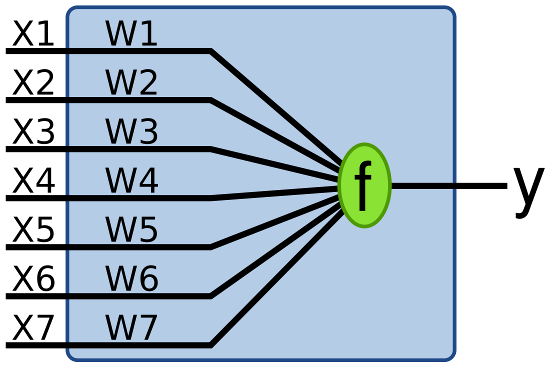
To represent the weights:

- w_i is the i th value in the *weight vector*, to be multiplied by the value of the i th input feature.
- Because $x_{j,0} = 1$, the w_0 is effectively a learned bias that we use instead of the bias constant b .

To show the time-dependence of $\mathbf{w}$, we use:

- $w_i(t)$ is the weight i at time t .
- α is the *learning rate*, where $0 < \alpha \leq 1$.

Too high a learning rate makes the perceptron periodically oscillate around the solution unless additional steps are taken.



The appropriate weights are applied to the inputs, and the resulting weighted sum passed to a function that produces the output y.

4.3.2 Steps

1. Initialize the weights and the threshold. Weights may be initialized to 0 or to a small random value. In the example below, we use 0.

2. For each example j in our training set D, perform the following steps over the input $\mathbf{x}_j$ and desired output d_j :

2a. Calculate the actual output:

$$y_j(t) = f[\mathbf{w}(t) \cdot \mathbf{x}_j] = f[w_0(t) + w_1(t)x_{j,1} + w_2(t)x_{j,2} + \cdots + w_n(t)x_{j,n}]$$

2b. Update the weights:

$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i}$, for all feature $0 \leq i \leq n$.

3. For offline learning, the step 2 may be repeated until the iteration error $\frac{1}{s}\sum_{j=1}^{s}|d_j - y_j(t)|$ is less than a user-specified error threshold γ , or a predetermined number of iterations have been completed.

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

4.3.3 Convergence

The perceptron is a linear classifier, therefore it will never get to the state with all the input vectors classified correctly if the training set D is not linearly separable, i.e. if the positive examples can not be separated from the negative examples by a hyperplane.

But if the training set *is* linearly separable, then the perceptron is guaranteed to converge, and there is an upper bound on the number of times the perceptron will adjust its weights during the training.

Suppose that the input vectors from the two classes can be separated by a hyperplane with a margin γ , i.e. there exists a weight vector $\mathbf{w}$, $||\mathbf{w}|| = 1$, and a bias term b such that $\mathbf{w} \cdot \mathbf{x}_j + b > \gamma$ for all $j : d_j = 1$ and $\mathbf{w} \cdot \mathbf{x}_j + b < -\gamma$ for all $j : d_j = 0$. And also let R denote the maximum norm of an input vector. Novikoff (1962) proved that in this case the perceptron algorithm converges after making $O(R^2/\gamma^2)$ updates. The idea of the proof is that the weight vector is always adjusted by a bounded amount in a direction that it has a negative dot product with, and thus can be bounded above by $O(\sqrt{t})$ where t is the number of changes to the weight vector. But it can also be bounded below by $O(t)$ because if there exists an (unknown) satisfactory weight vector, then every change makes progress in this (unknown) direction by a positive amount that depends only on the input vector.

The decision boundary of a perceptron is invariant with respect to scaling of the weight vector; that is, a perceptron trained with initial weight vector $\mathbf{w}$ and learning rate α behaves identically to a perceptron trained with initial weight vector $\mathbf{w}/\alpha$ and learning rate 1. Thus, since the initial weights become irrelevant with increasing number of iterations, the learning rate does not matter in the case of the perceptron and is usually just set to 1.

4.4 Variants

The pocket algorithm with ratchet (Gallant, 1990) solves the stability problem of perceptron learning by keeping the best solution seen so far "in its pocket". The pocket algorithm then returns the solution in the pocket, rather than the last solution. It can be used also for non-separable data sets, where the aim is to find a perceptron with a small number of misclassifications.

In separable problems, perceptron training can also aim at finding the largest separating margin between the classes. The so-called perceptron of optimal stability can be determined by means of iterative training and optimization schemes, such as the Min-Over algorithm (Krauth and Mezard, 1987)[8] or the AdaTron (Anlauf and Biehl, 1989)) .[9] AdaTron uses the fact that the corresponding quadratic optimization problem is convex. The perceptron of optimal stability, together with the kernel trick, are the conceptual foundations of the support vector machine.

The α -perceptron further used a pre-processing layer of fixed random weights, with thresholded output units. This enabled the perceptron to classify analogue patterns, by projecting them into a binary space. In fact, for a projection space of sufficiently high dimension, patterns can become linearly separable.

For example, consider the case of having to classify data into two classes. Here is a small such data set, consisting of points coming from two Gaussian distributions.

- Two-class Gaussian data
- A linear classifier operating on the original space
- A linear classifier operating on a high-dimensional projection

A linear classifier can only separate points with a hyperplane, so no linear classifier can classify all the points here perfectly. On the other hand, the data can be projected into a large number of dimensions. In our example, a random matrix was used to project the data linearly to a 1000-dimensional space; then each resulting data point was transformed through the hyperbolic tangent function. A linear classifier can then separate the data, as shown in the third figure. However the data may still not be completely separable in this space, in which the perceptron algorithm would not converge. In the example shown, stochastic steepest gradient descent was used to adapt the parameters.

Another way to solve nonlinear problems without using multiple layers is to use higher order networks (sigma-pi unit). In this type of network, each element in the input vector is extended with each pairwise combination of multiplied inputs (second order). This can be extended to an *n*-order network.

It should be kept in mind, however, that the best classifier is not necessarily that which classifies all the training data perfectly. Indeed, if we had the prior constraint that the data come from equi-variant Gaussian distributions, the linear separation in the input space is optimal.

Other linear classification algorithms include Winnow, support vector machine and logistic regression.

4.5 Example

A perceptron learns to perform a binary NAND function on inputs x_1 and x_2 .

Inputs: x_0 , x_1 , x_2 , with input x_0 held constant at 1.

Threshold (t): 0.5

Bias (b): 1

Learning rate (r): 0.1

Training set, consisting of four samples: $\{((1,0,0),1),((1,0,1),1),((1,1,0),1),((1,1,1),0)\}$

In the following, the final weights of one iteration become the initial weights of the next. Each cycle over all the samples in the training set is demarcated with heavy lines.

This example can be implemented in the following Python code.

threshold = 0.5 learning_rate = 0.1 weights = [0, 0, 0] training_set = [((1, 0, 0), 1), ((1, 0, 1), 1), ((1, 1, 0), 1), ((1, 1, 1), 0)] def dot_product(values, weights): return sum(value * weight for value, weight in zip(values, weights)) while True: print('-' * 60) error_count = 0 for input_vector, desired_output in training_set: print(weights) result = dot_product(input_vector, weights) > threshold error = desired_output - result if error != 0: error_count += 1 for index, value in enumerate(input_vector): weights[index] += learning_rate * error * value if error_count == 0: break

4.6 Multiclass perceptron

Like most other techniques for training linear classifiers, the perceptron generalizes naturally to multiclass classification. Here, the input x and the output y are drawn from arbitrary sets. A feature representation function $f(x,y)$ maps each possible input/output pair to a finite-dimensional real-valued feature vector. As before, the feature vector is multiplied by a weight vector w , but now the resulting score is used to choose among many possible outputs:

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

Learning again iterates over the examples, predicting an output for each, leaving the weights unchanged when the predicted output matches the target, and changing them when it does not. The update becomes:

$$w_{t+1} = w_t + f(x,y) - f(x,\hat{y}).$$

This multiclass formulation reduces to the original perceptron when x is a real-valued vector, y is chosen from $\{0,1\}$, and $f(x,y) = yx$.

For certain problems, input/output representations and features can be chosen so that $\operatorname{argmax}_y f(x,y) \cdot w$ can be found efficiently even though y is chosen from a very large or even infinite set.

In recent years, perceptron training has become popular in the field of natural language processing for such tasks as part-of-speech tagging and syntactic parsing (Collins, 2002).

4.7 References

[1] Freund, Y.; Schapire, R. E. (1999). “Large margin classification using the perceptron algorithm” (PDF). *Machine Learning* **37** (3): 277–296. doi:10.1023/A:1007662407062.

[2] Rosenblatt, Frank (1957), The Perceptron--a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.

[3] Mikel Olazaran (1996). “A Sociological Study of the Official History of the Perceptrons Controversy”. *Social Studies of Science* **26** (3). JSTOR 285702.

[4] Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[5] Aizerman, M. A.; Braverman, E. M.; Rozonoer, L. I. (1964). “Theoretical foundations of the potential function method in pattern recognition learning”. *Automation and Remote Control* **25**: 821–837.

[6] Mohri, Mehryar and Rostamizadeh, Afshin (2013). Perceptron Mistake Bounds arXiv:1305.0208, 2013.

[7] Liou, D.-R.; Liou, J.-W.; Liou, C.-Y. (2013). “Learning Behaviors of Perceptron”. *ISBN 978-1-477554-73-9*. *iConcept Press*.

[8] W. Krauth and M. Mezard. Learning algorithms with optimal stabilty in neural networks. J. of Physics A: Math. Gen. 20: L745-L752 (1987)

[9] J.K. Anlauf and M. Biehl. The AdaTron: an Adaptive Perceptron algorithm. Europhysics Letters 10: 687-692 (1989)

- Aizerman, M. A. and Braverman, E. M. and Lev I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, 25:821–837, 1964.
- Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408. doi:10.1037/h0042519.
- Rosenblatt, Frank (1962), Principles of Neurodynamics. Washington, DC:Spartan Books.
- Minsky M. L. and Papert S. A. 1969. *Perceptrons*. Cambridge, MA: MIT Press.
- Gallant, S. I. (1990). Perceptron-based learning algorithms. IEEE Transactions on Neural Networks, vol. 1, no. 2, pp. 179–191.
- Mohri, Mehryar and Rostamizadeh, Afshin (2013). Perceptron Mistake Bounds arXiv:1305.0208, 2013.
- Novikoff, A. B. (1962). On convergence proofs on perceptrons. Symposium on the Mathematical Theory of Automata, 12, 615-622. Polytechnic Institute of Brooklyn.
- Widrow, B., Lehr, M.A., “30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation,” *Proc. IEEE*, vol 78, no 9, pp. 1415–1442, (1990).
- Collins, M. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '02).
- Yin, Hongfeng (1996), Perceptron-Based Algorithms and Analysis, Spectrum Library, Concordia University, Canada

4.8 External links

- A Perceptron implemented in MATLAB to learn binary NAND function
- Chapter 3 Weighted networks - the perceptron and chapter 4 Perceptron learning of *Neural Networks - A Systematic Introduction* by Raúl Rojas (ISBN 978-3-540-60505-8)
- Explanation of the update rule by Charles Elkan
- History of perceptrons
- Mathematics of perceptrons

Chapter 5

Linear regression

In statistics, **linear regression** is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variable) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.[1] (This term should be distinguished from *multivariate linear regression*, where multiple correlated dependent variables are predicted, rather than a single scalar variable.)[2]

In linear regression, data are modeled using linear predictor functions, and unknown model parameters are estimated from the data. Such models are called *linear models*.[3] Most commonly, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X. Less commonly, linear regression could refer to a model in which the median, or some other quantile of the conditional distribution of y given X is expressed as a linear function of X. Like all forms of regression analysis, *linear regression* focuses on the conditional probability distribution of y given X, rather than on the joint probability distribution of y and X, which is the domain of multivariate analysis.

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

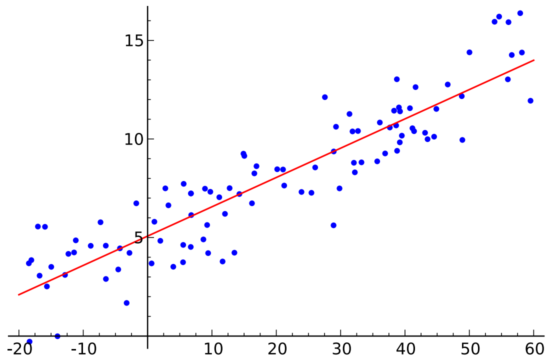
Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- If the goal is prediction, or forecasting, or reduction, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y.

- Given a variable y and a number of variables X_1, ..., Xp that may be related to y, linear regression analysis can be applied to quantify the strength of the relationship between y and the Xj, to assess which Xj may have no relationship with y at all, and to identify which subsets of the Xj contain redundant information about y.

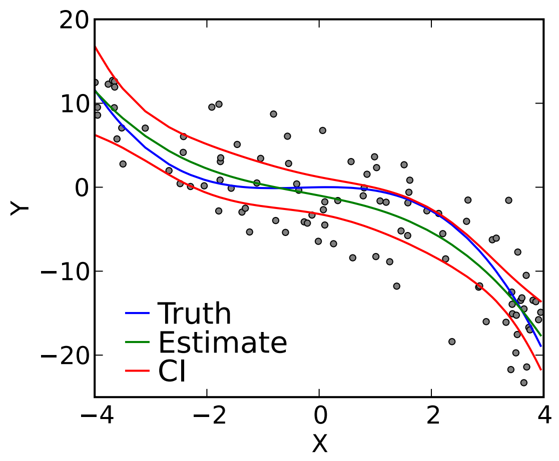
Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares loss function as in ridge regression (L2-norm penalty) and lasso (L1-norm penalty). Conversely, the least squares approach can be used to fit models that are not linear models. Thus, although the terms "least squares" and "linear model" are closely linked, they are not synonymous.

5.1 Introduction to linear regression



Example of simple linear regression, which has one independent variable

Given a data set $\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable yi and the p-vector of regressors xi is linear. This relationship is modeled through a *disturbance term* or *error variable* εi — an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form



Example of a cubic polynomial regression, which is a type of linear regression.

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

where $^{\mathrm{T}}$ denotes the transpose, so that $\boldsymbol{xi}^{\mathrm{T}}\boldsymbol\beta$ is the inner product between vectors $\boldsymbol{xi}$ and $\boldsymbol\beta$.

Often these n equations are stacked together and written in vector form as

$$\mathbf{y} = \mathbf{X}\boldsymbol\beta + \boldsymbol\varepsilon,$$

where

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

Some remarks on terminology and general use:

- y_i is called the *regressand*, *endogenous variable*, *response variable*, *measured variable*, *criterion variable*, or *dependent variable* (see dependent and independent variables.) The decision as to which variable in a data set is modeled as the dependent variable and which are modeled as the independent variables may be based on a presumption that the value of one of the variables is caused by, or directly influenced by the other variables. Alternatively, there may be an operational reason to model one of the variables in terms of the others, in which case there need be no presumption of causality.
- $x_{i1}, x_{i2}, \ldots, x_{ip}$ are called *regressors*, *exogenous variables*, *explanatory variables*, *covariates*, *input variables*, *predictor variables*, or *independent variables* (see dependent and independent variables, but not to be confused with independent random variables). The matrix $\mathbf{X}$ is sometimes called the design matrix.
  - Usually a constant is included as one of the regressors. For example we can take $xi_1 = 1$ for $i = 1, ..., n$. The corresponding element of $\boldsymbol\beta$ is called the *intercept*. Many statistical inference procedures for linear models require an intercept to be present, so it is often included even if theoretical considerations suggest that its value should be zero.
  - Sometimes one of the regressors can be a non-linear function of another regressor or of the data, as in polynomial regression and segmented regression. The model remains linear as long as it is linear in the parameter vector $\boldsymbol\beta$.
  - The regressors *xij* may be viewed either as random variables, which we simply observe, or they can be considered as predetermined fixed values which we can choose. Both interpretations may be appropriate in different cases, and they generally lead to the same estimation procedures; however different approaches to asymptotic analysis are used in these two situations.
- $\boldsymbol\beta$ is a p-dimensional *parameter vector*. Its elements are also called *effects*, or *regression coefficients*. Statistical estimation and inference in linear regression focuses on $\boldsymbol\beta$. The elements of this parameter vector are interpreted as the partial derivatives of the dependent variable with respect to the various independent variables.
- ε_i is called the *error term*, *disturbance term*, or *noise*. This variable captures all other factors which influence the dependent variable *yi* other than the regressors $\boldsymbol{xi}$. The relationship between the error term and the regressors, for example whether they are correlated, is a crucial step in formulating a linear regression model, as it will determine the method to use for estimation.

Example. Consider a situation where a small ball is being tossed up in the air and then we measure its heights of ascent *hi* at various moments in time *ti*. Physics tells us that, ignoring the drag, the relationship can be modeled as

$$h_i = \beta_1 t_i + \beta_2 t_i^2 + \varepsilon_i,$$

where β_1 determines the initial velocity of the ball, β_2 is proportional to the standard gravity, and εi is due to measurement errors. Linear regression can be used to estimate the values of β_1 and β_2 from the measured data.

This model is non-linear in the time variable, but it is linear in the parameters β_1 and β_2; if we take regressors ***xi*** = (xi_1, xi_2) = (ti, ti^2), the model takes on the standard form

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

5.1.1 Assumptions

Standard linear regression models with standard estimation techniques make a number of assumptions about the predictor variables, the response variables and their relationship. Numerous extensions have been developed that allow each of these assumptions to be relaxed (i.e. reduced to a weaker form), and in some cases eliminated entirely. Some methods are general enough that they can relax multiple assumptions at once, and in other cases this can be achieved by combining different extensions. Generally these extensions make the estimation procedure more complex and time-consuming, and may also require more data in order to produce an equally precise model.

The following are the major assumptions made by standard linear regression models with standard estimation techniques (e.g. ordinary least squares):

- **Weak exogeneity**. This essentially means that the predictor variables *x* can be treated as fixed values, rather than random variables. This means, for example, that the predictor variables are assumed to be error-free—that is, not contaminated with measurement errors. Although this assumption is not realistic in many settings, dropping it leads to significantly more difficult errors-in-variables models.

- **Linearity**. This means that the mean of the response variable is a linear combination of the parameters (regression coefficients) and the predictor variables. Note that this assumption is much less restrictive than it may at first seem. Because the predictor variables are treated as fixed values (see above), linearity is really only a restriction on the parameters. The predictor variables themselves can be arbitrarily transformed, and in fact multiple copies of the same underlying predictor variable can be added, each one transformed differently. This trick is used, for example, in polynomial regression, which uses linear regression to fit the response variable as an arbitrary polynomial function (up to a given rank) of a predictor variable. This makes linear regression an extremely powerful inference method. In fact, models such as polynomial regression are often "too powerful", in that they tend to overfit the data. As a result, some kind of regularization must typically be used to prevent unreasonable solutions coming out of the estimation process. Common examples are ridge regression and lasso regression. Bayesian linear regression can also be used, which by its nature is more or less immune to the problem of overfitting. (In fact, ridge regression and lasso regression can both be viewed as special cases of Bayesian linear regression, with particular types of prior distributions placed on the regression coefficients.)

- **Constant variance** (a.k.a. **homoscedasticity**). This means that different response variables have the same variance in their errors, regardless of the values of the predictor variables. In practice this assumption is invalid (i.e. the errors are heteroscedastic) if the response variables can vary over a wide scale. In order to determine for heterogeneous error variance, or when a pattern of residuals violates model assumptions of homoscedasticity (error is equally variable around the 'best-fitting line' for all points of x), it is prudent to look for a "fanning effect" between residual error and predicted values. This is to say there will be a systematic change in the absolute or squared residuals when plotted against the predicting outcome. Error will not be evenly distributed across the regression line. Heteroscedasticity will result in the averaging over of distinguishable variances around the points to get a single variance that is inaccurately representing all the variances of the line. In effect, residuals appear clustered and spread apart on their predicted plots for larger and smaller values for points along the linear regression line, and the mean squared error for the model will be wrong. Typically, for example, a response variable whose mean is large will have a greater variance than one whose mean is small. For example, a given person whose income is predicted to be $100,000 may easily have an actual income of $80,000 or $120,000 (a standard deviation of around $20,000), while another person with a predicted income of $10,000 is unlikely to have the same $20,000 standard deviation, which would imply their actual income would vary anywhere between -$10,000 and $30,000. (In fact, as this shows, in many cases—often the same cases where the assumption of normally distributed errors fails—the variance or standard deviation should be predicted to be proportional to the mean, rather than constant.) Simple linear regression estimation methods give less precise parameter estimates and misleading inferential quantities such as standard errors when substantial heteroscedasticity is present. However, various estimation techniques (e.g. weighted least squares and heteroscedasticity-consistent standard errors) can handle heteroscedasticity in a quite general way. Bayesian linear regression techniques can also be used when the variance is assumed to be a function of the mean. It is also possible in some cases to fix the problem by applying a transformation to the response variable (e.g. fit the logarithm of the response variable using a linear regression model, which implies that the response variable has a log-normal distribution rather than a normal distribution).

- **Independence** of errors. This assumes that the errors of the response variables are uncorrelated with each other. (Actual statistical independence is a stronger condition than mere lack of correlation and is often not needed, although it can be exploited if it is known to hold.) Some methods (e.g. generalized least squares) are capable of handling correlated errors, although they typically require significantly more data unless some sort of regularization is used to bias the model towards assuming uncorrelated errors. Bayesian linear regression is a general way of handling this issue.

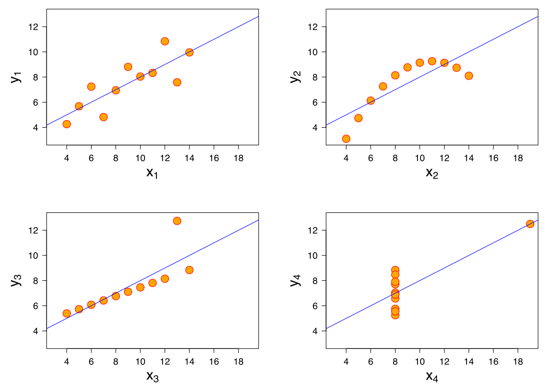
- **Lack of multicollinearity** in the predictors. For standard least squares estimation methods, the design matrix X must have full column rank p,; otherwise, we have a condition known as multicollinearity in the predictor variables. This can be triggered by having two or more perfectly correlated predictor variables (e.g. if the same predictor variable is mistakenly given twice, either without transforming one of the copies or by transforming one of the copies linearly). It can also happen if there is too little data available compared to the number of parameters to be estimated (e.g. fewer data points than regression coefficients). In the case of multicollinearity, the parameter vector β will be non-identifiable—it has no unique solution. At most we will be able to identify some of the parameters, i.e. narrow down its value to some linear subspace of $\mathbf{R}^p$. See partial least squares regression. Methods for fitting linear models with multicollinearity have been developed;[5][6][7][8] some require additional assumptions such as “effect sparsity”—that a large fraction of the effects are exactly zero. Note that the more computationally expensive iterated algorithms for parameter estimation, such as those used in generalized linear models, do not suffer from this problem—and in fact it’s quite normal to when handling categorically valued predictors to introduce a separate indicator variable predictor for each possible category, which inevitably introduces multicollinearity.

Beyond these assumptions, several other statistical properties of the data strongly influence the performance of different estimation methods:

- The statistical relationship between the error terms and the regressors plays an important role in determining whether an estimation procedure has desirable sampling properties such as being unbiased and consistent.

- The arrangement, or probability distribution of the predictor variables x has a major influence on the precision of estimates of β. Sampling and design of experiments are highly developed subfields of statistics that provide guidance for collecting data in such a way to achieve a precise estimate of β.

5.1.2 Interpretation



The sets in the Anscombe’s quartet have the same linear regression line but are themselves very different.

A fitted linear regression model can be used to identify the relationship between a single predictor variable xj and the response variable y when all the other predictor variables in the model are “held fixed”. Specifically, the interpretation of βj is the expected change in y for a one-unit change in xj when the other covariates are held fixed—that is, the expected value of the partial derivative of y with respect to xj. This is sometimes called the *unique effect* of xj on y. In contrast, the *marginal effect* of xj on y can be assessed using a correlation coefficient or simple linear regression model relating xj to y; this effect is the total derivative of y with respect to xj.

Care must be taken when interpreting regression results, as some of the regressors may not allow for marginal changes (such as dummy variables, or the intercept term), while others cannot be held fixed (recall the example from the introduction: it would be impossible to “hold ti fixed” and at the same time change the value of ti^2).

It is possible that the unique effect can be nearly zero even when the marginal effect is large. This may imply that some other covariate captures all the information in xj, so that once that variable is in the model, there is no contribution of xj to the variation in y. Conversely, the unique effect of xj can be large while its marginal effect is nearly zero. This would happen if the other covariates explained a great deal of the variation of y, but they mainly explain variation in a way that is complementary to what is captured by xj. In this case, including the other variables in the model reduces the part of the variability of y that is unrelated to xj, thereby strengthening the apparent relationship with xj.

The meaning of the expression "held fixed" may depend on how the values of the predictor variables arise. If the experimenter directly sets the values of the predictor variables according to a study design, the comparisons of interest may literally correspond to comparisons among units whose predictor variables have been "held fixed" by the experimenter. Alternatively, the expression "held fixed" can refer to a selection that takes place in the context of data analysis. In this case, we "hold a variable fixed" by restricting our attention to the subsets of the data that happen to have a common value for the given predictor variable. This is the only interpretation of "held fixed" that can be used in an observational study.

The notion of a "unique effect" is appealing when studying a complex system where multiple interrelated components influence the response variable. In some cases, it can literally be interpreted as the causal effect of an intervention that is linked to the value of a predictor variable. However, it has been argued that in many cases multiple regression analysis fails to clarify the relationships between the predictor variables and the response variable when the predictors are correlated with each other and are not assigned following a study design.[9] A commonality analysis may be helpful in disentangling the shared and unique impacts of correlated independent variables.[10]

5.2 Extensions

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

5.2.1 Simple and multiple regression

The very simplest case of a single scalar predictor variable x and a single scalar response variable y is known as *simple linear regression*. The extension to multiple and/or vector-valued predictor variables (denoted with a capital X) is known as *multiple linear regression*, also known as *multivariable linear regression*. Nearly all real-world regression models involve multiple predictors, and basic descriptions of linear regression are often phrased in terms of the multiple regression model. Note, however, that in these cases the response variable y is still a scalar. Another term *multivariate linear regression* refers to cases where y is a vector, i.e., the same as *general linear regression*. The difference between *multivariate* linear regression and *multivariable* linear regression should be emphasized as it causes much confusion and misunderstanding in the literature.

5.2.2 General linear models

The general linear model considers the situation when the response variable Y is not a scalar but a vector. Conditional linearity of $E(y|x) = Bx$ is still assumed, with a matrix B replacing the vector β of the classical linear regression model. Multivariate analogues of OLS and GLS have been developed. The term "general linear models" is equivalent to "multivariate linear models". It should be noted the difference of "multivariate linear models" and "multivariable linear models," where the former is the same as "general linear models" and the latter is the same as "multiple linear models."

5.2.3 Heteroscedastic models

Various models have been created that allow for heteroscedasticity, i.e. the errors for different response variables may have different variances. For example, weighted least squares is a method for estimating linear regression models when the response variables may have different error variances, possibly with correlated errors. (See also Weighted linear least squares, and generalized least squares.) Heteroscedasticity-consistent standard errors is an improved method for use with uncorrelated but potentially heteroscedastic errors.

5.2.4 Generalized linear models

Generalized linear models (GLMs) are a framework for modeling a response variable y that is bounded or discrete. This is used, for example:

- when modeling positive quantities (e.g. prices or populations) that vary over a large scale—which are better described using a skewed distribution such as the log-normal distribution or Poisson distribution (although GLMs are not used for log-normal data, instead the response variable is simply transformed using the logarithm function);
- when modeling categorical data, such as the choice of a given candidate in an election (which is better described using a Bernoulli distribution/binomial distribution for binary choices, or a categorical distribution/multinomial distribution for multi-way choices), where there are a fixed number of choices that cannot be meaningfully ordered;
- when modeling ordinal data, e.g. ratings on a scale from 0 to 5, where the different outcomes can be ordered but where the quantity itself may not have any absolute meaning (e.g. a rating of 4 may not be "twice as good" in any objective sense as a rating of 2, but simply indicates that it is better than 2 or 3 but not as good as 5).

Generalized linear models allow for an arbitrary *link function* g that relates the mean of the response variable to the predictors, i.e. $E(y) = g(\beta' x)$. The link function is often related to the distribution of the response, and in particular it typically has the effect of transforming between the $(-\infty, \infty)$ range of the linear predictor and the range of the response variable.

Some common examples of GLMs are:

- Poisson regression for count data.
- Logistic regression and probit regression for binary data.
- Multinomial logistic regression and multinomial probit regression for categorical data.
- Ordered probit regression for ordinal data.

Single index models allow some degree of nonlinearity in the relationship between x and y, while preserving the central role of the linear predictor $\beta' x$ as in the classical linear regression model. Under certain conditions, simply applying OLS to data from a single-index model will consistently estimate β up to a proportionality constant.[11]

5.2.5 Hierarchical linear models

Hierarchical linear models (or *multilevel regression*) organizes the data into a hierarchy of regressions, for example where A is regressed on B, and B is regressed on C. It is often used where the data have a natural hierarchical structure such as in educational statistics, where students are nested in classrooms, classrooms are nested in schools, and schools are nested in some administrative grouping, such as a school district. The response variable might be a measure of student achievement such as a test score, and different covariates would be collected at the classroom, school, and school district levels.

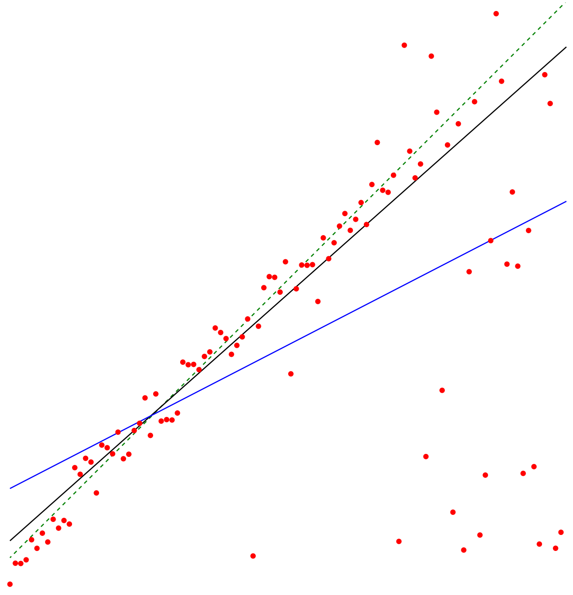
5.2.6 Errors-in-variables

Errors-in-variables models (or "measurement error models") extend the traditional linear regression model to allow the predictor variables X to be observed with error. This error causes standard estimators of β to become biased. Generally, the form of bias is an attenuation, meaning that the effects are biased toward zero.

5.2.7 Others

- In Dempster–Shafer theory, or a linear belief function in particular, a linear regression model may be represented as a partially swept matrix, which can be combined with similar matrices representing observations and other assumed normal distributions and state equations. The combination of swept or unswept matrices provides an alternative method for estimating linear regression models.

5.3 Estimation methods



Comparison of the Theil–Sen estimator (black) and simple linear regression (blue) for a set of points with outliers.

A large number of procedures have been developed for parameter estimation and inference in linear regression. These methods differ in computational simplicity of algorithms, presence of a closed-form solution, robustness with respect to heavy-tailed distributions, and theoretical assumptions needed to validate desirable statistical properties such as consistency and asymptotic efficiency.

Some of the more common estimation techniques for linear regression are summarized below.

5.3.1 Least-squares estimation and related techniques

- **Ordinary least squares** (OLS) is the simplest and thus most common estimator. It is conceptually simple and computationally straightforward. OLS estimates are commonly used to analyze both experimental and observational data.

  The OLS method minimizes the sum of squared residuals, and leads to a closed-form expression for the estimated value of the unknown parameter β:

  $$\hat{\boldsymbol\beta} = (\mathbf{X}^{\mathrm T}\mathbf{X})^{-1}\mathbf{X}^{\mathrm T}\mathbf{y} = \big(\textstyle\sum \mathbf{x}_i\mathbf{x}_i^{\mathrm T}\big)^{-1}\big(\textstyle\sum \mathbf{x}_i y_i\big).$$

  The estimator is unbiased and consistent if the errors have finite variance and are uncorrelated with the regressors[12]

  $$\mathrm{E}[\,\mathbf{x}_i\varepsilon_i\,] = 0.$$

It is also efficient under the assumption that the errors have finite variance and are homoscedastic, meaning that E[εi^2|**x**i] does not depend on i. The condition that the errors are uncorrelated with the regressors will generally be satisfied in an experiment, but in the case of observational data, it is difficult to exclude the possibility of an omitted covariate z that is related to both the observed covariates and the response variable. The existence of such a covariate will generally lead to a correlation between the regressors and the response variable, and hence to an inconsistent estimator of **β**. The condition of homoscedasticity can fail with either experimental or observational data. If the goal is either inference or predictive modeling, the performance of OLS estimates can be poor if multicollinearity is present, unless the sample size is large.

In simple linear regression, where there is only one regressor (with a constant), the OLS coefficient estimates have a simple form that is closely related to the correlation coefficient between the covariate and the response.

- **Generalized least squares** (GLS) is an extension of the OLS method, that allows efficient estimation of β when either heteroscedasticity, or correlations, or both are present among the error terms of the model, as long as the form of heteroscedasticity and correlation is known independently of the data. To handle heteroscedasticity when the error terms are uncorrelated with each other, GLS minimizes a weighted analogue to the sum of squared residuals from OLS regression, where the weight for the i^{th} case is inversely proportional to var(εi). This special case of GLS is called "weighted least squares". The GLS solution to estimation problem is

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

where **Ω** is the covariance matrix of the errors. GLS can be viewed as applying a linear transformation to the data so that the assumptions of OLS are met for the transformed data. For GLS to be applied, the covariance structure of the errors must be known up to a multiplicative constant.

- **Percentage least squares** focuses on reducing percentage errors, which is useful in the field of forecasting or time series analysis. It is also useful in situations where the dependent variable has a wide range without constant variance, as here the larger residuals at the upper end of the range would dominate if OLS were used. When the percentage or relative error is normally distributed, least squares percentage regression provides maximum likelihood estimates. Percentage regression is linked to a multiplicative error model, whereas OLS is linked to models containing an additive error term.[13]

- **Iteratively reweighted least squares** (IRLS) is used when heteroscedasticity, or correlations, or both are present among the error terms of the model, but where little is known about the covariance structure of the errors independently of the data.[14] In the first iteration, OLS, or GLS with a provisional covariance structure is carried out, and the residuals are obtained from the fit. Based on the residuals, an improved estimate of the covariance structure of the errors can usually be obtained. A subsequent GLS iteration is then performed using this estimate of the error structure to define the weights. The process can be iterated to convergence, but in many cases, only one iteration is sufficient to achieve an efficient estimate of β.[15][16]

- **Instrumental variables** regression (IV) can be performed when the regressors are correlated with the errors. In this case, we need the existence of some auxiliary *instrumental variables* **z***i* such that E[**z***i*εi] = 0. If **Z** is the matrix of instruments, then the estimator can be given in closed form as

$$\hat{\beta} = (\mathbf{X}^{\mathrm{T}}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{y}.$$

- **Optimal instruments** regression is an extension of classical IV regression to the situation where E[εi|**z***i*] = 0.

- **Total least squares** (TLS)[17] is an approach to least squares estimation of the linear regression model that treats the covariates and response variable in a more geometrically symmetric manner than OLS. It is one approach to handling the "errors in variables" problem, and is sometimes used when the covariates are assumed to be error-free.

5.3.2 Maximum-likelihood estimation and related techniques

- **Maximum likelihood estimation** can be performed when the distribution of the error terms is known to belong to a certain parametric family $f\theta$ of probability distributions.[18] When $f\theta$ is a normal distribution with zero mean and variance θ, the resulting estimate is identical to the OLS estimate. GLS estimates are maximum likelihood estimates when ε follows a multivariate normal distribution with a known covariance matrix.

- **Ridge regression**,[19][20][21] and other forms of penalized estimation such as **Lasso regression**,[5] deliberately introduce bias into the estimation of β in order to reduce the variability of the estimate. The resulting estimators generally have lower mean squared error than the OLS estimates, particularly when multicollinearity is present. They are generally used when the goal is to predict the value of the response variable y for values of the predictors x that have not yet been observed. These methods are not as commonly used when the goal is inference, since it is difficult to account for the bias.

- **Least absolute deviation** (LAD) regression is a robust estimation technique in that it is less sensitive to the presence of outliers than OLS (but is less efficient than OLS when no outliers are present). It is equivalent to maximum likelihood estimation under a Laplace distribution model for ε.[22]
- **Adaptive estimation**. If we assume that error terms are independent from the regressors $\varepsilon_i \perp \mathbf{x}_i$, the optimal estimator is the 2-step MLE, where the first step is used to non-parametrically estimate the distribution of the error term.[23]

5.3.3 Other estimation techniques

- **Bayesian linear regression** applies the framework of Bayesian statistics to linear regression. (See also Bayesian multivariate linear regression.) In particular, the regression coefficients β are assumed to be random variables with a specified prior distribution. The prior distribution can bias the solutions for the regression coefficients, in a way similar to (but more general than) ridge regression or lasso regression. In addition, the Bayesian estimation process produces not a single point estimate for the “best” values of the regression coefficients but an entire posterior distribution, completely describing the uncertainty surrounding the quantity. This can be used to estimate the “best” coefficients using the mean, mode, median, any quantile (see quantile regression), or any other function of the posterior distribution.
- **Quantile regression** focuses on the conditional quantiles of y given X rather than the conditional mean of y given X. Linear quantile regression models a particular conditional quantile, for example the conditional median, as a linear function $\beta^T x$ of the predictors.
- **Mixed models** are widely used to analyze linear regression relationships involving dependent data when the dependencies have a known structure. Common applications of mixed models include analysis of data involving repeated measurements, such as longitudinal data, or data obtained from cluster sampling. They are generally fit as parametric models, using maximum likelihood or Bayesian estimation. In the case where the errors are modeled as normal random variables, there is a close connection between mixed models and generalized least squares.[24] Fixed effects estimation is an alternative approach to analyzing this type of data.
- **Principal component regression** (PCR)[7][8] is used when the number of predictor variables is large, or when strong correlations exist among the predictor variables. This two-stage procedure first reduces the predictor variables using principal component analysis then uses the reduced variables in an OLS regression fit. While it often works well in practice, there is no general theoretical reason that the most informative linear function of the predictor variables should lie among the dominant principal components of the multivariate distribution of the predictor variables. The partial least squares regression is the extension of the PCR method which does not suffer from the mentioned deficiency.
- **Least-angle regression**[6] is an estimation procedure for linear regression models that was developed to handle high-dimensional covariate vectors, potentially with more covariates than observations.
- The **Theil–Sen estimator** is a simple robust estimation technique that chooses the slope of the fit line to be the median of the slopes of the lines through pairs of sample points. It has similar statistical efficiency properties to simple linear regression but is much less sensitive to outliers.[25]
- Other robust estimation techniques, including the **α-trimmed mean** approach, and **L-, M-, S-, and R-estimators** have been introduced.

5.3.4 Further discussion

In statistics and numerical analysis, the problem of **numerical methods for linear least squares** is an important one because linear regression models are one of the most important types of model, both as formal statistical models and for exploration of data sets. The majority of statistical computer packages contain facilities for regression analysis that make use of linear least squares computations. Hence it is appropriate that considerable effort has been devoted to the task of ensuring that these computations are undertaken efficiently and with due regard to numerical precision.

Individual statistical analyses are seldom undertaken in isolation, but rather are part of a sequence of investigatory steps. Some of the topics involved in considering numerical methods for linear least squares relate to this point. Thus important topics can be

- Computations where a number of similar, and often nested, models are considered for the same data set. That is, where models with the same dependent variable but different sets of independent variables are to be considered, for essentially the same set of data points.
- Computations for analyses that occur in a sequence, as the number of data points increases.
- Special considerations for very extensive data sets.

Fitting of linear models by least squares often, but not always, arises in the context of statistical analysis. It can therefore be important that considerations of computational efficiency for such problems extend to all of the auxiliary quantities required for such analyses, and are not restricted to the formal solution of the linear least squares problem.

Matrix calculations, like any others, are affected by rounding errors. An early summary of these effects, regarding the choice of computational methods for matrix inversion, was provided by Wilkinson.[26]

5.4 Applications of linear regression

Linear regression is widely used in biological, behavioral and social sciences to describe possible relationships between variables. It ranks as one of the most important tools used in these disciplines.

5.4.1 Trend line

Main article: Trend estimation

A **trend line** represents a trend, the long-term movement in time series data after other components have been accounted for. It tells whether a particular data set (say GDP, oil prices or stock prices) have increased or decreased over the period of time. A trend line could simply be drawn by eye through a set of data points, but more properly their position and slope is calculated using statistical techniques like linear regression. Trend lines typically are straight lines, although some variations use higher degree polynomials depending on the degree of curvature desired in the line.

Trend lines are sometimes used in business analytics to show changes in data over time. This has the advantage of being simple. Trend lines are often used to argue that a particular action or event (such as training, or an advertising campaign) caused observed changes at a point in time. This is a simple technique, and does not require a control group, experimental design, or a sophisticated analysis technique. However, it suffers from a lack of scientific validity in cases where other potential changes can affect the data.

5.4.2 Epidemiology

Early evidence relating tobacco smoking to mortality and morbidity came from observational studies employing regression analysis. In order to reduce spurious correlations when analyzing observational data, researchers usually include several variables in their regression models in addition to the variable of primary interest. For example, suppose we have a regression model in which cigarette smoking is the independent variable of interest, and the dependent variable is lifespan measured in years. Researchers might include socio-economic status as an additional independent variable, to ensure that any observed effect of smoking on lifespan is not due to some effect of education or income. However, it is never possible to include all possible confounding variables in an empirical analysis. For example, a hypothetical gene might increase mortality and also cause people to smoke more. For this reason, randomized controlled trials are often able to generate more compelling evidence of causal relationships than can be obtained using regression analyses of observational data. When controlled experiments are not feasible, variants of regression analysis such as instrumental variables regression may be used to attempt to estimate causal relationships from observational data.

5.4.3 Finance

The capital asset pricing model uses linear regression as well as the concept of beta for analyzing and quantifying the systematic risk of an investment. This comes directly from the beta coefficient of the linear regression model that relates the return on the investment to the return on all risky assets.

5.4.4 Economics

Main article: Econometrics

Linear regression is the predominant empirical tool in economics. For example, it is used to predict consumption spending,[27] fixed investment spending, inventory investment, purchases of a country's exports,[28] spending on imports,[28] the demand to hold liquid assets,[29] labor demand,[30] and labor supply.[30]

5.4.5 Environmental science

Linear regression finds application in a wide range of environmental science applications. In Canada, the Environmental Effects Monitoring Program uses statistical analyses on fish and benthic surveys to measure the effects of pulp mill or metal mine effluent on the aquatic ecosystem.[31]

5.5 See also

- Analysis of variance
- Censored regression model
- Cross-sectional regression

- Curve fitting
- Empirical Bayes methods
- Lack-of-fit sum of squares
- Logistic regression
- M-estimator
- MLPACK contains a C++ implementation of linear regression
- Multivariate adaptive regression splines
- Nonlinear regression
- Nonparametric regression
- Normal equations
- Projection pursuit regression
- Segmented linear regression
- Stepwise regression
- Support vector machine
- Truncated regression model

5.6 Notes

[1] David A. Freedman (2009). *Statistical Models: Theory and Practice*. Cambridge University Press. p. 26. A simple regression equation has on the right hand side an intercept and an explanatory variable with a slope coefficient. A multiple regression equation has several explanatory variables on the right hand side, each with its own slope coefficient

[2] Rencher, Alvin C.; Christensen, William F. (2012), "Chapter 10, Multivariate regression – Section 10.1, Introduction", *Methods of Multivariate Analysis*, Wiley Series in Probability and Statistics **709** (3rd ed.), John Wiley & Sons, p. 19, ISBN 9781118391679.

[3] Hilary L. Seal (1967). "The historical development of the Gauss linear model". *Biometrika* **54** (1/2): 1–24. doi:10.1093/biomet/54.1-2.1.

[4] Yan, Xin (2009), *Linear Regression Analysis: Theory and Computing*, World Scientific, pp. 1–2, ISBN 9789812834119, Regression analysis ... is probably one of the oldest topics in mathematical statistics dating back to about two hundred years ago. The earliest form of the linear regression was the least squares method, which was published by Legendre in 1805, and by Gauss in 1809 ... Legendre and Gauss both applied the method to the problem of determining, from astronomical observations, the orbits of bodies about the sun.

[5] Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". *Journal of the Royal Statistical Society, Series B* **58** (1): 267–288. JSTOR 2346178.

[6] Efron, Bradley; Hastie, Trevor; Johnstone,Iain; Tibshirani,Robert (2004). "Least Angle Regression". *The Annals of Statistics* **32** (2): 407–451. doi:10.1214/009053604000000067. JSTOR 3448465.

[7] Hawkins, Douglas M. (1973). "On the Investigation of Alternative Regressions by Principal Component Analysis". *Journal of the Royal Statistical Society, Series C* **22** (3): 275–286. JSTOR 2346776.

[8] Jolliffe, Ian T. (1982). "A Note on the Use of Principal Components in Regression". *Journal of the Royal Statistical Society, Series C* **31** (3): 300–303. JSTOR 2348005.

[9] Berk, Richard A. *Regression Analysis: A Constructive Critique*. Sage. doi:10.1177/0734016807304871.

[10] Warne, R. T. (2011). Beyond multiple regression: Using commonality analysis to better understand R2 results. *Gifted Child Quarterly, 55*, 313-318. doi:10.1177/0016986211422217

[11] Brillinger, David R. (1977). "The Identification of a Particular Nonlinear Time Series System". *Biometrika* **64** (3): 509–515. doi:10.1093/biomet/64.3.509. JSTOR 2345326.

[12] Lai, T.L.; Robbins; Wei; Robbins,H; Wei, C.Z. (1978). "Strong consistency of least squares estimates in multiple regression". *PNAS* **75** (7): 3034–3036. Bibcode:1978PNAS...75.3034L. doi:10.1073/pnas.75.7.3034. JSTOR 68164.

[13] Tofallis, C (2009). "Least Squares Percentage Regression". *Journal of Modern Applied Statistical Methods* **7**: 526–534. doi:10.2139/ssrn.1406472.

[14] del Pino, Guido (1989). "The Unifying Role of Iterative Generalized Least Squares in Statistical Algorithms". *Statistical Science* **4** (4): 394–403. doi:10.1214/ss/1177012408. JSTOR 2245853.

[15] Carroll, Raymond J. (1982). "Adapting for Heteroscedasticity in Linear Models". *The Annals of Statistics* **10** (4): 1224–1233. doi:10.1214/aos/1176345987. JSTOR 2240725.

[16] Cohen, Michael; Dalal, Siddhartha R.; Tukey,John W. (1993). "Robust, Smoothly Heterogeneous Variance Regression". *Journal of the Royal Statistical Society, Series C* **42** (2): 339–353. JSTOR 2986237.

[17] Nievergelt, Yves (1994). "Total Least Squares: State-of-the-Art Regression in Numerical Analysis". *SIAM Review* **36** (2): 258–264. doi:10.1137/1036055. JSTOR 2132463.

[18] Lange, Kenneth L.; Little, Roderick J. A.; Taylor,Jeremy M. G. (1989). "Robust Statistical Modeling Using the t Distribution". *Journal of the American Statistical Association* **84** (408): 881–896. doi:10.2307/2290063. JSTOR 2290063.

[19] Swindel, Benee F. (1981). "Geometry of Ridge Regression Illustrated". *The American Statistician* **35** (1): 12–15. doi:10.2307/2683577. JSTOR 2683577.

[20] Draper, Norman R.; van Nostrand; R. Craig (1979). “Ridge Regression and James-Stein Estimation: Review and Comments”. *Technometrics* **21** (4): 451–466. doi:10.2307/1268284. JSTOR 1268284.

[21] Hoerl, Arthur E.; Kennard,Robert W.; Hoerl,Roger W. (1985). “Practical Use of Ridge Regression: A Challenge Met”. *Journal of the Royal Statistical Society, Series C* **34** (2): 114–120. JSTOR 2347363.

[22] Narula, Subhash C.; Wellington, John F. (1982). “The Minimum Sum of Absolute Errors Regression: A State of the Art Survey”. *International Statistical Review* **50** (3): 317–326. doi:10.2307/1402501. JSTOR 1402501.

[23] Stone, C. J. (1975). “Adaptive maximum likelihood estimators of a location parameter”. *The Annals of Statistics* **3** (2): 267–284. doi:10.1214/aos/1176343056. JSTOR 2958945.

[24] Goldstein, H. (1986). “Multilevel Mixed Linear Model Analysis Using Iterative Generalized Least Squares”. *Biometrika* **73** (1): 43–56. doi:10.1093/biomet/73.1.43. JSTOR 2336270.

[25] Theil, H. (1950). “A rank-invariant method of linear and polynomial regression analysis. I, II, III”. *Nederl. Akad. Wetensch., Proc.* **53**: 386–392, 521–525, 1397–1412. MR 0036489; Sen, Pranab Kumar (1968). “Estimates of the regression coefficient based on Kendall’s tau”. *Journal of the American Statistical Association* **63** (324): 1379–1389. doi:10.2307/2285891. JSTOR 2285891. MR 0258201.

[26] Wilkinson, J.H. (1963) “Chapter 3: Matrix Computations”, *Rounding Errors in Algebraic Processes*, London: Her Majesty’s Stationery Office (National Physical Laboratory, Notes in Applied Science, No.32)

[27] Deaton, Angus (1992). *Understanding Consumption*. Oxford University Press. ISBN 0-19-828824-7.

[28] Krugman, Paul R.; Obstfeld, M.; Melitz, Marc J. (2012). *International Economics: Theory and Policy* (9th global ed.). Harlow: Pearson. ISBN 9780273754091.

[29] Laidler, David E. W. (1993). “The Demand for Money: Theories, Evidence, and Problems” (4th ed.). New York: Harper Collins. ISBN 0065010981.

[30] Ehrenberg; Smith (2008). *Modern Labor Economics* (10th international ed.). London: Addison-Wesley. ISBN 9780321538963.

[31] EEMP webpage

5.7 References

- Cohen, J., Cohen P., West, S.G., & Aiken, L.S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences.* (2nd ed.) Hillsdale, NJ: Lawrence Erlbaum Associates
- Charles Darwin. *The Variation of Animals and Plants under Domestication.* (1868) *(Chapter XIII describes what was known about reversion in Galton’s time. Darwin uses the term “reversion”.)*
- Draper, N.R.; Smith, H. (1998). *Applied Regression Analysis* (3rd ed.). John Wiley. ISBN 0-471-17082-8.
- Francis Galton. “Regression Towards Mediocrity in Hereditary Stature,” *Journal of the Anthropological Institute*, 15:246-263 (1886). *(Facsimile at:)*
- Robert S. Pindyck and Daniel L. Rubinfeld (1998, 4h ed.). *Econometric Models and Economic Forecasts*, ch. 1 (Intro, incl. appendices on Σ operators & derivation of parameter est.) & Appendix 4.3 (mult. regression in matrix form).

5.8 Further reading

- Barlow, Jesse L. (1993). “Chapter 9: Numerical aspects of Solving Linear Least Squares Problems”. In Rao, C.R. *Computational Statistics*. Handbook of Statistics **9**. North-Holland. ISBN 0-444-88096-8
- Björck, Åke (1996). *Numerical methods for least squares problems*. Philadelphia: SIAM. ISBN 0-89871-360-9.
- Goodall, Colin R. (1993). “Chapter 13: Computation using the QR decomposition”. In Rao, C.R. *Computational Statistics*. Handbook of Statistics **9**. North-Holland. ISBN 0-444-88096-8
- Pedhazur, Elazar J (1982). “Multiple regression in behavioral research: Explanation and prediction” (2nd ed.). New York: Holt, Rinehart and Winston. ISBN 0-03-041760-0.
- National Physical Laboratory (1961). “Chapter 1: Linear Equations and Matrices: Direct Methods”. *Modern Computing Methods*. Notes on Applied Science **16** (2nd ed.). Her Majesty’s Stationery Office
- National Physical Laboratory (1961). “Chapter 2: Linear Equations and Matrices: Direct Methods on Automatic Computers”. *Modern Computing Methods*. Notes on Applied Science **16** (2nd ed.). Her Majesty’s Stationery Office

5.9 External links

- Online Linear Regression Calculator & Trend Line Graphing Tool
- Using gradient descent in C++, Boost, Ublas for linear regression

- Lecture notes on linear regression analysis (Robert Nau, Duke University)

Chapter 6

Logistic regression

In statistics, **logistic regression**, or **logit regression**, or **logit model**[1] is a direct probability model that was developed by statistician D. R. Cox in 1958[2] [3] although much work was done in the single independent variable case almost two decades earlier. The binary logistic model is used to predict a binary response based on one or more predictor variables (features). That is, it is used in estimating the parameters of a qualitative response model. The probabilities describing the possible outcomes of a single trial are modeled, as a function of the explanatory (predictor) variables, using a logistic function. Frequently (and hereafter in this article) "logistic regression" is used to refer specifically to the problem in which the dependent variable is binary—that is, the number of available categories is two—while problems with more than two categories are referred to as multinomial logistic regression or **polytomous logistic regression**, or, if the multiple categories are ordered, as ordinal logistic regression.[3]

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables, which are usually (but not necessarily) continuous, by estimating probabilities. Thus, it treats the same set of problems as does probit regression using similar techniques; the first assumes a logistic function and the second a standard normal distribution function.

Logistic regression can be seen as a special case of generalized linear model and thus analogous to linear regression. The model of logistic regression, however, is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression. In particular the key differences of these two models can be seen in the following two features of logistic regression. First, the conditional distribution $p(y \mid x)$ is a Bernoulli distribution rather than a Gaussian distribution, because the dependent variable is binary. Second, the estimated probabilities are restricted to [0,1] through the logistic distribution function because logistic regression predicts the **probability** of the instance being positive.

Logistic regression is an alternative to Fisher's 1936 classification method, linear discriminant analysis.[4] If the assumptions of linear discriminant analysis hold, application of Bayes' rule to reverse the conditioning results in the logistic model, so if linear discriminant assumptions are true, logistic regression assumptions must hold. The converse is not true, so the logistic model has fewer assumptions than discriminant analysis and makes no assumption on the distribution of the independent variables.

6.1 Fields and example applications

Logistic regression is used widely in many fields, including the medical and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression.[5] Many other medical scales used to assess severity of a patient have been developed using logistic regression.[6][7][8][9] Logistic regression may be used to predict whether a patient has a given disease (e.g. diabetes; coronary heart disease), based on observed characteristics of the patient (age, sex, body mass index, results of various blood tests, etc.; age, blood cholesterol level, systolic blood pressure, relative weight, blood hemoglobin level, smoking (at 3 levels), and abnormal electrocardiogram.).[1][10] Another example might be to predict whether an American voter will vote Democratic or Republican, based on age, income, sex, race, state of residence, votes in previous elections, etc.[11] The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product.[12][13] It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc. In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a mortgage. Conditional random fields, an extension of logistic regression to sequential data, are used in natural language processing.

6.2 Basics

Logistic regression can be binomial or multinomial. Binomial or binary logistic regression deals with situations in which the observed outcome for a dependent variable can have only two possible types (for example, "dead" vs. "alive" or "win" vs. "loss"). Multinomial logistic regression deals with situations where the outcome can have three or more possible types (e.g., "disease A" vs. "disease B" vs. "disease C"). In binary logistic regression, the outcome is usually coded as "0" or "1", as this leads to the most straightforward interpretation.[14] If a particular observed outcome for the dependent variable is the noteworthy possible outcome (referred to as a "success" or a "case") it is usually coded as "1" and the contrary outcome (referred to as a "failure" or a "noncase") as "0". Logistic regression is used to predict the odds of being a case based on the values of the independent variables (predictors). The odds are defined as the probability that a particular outcome is a case divided by the probability that it is a noncase.

Like other forms of regression analysis, logistic regression makes use of one or more predictor variables that may be either continuous or categorical data. Unlike ordinary linear regression, however, logistic regression is used for predicting binary outcomes of the dependent variable (treating the dependent variable as the outcome of a Bernoulli trial) rather than a continuous outcome. Given this difference, it is necessary that logistic regression take the natural logarithm of the odds of the dependent variable being a case (referred to as the logit or log-odds) to create a continuous criterion as a transformed version of the dependent variable. Thus the logit transformation is referred to as the *link function* in logistic regression—although the dependent variable in logistic regression is binomial, the logit is the continuous criterion upon which linear regression is conducted.[14]

The logit of success is then fitted to the predictors using linear regression analysis. The predicted value of the logit is converted back into predicted odds via the inverse of the natural logarithm, namely the exponential function. Thus, although the observed dependent variable in logistic regression is a zero-or-one variable, the logistic regression estimates the odds, as a continuous variable, that the dependent variable is a success (a case). In some applications the odds are all that is needed. In others, a specific yes-or-no prediction is needed for whether the dependent variable is or is not a case; this categorical prediction can be based on the computed odds of a success, with predicted odds above some chosen cutoff value being translated into a prediction of a success.

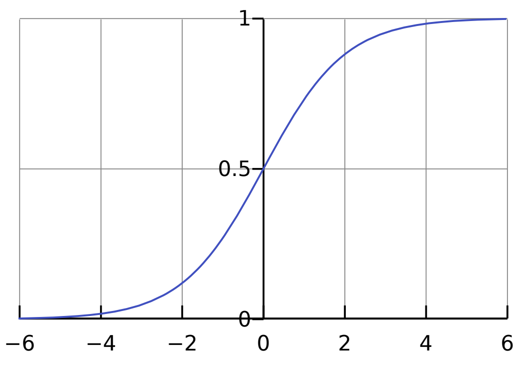


Figure 1. The logistic function $\sigma(t)$; note that $\sigma(t) \in [0,1]$ for all t .

6.3 Logistic function, odds, odds ratio, and logit

6.3.1 Definition of the logistic function

An explanation of logistic regression begins with an explanation of the logistic function. The logistic function is useful because it can take an input with any value from negative to positive infinity, whereas the output always takes values between zero and one[14] and hence is interpretable as a probability. The logistic function $\sigma(t)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}},$$

A graph of the logistic function is shown in Figure 1.

If t is viewed as a linear function of an explanatory variable x (or of a linear combination of explanatory variables), then we express t as follows:

$$t = \beta_0 + \beta_1 x$$

And the logistic function can now be written as:

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

Note that $F(x)$ is interpreted as the probability of the dependent variable equaling a "success" or "case" rather than a failure or non-case. It's clear that the response variables Y_i are not identically distributed: $P(Y_i = 1 \mid X)$ differs from one data point X_i to another, though they are independent given design matrix X and shared with parameters β .[1]

6.3.2 Definition of the inverse of the logistic function

We can now define the inverse of the logistic function, g , the logit (log odds):

$$g(F(x)) = \ln \frac{F(x)}{1 - F(x)} = \beta_0 + \beta_1 x,$$

and equivalently:

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

6.3.3 Interpretation of these terms

In the above equations, the terms are as follows:

- $g(\cdot)$ refers to the logit function. The equation for $g(F(x))$ illustrates that the logit (i.e., log-odds or natural logarithm of the odds) is equivalent to the linear regression expression.
- ln denotes the natural logarithm.
- $F(x)$ is the probability that the dependent variable equals a case, given some linear combination x of the predictors. The formula for $F(x)$ illustrates that the probability of the dependent variable equaling a case is equal to the value of the logistic function of the linear regression expression. This is important in that it shows that the value of the linear regression expression can vary from negative to positive infinity and yet, after transformation, the resulting expression for the probability $F(x)$ ranges between 0 and 1.
- β_0 is the intercept from the linear regression equation (the value of the criterion when the predictor is equal to zero).
- $\beta_1 x$ is the regression coefficient multiplied by some value of the predictor.
- base e denotes the exponential function.

6.3.4 Definition of the odds

The odds of the dependent variable equaling a case (given some linear combination x of the predictors) is equivalent to the exponential function of the linear regression expression. This illustrates how the logit serves as a link function between the probability and the linear regression expression. Given that the logit ranges between negative and positive infinity, it provides an adequate criterion upon which to conduct linear regression and the logit is easily converted back into the odds.[14]

So we define odds of the dependent variable equaling a case (given some linear combination x of the predictors) as follows:

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

6.3.5 Definition of the odds ratio

The odds ratio can be defined as:

$$OR = \text{odds}(x+1)/\,\text{odds}(x) = \frac{\frac{F(x+1)}{1-F(x+1)}}{\frac{F(x)}{1-F(x)}} = e^{\beta_0+\beta_1(x+1)}/e^{\beta_0+\beta_1 x} = e^{\beta_1}$$

or for binary variable F(0) instead of F(x) and F(1) for F(x+1). This exponential relationship provides an interpretation for β_1 : The odds multiply by e^{β_1} for every 1-unit increase in x.[15]

6.3.6 Multiple explanatory variables

If there are multiple explanatory variables, the above expression $\beta_0 + \beta_1 x$ can be revised to $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$. Then when this is used in the equation relating the logged odds of a success to the values of the predictors, the linear regression will be a multiple regression with *m* explanators; the parameters β_j for all *j* = 0, 1, 2, ..., *m* are all estimated.

6.4 Model fitting

6.4.1 Estimation

Because the model can be expressed as a generalized linear model (see below), for 0<p<1, ordinary least squares can suffice, with R-squared as the measure of goodness of fit in the fitting space. When p=0 or 1, more complex methods are required.

Maximum likelihood estimation

The regression coefficients are usually estimated using maximum likelihood estimation.[16] Unlike linear regression with normally distributed residuals, it is not possible to find a closed-form expression for the coefficient values that maximize the likelihood function, so that an iterative process must be used instead; for example Newton's method. This process begins with a tentative solution, revises it slightly to see if it can be improved, and repeats this revision until improvement is minute, at which point the process is said to have converged.[17]

In some instances the model may not reach convergence. Nonconvergence of a model indicates that the coefficients are not meaningful because the iterative process was unable to find appropriate solutions. A failure to converge may occur for a number of reasons: having a large ratio of predictors to cases, multicollinearity, sparseness, or complete separation.

- Having a large ratio of variables to cases results in an overly conservative Wald statistic (discussed below) and can lead to nonconvergence.
- Multicollinearity refers to unacceptably high correlations between predictors. As multicollinearity increases, coefficients remain unbiased but standard errors increase and the likelihood of model convergence decreases.[16] To detect multicollinearity amongst the predictors, one can conduct a linear regression analysis with the predictors of interest for the sole purpose of examining the tolerance statistic [16] used to assess whether multicollinearity is unacceptably high.
- Sparseness in the data refers to having a large proportion of empty cells (cells with zero counts). Zero cell counts are particularly problematic with categorical predictors. With continuous predictors, the model can infer values for the zero cell counts, but this is not the case with categorical predictors. The model will not converge with zero cell counts for categorical predictors because the natural logarithm of zero is an undefined value, so that final solutions to the model cannot be reached. To remedy this problem, researchers may collapse categories in a theoretically meaningful way or add a constant to all cells.[16]
- Another numerical problem that may lead to a lack of convergence is complete separation, which refers to the instance in which the predictors perfectly predict the criterion – all cases are accurately classified. In such instances, one should reexamine the data, as there is likely some kind of error.[14]

As a general rule of thumb, logistic regression models require a minimum of about 10 events per explaining variable (where *event* denotes the cases belonging to the less frequent category in the dependent variable).[18]

Minimum chi-squared estimator for grouped data

While individual data will have a dependent variable with a value of zero or one for every observation, with grouped data one observation is on a group of people who all share the same characteristics (e.g., demographic characteristics); in this case the researcher observes the proportion of people in the group for whom the response variable falls into one category or the other. If this proportion is neither zero nor one for any group, the minimum chi-squared estimator involves using weighted least squares to estimate a linear model in which the dependent variable is the logit of the proportion: that is, the log of the ratio of the fraction in one group to the fraction in the other group.[19]:pp.686–9

6.4.2 Evaluating goodness of fit

Goodness of fit in linear regression models is generally measured using the R^2. Since this has no direct analog in logistic regression, various methods[19]:ch.21 including the following can be used instead.

Deviance and likelihood ratio tests

In linear regression analysis, one is concerned with partitioning variance via the sum of squares calculations – variance in the criterion is essentially divided into variance accounted for by the predictors and residual variance. In logistic regression analysis, deviance is used in lieu of sum of squares calculations.[20] Deviance is analogous to the sum of squares calculations in linear regression[14] and is a measure of the lack of fit to the data in a logistic regression model.[20] When a "saturated" model is available (a model with a theoretically perfect fit), deviance is calculated by comparing a given model with the saturated model.[14] This computation give the likelihood-ratio test:.[14]

$$D = -2 \ln \frac{\text{model fitted the of likelihood}}{\text{model saturated the of likelihood}}.$$

In the above equation D represents the deviance and ln represents the natural logarithm. The log of the likelihood ratio (the ratio of the fitted model to the saturated model) will produce a negative value, so the product is multiplied by negative two times its natural logarithm to produce a value with an approximate chi-squared distribution.[14] Smaller values indicate better fit as the fitted model deviates less from the saturated model. When assessed upon a chi-square distribution, nonsignificant chi-square values indicate very little unexplained variance and thus, good model fit. Conversely, a significant chi-square value indicates that a significant amount of the variance is unexplained.

When the saturated model is not available (a common case), deviance is calculated simply as (-2)x(log likelihood of the fitted model), and the reference to the saturated model's log likelihood can be removed from all that follows without harm.

Two measures of deviance are particularly important in logistic regression: null deviance and model deviance. The null deviance represents the difference between a model with only the intercept (which means "no predictors") and the saturated model. The model deviance represents the difference between a model with at least one predictor and the saturated model.[20] In this respect, the null model provides a baseline upon which to compare predictor models. Given that deviance is a measure of the difference between a given model and the saturated model, smaller values indicate better fit. Thus, to assess the contribution of a predictor or set of predictors, one can subtract the model deviance from the null deviance and assess the difference on a χ^2_{s-p}, chi-square distribution with degrees of freedom[14] equal to the difference in the number of parameters estimated.

Let

$$D_{\text{null}} = -2 \ln \frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}}$$

$$D_{\text{fitted}} = -2 \ln \frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}.$$

Then

$$\begin{aligned} D_{\text{null}} - D_{\text{fitted}} &= \left(-2 \ln \frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}}\right) - \left(-2 \ln \frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}\right) \\ &= -2 \left(\ln \frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}} - \ln \frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}\right) \\ &= -\,2 \ln \frac{\left(\frac{\text{model null of likelihood}}{\text{model saturated the of likelihood}}\right)}{\left(\frac{\text{model fitted of likelihood}}{\text{model saturated the of likelihood}}\right)} \\ &= -\,2 \ln \frac{\text{model null the of likelihood}}{\text{model fitted of likelihood}}. \end{aligned}$$

If the model deviance is significantly smaller than the null deviance then one can conclude that the predictor or set of predictors significantly improved model fit. This is analogous to the F-test used in linear regression analysis to assess the significance of prediction.[20]

Pseudo-R^{2}s

In linear regression the squared multiple correlation, R^2 is used to assess goodness of fit as it represents the proportion of variance in the criterion that is explained by the predictors.[20] In logistic regression analysis, there is no agreed upon analogous measure, but there are several competing measures each with limitations.[20] Three of the most commonly used indices are examined on this page beginning with the likelihood ratio R^2, R^2L:[20]

$$R^2_{\text{L}} = \frac{D_{\text{null}} - D_{\text{fitted}}}{D_{\text{null}}}.$$

This is the most analogous index to the squared multiple correlation in linear regression.[16] It represents the proportional reduction in the deviance wherein the deviance is treated as a measure of variation analogous but not identical to the variance in linear regression analysis.[16] One limitation of the likelihood ratio R^2 is that it is not monotonically related to the odds ratio,[20] meaning that it does not necessarily increase as the odds ratio increases and does not necessarily decrease as the odds ratio decreases.

The Cox and Snell R^2 is an alternative index of goodness of fit related to the R^2 value from linear regression. The Cox and Snell index is problematic as its maximum value is .75, when the variance is at its maximum (.25). The Nagelkerke R^2 provides a correction to the Cox and Snell R^2 so that the maximum value is equal to one. Nevertheless, the Cox and Snell and likelihood ratio R^2s show greater agreement with each other than either does with the Nagelkerke R^2.[20] Of course, this might not be the case for values exceeding .75 as the Cox and Snell index is capped at this value. The likelihood ratio R^2 is often preferred to the alternatives as it is most analogous to R^2 in linear regression, is independent of the base rate (both Cox and Snell and Nagelkerke R^2s increase as the proportion of cases increase from 0 to .5) and varies between 0 and 1.

A word of caution is in order when interpreting pseudo-R^2 statistics. The reason these indices of fit are referred to as *pseudo* R^2 is that they do not represent the proportionate reduction in error as the R^2 in linear regression does.[20] Linear regression assumes homoscedasticity, that the error variance is the same for all values of the criterion. Logistic regression will always be heteroscedastic – the error variances differ for each value of the predicted score. For each value of the predicted score there would be a different value of the proportionate reduction in error. Therefore, it is inappropriate to think of R^2 as a proportionate reduction in error in a universal sense in logistic regression.[20]

Hosmer–Lemeshow test

The Hosmer–Lemeshow test uses a test statistic that asymptotically follows a χ^2 distribution to assess whether or not the observed event rates match expected event rates in subgroups of the model population.

Evaluating binary classification performance

If the estimated probabilities are to be used to classify each observation of independent variable values as predicting the category that the dependent variable is found in, the various methods below for judging the model's suitability in out-of-sample forecasting can also be used on the data that were used for estimation—accuracy, precision (also called positive predictive value), recall (also called sensitivity), specificity and negative predictive value. In each of these evaluative methods, an aspect of the model's effectiveness in assigning instances to the correct categories is measured.

6.5 Coefficients

After fitting the model, it is likely that researchers will want to examine the contribution of individual predictors. To do so, they will want to examine the regression coefficients. In linear regression, the regression coefficients represent the change in the criterion for each unit change in the predictor.[20] In logistic regression, however, the regression coefficients represent the change in the logit for each unit change in the predictor. Given that the logit is not intuitive, researchers are likely to focus on a predictor's effect on the exponential function of the regression coefficient – the odds ratio (see definition). In linear regression, the significance of a regression coefficient is assessed by computing a *t* test. In logistic regression, there are several different tests designed to assess the significance of an individual predictor, most notably the likelihood ratio test and the Wald statistic.

6.5.1 Likelihood ratio test

The likelihood-ratio test discussed above to assess model fit is also the recommended procedure to assess the contribution of individual "predictors" to a given model.[14][16][20] In the case of a single predictor model, one simply compares the deviance of the predictor model with that of the null model on a chi-square distribution with a single degree of freedom. If the predictor model has a significantly smaller deviance (c.f chi-square using the difference in degrees of freedom of the two models), then one can conclude that there is a significant association between the "predictor" and the outcome. Although some common statistical packages (e.g. SPSS) do provide likelihood ratio test statistics, without this computationally intensive test it would be more difficult to assess the contribution of individual predictors in the multiple logistic regression case. To assess the contribution of individual predictors one can enter the predictors hierarchically, comparing each new model with the previous to determine the contribution of each predictor.[20] There is some debate among statisticians about the appropriateness of so-called "stepwise" procedures. The fear is that they may not preserve nominal statistical properties and may become misleading.

6.5.2 Wald statistic

Alternatively, when assessing the contribution of individual predictors in a given model, one may examine the significance of the Wald statistic. The Wald statistic, analogous to the *t*-test in linear regression, is used to assess the significance of coefficients. The Wald statistic is the ratio of the square of the regression coefficient to the square of the standard error of the coefficient and is asymptotically distributed as a chi-square distribution.[16]

$W_j = \frac{B_j^2}{SE_{B_j}^2}$

Although several statistical packages (e.g., SPSS, SAS) report the Wald statistic to assess the contribution of individual predictors, the Wald statistic has limitations. When the regression coefficient is large, the standard error of the regression coefficient also tends to be large increasing the probability of Type-II error. The Wald statistic also tends to be biased when data are sparse.[20]

6.5.3 Case-control sampling

Suppose cases are rare. Then we might wish to sample them more frequently than their prevalence in the population. For example, suppose there is a disease that affects 1 person in 10,000 and to collect our data we need to do a complete physical. It may be too expensive to do thousands of physicals of healthy people in order to obtain data for only a few diseased individuals. Thus, we may evaluate more diseased individuals. This is also called unbalanced data. As a rule of thumb, sampling controls at a rate of five times the number of cases will produce sufficient control data.[21]

If we form a logistic model from such data, if the model is correct, the β_j parameters are all correct except for β_0 . We can correct β_0 if we know the true prevalence as follows:[21]

$\hat{\beta}_0^* = \hat{\beta}_0 + \log \frac{\pi}{1-\pi} - \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$

where π is the true prevalence and $\tilde{\pi}$ is the prevalence in the sample.

6.6 Formal mathematical specification

There are various equivalent specifications of logistic regression, which fit into different types of more general models. These different specifications allow for different sorts of useful generalizations.

6.6.1 Setup

The basic setup of logistic regression is the same as for standard linear regression.

It is assumed that we have a series of *N* observed data points. Each data point *i* consists of a set of *m* explanatory variables x_1,*i* ... *xm,i* (also called independent variables, predictor variables, input variables, features, or attributes), and an associated binary-valued outcome variable *Yi* (also known as a dependent variable, response variable, output variable, outcome variable or class variable), i.e. it can assume only the two possible values 0 (often meaning "no" or "failure") or 1 (often meaning "yes" or "success"). The goal of logistic regression is to explain the relationship between the explanatory variables and the outcome, so that an outcome can be predicted for a new set of explanatory variables.

Some examples:

- The observed outcomes are the presence or absence of a given disease (e.g. diabetes) in a set of patients, and the explanatory variables might be characteristics of the patients thought to be pertinent (sex, race, age, blood pressure, body-mass index, etc.).

- The observed outcomes are the votes (e.g. Democratic or Republican) of a set of people in an election, and the explanatory variables are the demographic characteristics of each person (e.g. sex, race, age, income, etc.). In such a case, one of the two outcomes is arbitrarily coded as 1, and the other as 0.

As in linear regression, the outcome variables *Yi* are assumed to depend on the explanatory variables *x*1*,i* ... *xm,i*.

Explanatory variables

As shown above in the above examples, the explanatory variables may be of any type: real-valued, binary, categorical, etc. The main distinction is between continuous variables (such as income, age and blood pressure) and discrete variables (such as sex or race). Discrete variables referring to more than two possible choices are typically coded using dummy variables (or indicator variables), that is, separate explanatory variables taking the value 0 or 1 are created for each possible value of the discrete variable, with a 1 meaning "variable does have the given value" and a 0 meaning "variable does not have that value". For example, a four-way discrete variable of blood type with the possible values "A, B, AB, O" can be converted to four separate two-way dummy variables, "is-A, is-B, is-AB, is-O", where only one of them has the value 1 and all the rest have the value 0. This allows for separate regression coefficients to be matched for each possible value of the discrete variable. (In a case like this, only three of the four dummy variables are independent of each other, in the sense that once the values of three of the variables are known, the fourth is automatically determined. Thus, it is necessary to encode only three of the four possibilities as dummy variables. This also means that when all four possibilities are encoded, the overall model is not identifiable in the absence of additional constraints such as a regularization constraint. Theoretically, this could cause problems, but in reality almost all logistic regression models are fitted with regularization constraints.)

Outcome variables

Formally, the outcomes *Yi* are described as being Bernoulli-distributed data, where each outcome is determined by an unobserved probability *pi* that is specific to the outcome at hand, but related to the explanatory variables. This can be expressed in any of the following equivalent forms:

$$\begin{aligned}
Y_i \mid x_{1,i},\ldots,x_{m,i} \ &\sim \text{Bernoulli}(p_i) \\
\mathbb{E}[Y_i \mid x_{1,i},\ldots,x_{m,i}] &= p_i \\
\Pr(Y_i = y_i \mid x_{1,i},\ldots,x_{m,i}) &= \begin{cases} p_i & \text{if} y_i = 1 \\ 1-p_i & \text{if} y_i = 0 \end{cases} \\
\Pr(Y_i = y_i \mid x_{1,i},\ldots,x_{m,i}) &= p_i^{y_i}(1-p_i)^{(1-y_i)}
\end{aligned}$$

The meanings of these four lines are:

1. The first line expresses the probability distribution of each *Yi*: Conditioned on the explanatory variables, it follows a Bernoulli distribution with parameters *pi*, the probability of the outcome of 1 for trial *i*. As noted above, each separate trial has its own probability of success, just as each trial has its own explanatory variables. The probability of success *pi* is not observed, only the outcome of an individual Bernoulli trial using that probability.

2. The second line expresses the fact that the expected value of each *Yi* is equal to the probability of success *pi*, which is a general property of the Bernoulli distribution. In other words, if we run a large number of Bernoulli trials using the same probability of success *pi*, then take the average of all the 1 and 0 outcomes, then the result would be close to *pi*. This is because doing an average this way simply computes the proportion of successes seen, which we expect to converge to the underlying probability of success.

3. The third line writes out the probability mass function of the Bernoulli distribution, specifying the probability of seeing each of the two possible outcomes.

4. The fourth line is another way of writing the probability mass function, which avoids having to write separate cases and is more convenient for certain types of calculations. This relies on the fact that *Yi* can take only the value 0 or 1. In each case, one of the exponents will be 1, "choosing" the value under it, while the other is 0, "canceling out" the value under it. Hence, the outcome is either *pi* or 1 − *pi*, as in the previous line.

Linear predictor function

The basic idea of logistic regression is to use the mechanism already developed for linear regression by modeling the probability *pi* using a linear predictor function, i.e. a linear combination of the explanatory variables and a set of regression coefficients that are specific to the model at hand but the same for all trials. The linear predictor function $f(i)$ for a particular data point *i* is written as:

$$f(i) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_m x_{m,i},$$

where $\beta_0, \ldots, \beta_m$ are regression coefficients indicating the relative effect of a particular explanatory variable on the outcome.

The model is usually put into a more compact form as follows:

- The regression coefficients β_0, β_1, ..., βm are grouped into a single vector $\boldsymbol{\beta}$ of size *m* + 1.
- For each data point *i*, an additional explanatory pseudo-variable x_0,*i* is added, with a fixed value of 1, corresponding to the intercept coefficient β_0.
- The resulting explanatory variables x_0,*i*, x_1,*i*, ..., *xm,i* are then grouped into a single vector ***Xi*** of size *m* + 1.

This makes it possible to write the linear predictor function as follows:

$$f(i) = \boldsymbol{\beta} \cdot \mathbf{X}_i,$$

using the notation for a dot product between two vectors.

6.6.2 As a generalized linear model

The particular model used by logistic regression, which distinguishes it from standard linear regression and from other types of regression analysis used for binary-valued outcomes, is the way the probability of a particular outcome is linked to the linear predictor function:

$$\operatorname{logit}(\mathbb{E}[Y_i \mid x_{1,i}, \ldots, x_{m,i}]) = \operatorname{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_m x_{m,i}$$

Written using the more compact notation described above, this is:

$$\operatorname{logit}(\mathbb{E}[Y_i \mid \mathbf{X}_i]) = \operatorname{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \boldsymbol{\beta} \cdot \mathbf{X}_i$$

This formulation expresses logistic regression as a type of generalized linear model, which predicts variables with various types of probability distributions by fitting a linear predictor function of the above form to some sort of arbitrary transformation of the expected value of the variable.

The intuition for transforming using the logit function (the natural log of the odds) was explained above. It also has the practical effect of converting the probability (which is bounded to be between 0 and 1) to a variable that ranges over $(-\infty, +\infty)$ — thereby matching the potential range of the linear prediction function on the right side of the equation.

Note that both the probabilities *pi* and the regression coefficients are unobserved, and the means of determining them is not part of the model itself. They are typically determined by some sort of optimization procedure, e.g. maximum likelihood estimation, that finds values that best fit the observed data (i.e. that give the most accurate predictions for the data already observed), usually subject to regularization conditions that seek to exclude unlikely values, e.g. extremely large values for any of the regression coefficients. The use of a regularization condition is equivalent to doing maximum a posteriori (MAP) estimation, an extension of maximum likelihood. (Regularization is most commonly done using a squared regularizing function, which is equivalent to placing a zero-mean Gaussian prior distribution on the coefficients, but other regularizers are also possible.) Whether or not regularization is used, it is usually not possible to find a closed-form solution; instead, an iterative numerical method must be used, such as iteratively reweighted least squares (IRLS) or, more commonly these days, a quasi-Newton method such as the L-BFGS method.

The interpretation of the βj parameter estimates is as the additive effect on the log of the odds for a unit change in the *j*th explanatory variable. In the case of a dichotomous explanatory variable, for instance gender, e^β is the estimate of the odds of having the outcome for, say, males compared with females.

An equivalent formula uses the inverse of the logit function, which is the logistic function, i.e.:

$$\mathbb{E}[Y_i \mid \mathbf{X}_i] = p_i = \operatorname{logit}^{-1}(\boldsymbol{\beta} \cdot \mathbf{X}_i) = \frac{1}{1+e^{-\boldsymbol{\beta}\cdot\mathbf{X}_i}}$$

The formula can also be written (somewhat awkwardly) as a probability distribution (specifically, using a probability mass function):

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

6.6.3 As a latent-variable model

The above model has an equivalent formulation as a latent-variable model. This formulation is common in the theory of discrete choice models, and makes it easier to extend to certain more complicated models with multiple, correlated choices, as well as to compare logistic regression to the closely related probit model.

Imagine that, for each trial *i*, there is a continuous latent variable *Yi** (i.e. an unobserved random variable) that is distributed as follows:

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

where

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

i.e. the latent variable can be written directly in terms of the linear predictor function and an additive random error variable that is distributed according to a standard logistic distribution.

Then *Yi* can be viewed as an indicator for whether this latent variable is positive:

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

The choice of modeling the error variable specifically with a standard logistic distribution, rather than a general logistic distribution with the location and scale set to arbitrary values, seems restrictive, but in fact it is not. It must be kept in mind that we can choose the regression coefficients ourselves, and very often can use them to offset changes in the parameters of the error variable's distribution. For example, a logistic error-variable distribution with a non-zero location parameter *μ* (which sets the mean) is equivalent to a distribution with a zero location parameter, where *μ* has been added to the intercept coefficient. Both situations produce the same value for *Yi** regardless of settings of explanatory variables. Similarly, an arbitrary scale parameter *s* is equivalent to setting the scale parameter to 1 and then dividing all regression coefficients by *s*. In the latter case, the resulting value of *Yi** will be smaller by a factor of *s* than in the former case, for all sets of explanatory variables — but critically, it will always remain on the same side of 0, and hence lead to the same *Yi* choice.

(Note that this predicts that the irrelevancy of the scale parameter may not carry over into more complex models where more than two choices are available.)

It turns out that this formulation is exactly equivalent to the preceding one, phrased in terms of the generalized linear model and without any latent variables. This can be shown as follows, using the fact that the cumulative distribution function (CDF) of the standard logistic distribution is the logistic function, which is the inverse of the logit function, i.e.

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

Then:

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

This formulation—which is standard in discrete choice models—makes clear the relationship between logistic regression (the "logit model") and the probit model, which uses an error variable distributed according to a standard normal distribution instead of a standard logistic distribution. Both the logistic and normal distributions are symmetric with a basic unimodal, "bell curve" shape. The only difference is that the logistic distribution has somewhat heavier tails, which means that it is less sensitive to outlying data (and hence somewhat more robust to model mis-specifications or erroneous data).

6.6.4 As a two-way latent-variable model

Yet another formulation uses two separate latent variables:

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

where

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

where $EV_1(0,1)$ is a standard type-1 extreme value distribution: i.e.

$$\Pr(\varepsilon_0 = x) = \Pr(\varepsilon_1 = x) = e^{-x}e^{-e^{-x}}$$

Then

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

This model has a separate latent variable and a separate set of regression coefficients for each possible outcome of the dependent variable. The reason for this separation is that it makes it easy to extend logistic regression to multi-outcome categorical variables, as in the multinomial logit model. In such a model, it is natural to model each possible outcome using a different set of regression coefficients. It is also possible to motivate each of the separate latent variables as the theoretical utility associated with making the associated choice, and thus motivate logistic regression in terms of utility theory. (In terms of utility theory, a rational actor always chooses the choice with the greatest associated utility.) This is the approach taken by economists when formulating discrete choice models, because it both provides a theoretically strong foundation and facilitates intuitions about the model, which in turn makes it easy to consider various sorts of extensions. (See the example below.)

The choice of the type-1 extreme value distribution seems fairly arbitrary, but it makes the mathematics work out, and it may be possible to justify its use through rational choice theory.

It turns out that this model is equivalent to the previous model, although this seems non-obvious, since there are now two sets of regression coefficients and error variables, and the error variables have a different distribution. In fact, this model reduces directly to the previous one with the following substitutions:

$$\boldsymbol\beta = \boldsymbol\beta_1 - \boldsymbol\beta_0$$

$$\varepsilon = \varepsilon_1 - \varepsilon_0$$

An intuition for this comes from the fact that, since we choose based on the maximum of two values, only their difference matters, not the exact values — and this effectively removes one degree of freedom. Another critical fact is that the difference of two type-1 extreme-value-distributed variables is a logistic distribution, i.e. if $\varepsilon = \varepsilon_1 - \varepsilon_0 \sim \text{Logistic}(0, 1)$.

We can demonstrate the equivalent as follows:

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

Example

As an example, consider a province-level election where the choice is between a right-of-center party, a left-of-center party, and a secessionist party (e.g. the Parti Québécois, which wants Quebec to secede from Canada). We would then use three latent variables, one for each choice. Then, in accordance with utility theory, we can then interpret the latent variables as expressing the utility that results from making each of the choices. We can also interpret the regression coefficients as indicating the strength that the associated factor (i.e. explanatory variable) has in contributing to the utility — or more correctly, the amount by which a unit change in an explanatory variable changes the utility of a given choice. A voter might expect that the right-of-center party would lower taxes, especially on rich people. This would give low-income people no benefit, i.e. no change in utility (since they usually don't pay taxes); would cause moderate benefit (i.e. somewhat more money, or moderate utility increase) for middle-incoming people; and would cause significant benefits for high-income people. On the other hand, the left-of-center party might be expected to raise taxes and offset it with increased welfare and other assistance for the lower and middle classes. This would cause significant positive benefit to low-income people, perhaps weak benefit to middle-income people, and significant negative benefit to high-income people. Finally, the secessionist party would take no direct actions on the economy, but simply secede. A low-income or middle-income voter might expect basically no clear utility gain or loss from this, but a high-income voter might expect negative utility, since he/she is likely to own companies, which will have a harder time doing business in such an environment and probably lose money.

These intuitions can be expressed as follows:

This clearly shows that

1. Separate sets of regression coefficients need to exist for each choice. When phrased in terms of utility, this can be seen very easily. Different choices have different effects on net utility; furthermore, the effects vary in complex ways that depend on the characteristics of each individual, so there need to be separate sets of coefficients for each characteristic, not simply a single extra per-choice characteristic.

2. Even though income is a continuous variable, its effect on utility is too complex for it to be treated as a single variable. Either it needs to be directly split up into ranges, or higher powers of income need to be added so that polynomial regression on income is effectively done.

6.6.5 As a "log-linear" model

Yet another formulation combines the two-way latent variable formulation above with the original formulation higher up without latent variables, and in the process provides a link to one of the standard formulations of the multinomial logit.

Here, instead of writing the logit of the probabilities *pi* as a linear predictor, we separate the linear predictor into two, one for each of the two outcomes:

$$\ln \Pr(Y_i = 0) = \boldsymbol\beta_0 \cdot \mathbf{X}_i - \ln Z$$
$$\ln \Pr(Y_i = 1) = \boldsymbol\beta_1 \cdot \mathbf{X}_i - \ln Z$$

Note that two separate sets of regression coefficients have been introduced, just as in the two-way latent variable model, and the two equations appear a form that writes the logarithm of the associated probability as a linear predictor, with an extra term $-lnZ$ at the end. This term, as it turns out, serves as the normalizing factor ensuring that the result is a distribution. This can be seen by exponentiating both sides:

$$\Pr(Y_i = 0) = \frac{1}{Z} e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i}$$
$$\Pr(Y_i = 1) = \frac{1}{Z} e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}$$

In this form it is clear that the purpose of *Z* is to ensure that the resulting distribution over *Yi* is in fact a probability distribution, i.e. it sums to 1. This means that *Z* is simply the sum of all un-normalized probabilities, and by dividing each probability by *Z*, the probabilities become "normalized". That is:

$$Z = e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}$$

and the resulting equations are

$$\Pr(Y_i = 0) = \frac{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}$$
$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}$$

Or generally:

$$\Pr(Y_i = c) = \frac{e^{\boldsymbol\beta_c \cdot \mathbf{X}_i}}{\sum_h e^{\boldsymbol\beta_h \cdot \mathbf{X}_i}}$$

This shows clearly how to generalize this formulation to more than two outcomes, as in multinomial logit. Note that this general formulation is exactly the Softmax function as in

$$\Pr(Y_i = c) = \operatorname{softmax}(c, \boldsymbol\beta_0 \cdot \mathbf{X}_i, \boldsymbol\beta_1 \cdot \mathbf{X}_i, \dots).$$

In order to prove that this is equivalent to the previous model, note that the above model is overspecified, in that $\Pr(Y_i = 0)$ and $\Pr(Y_i = 1)$ cannot be independently specified: rather $\Pr(Y_i = 0) + \Pr(Y_i = 1) = 1$ so knowing one automatically determines the other. As a result, the model is nonidentifiable, in that multiple combinations of $\boldsymbol\beta_0$ and $\boldsymbol\beta_1$ will produce the same probabilities for all possible explanatory variables. In fact, it can be seen that adding any constant vector to both of them will produce the same probabilities:

$$\begin{aligned}
\Pr(Y_i = 1) &= \frac{e^{(\boldsymbol\beta_1 + \mathbf{C}) \cdot \mathbf{X}_i}}{e^{(\boldsymbol\beta_0 + \mathbf{C}) \cdot \mathbf{X}_i} + e^{(\boldsymbol\beta_1 + \mathbf{C}) \cdot \mathbf{X}_i}} \\
&= \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i} e^{\mathbf{C} \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} e^{\mathbf{C} \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i} e^{\mathbf{C} \cdot \mathbf{X}_i}} \\
&= \frac{e^{\mathbf{C} \cdot \mathbf{X}_i} e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{e^{\mathbf{C} \cdot \mathbf{X}_i} (e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i})} \\
&= \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}
\end{aligned}$$

As a result, we can simplify matters, and restore identifiability, by picking an arbitrary value for one of the two vectors. We choose to set $\boldsymbol\beta_0 = \mathbf{0}$. Then,

$$e^{\boldsymbol\beta_0 \cdot \mathbf{X}_i} = e^{\mathbf{0} \cdot \mathbf{X}_i} = 1$$

and so

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}}{1 + e^{\boldsymbol\beta_1 \cdot \mathbf{X}_i}} = \frac{1}{1 + e^{-\boldsymbol\beta_1 \cdot \mathbf{X}_i}} = p_i$$

which shows that this formulation is indeed equivalent to the previous formulation. (As in the two-way latent variable formulation, any settings where $\boldsymbol\beta = \boldsymbol\beta_1 - \boldsymbol\beta_0$ will produce equivalent results.)

Note that most treatments of the multinomial logit model start out either by extending the “log-linear” formulation presented here or the two-way latent variable formulation presented above, since both clearly show the way that the model could be extended to multi-way outcomes. In general, the presentation with latent variables is more common in econometrics and political science, where discrete choice models and utility theory reign, while the “log-linear” formulation here is more common in computer science, e.g. machine learning and natural language processing.

6.6.6 As a single-layer perceptron

The model has an equivalent formulation

$$p_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i})}}.$$

This functional form is commonly called a single-layer perceptron or single-layer artificial neural network. A single-layer neural network computes a continuous output instead of a step function. The derivative of *pi* with respect to $X = (x_1, ..., xk)$ is computed from the general form:

$$y = \frac{1}{1+e^{-f(X)}}$$

where $f(X)$ is an analytic function in X. With this choice, the single-layer neural network is identical to the logistic regression model. This function has a continuous derivative, which allows it to be used in backpropagation. This function is also preferred because its derivative is easily calculated:

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

6.6.7 In terms of binomial data

A closely related model assumes that each i is associated not with a single Bernoulli trial but with ni independent identically distributed trials, where the observation Yi is the number of successes observed (the sum of the individual Bernoulli-distributed random variables), and hence follows a binomial distribution:

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

An example of this distribution is the fraction of seeds (pi) that germinate after ni are planted.

In terms of expected values, this model is expressed as follows:

$$p_i = \mathbb{E}\left[\left.\frac{Y_i}{n_i}\,\right|\,\mathbf{X}_i\right],$$

so that

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

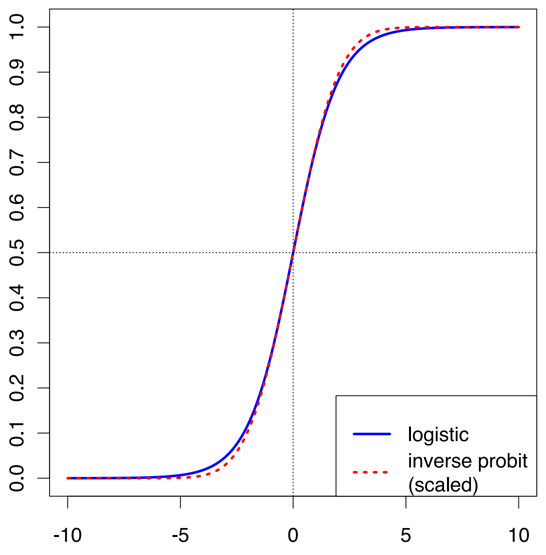
Or equivalently:

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

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

6.7 Bayesian logistic regression

In a Bayesian statistics context, prior distributions are normally placed on the regression coefficients, usually in the form of Gaussian distributions. Unfortunately, the Gaussian distribution is not the conjugate prior of the likelihood function in logistic regression. As a result, the posterior distribution is difficult to calculate, even using standard simulation algorithms (e.g. Gibbs sampling).



Comparison of logistic function with a scaled inverse probit function (i.e. the CDF of the normal distribution), comparing $\sigma(x)$ vs. $\Phi(\sqrt{\frac{\pi}{8}}x)$, which makes the slopes the same at the origin. This shows the heavier tails of the logistic distribution.

There are various possibilities:

- Don't do a proper Bayesian analysis, but simply compute a maximum a posteriori point estimate of the parameters. This is common, for example, in “maximum entropy” classifiers in machine learning.
- Use a more general approximation method such as the Metropolis–Hastings algorithm.
- Draw a Markov chain Monte Carlo sample from the exact posterior by using the Independent Metropolis–Hastings algorithm with heavy-tailed multivariate candidate distribution found by matching the mode and curvature at the mode of the normal approximation to the posterior and then using the Student’s t shape with low degrees of freedom.[22] This is shown to have excellent convergence properties.
- Use a latent variable model and approximate the logistic distribution using a more tractable distribution, e.g. a Student’s t-distribution or a mixture of normal distributions.
- Do probit regression instead of logistic regression. This is actually a special case of the previous situation, using a normal distribution in place of a Student’s t, mixture of normals, etc. This will be less accurate but has the advantage that probit regression

is extremely common, and a ready-made Bayesian implementation may already be available.

- Use the Laplace approximation of the posterior distribution.[23] This approximates the posterior with a Gaussian distribution. This is not a terribly good approximation, but it suffices if all that is desired is an estimate of the posterior mean and variance. In such a case, an approximation scheme such as variational Bayes can be used.[24]

6.7.1 Gibbs sampling with an approximating distribution

As shown above, logistic regression is equivalent to a latent variable model with an error variable distributed according to a standard logistic distribution. The overall distribution of the latent variable Y_i* is also a logistic distribution, with the mean equal to $\boldsymbol\beta \cdot \mathbf{X}_i$ (i.e. the fixed quantity added to the error variable). This model considerably simplifies the application of techniques such as Gibbs sampling. However, sampling the regression coefficients is still difficult, because of the lack of conjugacy between the normal and logistic distributions. Changing the prior distribution over the regression coefficients is of no help, because the logistic distribution is not in the exponential family and thus has no conjugate prior.

One possibility is to use a more general Markov chain Monte Carlo technique, such as the Metropolis–Hastings algorithm, which can sample arbitrary distributions. Another possibility, however, is to replace the logistic distribution with a similar-shaped distribution that is easier to work with using Gibbs sampling. In fact, the logistic and normal distributions have a similar shape, and thus one possibility is simply to have normally distributed errors. Because the normal distribution is conjugate to itself, sampling the regression coefficients becomes easy. In fact, this model is exactly the model used in probit regression.

However, the normal and logistic distributions differ in that the logistic has heavier tails. As a result, it is more robust to inaccuracies in the underlying model (which are inevitable, in that the model is essentially always an approximation) or to errors in the data. Probit regression loses some of this robustness.

Another alternative is to use errors distributed as a Student's t-distribution. The Student's t-distribution has heavy tails, and is easy to sample from because it is the compound distribution of a normal distribution with variance distributed as an inverse gamma distribution. In other words, if a normal distribution is used for the error variable, and another latent variable, following an inverse gamma distribution, is added corresponding to the variance of this error variable, the marginal distribution of the error variable will follow a Student's t distribution. Because of the various conjugacy relationships, all variables in this model are easy to sample from.

The Student's t distribution that best approximates a standard logistic distribution can be determined by matching the moments of the two distributions. The Student's t distribution has three parameters, and since the skewness of both distributions is always 0, the first four moments can all be matched, using the following equations:

$$\begin{aligned}\mu &= 0\\ \frac{\nu}{\nu-2}s^2 &= \frac{\pi^2}{3}\\ \frac{6}{\nu-4} &= \frac{6}{5}\end{aligned}$$

This yields the following values:

$$\begin{aligned}&\mu = 0\\ &s = \sqrt{\frac{7}{9}\frac{\pi^2}{3}}\\ &\nu = 9\end{aligned}$$

The following graphs compare the standard logistic distribution with the Student's t distribution that matches the first four moments using the above-determined values, as well as the normal distribution that matches the first two moments. Note how much closer the Student's t distribution agrees, especially in the tails. Beyond about two standard deviations from the mean, the logistic and normal distributions diverge rapidly, but the logistic and Student's t distributions don't start diverging significantly until more than 5 standard deviations away.

(Another possibility, also amenable to Gibbs sampling, is to approximate the logistic distribution using a mixture density of normal distributions.)

6.8 Extensions

There are large numbers of extensions:

- Multinomial logistic regression (or **multinomial logit**) handles the case of a multi-way categorical dependent variable (with unordered values, also called "classification"). Note that the general case of having dependent variables with more than two values is termed *polytomous regression*.
- Ordered logistic regression (or **ordered logit**) handles ordinal dependent variables (ordered values).
- Mixed logit is an extension of multinomial logit that allows for correlations among the choices of the dependent variable.
- An extension of the logistic model to sets of interdependent variables is the conditional random field.

6.9 Model suitability

A way to measure a model's suitability is to assess the model against a set of data that was not used to create the model.[25] The class of techniques is called cross-validation. This holdout model assessment method is particularly valuable when data are collected in different settings (e.g., at different times or places) or when models are assumed to be generalizable.

To measure the suitability of a binary regression model, one can classify both the actual value and the predicted value of each observation as either 0 or 1.[26] The predicted value of an observation can be set equal to 1 if the estimated probability that the observation equals 1 is above $\frac{1}{2}$, and set equal to 0 if the estimated probability is below $\frac{1}{2}$. Here logistic regression is being used as a binary classification model. There are four possible combined classifications:

1. prediction of 0 when the holdout sample has a 0 (True Negatives, the number of which is TN)
2. prediction of 0 when the holdout sample has a 1 (False Negatives, the number of which is FN)
3. prediction of 1 when the holdout sample has a 0 (False Positives, the number of which is FP)
4. prediction of 1 when the holdout sample has a 1 (True Positives, the number of which is TP)

These classifications are used to calculate accuracy, precision (also called positive predictive value), recall (also called sensitivity), specificity and negative predictive value:

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

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

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

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

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

6.10 See also

- Logistic function
- Discrete choice
- Jarrow–Turnbull model
- Limited dependent variable
- Multinomial logit model
- Ordered logit
- Hosmer–Lemeshow test
- Brier score
- MLPACK - contains a C++ implementation of logistic regression

6.11 References

[1] David A. Freedman (2009). *Statistical Models: Theory and Practice*. Cambridge University Press. p. 128.

[2] Cox, DR (1958). “The regression analysis of binary sequences (with discussion)". *J Roy Stat Soc B* **20**: 215–242.

[3] Walker, SH; Duncan, DB (1967). “Estimation of the probability of an event as a function of several independent variables”. *Biometrika* **54**: 167–178.

[4] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Springer. p. 6.

[5] Boyd, C. R.; Tolson, M. A.; Copes, W. S. (1987). “Evaluating trauma care: The TRISS method. Trauma Score and the Injury Severity Score”. *The Journal of trauma* **27** (4): 370–378. doi:10.1097/00005373-198704000-00005. PMID 3106646.

[6] Kologlu M., Elker D., Altun H., Sayek I. Valdation of MPI and OIA II in two different groups of patients with secondary peritonitis // Hepato-Gastroenterology. – 2001. – Vol. 48, № 37. – P. 147-151.

[7] Biondo S., Ramos E., Deiros M. et al. Prognostic factors for mortality in left colonic peritonitis: a new scoring system // J. Am. Coll. Surg. – 2000. – Vol. 191, № 6. – P. 635-642.

[8] Marshall J.C., Cook D.J., Christou N.V. et al. Multiple Organ Dysfunction Score: A reliable descriptor of a complex clinical outcome // Crit. Care Med. – 1995. – Vol. 23. – P. 1638-1652.

[9] Le Gall J.-R., Lemeshow S., Saulnier F. A new Simplified Acute Physiology Score (SAPS II) based on a European/North American multicenter study // JAMA. – 1993. – Vol. 270. – P. 2957-2963.

[10] Truett, J; Cornfield, J; Kannel, W (1967). “A multivariate analysis of the risk of coronary heart disease in Framingham”. *Journal of chronic diseases* **20** (7): 511–24. PMID 6028270.

[11] Harrell, Frank E. (2001). *Regression Modeling Strategies*. Springer-Verlag. ISBN 0-387-95232-2.

[12] M. Strano; B.M. Colosimo (2006). “Logistic regression analysis for experimental determination of forming limit diagrams”. *International Journal of Machine Tools and Manufacture* **46** (6). doi:10.1016/j.ijmachtools.2005.07.005.

[13] Palei, S. K.; Das, S. K. (2009). "Logistic regression model for prediction of roof fall risks in bord and pillar workings in coal mines: An approach". *Safety Science* **47**: 88. doi:10.1016/j.ssci.2008.01.002.

[14] Hosmer, David W.; Lemeshow, Stanley (2000). *Applied Logistic Regression* (2nd ed.). Wiley. ISBN 0-471-35632-8.

[15] http://www.planta.cn/forum/files_planta/introduction_to_categorical_data_analysis_805.pdf

[16] Menard, Scott W. (2002). *Applied Logistic Regression* (2nd ed.). SAGE. ISBN 978-0-7619-2208-7.

[17] Menard ch 1.3

[18] Peduzzi, P; Concato, J; Kemper, E; Holford, TR; Feinstein, AR (December 1996). "A simulation study of the number of events per variable in logistic regression analysis.". *Journal of Clinical Epidemiology* **49** (12): 1373–9. doi:10.1016/s0895-4356(96)00236-3. PMID 8970487.

[19] Greene, William N. (2003). *Econometric Analysis* (Fifth ed.). Prentice-Hall. ISBN 0-13-066189-9.

[20] Cohen, Jacob; Cohen, Patricia; West, Steven G.; Aiken, Leona S. (2002). *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences* (3rd ed.). Routledge. ISBN 978-0-8058-2223-6.

[21] https://class.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/classification.pdf slide 16

[22] Bolstad, William M. (2010). *Understandeing Computational Bayesian Statistics*. Wiley. ISBN 978-0-470-04609-8.

[23] Bishop, Christopher M. "Chapter 4. Linear Models for Classification". *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC. pp. 217–218. ISBN 978-0387-31073-2.

[24] Bishop, Christopher M. "Chapter 10. Approximate Inference". *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC. pp. 498–505. ISBN 978-0387-31073-2.

[25] Jonathan Mark and Michael A. Goldberg (2001). Multiple Regression Analysis and Mass Assessment: A Review of the Issues. The Appraisal Journal, Jan. pp. 89–109

[26] Myers, J. H.; Forgy, E. W. (1963). "The Development of Numerical Credit Evaluation Systems". *J. Amer. Statist. Assoc.* **58** (303): 799–806. doi:10.1080/01621459.1963.10500889.

6.12 Further reading

- Agresti, Alan. (2002). *Categorical Data Analysis*. New York: Wiley-Interscience. ISBN 0-471-36093-7.
- Amemiya, T. (1985). *Advanced Econometrics*. Harvard University Press. ISBN 0-674-00560-0.
- Balakrishnan, N. (1991). *Handbook of the Logistic Distribution*. Marcel Dekker, Inc. ISBN 978-0-8247-8587-1.
- Greene, William H. (2003). *Econometric Analysis, fifth edition*. Prentice Hall. ISBN 0-13-066189-9.
- Hilbe, Joseph M. (2009). *Logistic Regression Models*. Chapman & Hall/CRC Press. ISBN 978-1-4200-7575-5.
- Howell, David C. (2010). *Statistical Methods for Psychology, 7th ed*. Belmont, CA; Thomson Wadsworth. ISBN 978-0-495-59786-5.
- Peduzzi, P.; J. Concato, E. Kemper, T.R. Holford, A.R. Feinstein (1996). "A simulation study of the number of events per variable in logistic regression analysis". *Journal of Clinical Epidemiology* **49** (12): 1373–1379. doi:10.1016/s0895-4356(96)00236-3. PMID 8970487.

6.13 External links

- Econometrics Lecture (topic: Logit model) on YouTube by Mark Thoma
- Logistic Regression Interpretation
- Logistic Regression tutorial
- Using open source software for building Logistic Regression models
- Logistic regression. Biomedical statistics

Chapter 7

Support vector machine

Not to be confused with Secure Virtual Machine.

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**[1]) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

7.1 Definition

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

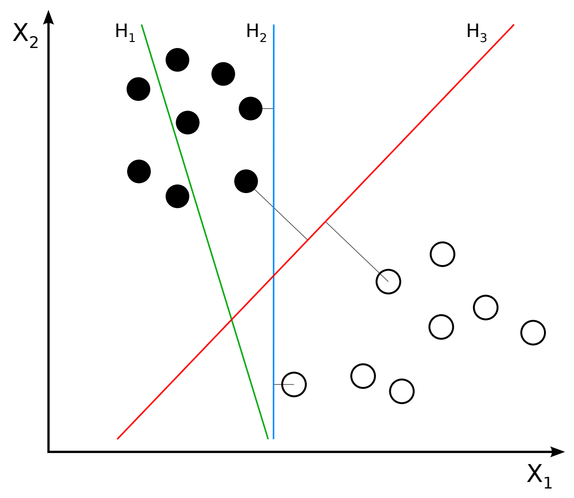
Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem.[2] The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vectors that occur in the data base. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation: $\sum_i \alpha_i k(x_i, x) = \text{constant}.$ Note that if $k(x, y)$ becomes small as y grows further away from x , each term in the sum measures the degree of closeness of the test point x to the corresponding data base point x_i . In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets which are not convex at all in the original space.

7.2 History

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes.[3] The current standard incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995.[1]

7.3 Motivation

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a *new* data point will be in. In the case of support vector machines, a data point is viewed as a p -dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$ -dimensional hyperplane. This is called a linear classifier. There are many hyperplanes



H_1 does not separate the classes. H_2 does, but only with a small margin. H_3 separates them with the maximum margin.

that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the *maximum-margin hyperplane* and the linear classifier it defines is known as a *maximum margin classifier*; or equivalently, the *perceptron of optimal stability.*

7.4 Linear SVM

Given some training data $\mathcal{D}$, a set of n points of the form

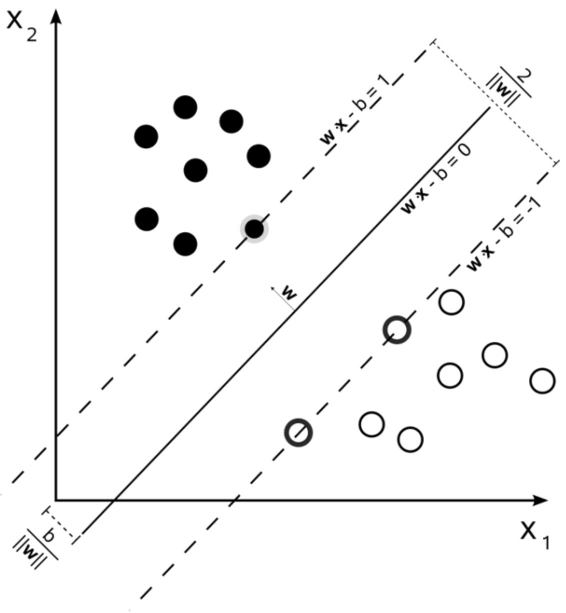
$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1, 1\}\}_{i=1}^{n}$$

where the y_i is either 1 or −1, indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p -dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0,$$

where $\cdot$ denotes the dot product and $\mathbf{w}$ the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called “the margin”. These hyperplanes can be described by the equations



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

By using geometry, we find the distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we want to minimize $\|\mathbf{w}\|$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i$$

or

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i$$

This can be rewritten as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{all for } 1 \leq i \leq n. \qquad (1)$$

We can put this together to get the optimization problem:

Minimize (in $\mathbf{w}, b$)

$$\|\mathbf{w}\|$$

subject to (for any $i = 1, \dots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

7.4.1 Primal form

The optimization problem presented in the preceding section is difficult to solve because it depends on $||\mathbf{w}||$, the norm of $\mathbf{w}$, which involves a square root. Fortunately it is possible to alter the equation by substituting $||\mathbf{w}||$ with $\frac{1}{2}||\mathbf{w}||^2$ (the factor of $\frac{1}{2}$ being used for mathematical convenience) without changing the solution (the minimum of the original and the modified equation have the same $\mathbf{w}$ and b). This is a quadratic programming optimization problem. More clearly:

$$\arg\min_{(\mathbf{w},b)} \frac{1}{2}||\mathbf{w}||^2$$

subject to (for any $i = 1, \dots, n$)

$$y_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1.$$

By introducing Lagrange multipliers $\boldsymbol{\alpha}$, the previous constrained problem can be expressed as

$$\arg\min_{\mathbf{w},b}\max_{\boldsymbol{\alpha}\geq 0}\left\{\frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x_i} - b) - 1]\right\}$$

that is we look for a saddle point. In doing so all the points which can be separated as $y_i(\mathbf{w}\cdot\mathbf{x_i} - b) - 1 > 0$ do not matter since we must set the corresponding α_i to zero.

This problem can now be solved by standard quadratic programming techniques and programs. The “stationary” Karush–Kuhn–Tucker condition implies that the solution can be expressed as a linear combination of the training vectors

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x_i}.$$

Only a few α_i will be greater than zero. The corresponding $\mathbf{x_i}$ are exactly the *support vectors*, which lie on the margin and satisfy $y_i(\mathbf{w}\cdot\mathbf{x_i} - b) = 1$. From this one can derive that the support vectors also satisfy

$$\mathbf{w}\cdot\mathbf{x_i} - b = \frac{1}{y_i} = y_i \iff b = \mathbf{w}\cdot\mathbf{x_i} - y_i$$

which allows one to define the offset b . The b depends on y_i and x_i , so it will vary for each data point in the sample. In practice, it is more robust to average over all N_{SV} support vectors, since the average over the sample is an unbiased estimator of the population mean:

$$b = \frac{1}{N_{SV}}\sum_{i=1}^{N_{SV}}(\mathbf{w}\cdot\mathbf{x_i} - y_i)$$

7.4.2 Dual form

Writing the classification rule in its unconstrained dual form reveals that the *maximum-margin hyperplane* and therefore the classification task is only a function of the *support vectors*, the subset of the training data that lie on the margin.

Using the fact that $||\mathbf{w}||^2 = \mathbf{w}^T\cdot\mathbf{w}$ and substituting $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x_i}$, one can show that the dual of the SVM reduces to the following optimization problem:

Maximize (in α_i)

$$\tilde{L}(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j$$

subject to (for any $i = 1, \dots, n$)

$$\alpha_i \geq 0,$$

and to the constraint from the minimization in b

$$\sum_{i=1}^{n}\alpha_i y_i = 0.$$

Here the kernel is defined by $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i\cdot\mathbf{x}_j$.

W can be computed thanks to the α terms:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i.$$

7.4.3 Biased and unbiased hyperplanes

For simplicity reasons, sometimes it is required that the hyperplane pass through the origin of the coordinate system. Such hyperplanes are called *unbiased*, whereas general hyperplanes not necessarily passing through the origin are called *biased*. An unbiased hyperplane can be enforced by setting $b = 0$ in the primal optimization problem. The corresponding dual is identical to the dual given above without the equality constraint

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

7.5 Soft margin

In 1995, Corinna Cortes and Vladimir N. Vapnik suggested a modified maximum margin idea that allows for mislabeled examples.[1] If there exists no hyperplane that can split the "yes" and "no" examples, the *Soft Margin* method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces non-negative slack variables, ξ_i , which measure the degree of misclassification of the data x_i

$$y_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i \quad 1\leq i\leq n. \qquad (2)$$

The objective function is then increased by a function which penalizes non-zero ξ_i , and the optimization becomes a trade off between a large margin and a small error penalty. If the penalty function is linear, the optimization problem becomes:

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

subject to (for any $i=1,\dots n$)

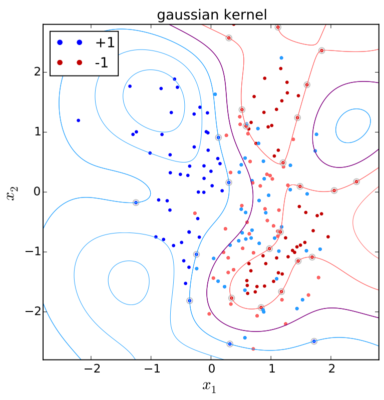
$$y_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i, \quad \xi_i\geq 0$$

Using the hinge function notation like that in MARS, this optimization problem can be rewritten as $\sum_i[1-y_i(w\cdot x_i+b)]_+ +\lambda\|w\|^2$, wherein let $[1-y_i(w\cdot x_i+b)]_+=[\xi_i]_+=\xi_i, \quad \lambda=1/2C$.

This constraint in (2) along with the objective of minimizing $\|\mathbf{w}\|$ can be solved using Lagrange multipliers as done above. One then has to solve the following problem:

$$\arg\min_{\mathbf{w},\xi,b}\max_{\boldsymbol{\alpha},\boldsymbol{\beta}}\left\{\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\xi_i-\sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x_i}-b)-\right.$$

with $\alpha_i,\beta_i\geq 0$.



An example for a result of soft-margin SVM

7.5.1 Dual form

Maximize (in α_i)

$$\tilde{L}(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)$$

subject to (for any $i=1,\dots,n$)

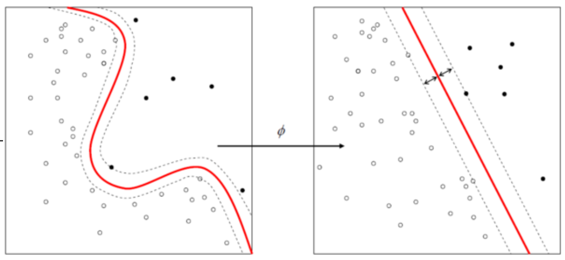
$$0\leq\alpha_i\leq C,$$

and

$$\sum_{i=1}^{n}\alpha_i y_i=0.$$

The key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant *C* appearing only as an additional constraint on the Lagrange multipliers. For the above formulation and its huge impact in practice, Cortes and Vapnik received the 2008 ACM Paris Kanellakis Award.[4] Nonlinear penalty functions have been used, particularly to reduce the effect of outliers on the classifier, but unless care is taken the problem becomes non-convex, and thus it is considerably more difficult to find a global solution.

7.6 Nonlinear classification



Kernel machine

The original optimal hyperplane algorithm proposed by Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick (originally proposed by Aizerman et al.[5]) to maximum-margin hyperplanes.[6] The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimensions. Maximum margin classifiers are well regularized, and previously it was widely believed that the infinite dimensions do not spoil the results. However, it has been shown that higher dimensions do increase the generalization error, although the amount is bounded.[7]

Some common kernels include:

- Polynomial (homogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j})^d$
- Polynomial (inhomogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$
- Gaussian radial basis function: $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$, for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/2\sigma^2$
- Hyperbolic tangent: $k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$, for some (not every) $\kappa > 0$ and $c < 0$

The kernel is related to the transform $\varphi(\mathbf{x_i})$ by the equation $k(\mathbf{x_i}, \mathbf{x_j}) = \varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x_j})$. The value **w** is also in the transformed space, with $\mathbf{w} = \sum_i \alpha_i y_i \varphi(\mathbf{x}_i)$. Dot products with **w** for classification can again be computed by the kernel trick, i.e. $\mathbf{w} \cdot \varphi(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$. However, there does not in general exist a value **w'** such that $\mathbf{w} \cdot \varphi(\mathbf{x}) = k(\mathbf{w}', \mathbf{x})$.

7.7 Properties

SVMs belong to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron. They can also be considered a special case of Tikhonov regularization. A special property is that they simultaneously minimize the empirical *classification error* and maximize the *geometric margin*; hence they are also known as **maximum margin classifiers**.

A comparison of the SVM to other classifiers has been made by Meyer, Leisch and Hornik.[8]

7.7.1 Parameter selection

The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C. A common choice is a Gaussian kernel, which has a single parameter γ. The best combination of C and γ is often selected by a grid search with exponentially growing sequences of C and γ, for example, $C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}$; $\gamma \in \{2^{-15}, 2^{-13}, \dots, 2^{1}, 2^{3}\}$. Typically, each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are picked. Alternatively, recent work in Bayesian optimization can be used to select C and γ, often requiring the evaluation of far fewer parameter combinations than grid search. The final model, which is used for testing and for classifying new data, is then trained on the whole training set using the selected parameters.[9]

7.7.2 Issues

Potential drawbacks of the SVM are the following three aspects:

- Uncalibrated class membership probabilities
- The SVM is only directly applicable for two-class tasks. Therefore, algorithms that reduce the multi-class task to several binary problems have to be applied; see the multi-class SVM section.
- Parameters of a solved model are difficult to interpret.

7.8 Extensions

7.8.1 Multiclass SVM

Multiclass SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements.

The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems.[10] Common methods for such reduction include:[10] [11]

- Building binary classifiers which distinguish between (i) one of the labels and the rest (*one-versus-all*) or (ii) between every pair of classes (*one-versus-one*). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.
- Directed acyclic graph SVM (DAGSVM)[12]
- Error-correcting output codes[13]

Crammer and Singer proposed a multiclass SVM method which casts the multiclass classification problem into a single optimization problem, rather than decomposing it into multiple binary classification problems.[14] See also Lee, Lin and Wahba.[15][16]

7.8.2 Transductive support vector machines

Transductive support vector machines extend SVMs in that they could also treat partially labeled data in semi-supervised learning by following the principles of transduction. Here, in addition to the training set $\mathcal{D}$, the learner is also given a set

$$\mathcal{D}^\star = \{\mathbf{x}_i^\star | \mathbf{x}_i^\star \in \mathbb{R}^p\}_{i=1}^k$$

of test examples to be classified. Formally, a transductive support vector machine is defined by the following primal optimization problem:[17]

Minimize (in $\mathbf{w}, b, \mathbf{y}^\star$)

$$\frac{1}{2}\|\mathbf{w}\|^2$$

subject to (for any $i = 1, \dots, n$ and any $j = 1, \dots, k$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1,$$

$$y_j^\star(\mathbf{w} \cdot \mathbf{x_j^\star} - b) \geq 1,$$

and

$$y_j^\star \in \{-1, 1\}.$$

Transductive support vector machines were introduced by Vladimir N. Vapnik in 1998.

7.8.3 Structured SVM

SVMs have been generalized to structured SVMs, where the label space is structured and of possibly infinite size.

7.8.4 Regression

A version of SVM for regression was proposed in 1996 by Vladimir N. Vapnik, Harris Drucker, Christopher J. C. Burges, Linda Kaufman and Alexander J. Smola.[18] This method is called support vector regression (SVR). The model produced by support vector classification (as described above) depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. Another SVM version known as least squares support vector machine (LS-SVM) has been proposed by Suykens and Vandewalle.[19]

Training the original SVR means solving[20]

$$\frac{1}{2}\|w\|_2$$

$$\begin{cases} y_i - \langle w, x_i \rangle - b \leq \epsilon \\ \langle w, x_i \rangle + b - y_i \leq \epsilon \end{cases}$$

where x_i is a training sample with target value y_i . The inner product plus intercept $\langle w, x_i \rangle + b$ is the prediction for that sample, and ϵ is a free parameter that serves as a threshold: all predictions have to be within an ϵ range of the true predictions. Slack variables are usually added into the above to allow for errors and to allow approximation in the case the above problem is infeasible.

7.9 Interpreting SVM models

The SVM algorithm has been widely applied in the biological and other sciences. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models.[21][22] Support vector machine weights have also been used to interpret SVM models in the past.[23] Posthoc interpretation of support vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

7.10 Implementation

The parameters of the maximum-margin hyperplane are derived by solving the optimization. There exist several specialized algorithms for quickly solving the QP problem that arises from SVMs, mostly relying on heuristics for breaking the problem down into smaller, more-manageable chunks. A common method is Platt's sequential minimal optimization (SMO) algorithm, which breaks the problem down into 2-dimensional sub-problems that may be solved analytically, eliminating the need for a numerical optimization algorithm.[24]

Another approach is to use an interior point method that uses Newton-like iterations to find a solution of the Karush–Kuhn–Tucker conditions of the primal and dual problems.[25] Instead of solving a sequence of broken down problems, this approach directly solves the problem as a whole. To avoid solving a linear system involving the large kernel matrix, a low rank approximation to the matrix is often used in the kernel trick.

The special case of linear support vector machines can be solved more efficiently by the same kind of algorithms used to optimize its close cousin, logistic regression; this class of algorithms includes sub-gradient descent (e.g., PEGASOS[26]) and coordinate descent (e.g., LIBLINEAR[27]). The general kernel SVMs can also be solved more efficiently using sub-gradient descent (e.g. P-packSVM[28]), especially when parallelization is allowed.

Kernel SVMs are available in many machine learning toolkits, including LIBSVM, MATLAB, SVM-light, kernlab, scikit-learn, Shogun, Weka, Shark, JKernelMachines and others.

7.11 Applications

SVMs can be used to solve various real world problems:

- SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.
- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.
- SVMs are also useful in medical science to classify proteins with up to 90% of the compounds classified correctly.
- Hand-written characters can be recognized using SVM

7.12 See also

- In situ adaptive tabulation
- Kernel machines
- Fisher kernel
- Platt scaling
- Polynomial kernel
- Predictive analytics
- Regularization perspectives on support vector machines
- Relevance vector machine, a probabilistic sparse kernel model identical in functional form to SVM
- Sequential minimal optimization
- Winnow (algorithm)

7.13 References

[1] Cortes, C.; Vapnik, V. (1995). “Support-vector networks”. *Machine Learning* **20** (3): 273. doi:10.1007/BF00994018.

[2] • Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, B. P. (2007). “Section 16.5. Support Vector Machines”. *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

[3] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992). “A training algorithm for optimal margin classifiers”. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. p. 144. doi:10.1145/130385.130401. ISBN 089791497X.

[4] ACM Website, Press release of March 17th 2009. http://www.acm.org/press-room/news-releases/awards-08-groupa

[5] Aizerman, Mark A.; Braverman, Emmanuel M.; and Rozonoer, Lev I. (1964). “Theoretical foundations of the potential function method in pattern recognition learning”. *Automation and Remote Control* **25**: 821–837.

[6] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992). “A training algorithm for optimal margin classifiers”. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. p. 144. doi:10.1145/130385.130401. ISBN 089791497X.

[7] Jin, Chi; Wang, Liwei (2012). *Dimensionality dependent PAC-Bayes margin bound*. Advances in Neural Information Processing Systems.

[8] Meyer, D.; Leisch, F.; Hornik, K. (2003). “The support vector machine under test”. *Neurocomputing* **55**: 169. doi:10.1016/S0925-2312(03)00431-4.

[9] Hsu, Chih-Wei; Chang, Chih-Chung; and Lin, Chih-Jen (2003). *A Practical Guide to Support Vector Classification* (PDF) (Technical report). Department of Computer Science and Information Engineering, National Taiwan University.

[10] Duan, K. B.; Keerthi, S. S. (2005). “Which Is the Best Multiclass SVM Method? An Empirical Study”. *Multiple Classifier Systems* (PDF). LNCS **3541**. pp. 278–285. doi:10.1007/11494683_28. ISBN 978-3-540-26306-7.

[11] Hsu, Chih-Wei; and Lin, Chih-Jen (2002). “A Comparison of Methods for Multiclass Support Vector Machines”. *IEEE Transactions on Neural Networks*.

[12] Platt, John; Cristianini, N.; and Shawe-Taylor, J. (2000). “Large margin DAGs for multiclass classification”. In Solla, Sara A.; Leen, Todd K.; and Müller, Klaus-Robert; eds. *Advances in Neural Information Processing Systems* (PDF). MIT Press. pp. 547–553.

[13] Dietterich, Thomas G.; and Bakiri, Ghulum; Bakiri (1995). “Solving Multiclass Learning Problems via Error-Correcting Output Codes” (PDF). *Journal of Artificial Intelligence Research, Vol. 2* **2**: 263–286. arXiv:cs/9501101. Bibcode:1995cs........1101D.

[14] Crammer, Koby; and Singer, Yoram (2001). “On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines” (PDF). *J. of Machine Learning Research* **2**: 265–292.

[15] Lee, Y.; Lin, Y.; and Wahba, G. (2001). "Multicategory Support Vector Machines" (PDF). *Computing Science and Statistics 33*.

[16] Lee, Y.; Lin, Y.; Wahba, G. (2004). "Multicategory Support Vector Machines". *Journal of the American Statistical Association* **99** (465): 67. doi:10.1198/016214504000000098.

[17] Joachims, Thorsten; "Transductive Inference for Text Classification using Support Vector Machines", Proceedings of the 1999 International Conference on Machine Learning (ICML 1999), pp. 200-209.

[18] Drucker, Harris; Burges, Christopher J. C.; Kaufman, Linda; Smola, Alexander J.; and Vapnik, Vladimir N. (1997); "Support Vector Regression Machines", in *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155–161, MIT Press.

[19] Suykens, Johan A. K.; Vandewalle, Joos P. L.; *Least squares support vector machine classifiers*, Neural Processing Letters, vol. 9, no. 3, Jun. 1999, pp. 293–300.

[20] Smola, Alex J.; Schölkopf, Bernhard (2004). "A tutorial on support vector regression" (PDF). *Statistics and Computing* **14** (3): 199–222.

[21] Bilwaj Gaonkar, Christos Davatzikos Analytic estimation of statistical significance maps for support vector machine based multi-variate image analysis and classification

[22] R. Cuingnet, C. Rosso, M. Chupin, S. Lehéricy, D. Dormont, H. Benali, Y. Samson and O. Colliot, Spatial regularization of SVM for the detection of diffusion alterations associated with stroke outcome, Medical Image Analysis, 2011, 15 (5): 729-737

[23] Statnikov, A., Hardin, D., & Aliferis, C. (2006). Using SVM weight-based methods to identify causally relevant and non-causally relevant variables. sign, 1, 4.

[24] John C. Platt (1999). *Using Analytic QP and Sparseness to Speed Training of Support Vector Machines* (PDF). NIPS.

[25] Ferris, M. C.; Munson, T. S. (2002). "Interior-Point Methods for Massive Support Vector Machines". *SIAM Journal on Optimization* **13** (3): 783. doi:10.1137/S1052623400374379.

[26] Shai Shalev-Shwartz; Yoram Singer; Nathan Srebro (2007). *Pegasos: Primal Estimated sub-GrAdient SOlver for SVM* (PDF). ICML.

[27] R.-E. Fan; K.-W. Chang; C.-J. Hsieh; X.-R. Wang; C.-J. Lin (2008). "LIBLINEAR: A library for large linear classification". *Journal of Machine Learning Research* **9**: 1871–1874.

[28] Zeyuan Allen Zhu et al. (2009). *P-packSVM: Parallel Primal grAdient desCent Kernel SVM* (PDF). ICDM.

7.14 External links

- www.support-vector.net The key book about the method, "An Introduction to Support Vector Machines" with online software
- Burges, Christopher J. C.; A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2:121–167, 1998
- www.kernel-machines.org *(general information and collection of research papers)*
- www.support-vector-machines.org *(Literature, Review, Software, Links related to Support Vector Machines — Academic Site)*
- videolectures.net *(SVM-related video lectures)*
- Karatzoglou, Alexandros et al.; Support Vector Machines in R, Journal of Statistical Software April 2006, Volume 15, Issue 9.
- libsvm LIBSVM is a popular library of SVM learners
- liblinear liblinear is a library for large linear classification including some SVMs
- Shark Shark is a C++ machine learning library implementing various types of SVMs
- dlib dlib is a C++ library for working with kernel methods and SVMs
- SVM light is a collection of software tools for learning and classification using SVM.
- SVMJS live demo is a GUI demo for Javascript implementation of SVMs
- Gesture Recognition Toolkit contains an easy to use wrapper for libsvm

7.15 Bibliography

- Theodoridis, Sergios; and Koutroumbas, Konstantinos; "Pattern Recognition", 4th Edition, Academic Press, 2009, ISBN 978-1-59749-272-0
- Cristianini, Nello; and Shawe-Taylor, John; *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000. ISBN 0-521-78019-5 *(SVM Book)*
- Huang, Te-Ming; Kecman, Vojislav; and Kopriva, Ivica (2006); *Kernel Based Algorithms for Mining Huge Data Sets*, in *Supervised, Semi-supervised, and Unsupervised Learning*, Springer-Verlag, Berlin, Heidelberg, 260 pp. 96 illus., Hardcover, ISBN 3-540-31681-7

- Kecman, Vojislav; *Learning and Soft Computing — Support Vector Machines, Neural Networks, Fuzzy Logic Systems*, The MIT Press, Cambridge, MA, 2001.

- Schölkopf, Bernhard; and Smola, Alexander J.; *Learning with Kernels*, MIT Press, Cambridge, MA, 2002. ISBN 0-262-19475-9

- Schölkopf, Bernhard; Burges, Christopher J. C.; and Smola, Alexander J. (editors); *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge, MA, 1999. ISBN 0-262-19416-3.

- Shawe-Taylor, John; and Cristianini, Nello; *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004. ISBN 0-521-81397-2 *(Kernel Methods Book)*

- Steinwart, Ingo; and Christmann, Andreas; *Support Vector Machines*, Springer-Verlag, New York, 2008. ISBN 978-0-387-77241-7 *(SVM Book)*

- Tan, Peter Jing; and Dowe, David L. (2004); *MML Inference of Oblique Decision Trees*, Lecture Notes in Artificial Intelligence (LNAI) 3339, Springer-Verlag, pp1082-1088. (This paper uses minimum message length (MML) and actually incorporates probabilistic support vector machines in the leaves of decision trees.)

- Vapnik, Vladimir N.; *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995. ISBN 0-387-98780-0

- Vapnik, Vladimir N.; and Kotz, Samuel; *Estimation of Dependences Based on Empirical Data*, Springer, 2006. ISBN 0-387-30865-2, 510 pages [this is a reprint of Vapnik's early book describing philosophy behind SVM approach. The 2006 Appendix describes recent development].

- Fradkin, Dmitriy; and Muchnik, Ilya; *Support Vector Machines for Classification* in Abello, J.; and Carmode, G. (Eds); *Discrete Methods in Epidemiology*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 70, pp. 13–20, 2006. . Succinctly describes theoretical ideas behind SVM.

- Bennett, Kristin P.; and Campbell, Colin; *Support Vector Machines: Hype or Hallelujah?*, SIGKDD Explorations, 2, 2, 2000, 1–13. . Excellent introduction to SVMs with helpful figures.

- Ivanciuc, Ovidiu; *Applications of Support Vector Machines in Chemistry*, in *Reviews in Computational Chemistry*, Volume 23, 2007, pp. 291–400. Reprint available:

- Catanzaro, Bryan; Sundaram, Narayanan; and Keutzer, Kurt; *Fast Support Vector Machine Training and Classification on Graphics Processors*, in *International Conference on Machine Learning*, 2008

- Campbell, Colin; and Ying, Yiming; *Learning with Support Vector Machines*, 2011, Morgan and Claypool. ISBN 978-1-60845-616-1.

Chapter 8

Naive Bayes classifier

In machine learning, **naive Bayes classifiers** are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s,[1]:488 and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate preprocessing, it is competitive in this domain with more advanced methods including support vector machines.[2] It also finds application in automatic medical diagnosis.[3]

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression,[1]:718 which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, Naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**.[4] All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method;[4] Russell and Norvig note that "[naive Bayes] is sometimes called a **Bayesian classifier**, a somewhat careless usage that has prompted true Bayesians to call it the **idiot Bayes** model."[1]:482

8.1 Introduction

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.[5] Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.[6]

An advantage of naive Bayes is that it only requires a small amount of training data to estimate the parameters necessary for classification.

8.2 Probabilistic model

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (dependent variables), it assigns to this instance probabilities

$$p(C_k | x_1, \dots, x_n)$$

for each of k possible outcomes or *classes*.[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(C_k)\ p(x_1, \dots, x_n|C_k) \\ &= p(C_k)\ p(x_1|C_k)\ p(x_2, \dots, x_n|C_k, x_1) \\ &= p(C_k)\ p(x_1|C_k)\ p(x_2|C_k, x_1)\ p(x_3, \dots, x_n|C_k, x_1, x_2) \\ &= p(C_k)\ p(x_1|C_k)\ p(x_2|C_k, x_1)\ \dots p(x_n|C_k, x_1, x_2, x_3, \dots, x_{n-1}) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$, given the category C. This means that

$$p(x_i|C_k, x_j) = p(x_i|C_k)$$

$$p(x_i|C_k, x_j, x_k) = p(x_i|C_k)$$

$$p(x_i|C_k, x_j, x_k, x_l) = p(x_i|C_k)$$

and so on, for $i \neq j, k, l$. Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k|x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k)\ p(x_1|C_k)\ p(x_2|C_k)\ p(x_3|C_k)\ \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k)\,. \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k|x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

where the evidence $Z = p(\mathbf{x})$ is a scaling factor dependent only on $x_1, \dots, x_n$, that is, a constant if the values of the feature variables are known.

8.2.1 Constructing a classifier from the probability model

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

8.3 Parameter estimation and event models

A class' prior may be calculated by assuming equiprobable classes (i.e., priors = 1 / (number of classes)), or by calculating an estimate for the class probability from the training set (i.e., (prior for a given class) = (number of samples in the class) / (total number of samples)). To estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set.[8]

The assumptions on distributions of features are called the *event model* of the Naive Bayes classifier. For discrete features like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular. These assumptions lead to two distinct models, which are often confused.[9][10]

8.3.1 Gaussian naive Bayes

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contain a continuous attribute, x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_c be the mean of the values in x associated with class c, and let σ_c^2 be the variance of the values in x associated with class c. Then, the probability *distribution* of some value given a class, $p(x = v|c)$, can be computed by plugging v into the equation for a Normal distribution parameterized by μ_c and σ_c^2 . That is,

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}}\, e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features; some literature in fact suggests that this is necessary to apply naive Bayes, but it is not, and the discretization may throw away discriminative information.[4]

8.3.2 Multinomial naive Bayes

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1, \dots, p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector $\mathbf{x} = (x_1, \dots, x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram $\mathbf{x}$ is given by

$$p(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:[2]

$$\begin{aligned}\log p(C_k|\mathbf{x}) &\propto \log \left(p(C_k) \prod_{i=1}^{n} {p_{ki}}^{x_i} \right) \\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki} \\ &= b + \mathbf{w}_k^\top \mathbf{x}\end{aligned}$$

where $b = \log p(C_k)$ and $w_{ki} = \log p_{ki}$.

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case.

Rennie *et al.* discuss problems with the multinomial assumption in the context of document classification and possible ways to alleviate those problems, including the use of tf–idf weights instead of raw term frequencies and document length normalization, to produce a naive Bayes classifier that is competitive with support vector machines.[2]

8.3.3 Bernoulli naive Bayes

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks,[9] where binary term occurrence features are used rather than term frequencies. If x_i is a boolean expressing the occurrence or absence of the i'th term from the vocabulary, then the likelihood of a document given a class C_k is given by[9]

$$p(\mathbf{x}|C_k) = \prod_{i=1}^{n} p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

where p_{ki} is the probability of class C_k generating the term w_i . This event model is especially popular for classifying short texts. It has the benefit of explicitly modelling the absence of terms. Note that a naive Bayes classifier with a Bernoulli event model is not the same as a multinomial NB classifier with frequency counts truncated to one.

8.3.4 Semi-supervised parameter estimation

Given a way to train a naive Bayes classifier from labeled data, it's possible to construct a semi-supervised training algorithm that can learn from a combination of labeled and unlabeled data by running the supervised learning algorithm in a loop:[11]

```
Given a collection $D = L \uplus U$ of labeled samples L and unlabeled samples U, start by training a naive Bayes classifier on L.
Until convergence, do:
    Predict class probabilities $P(C|x)$ for all examples x in $D$ .
    Re-train the model based on the *probabilities* (not the labels) predicted in the previous step.
```

Convergence is determined based on improvement to the model likelihood $P(D|\theta)$, where θ denotes the parameters of the naive Bayes model.

This training algorithm is an instance of the more general expectation–maximization algorithm (EM): the prediction step inside the loop is the *E*-step of EM, while the re-training of naive Bayes is the *M*-step. The algorithm is formally justified by the assumption that the data are generated by a mixture model, and the components of this mixture model are exactly the classes of the classification problem.[11]

8.4 Discussion

Despite the fact that the far-reaching independence assumptions are often inaccurate, the naive Bayes classifier has several properties that make it surprisingly useful in practice. In particular, the decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This helps alleviate problems stemming from the curse of dimensionality, such as the need for data sets that scale exponentially with the number of features. While naive Bayes often fails to produce a good estimate for the correct class probabilities,[12] this may not be a requirement for many applications. For example, the naive Bayes classifier will make the correct MAP decision rule classification so long as the correct class is more probable than any other class. This is true regardless of whether the probability estimate is slightly, or even grossly inaccurate. In this manner, the overall classifier can be robust enough to ignore serious deficiencies in its underlying naive probability model.[3] Other reasons for the observed success of the naive Bayes classifier are discussed in the literature cited below.

8.4.1 Relation to logistic regression

In the case of discrete inputs (indicator or frequency features for discrete events), naive Bayes classifiers form a *generative-discriminative* pair with (multinomial) logistic regression classifiers: each naive Bayes classifier can be considered a way of fitting a probability model that optimizes the joint likelihood $p(C, \mathbf{x})$, while logistic regression fits the same probability model to optimize the conditional $p(C|\mathbf{x})$.[13]

The link between the two can be seen by observing that the decision function for naive Bayes (in the binary case) can be rewritten as "predict class C_1 if the odds of $p(C_1|\mathbf{x})$ exceed those of $p(C_2|\mathbf{x})$ ". Expressing this in log-space gives:

$$\log\frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} = \log p(C_1|\mathbf{x}) - \log p(C_2|\mathbf{x}) > 0$$

The left-hand side of this equation is the log-odds, or *logit*, the quantity predicted by the linear model that underlies logistic regression. Since naive Bayes is also a linear model for the two "discrete" event models, it can be reparametrised as a linear function $b + \mathbf{w}^\top x > 0$. Obtaining the probabilities is then a matter of applying the logistic function to $b + \mathbf{w}^\top x$, or in the multiclass case, the softmax function.

Discriminative classifiers have lower asymptotic error than generative ones; however, research by Ng and Jordan has shown that in some practical cases naive Bayes can outperform logistic regression because it reaches its asymptotic error faster.[13]

8.5 Examples

8.5.1 Gender classification

Problem: classify whether a given person is a male or a female based on the measured features. The features include height, weight, and foot size.

Training

Example training set below.

The classifier created from the training set using a Gaussian distribution assumption would be (given variances are *unbiased* sample variances):

Let's say we have equiprobable classes so P(male)= P(female) = 0.5. This prior probability distribution might be based on our knowledge of frequencies in the larger population, or on frequency in the training set.

Testing

Below is a sample to be classified as a male or female.

We wish to determine which posterior is greater, male or female. For the classification as male the posterior is given by

$$posterior(male) = \frac{P(male)\,p(height|male)\,p(weight|male)\,p(foots}{evidence}$$

For the classification as female the posterior is given by

$$posterior(female) = \frac{P(female)\,p(height|female)\,p(weight|femal}{evidence}$$

The evidence (also termed normalizing constant) may be calculated:

$$evidence = P(male)\,p(height|male)\,p(weight|male)\,p(footsize|mal$$

$$+P(female)\,p(height|female)\,p(weight|female)\,p(footsize|femal$$

However, given the sample the evidence is a constant and thus scales both posteriors equally. It therefore does not affect classification and can be ignored. We now determine the probability distribution for the sex of the sample.

$$P(male) = 0.5$$

$$p(\text{height}|\text{male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6-\mu)^2}{2\sigma^2}\right) \approx 1.5789$$

where $\mu = 5.855$ and $\sigma^2 = 3.5033 \cdot 10^{-2}$ are the parameters of normal distribution which have been previously determined from the training set. Note that a value greater than 1 is OK here – it is a probability density rather than a probability, because height is a continuous variable.

$$p(\text{weight}|\text{male}) = 5.9881 \cdot 10^{-6}$$

$$p(\text{foot size}|\text{male}) = 1.3112 \cdot 10^{-3}$$

$$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$$

$$P(\text{female}) = 0.5$$

$$p(\text{height}|\text{female}) = 2.2346 \cdot 10^{-1}$$

$$p(\text{weight}|\text{female}) = 1.6789 \cdot 10^{-2}$$

$$p(\text{foot size}|\text{female}) = 2.8669 \cdot 10^{-1}$$

$$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$$

Since posterior numerator is greater in the female case, we predict the sample is female.

8.5.2 Document classification

Here is a worked example of naive Bayesian classification to the document classification problem. Consider the problem of classifying documents by their content, for example into spam and non-spam e-mails. Imagine that documents are drawn from a number of classes of documents which can be modelled as sets of words where the (independent) probability that the i-th word of a given document occurs in a document from class *C* can be written as

$$p(w_i|C)$$

(For this treatment, we simplify things further by assuming that words are randomly distributed in the document - that is, words are not dependent on the length of the document, position within the document with relation to other words, or other document-context.)

Then the probability that a given document *D* contains all of the words w_i , given a class *C*, is

$$p(D|C) = \prod_i p(w_i|C)$$

The question that we desire to answer is: "what is the probability that a given document *D* belongs to a given class *C*?" In other words, what is $p(C|D)$?

Now by definition

$$p(D|C) = \frac{p(D \cap C)}{p(C)}$$

and

$$p(C|D) = \frac{p(D \cap C)}{p(D)}$$

Bayes' theorem manipulates these into a statement of probability in terms of likelihood.

$$p(C|D) = \frac{p(C)}{p(D)} p(D|C)$$

Assume for the moment that there are only two mutually exclusive classes, *S* and ¬*S* (e.g. spam and not spam), such that every element (email) is in either one or the other;

$$p(D|S) = \prod_i p(w_i|S)$$

and

$$p(D|\neg S) = \prod_i p(w_i|\neg S)$$

Using the Bayesian result above, we can write:

$$p(S|D) = \frac{p(S)}{p(D)} \prod_i p(w_i|S)$$

$$p(\neg S|D) = \frac{p(\neg S)}{p(D)} \prod_i p(w_i|\neg S)$$

Dividing one by the other gives:

$$\frac{p(S|D)}{p(\neg S|D)} = \frac{p(S) \prod_i p(w_i|S)}{p(\neg S) \prod_i p(w_i|\neg S)}$$

Which can be re-factored as:

$$\frac{p(S|D)}{p(\neg S|D)} = \frac{p(S)}{p(\neg S)} \prod_i \frac{p(w_i|S)}{p(w_i|\neg S)}$$

Thus, the probability ratio p(*S* | *D*) / p(¬*S* | *D*) can be expressed in terms of a series of likelihood ratios. The actual probability p(*S* | *D*) can be easily computed from log (p(S | D) / p(¬S | D)) based on the observation that p(S | D) + p(¬S | D) = 1.

Taking the logarithm of all these ratios, we have:

$$\ln \frac{p(S|D)}{p(\neg S|D)} = \ln \frac{p(S)}{p(\neg S)} + \sum_i \ln \frac{p(w_i|S)}{p(w_i|\neg S)}$$

(This technique of "log-likelihood ratios" is a common technique in statistics. In the case of two mutually exclusive alternatives (such as this example), the conversion of a log-likelihood ratio to a probability takes the form of a sigmoid curve: see logit for details.)

Finally, the document can be classified as follows. It is spam if $p(S|D) > p(\neg S|D)$ (i.e., $\ln \frac{p(S|D)}{p(\neg S|D)} > 0$), otherwise it is not spam.

8.6 See also

- AODE
- Bayesian spam filtering
- Bayesian network
- Random naive Bayes
- Linear classifier
- Logistic regression
- Perceptron
- Take-the-best heuristic

8.7 References

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

[2] Rennie, J.; Shih, L.; Teevan, J.; Karger, D. (2003). *Tackling the poor assumptions of Naive Bayes classifiers* (PDF). ICML.

[3] Rish, Irina (2001). *An empirical study of the naive Bayes classifier* (PDF). IJCAI Workshop on Empirical Methods in AI.

[4] Hand, D. J.; Yu, K. (2001). “Idiot’s Bayes — not so stupid after all?". *International Statistical Review* **69** (3): 385–399. doi:10.2307/1403452. ISSN 0306-7734.

[5] Zhang, Harry. *The Optimality of Naive Bayes* (PDF). FLAIRS2004 conference.

[6] Caruana, R.; Niculescu-Mizil, A. (2006). *An empirical comparison of supervised learning algorithms*. Proc. 23rd International Conference on Machine Learning. CiteSeerX: 10.1.1.122.5901.

[7] Narasimha Murty, M.; Susheela Devi, V. (2011). *Pattern Recognition: An Algorithmic Approach*. ISBN 0857294946.

[8] John, George H.; Langley, Pat (1995). *Estimating Continuous Distributions in Bayesian Classifiers*. Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann. pp. 338–345.

[9] McCallum, Andrew; Nigam, Kamal (1998). *A comparison of event models for Naive Bayes text classification* (PDF). AAAI-98 workshop on learning for text categorization **752**.

[10] Metsis, Vangelis; Androutsopoulos, Ion; Paliouras, Georgios (2006). *Spam filtering with Naive Bayes—which Naive Bayes?*. Third conference on email and anti-spam (CEAS) **17**.

[11] Nigam, Kamal; McCallum, Andrew; Thrun, Sebastian; Mitchell, Tom (2000). “Learning to classify text from labeled and unlabeled documents using EM” (PDF). *Machine Learning*.

[12] Niculescu-Mizil, Alexandru; Caruana, Rich (2005). *Predicting good probabilities with supervised learning* (PDF). ICML. doi:10.1145/1102351.1102430.

[13] Ng, Andrew Y.; Jordan, Michael I. (2002). *On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes*. NIPS **14**.

8.7.1 Further reading

- Domingos, Pedro; Pazzani, Michael (1997). “On the optimality of the simple Bayesian classifier under zero-one loss”. *Machine Learning* **29**: 103–137.
- Webb, G. I.; Boughton, J.; Wang, Z. (2005). “Not So Naive Bayes: Aggregating One-Dependence Estimators”. *Machine Learning* (Springer) **58** (1): 5–24. doi:10.1007/s10994-005-4258-6.
- Mozina, M.; Demsar, J.; Kattan, M.; Zupan, B. (2004). *Nomograms for Visualization of Naive Bayesian Classifier* (PDF). Proc. PKDD-2004. pp. 337–348.
- Maron, M. E. (1961). “Automatic Indexing: An Experimental Inquiry”. *JACM* **8** (3): 404–417. doi:10.1145/321075.321084.
- Minsky, M. (1961). *Steps toward Artificial Intelligence*. Proc. IRE **49** (1). pp. 8–30.

8.8 External links

- Book Chapter: Naive Bayes text classification, Introduction to Information Retrieval
- Naive Bayes for Text Classification with Unbalanced Classes

- Benchmark results of Naive Bayes implementations
- Hierarchical Naive Bayes Classifiers for uncertain data (an extension of the Naive Bayes classifier).

Software

- Naive Bayes classifiers are available in many general-purpose machine learning and NLP packages, including Apache Mahout, Mallet, NLTK, Orange, scikit-learn and Weka.
- IMSL Numerical Libraries Collections of math and statistical algorithms available in C/C++, Fortran, Java and C#/.NET. Data mining routines in the IMSL Libraries include a Naive Bayes classifier.
- Winnow content recommendation Open source Naive Bayes text classifier works with very small training and unbalanced training sets. High performance, C, any Unix.
- An interactive Microsoft Excel spreadsheet Naive Bayes implementation using VBA (requires enabled macros) with viewable source code.
- jBNC - Bayesian Network Classifier Toolbox
- Statistical Pattern Recognition Toolbox for Matlab.
- ifile - the first freely available (Naive) Bayesian mail/spam filter
- NClassifier - NClassifier is a .NET library that supports text classification and text summarization. It is a port of Classifier4J.
- Classifier4J - Classifier4J is a Java library designed to do text classification. It comes with an implementation of a Bayesian classifier.

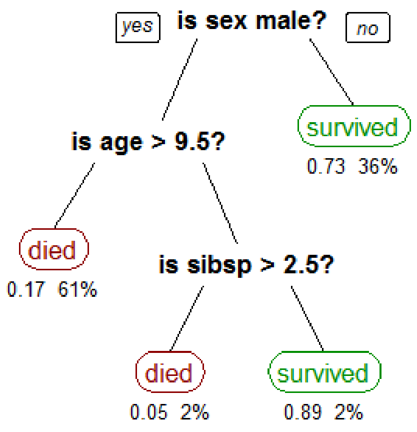
Chapter 9

Decision tree learning

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

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called **classification trees**. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called **regression trees**.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. This page deals with decision trees in data mining.



A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

9.1 General

Decision tree learning is a method commonly used in data mining.[1] The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown on the right. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

A decision tree is a simple representation for classifying examples. Decision tree learning is one of the most successful techniques for supervised classification learning. For this section, assume that all of the features have finite discrete domains, and there is a single target feature called the classification. Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of *top-down induction of decision trees* (TDIDT) [2] is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.

In data mining, decision trees can be described also as the combination of mathematical and computational techniques to aid the description, categorisation and generalisation of a given set of data.

Data comes in records of the form:

$(\mathbf{x}, Y) = (x_1, x_2, x_3, ..., x_k, Y)$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector **x** is composed of the input variables, x_1, x_2, x_3 etc., that are used for that task.

9.2 Types

Decision trees used in data mining are of two main types:

- **Classification tree** analysis is when the predicted outcome is the class to which the data belongs.
- **Regression tree** analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

The term **Classification And Regression Tree (CART)** analysis is an umbrella term used to refer to both of the above procedures, first introduced by Breiman et al.[3] Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split.[3]

Some techniques, often called *ensemble* methods, construct more than one decision tree:

- **Bagging** decision trees, an early ensemble method, builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction.[4]
- A **Random Forest** classifier uses a number of decision trees, in order to improve the classification rate.
- **Boosted Trees** can be used for regression-type and classification-type problems.[5][6]
- **Rotation forest** - in which every decision tree is trained by first applying principal component analysis (PCA) on a random subset of the input features.[7]

Decision tree learning is the construction of a decision tree from class-labeled training tuples. A decision tree is a flow-chart-like structure, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf (or terminal) node holds a class label. The topmost node in a tree is the root node.

There are many specific decision-tree algorithms. Notable ones include:

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.[8]
- MARS: extends decision trees to handle numerical data better.
- Conditional Inference Trees. Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning.[9][10]

ID3 and CART were invented independently at around the same time (between 1970 and 1980), yet follow a similar approach for learning decision tree from training tuples.

9.3 Metrics

Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items.[11] Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets. Some examples are given below. These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

9.3.1 Gini impurity

Not to be confused with Gini coefficient.

Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. Gini impurity can be computed by summing the probability of each item being chosen times the probability of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

To compute Gini impurity for a set of items, suppose $i \in \{1, 2, ..., m\}$, and let f_i be the fraction of items labeled with value i in the set.

$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m}(f_i - {f_i}^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} {f_i}^2 = 1 - \sum_{i=1}^{m} {f_i}^2$

9.3.2 Information gain

Main article: Information gain in decision trees

Used by the ID3, C4.5 and C5.0 tree-generation algorithms. Information gain is based on the concept of entropy from information theory.

$I_E(f) = -\sum_{i=1}^{m} f_i \log_2 f_i$

9.3.3 Variance reduction

Introduced in CART,[3] variance reduction is often employed in cases where the target variable is continuous (regression tree), meaning that use of many other metrics would first require discretization before being applied. The variance reduction of a node N is defined as the total reduction of the variance of the target variable x due to the split at this node:

$$I_V(N) = \frac{1}{|S|}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i - x_j)^2\right)$$

where S , S_t , and S_f are the set of presplit sample indices, set of sample indices for which the split test is true, and set of sample indices for which the split test is false, respectively.

9.4 Decision tree advantages

Amongst other data mining methods, decision trees have various advantages:

- **Simple to understand and interpret.** People are able to understand decision tree models after a brief explanation.
- **Requires little data preparation.** Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed.
- **Able to handle both numerical and categorical data.** Other techniques are usually specialised in analysing datasets that have only one type of variable. (For example, relation rules can be used only with nominal variables while neural networks can be used only with numerical variables.)
- **Uses a white box model.** If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. (An example of a black box model is an artificial neural network since the explanation for the results is difficult to understand.)
- **Possible to validate a model using statistical tests.** That makes it possible to account for the reliability of the model.
- **Robust.** Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.
- **Performs well with large datasets.** Large amounts of data can be analysed using standard computing resources in reasonable time.

9.5 Limitations

- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts.[12][13] Consequently, practical decision-tree learning algorithms are based on heuristics such as the greedy algorithm where locally-optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally-optimal decision tree. To reduce the greedy effect of local-optimality some methods such as the dual information distance (DID) tree were proposed.[14]
- Decision-tree learners can create over-complex trees that do not generalise well from the training data. (This is known as overfitting.[15]) Mechanisms such as pruning are necessary to avoid this problem (with the exception of some algorithms such as the Conditional Inference approach, that does not require pruning [9][10]).
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems. In such cases, the decision tree becomes prohibitively large. Approaches to solve the problem involve either changing the representation of the problem domain (known as propositionalisation)[16] or using learning algorithms based on more expressive representations (such as statistical relational learning or inductive logic programming).
- For data including categorical variables with different numbers of levels, information gain in decision trees is biased in favor of those attributes with more levels.[17] However, the issue of biased predictor selection is avoided by the Conditional Inference approach.[9]

9.6 Extensions

9.6.1 Decision graphs

In a decision tree, all paths from the root node to the leaf node proceed by way of conjunction, or *AND*. In a decision graph, it is possible to use disjunctions (ORs) to join two more paths together using Minimum message length (MML).[18] Decision graphs have been further extended to allow for previously unstated new attributes to be learnt dynamically and used at different places within the graph.[19] The more general coding scheme results in better predictive accuracy and log-loss probabilistic scoring. In general, decision graphs infer models with fewer leaves than decision trees.

9.6.2 Alternative search methods

Evolutionary algorithms have been used to avoid local optimal decisions and search the decision tree space with little *a priori* bias.[20][21]

It is also possible for a tree to be sampled using MCMC.[22]

The tree can be searched for in a bottom-up fashion.[23]

9.7 See also

- Decision tree pruning
- Binary decision diagram
- CHAID
- CART
- ID3 algorithm
- C4.5 algorithm
- Decision stump
- Incremental decision tree
- Alternating decision tree
- Structured data analysis (statistics)

9.8 Implementations

Many data mining software packages provide implementations of one or more decision tree algorithms. Several examples include Salford Systems CART (which licensed the proprietary code of the original CART authors[3]), IBM SPSS Modeler, RapidMiner, SAS Enterprise Miner, Matlab, R (an open source software environment for statistical computing which includes several CART implementations such as rpart, party and randomForest packages), Weka (a free and open-source data mining suite, contains many decision tree algorithms), Orange (a free data mining software suite, which includes the tree module orngTree), KNIME, Microsoft SQL Server , and scikit-learn (a free and open-source machine learning library for the Python programming language).

9.9 References

[1] Rokach, Lior; Maimon, O. (2008). *Data mining with decision trees: theory and applications*. World Scientific Pub Co Inc. ISBN 978-9812771711.

[2] Quinlan, J. R., (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers

[3] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. ISBN 978-0-412-04841-8.

[4] Breiman, L. (1996). Bagging Predictors. “Machine Learning, 24": pp. 123-140.

[5] Friedman, J. H. (1999). *Stochastic gradient boosting.* Stanford University.

[6] Hastie, T., Tibshirani, R., Friedman, J. H. (2001). *The elements of statistical learning : Data mining, inference, and prediction.* New York: Springer Verlag.

[7] Rodriguez, J.J. and Kuncheva, L.I. and Alonso, C.J. (2006), Rotation forest: A new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(10):1619-1630.

[8] Kass, G. V. (1980). “An exploratory technique for investigating large quantities of categorical data”. *Applied Statistics* **29** (2): 119–127. doi:10.2307/2986296. JSTOR 2986296.

[9] Hothorn, T.; Hornik, A.; Zeileis (2006). “Unbiased Recursive Partitioning: A Conditional Inference Framework”. *Journal of Computational and Graphical Statistics* **15** (3): 651–674. doi:10.1198/106186006X133933. JSTOR 27594202.

[10] Strobl, C.; Malley, G.; Tutz (2009). “An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classification and Regression Trees, Bagging and Random Forests”. *Psychological Methods* **14** (4): 323–348. doi:10.1037/a0016973.

[11] Rokach, L.; Maimon, O. (2005). “Top-down induction of decision trees classifiers-a survey”. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* **35** (4): 476–487. doi:10.1109/TSMCC.2004.843247.

[12] Hyafil, Laurent; Rivest, RL (1976). “Constructing Optimal Binary Decision Trees is NP-complete”. *Information Processing Letters* **5** (1): 15–17. doi:10.1016/0020-0190(76)90095-8.

[13] Murthy S. (1998). Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*

[14] Ben-Gal I. Dana A., Shkolnik N. and Singer (20). “Efficient Construction of Decision Trees by the Dual Information Distance Method” (PDF). Quality Technology & Quantitative Management (QTQM), 11(1), 133-147. Check date values in: |date= (help)

[15] “Principles of Data Mining”. 2007. doi:10.1007/978-1-84628-766-4. ISBN 978-1-84628-765-7.

[16] Horváth, Tamás; Yamamoto, Akihiro, eds. (2003). “Inductive Logic Programming”. Lecture Notes in Computer Science **2835**. doi:10.1007/b13700. ISBN 978-3-540-20144-1.

[17] Deng,H.; Runger, G.; Tuv, E. (2011). *Bias of importance measures for multi-valued attributes and solutions*. Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN). pp. 293–300.

[18] http://citeseer.ist.psu.edu/oliver93decision.html

[19] Tan & Dowe (2003)

[20] Papagelis A., Kalles D.(2001). Breeding Decision Trees Using Evolutionary Techniques, Proceedings of the Eighteenth International Conference on Machine Learning, p.393-400, June 28-July 01, 2001

[21] Barros, Rodrigo C., Basgalupp, M. P., Carvalho, A. C. P. L. F., Freitas, Alex A. (2011). A Survey of Evolutionary Algorithms for Decision-Tree Induction. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, vol. 42, n. 3, p. 291-312, May 2012.

[22] Chipman, Hugh A., Edward I. George, and Robert E. McCulloch. “Bayesian CART model search.” Journal of the American Statistical Association 93.443 (1998): 935-948.

[23] Barros R. C., Cerri R., Jaskowiak P. A., Carvalho, A. C. P. L. F., A bottom-up oblique decision tree induction algorithm. Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011).

9.10 External links

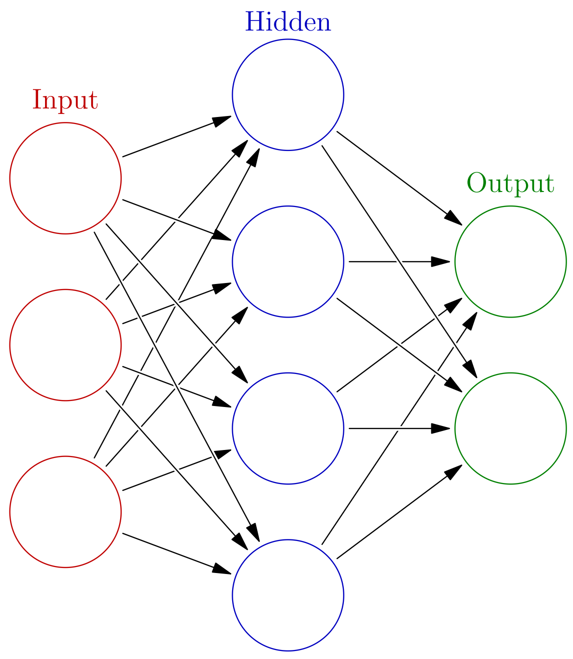
- Building Decision Trees in Python From O'Reilly.
- An Addendum to “Building Decision Trees in Python” From O'Reilly.
- Decision Trees Tutorial using Microsoft Excel.
- Decision Trees page at aitopics.org, a page with commented links.
- Decision tree implementation in Ruby (AI4R)
- Evolutionary Learning of Decision Trees in C++
- Java implementation of Decision Trees based on Information Gain
- A very explicit explanation of information gain as splitting criterion

Chapter 10

Artificial neural network

“Neural network” redirects here. For networks of living neurons, see Biological neural network. For the journal, see Neural Networks (journal). For the evolutionary concept, see Neutral network (evolution).

In machine learning and cognitive science, **artificial neural networks** (**ANNs**) are a family of statistical learning models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which send messages to each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.



An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

For example, a neural network for handwriting recognition is defined by a set of input neurons which may be activated by the pixels of an input image. After being weighted and transformed by a function (determined by the network’s designer), the activations of these neurons are then passed on to other neurons. This process is repeated until finally, an output neuron is activated. This determines which character was read.

Like other machine learning methods - systems that learn from data - neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition.

10.1 Background

Examinations of the human’s central nervous system inspired the concept of neural networks. In an Artificial Neural Network, simple artificial nodes, known as "neurons", “neurodes”, “processing elements” or “units”, are connected together to form a network which mimics a biological neural network.

There is no single formal definition of what an artificial neural network is. However, a class of statistical models may commonly be called “Neural” if they possess the following characteristics:

1. consist of sets of adaptive weights, i.e. numerical parameters that are tuned by a learning algorithm, and

2. are capable of approximating non-linear functions of their inputs.

The adaptive weights are conceptually connection strengths between neurons, which are activated during training and prediction.

Neural networks are similar to biological neural networks in performing functions collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned. The term “neural network” usually refers to models employed in statistics, cognitive psychology and artificial intelligence.

Neural network models which emulate the central nervous system are part of theoretical neuroscience and computational neuroscience.

In modern software implementations of artificial neural networks, the approach inspired by biology has been largely abandoned for a more practical approach based on statistics and signal processing. In some of these systems, neural networks or parts of neural networks (like artificial neurons) form components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such systems is more suitable for real-world problem solving, it has little to do with the traditional artificial intelligence connectionist models. What they do have in common, however, is the principle of non-linear, distributed, parallel and local processing and adaptation. Historically, the use of neural networks models marked a paradigm shift in the late eighties from high-level (symbolic) AI, characterized by expert systems with knowledge embodied in *if-then* rules, to low-level (sub-symbolic) machine learning, characterized by knowledge embodied in the parameters of a dynamical system.

10.2 History

Warren McCulloch and Walter Pitts[1] (1943) created a computational model for neural networks based on mathematics and algorithms called threshold logic. This model paved the way for neural network research to split into two distinct approaches. One approach focused on biological processes in the brain and the other focused on the application of neural networks to artificial intelligence.

In the late 1940s psychologist Donald Hebb[2] created a hypothesis of learning based on the mechanism of neural plasticity that is now known as Hebbian learning. Hebbian learning is considered to be a 'typical' unsupervised learning rule and its later variants were early models for long term potentiation. These ideas started being applied to computational models in 1948 with Turing's B-type machines.

Farley and Wesley A. Clark[3] (1954) first used computational machines, then called calculators, to simulate a Hebbian network at MIT. Other neural network computational machines were created by Rochester, Holland, Habit, and Duda[4] (1956).

Frank Rosenblatt[5] (1958) created the perceptron, an algorithm for pattern recognition based on a two-layer learning computer network using simple addition and subtraction. With mathematical notation, Rosenblatt also described circuitry not in the basic perceptron, such as the exclusive-or circuit, a circuit whose mathematical computation could not be processed until after the backpropagation algorithm was created by Paul Werbos[6] (1975).

Neural network research stagnated after the publication of machine learning research by Marvin Minsky and Seymour Papert[7] (1969), who discovered two key issues with the computational machines that processed neural networks. The first was that single-layer neural networks were incapable of processing the exclusive-or circuit. The second significant issue was that computers were not sophisticated enough to effectively handle the long run time required by large neural networks. Neural network research slowed until computers achieved greater processing power. Also key later advances was the backpropagation algorithm which effectively solved the exclusive-or problem (Werbos 1975).[6]

The parallel distributed processing of the mid-1980s became popular under the name connectionism. The text by David E. Rumelhart and James McClelland[8] (1986) provided a full exposition on the use of connectionism in computers to simulate neural processes.

Neural networks, as used in artificial intelligence, have traditionally been viewed as simplified models of neural processing in the brain, even though the relation between this model and brain biological architecture is debated, as it is not clear to what degree artificial neural networks mirror brain function.[9]

Neural networks were gradually overtaken in popularity in machine learning by support vector machines and other, much simpler methods such as linear classifiers. Renewed interest in neural nets was sparked in the late 2000s by the advent of deep learning.

10.2.1 Improvements since 2006

Computational devices have been created in CMOS, for both biophysical simulation and neuromorphic computing. More recent efforts show promise for creating nanodevices[10] for very large scale principal components analyses and convolution. If successful, these efforts could usher in a new era of neural computing[11] that is a step beyond digital computing, because it depends on learning rather than programming and because it is fundamentally analog rather than digital even though the first instantiations may in fact be with CMOS digital devices.

Between 2009 and 2012, the recurrent neural networks and deep feedforward neural networks developed in the research group of Jürgen Schmidhuber at the Swiss AI Lab IDSIA have won eight international competitions in pattern recognition and machine learning.[12][13] For example, the bi-directional and multi-dimensional long short term memory (LSTM)[14][15][16][17] of Alex Graves et al. won three competitions in connected handwriting recognition at the 2009 International Conference on Document Analysis and Recognition (ICDAR), without any prior knowledge about the three different languages to be learned.

Fast GPU-based implementations of this approach by

Dan Ciresan and colleagues at IDSIA have won several pattern recognition contests, including the IJCNN 2011 Traffic Sign Recognition Competition,[18][19] the ISBI 2012 Segmentation of Neuronal Structures in Electron Microscopy Stacks challenge,[20] and others. Their neural networks also were the first artificial pattern recognizers to achieve human-competitive or even superhuman performance[21] on important benchmarks such as traffic sign recognition (IJCNN 2012), or the MNIST handwritten digits problem of Yann LeCun at NYU.

Deep, highly nonlinear neural architectures similar to the 1980 neocognitron by Kunihiko Fukushima[22] and the “standard architecture of vision”,[23] inspired by the simple and complex cells identified by David H. Hubel and Torsten Wiesel in the primary visual cortex, can also be pre-trained by unsupervised methods[24][25] of Geoff Hinton's lab at University of Toronto.[26][27] A team from this lab won a 2012 contest sponsored by Merck to design software to help find molecules that might lead to new drugs.[28]

10.3 Models

Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function $f : X \to Y$ or a distribution over X or both X and Y , but sometimes models are also intimately associated with a particular learning algorithm or learning rule. A common use of the phrase ANN model really means the definition of a *class* of such functions (where members of the class are obtained by varying parameters, connection weights, or specifics of the architecture such as the number of neurons or their connectivity).

10.3.1 Network function

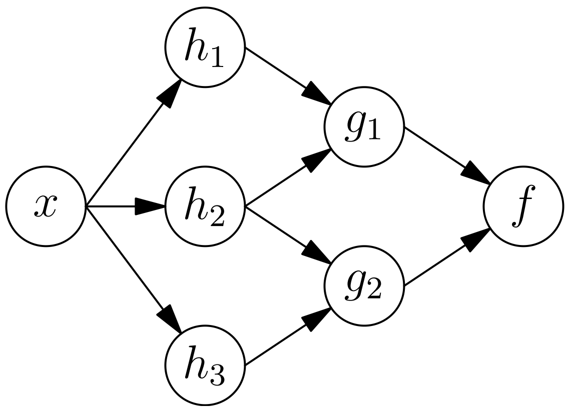
See also: Graphical models

The word *network* in the term 'artificial neural network' refers to the inter–connections between the neurons in the different layers of each system. An example system has three layers. The first layer has input neurons which send data via synapses to the second layer of neurons, and then via more synapses to the third layer of output neurons. More complex systems will have more layers of neurons with some having increased layers of input neurons and output neurons. The synapses store parameters called “weights” that manipulate the data in the calculations.

An ANN is typically defined by three types of parameters:

1. The interconnection pattern between the different layers of neurons
2. The learning process for updating the weights of the interconnections
3. The activation function that converts a neuron’s weighted input to its output activation.

Mathematically, a neuron’s network function $f(x)$ is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the *nonlinear weighted sum*, where $f(x) = K\left(\sum_i w_i g_i(x)\right)$, where K (commonly referred to as the activation function[29]) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$.



ANN dependency graph

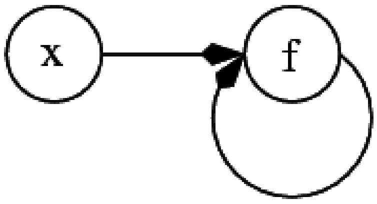
This figure depicts such a decomposition of f , with dependencies between variables indicated by arrows. These can be interpreted in two ways.

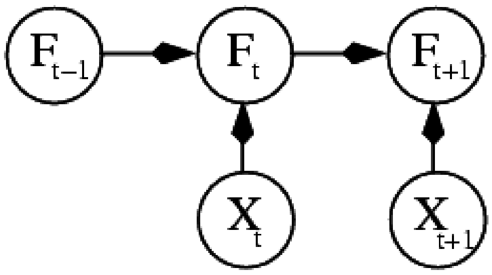
The first view is the functional view: the input x is transformed into a 3-dimensional vector h , which is then transformed into a 2-dimensional vector g , which is finally transformed into f . This view is most commonly encountered in the context of optimization.

The second view is the probabilistic view: the random variable $F = f(G)$ depends upon the random variable $G = g(H)$, which depends upon $H = h(X)$, which depends upon the random variable X . This view is most commonly encountered in the context of graphical models.

The two views are largely equivalent. In either case, for this particular network architecture, the components of individual layers are independent of each other (e.g., the components of g are independent of each other given their input h). This naturally enables a degree of parallelism in the implementation.

Networks such as the previous one are commonly called feedforward, because their graph is a directed acyclic graph. Networks with cycles are commonly called





Two separate depictions of the recurrent ANN dependency graph

recurrent. Such networks are commonly depicted in the manner shown at the top of the figure, where f is shown as being dependent upon itself. However, an implied temporal dependence is not shown.

10.3.2 Learning

What has attracted the most interest in neural networks is the possibility of *learning*. Given a specific *task* to solve, and a *class* of functions F , learning means using a set of *observations* to find $f^* \in F$ which solves the task in some *optimal* sense.

This entails defining a cost function $C : F \to \mathbb{R}$ such that, for the optimal solution f^* , $C(f^*) \leq C(f)$ $\forall f \in F$ – i.e., no solution has a cost less than the cost of the optimal solution (see Mathematical optimization).

The cost function C is an important concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Learning algorithms search through the solution space to find a function that has the smallest possible cost.

For applications where the solution is dependent on some data, the cost must necessarily be a *function of the observations*, otherwise we would not be modelling anything related to the data. It is frequently defined as a statistic to which only approximations can be made. As a simple example, consider the problem of finding the model f , which minimizes $C = E\left[(f(x) - y)^2\right]$, for data pairs (x, y) drawn from some distribution $\mathcal{D}$. In practical situations we would only have N samples from $\mathcal{D}$ and thus, for the above example, we would only minimize $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$. Thus, the cost is minimized over a sample of the data rather than the entire data set.

When $N \to \infty$ some form of online machine learning must be used, where the cost is partially minimized as each new example is seen. While online machine learning is often used when $\mathcal{D}$ is fixed, it is most useful in the case where the distribution changes slowly over time. In neural network methods, some form of online machine learning is frequently used for finite datasets.

See also: Mathematical optimization, Estimation theory and Machine learning

Choosing a cost function

While it is possible to define some arbitrary ad hoc cost function, frequently a particular cost will be used, either because it has desirable properties (such as convexity) or because it arises naturally from a particular formulation of the problem (e.g., in a probabilistic formulation the posterior probability of the model can be used as an inverse cost). Ultimately, the cost function will depend on the desired task. An overview of the three main categories of learning tasks is provided below:

10.3.3 Learning paradigms

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning

In supervised learning, we are given a set of example pairs $(x, y), x \in X, y \in Y$ and the aim is to find a function $f : X \to Y$ in the allowed class of functions that matches the examples. In other words, we wish to *infer* the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error, which tries to minimize the average squared error between the network's output, $f(x)$, and the target value y over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called multilayer perceptrons, one obtains the common and well-known backpropagation algorithm for training neural networks.

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a "teacher", in the form of a function that provides continuous feedback on the quality of solutions obtained thus far.

Unsupervised learning

In unsupervised learning, some data x is given and the cost function to be minimized, that can be any function of the data x and the network's output, f .

The cost function is dependent on the task (what we are trying to model) and our *a priori* assumptions (the implicit properties of our model, its parameters and the observed variables).

As a trivial example, consider the model $f(x) = a$ where a is a constant and the cost $C = E[(x - f(x))^2]$. Minimizing this cost will give us a value of a that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: for example, in compression it could be related to the mutual information between x and $f(x)$, whereas in statistical modeling, it could be related to the posterior probability of the model given the data (note that in both of those examples those quantities would be maximized rather than minimized).

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

Reinforcement learning

In reinforcement learning, data x are usually not given, but generated by an agent's interactions with the environment. At each point in time t , the agent performs an action y_t and the environment generates an observation x_t and an instantaneous cost c_t , according to some (usually unknown) dynamics. The aim is to discover a *policy* for selecting actions that minimizes some measure of a long-term cost; i.e., the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

More formally the environment is modelled as a Markov decision process (MDP) with states $s_1, ..., s_n \in S$ and actions $a_1, ..., a_m \in A$ with the following probability distributions: the instantaneous cost distribution $P(c_t|s_t)$, the observation distribution $P(x_t|s_t)$ and the transition $P(s_{t+1}|s_t, a_t)$, while a policy is defined as conditional distribution over actions given the observations. Taken together, the two then define a Markov chain (MC). The aim is to discover the policy that minimizes the cost; i.e., the MC for which the cost is minimal.

ANNs are frequently used in reinforcement learning as part of the overall algorithm.[30][31] Dynamic programming has been coupled with ANNs (Neuro dynamic programming) by Bertsekas and Tsitsiklis[32] and applied to multi-dimensional nonlinear problems such as those involved in vehicle routing,[33] natural resources management[34][35] or medicine[36] because of the ability of ANNs to mitigate losses of accuracy even when reducing the discretization grid density for numerically approximating the solution of the original control problems.

Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

See also: dynamic programming and stochastic control

10.3.4 Learning algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation.

Most of the algorithms used in training artificial neural networks employ some form of gradient descent, using backpropagation to compute the actual gradients. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Evolutionary methods,[37] gene expression programming,[38] simulated annealing,[39] expectation-maximization, non-parametric methods and particle swarm optimization[40] are some commonly used methods for training neural networks.

See also: machine learning

10.4 Employing artificial neural networks

Perhaps the greatest advantage of ANNs is their ability to be used as an arbitrary function approximation mechanism that 'learns' from observed data. However, using them is not so straightforward, and a relatively good understanding of the underlying theory is essential.

- Choice of model: This will depend on the data representation and the application. Overly complex models tend to lead to problems with learning.

- Learning algorithm: There are numerous trade-offs between learning algorithms. Almost any algorithm will work well with the *correct hyperparameters* for training on a particular fixed data set. However, selecting and tuning an algorithm for training on unseen data requires a significant amount of experimentation.

- Robustness: If the model, cost function and learning algorithm are selected appropriately the resulting ANN can be extremely robust.

With the correct implementation, ANNs can be used naturally in online learning and large data set applications. Their simple implementation and the existence of mostly local dependencies exhibited in the structure allows for fast, parallel implementations in hardware.

10.5 Applications

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical.

10.5.1 Real-life applications

The tasks artificial neural networks are applied to tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.
- Robotics, including directing manipulators, prosthesis.
- Control, including Computer numerical control.

Application areas include the system identification and control (vehicle control, process control, natural resources management), quantum chemistry,[41] game-playing and decision making (backgammon, chess, poker), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications (e.g. automated trading systems), data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

Artificial neural networks have also been used to diagnose several cancers. An ANN based hybrid lung cancer detection system named HLND improves the accuracy of diagnosis and the speed of lung cancer radiology.[42] These networks have also been used to diagnose prostate cancer. The diagnoses can be used to make specific models taken from a large group of patients compared to information of one given patient. The models do not depend on assumptions about correlations of different variables. Colorectal cancer has also been predicted using the neural networks. Neural networks could predict the outcome for a patient with colorectal cancer with more accuracy than the current clinical methods. After training, the networks could predict multiple patient outcomes from unrelated institutions.[43]

10.5.2 Neural networks and neuroscience

Theoretical and computational neuroscience is the field concerned with the theoretical analysis and the computational modeling of biological neural systems. Since neural systems are intimately related to cognitive processes and behavior, the field is closely related to cognitive and behavioral modeling.

The aim of the field is to create models of biological neural systems in order to understand how biological systems work. To gain this understanding, neuroscientists strive to make a link between observed biological processes (data), biologically plausible mechanisms for neural processing and learning (biological neural network models) and theory (statistical learning theory and information theory).

Types of models

Many models are used in the field, defined at different levels of abstraction and modeling different aspects of neural systems. They range from models of the short-term behavior of individual neurons, models of how the dynamics of neural circuitry arise from interactions between individual neurons and finally to models of how behavior can arise from abstract neural modules that represent complete subsystems. These include models of the long-term, and short-term plasticity, of neural systems and their relations to learning and memory from the individual neuron to the system level.

10.6 Neural network software

Main article: Neural network software

Neural network software is used to simulate, research, develop and apply artificial neural networks, biological neural networks and, in some cases, a wider array of adaptive systems.

10.7 Types of artificial neural networks

Main article: Types of artificial neural networks

Artificial neural network types vary from those with only one or two layers of single direction logic, to complicated multi–input many directional feedback loops and layers. On the whole, these systems use algorithms in their programming to determine control and organization of their functions. Most systems use “weights” to change the parameters of the throughput and the varying connections to the neurons. Artificial neural networks can be autonomous and learn by input from outside “teachers” or even self-teaching from written-in rules.

10.8 Theoretical properties

10.8.1 Computational power

The multi-layer perceptron (MLP) is a universal function approximator, as proven by the universal approximation theorem. However, the proof is not constructive regarding the number of neurons required or the settings of the weights.

Work by Hava Siegelmann and Eduardo D. Sontag has provided a proof that a specific recurrent architecture with rational valued weights (as opposed to full precision real number-valued weights) has the full power of a Universal Turing Machine[44] using a finite number of neurons and standard linear connections. Further, it has been shown that the use of irrational values for weights results in a machine with super-Turing power.[45]

10.8.2 Capacity

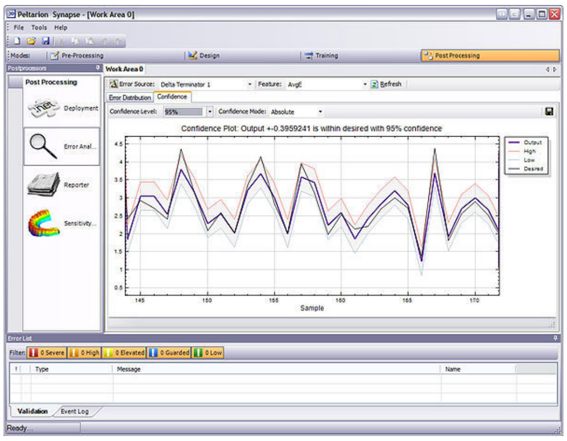
Artificial neural network models have a property called 'capacity', which roughly corresponds to their ability to model any given function. It is related to the amount of information that can be stored in the network and to the notion of complexity.

10.8.3 Convergence

Nothing can be said in general about convergence since it depends on a number of factors. Firstly, there may exist many local minima. This depends on the cost function and the model. Secondly, the optimization method used might not be guaranteed to converge when far away from a local minimum. Thirdly, for a very large amount of data or parameters, some methods become impractical. In general, it has been found that theoretical guarantees regarding convergence are an unreliable guide to practical application.

10.8.4 Generalization and statistics

In applications where the goal is to create a system that generalizes well in unseen examples, the problem of overtraining has emerged. This arises in convoluted or overspecified systems when the capacity of the network significantly exceeds the needed free parameters. There are two schools of thought for avoiding this problem: The first is to use cross-validation and similar techniques to check for the presence of overtraining and optimally select hyperparameters such as to minimize the generalization error. The second is to use some form of *regularization*. This is a concept that emerges naturally in a probabilistic (Bayesian) framework, where the regularization can be performed by selecting a larger prior probability over simpler models; but also in statistical learning theory, where the goal is to minimize over two quantities: the 'empirical risk' and the 'structural risk', which roughly corresponds to the error over the training set and the predicted error in unseen data due to overfitting.


Peltarion Synapse - [Work Area 0]

Error Source: Delta Terminator 1 Feature: AvgE

Confidence Level: 95% Confidence Mode: Absolute

Confidence Plot: Output +-0.3959241 is within desired with 95% confidence

Filter: 0 Severe 0 High 0 Elevated 0 Guarded 0 Low

Ready...

Confidence analysis of a neural network

Supervised neural networks that use a mean squared error (MSE) cost function can use formal statistical methods to determine the confidence of the trained model. The MSE on a validation set can be used as an estimate for variance. This value can then be used to calculate the confidence interval of the output of the network, assuming a normal distribution. A confidence analysis made this way is statistically valid as long as the output probability distribution stays the same and the network is not modified.

By assigning a softmax activation function, a generalization of the logistic function, on the output layer of the neural network (or a softmax component in a component-based neural network) for categorical target variables, the outputs can be interpreted as posterior probabilities. This is very useful in classification as it gives a certainty measure on classifications.

The softmax activation function is:

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

10.9 Controversies

10.9.1 Training issues

A common criticism of neural networks, particularly in robotics, is that they require a large diversity of training for real-world operation . This is not surprising, since any learning machine needs sufficient representative examples in order to capture the underlying structure that allows it to generalize to new cases. Dean Pomerleau, in his research presented in the paper "Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving," uses a neural network to train a robotic vehicle to drive on multiple types of roads (single lane, multi-lane, dirt, etc.). A large amount of his research is devoted to (1) extrapolating multiple training scenarios from a single training experience, and (2) preserving past training diversity so that the system does not become overtrained (if, for example, it is presented with a series of right turns – it should not learn to always turn right). These issues are common in neural networks that must decide from amongst a wide variety of responses, but can be dealt with in several ways, for example by randomly shuffling the training examples, by using a numerical optimization algorithm that does not take too large steps when changing the network connections following an example, or by grouping examples in so-called mini-batches.

A. K. Dewdney, a former *Scientific American* columnist, wrote in 1997, "Although neural nets do solve a few toy problems, their powers of computation are so limited that I am surprised anyone takes them seriously as a general problem-solving tool." (Dewdney, p. 82)

10.9.2 Hardware issues

To implement large and effective software neural networks, considerable processing and storage resources need to be committed . While the brain has hardware tailored to the task of processing signals through a graph of neurons, simulating even a most simplified form on Von Neumann technology may compel a neural network designer to fill many millions of database rows for its connections – which can consume vast amounts of computer memory and hard disk space. Furthermore, the designer of neural network systems will often need to simulate the transmission of signals through many of these connections and their associated neurons – which must often be matched with incredible amounts of CPU processing power and time. While neural networks often yield *effective* programs, they too often do so at the cost of *efficiency* (they tend to consume considerable amounts of time and money).

Computing power continues to grow roughly according to Moore's Law, which may provide sufficient resources to accomplish new tasks. Neuromorphic engineering addresses the hardware difficulty directly, by constructing non-Von-Neumann chips with circuits designed to implement neural nets from the ground up.

10.9.3 Practical counterexamples to criticisms

Arguments against Dewdney's position are that neural networks have been successfully used to solve many complex and diverse tasks, ranging from autonomously flying aircraft[46] to detecting credit card fraud .

Technology writer Roger Bridgman commented on Dewdney's statements about neural nets:

> Neural networks, for instance, are in the dock not only because they have been hyped to high heaven, (what hasn't?) but also because you could create a successful net without understanding how it worked: the bunch of numbers that captures its behaviour would in all probability be "an opaque, unreadable table...valueless as a scientific resource".
>
> In spite of his emphatic declaration that science is not technology, Dewdney seems here to pillory neural nets as bad science when most of those devising them are just trying to be good engineers. An unreadable table that a useful machine could read would still be well worth having.[47]

Although it is true that analyzing what has been learned by an artificial neural network is difficult, it is much easier to do so than to analyze what has been learned by a biological neural network. Furthermore, researchers involved in exploring learning algorithms for neural networks are gradually uncovering generic principles which allow a learning machine to be successful. For example, Bengio and LeCun (2007) wrote an article regarding local vs non-local learning, as well as shallow vs deep architecture.[48]

10.9.4 Hybrid approaches

Some other criticisms come from advocates of hybrid models (combining neural networks and symbolic approaches), who believe that the intermix of these two approaches can better capture the mechanisms of the human mind.[49][50]

10.10 Gallery

- A single-layer feedforward artificial neural network. Arrows originating from are omitted for clarity. There are p inputs to this network and q outputs. In this system, the value of the qth output, would be calculated as
- A two-layer feedforward artificial neural network.
-
-

10.11 See also

- 20Q
- ADALINE
- Adaptive resonance theory
- Artificial life
- Associative memory
- Autoencoder
- Backpropagation
- BEAM robotics
- Biological cybernetics
- Biologically inspired computing
- Blue brain
- Catastrophic interference
- Cerebellar Model Articulation Controller
- Cognitive architecture
- Cognitive science
- Convolutional neural network (CNN)
- Connectionist expert system
- Connectomics
- Cultured neuronal networks
- Deep learning
- Digital morphogenesis
- Encog
- Fuzzy logic
- Gene expression programming
- Genetic algorithm
- Group method of data handling
- Habituation
- In Situ Adaptive Tabulation
- Models of neural computation
- Multilinear subspace learning
- Neuroevolution
- Neural coding
- Neural gas
- Neural network software
- Neuroscience
- Ni1000 chip
- Nonlinear system identification
- Optical neural network
- Parallel Constraint Satisfaction Processes
- Parallel distributed processing
- Radial basis function network
- Recurrent neural networks
- Self-organizing map
- Spiking neural network
- Systolic array
- Tensor product network
- Time delay neural network (TDNN)

10.12 References

[1] McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics* **5** (4): 115–133. doi:10.1007/BF02478259.

[2] Hebb, Donald (1949). *The Organization of Behavior*. New York: Wiley.

[3] Farley, B.G.; W.A. Clark (1954). "Simulation of Self-Organizing Systems by Digital Computer". *IRE Transactions on Information Theory* **4** (4): 76–84. doi:10.1109/TIT.1954.1057468.

[4] Rochester, N.; J.H. Holland, L.H. Habit, and W.L. Duda (1956). "Tests on a cell assembly theory of the action of the brain, using a large digital computer". *IRE Transactions on Information Theory* **2** (3): 80–93. doi:10.1109/TIT.1956.1056810.

[5] Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". *Psychological Review* **65** (6): 386–408. doi:10.1037/h0042519. PMID 13602029.

[6] Werbos, P.J. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*.

[7] Minsky, M.; S. Papert (1969). *An Introduction to Computational Geometry*. MIT Press. ISBN 0-262-63022-2.

[8] Rumelhart, D.E; James McClelland (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge: MIT Press.

[9] Russell, Ingrid. "Neural Networks Module". Retrieved 2012.

[10] Yang, J. J.; Pickett, M. D.; Li, X. M.; Ohlberg, D. A. A.; Stewart, D. R.; Williams, R. S. Nat. Nanotechnol. 2008, 3, 429–433.

[11] Strukov, D. B.; Snider, G. S.; Stewart, D. R.; Williams, R. S. *Nature* 2008, 453, 80–83.

[12] 2012 Kurzweil AI Interview with Jürgen Schmidhuber on the eight competitions won by his Deep Learning team 2009–2012

[13] http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions 2012 Kurzweil AI Interview with Jürgen Schmidhuber on the eight competitions won by his Deep Learning team 2009–2012

[14] Graves, Alex; and Schmidhuber, Jürgen; *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks*, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), *Advances in Neural Information Processing Systems 22 (NIPS'22), 7–10 December 2009, Vancouver, BC*, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552.

[15] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.

[16] Graves, Alex; and Schmidhuber, Jürgen; *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks*, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), *Advances in Neural Information Processing Systems 22 (NIPS'22), December 7th–10th, 2009, Vancouver, BC*, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552

[17] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.

[18] D. C. Ciresan, U. Meier, J. Masci, J. Schmidhuber. Multi-Column Deep Neural Network for Traffic Sign Classification. Neural Networks, 2012.

[19] D. C. Ciresan, U. Meier, J. Masci, J. Schmidhuber. Multi-Column Deep Neural Network for Traffic Sign Classification. Neural Networks, 2012.

[20] D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In Advances in Neural Information Processing Systems (NIPS 2012), Lake Tahoe, 2012.

[21] D. C. Ciresan, U. Meier, J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. IEEE Conf. on Computer Vision and Pattern Recognition CVPR 2012.

[22] Fukushima, K. (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". *Biological Cybernetics* **36** (4): 93–202. doi:10.1007/BF00344251. PMID 7370364.

[23] M Riesenhuber, T Poggio. Hierarchical models of object recognition in cortex. Nature neuroscience, 1999.

[24] Deep belief networks at Scholarpedia.

[25] Hinton, G. E.; Osindero, S.; Teh, Y. W. (2006). "A Fast Learning Algorithm for Deep Belief Nets" (PDF). *Neural Computation* **18** (7): 1527–1554. doi:10.1162/neco.2006.18.7.1527. PMID 16764513.

[26] http://www.scholarpedia.org/article/Deep_belief_networks /

[27] Hinton, G. E.; Osindero, S.; Teh, Y. (2006). "A fast learning algorithm for deep belief nets" (PDF). *Neural Computation* **18** (7): 1527–1554. doi:10.1162/neco.2006.18.7.1527. PMID 16764513.

[28] John Markoff (November 23, 2012). "Scientists See Promise in Deep-Learning Programs". *New York Times*.

[29] "The Machine Learning Dictionary".

[30] Dominic, S., Das, R., Whitley, D., Anderson, C. (July 1991). "Genetic reinforcement learning for neural networks". *IJCNN-91-Seattle International Joint Conference on Neural Networks*. IJCNN-91-Seattle International Joint Conference on Neural Networks. Seattle, Washington, USA: IEEE. doi:10.1109/IJCNN.1991.155315. ISBN 0-7803-0164-1. Retrieved 29 July 2012.

[31] Hoskins, J.C.; Himmelblau, D.M. (1992). "Process control via artificial neural networks and reinforcement learning". *Computers & Chemical Engineering* **16** (4): 241–251. doi:10.1016/0098-1354(92)80045-B.

[32] Bertsekas, D.P., Tsitsiklis, J.N. (1996). *Neuro-dynamic programming*. Athena Scientific. p. 512. ISBN 1-886529-10-8.

[33] Secomandi, Nicola (2000). "Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands". *Computers & Operations Research* **27** (11–12): 1201–1225. doi:10.1016/S0305-0548(99)00146-X.

[34] de Rigo, D., Rizzoli, A. E., Soncini-Sessa, R., Weber, E., Zenesi, P. (2001). "Neuro-dynamic programming for the efficient management of reservoir networks" (PDF). *Proceedings of MODSIM 2001, International Congress on Modelling and Simulation*. MODSIM 2001, International

Congress on Modelling and Simulation. Canberra, Australia: Modelling and Simulation Society of Australia and New Zealand. doi:10.5281/zenodo.7481. ISBN 0-867405252. Retrieved 29 July 2012.

[35] Damas, M., Salmeron, M., Diaz, A., Ortega, J., Prieto, A., Olivares, G. (2000). "Genetic algorithms and neuro-dynamic programming: application to water supply networks". *Proceedings of 2000 Congress on Evolutionary Computation*. 2000 Congress on Evolutionary Computation. La Jolla, California, USA: IEEE. doi:10.1109/CEC.2000.870269. ISBN 0-7803-6375-2. Retrieved 29 July 2012.

[36] Deng, Geng; Ferris, M.C. (2008). "Neuro-dynamic programming for fractionated radiotherapy planning". *Springer Optimization and Its Applications* **12**: 47–70. doi:10.1007/978-0-387-73299-2_3.

[37] de Rigo, D., Castelletti, A., Rizzoli, A.E., Soncini-Sessa, R., Weber, E. (January 2005). "A selective improvement technique for fastening Neuro-Dynamic Programming in Water Resources Network Management". In Pavel Zítek. *Proceedings of the 16th IFAC World Congress – IFAC-PapersOnLine*. 16th IFAC World Congress **16**. Prague, Czech Republic: IFAC. doi:10.3182/20050703-6-CZ-1902.02172. ISBN 978-3-902661-75-3. Retrieved 30 December 2011.

[38] Ferreira, C. (2006). "Designing Neural Networks Using Gene Expression Programming" (PDF). In A. Abraham, B. de Baets, M. Köppen, and B. Nickolay, eds., Applied Soft Computing Technologies: The Challenge of Complexity, pages 517–536, Springer-Verlag.

[39] Da, Y., Xiurun, G. (July 2005). T. Villmann, ed. *An improved PSO-based ANN with simulated annealing technique*. New Aspects in Neurocomputing: 11th European Symposium on Artificial Neural Networks. Elsevier. doi:10.1016/j.neucom.2004.07.002.

[40] Wu, J., Chen, E. (May 2009). Wang, H., Shen, Y., Huang, T., Zeng, Z., ed. *A Novel Nonparametric Regression Ensemble for Rainfall Forecasting Using Particle Swarm Optimization Technique Coupled with Artificial Neural Network*. 6th International Symposium on Neural Networks, ISNN 2009. Springer. doi:10.1007/978-3-642-01513-7_6. ISBN 978-3-642-01215-0.

[41] Roman M. Balabin, Ekaterina I. Lomakina (2009). "Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies". *J. Chem. Phys.* **131** (7): 074104. doi:10.1063/1.3206326. PMID 19708729.

[42] Ganesan, N. "Application of Neural Networks in Diagnosing Cancer Disease Using Demographic Data" (PDF). International Journal of Computer Applications.

[43] Bottaci, Leonardo. "Artificial Neural Networks Applied to Outcome Prediction for Colorectal Cancer Patients in Separate Institutions" (PDF). The Lancet.

[44] Siegelmann, H.T.; Sontag, E.D. (1991). "Turing computability with neural nets" (PDF). *Appl. Math. Lett.* **4** (6): 77–80. doi:10.1016/0893-9659(91)90080-F.

[45] Balcázar, José (Jul 1997). "Computational Power of Neural Networks: A Kolmogorov Complexity Characterization". *Information Theory, IEEE Transactions on* **43** (4): 1175–1183. doi:10.1109/18.605580. Retrieved 3 November 2014.

[46] NASA - Dryden Flight Research Center - News Room: News Releases: NASA NEURAL NETWORK PROJECT PASSES MILESTONE. Nasa.gov. Retrieved on 2013-11-20.

[47] Roger Bridgman's defence of neural networks

[48] http://www.iro.umontreal.ca/~{ }lisa/publications2/index.php/publications/show/4

[49] Sun and Bookman (1990)

[50] Tahmasebi; Hezarkhani (2012). "A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation". *Computers & Geosciences* **42**: 18–27. doi:10.1016/j.cageo.2012.02.004.

10.13 Bibliography

- Bhadeshia H. K. D. H. (1999). "Neural Networks in Materials Science" (PDF). *ISIJ International* **39** (10): 966–979. doi:10.2355/isijinternational.39.966.
- Bishop, C.M. (1995) *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press. ISBN 0-19-853849-9 (hardback) or ISBN 0-19-853864-2 (paperback)
- Cybenko, G.V. (1989). Approximation by Superpositions of a Sigmoidal function, *Mathematics of Control, Signals, and Systems*, Vol. 2 pp. 303–314. electronic version
- Duda, R.O., Hart, P.E., Stork, D.G. (2001) *Pattern classification (2nd edition)*, Wiley, ISBN 0-471-05669-3
- Egmont-Petersen, M., de Ridder, D., Handels, H. (2002). "Image processing with neural networks – a review". *Pattern Recognition* **35** (10): 2279–2301. doi:10.1016/S0031-3203(01)00178-9.
- Gurney, K. (1997) *An Introduction to Neural Networks* London: Routledge. ISBN 1-85728-673-1 (hardback) or ISBN 1-85728-503-4 (paperback)
- Haykin, S. (1999) *Neural Networks: A Comprehensive Foundation*, Prentice Hall, ISBN 0-13-273350-1
- Fahlman, S, Lebiere, C (1991). *The Cascade-Correlation Learning Architecture*, created for National Science Foundation, Contract Number EET-8716324, and Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976 under Contract F33615-87-C-1499. electronic version

- Hertz, J., Palmer, R.G., Krogh. A.S. (1990) *Introduction to the theory of neural computation*, Perseus Books. ISBN 0-201-51560-1
- Lawrence, Jeanette (1994) *Introduction to Neural Networks*, California Scientific Software Press. ISBN 1-883157-00-5
- Masters, Timothy (1994) *Signal and Image Processing with Neural Networks*, John Wiley & Sons, Inc. ISBN 0-471-04963-8
- Ripley, Brian D. (1996) *Pattern Recognition and Neural Networks*, Cambridge
- Siegelmann, H.T. and Sontag, E.D. (1994). Analog computation via neural networks, *Theoretical Computer Science*, v. 131, no. 2, pp. 331–360. electronic version
- Sergios Theodoridis, Konstantinos Koutroumbas (2009) "Pattern Recognition", 4th Edition, Academic Press, ISBN 978-1-59749-272-0.
- Smith, Murray (1993) *Neural Networks for Statistical Modeling*, Van Nostrand Reinhold, ISBN 0-442-01310-8
- Wasserman, Philip (1993) *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, ISBN 0-442-00461-3
- *Computational Intelligence: A Methodological Introduction* by Kruse, Borgelt, Klawonn, Moewes, Steinbrecher, Held, 2013, Springer, ISBN 9781447150121
- *Neuro-Fuzzy-Systeme* (3rd edition) by Borgelt, Klawonn, Kruse, Nauck, 2003, Vieweg, ISBN 9783528252656

10.14 External links

- Neural Networks at DMOZ
- A brief introduction to Neural Networks (PDF), illustrated 250p textbook covering the common kinds of neural networks (CC license).

Chapter 11

Ensemble learning

For an alternative meaning, see variational Bayesian methods.

In statistics and machine learning, **ensemble methods** use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms.[1][2][3] Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble refers only to a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives.

11.1 Overview

Supervised learning algorithms are commonly described as performing the task of searching through a hypothesis space to find a suitable hypothesis that will make good predictions with a particular problem. Even if the hypothesis space contains hypotheses that are very well-suited for a particular problem, it may be very difficult to find a good one. Ensembles combine multiple hypotheses to form a (hopefully) better hypothesis. In other words, an ensemble is a technique for combining many *weak learners* in an attempt to produce a *strong learner*. The term *ensemble* is usually reserved for methods that generate multiple hypotheses using the same base learner. The broader term of *multiple classifier systems* also covers hybridization of hypotheses that are not induced by the same base learner.

Evaluating the prediction of an ensemble typically requires more computation than evaluating the prediction of a single model, so ensembles may be thought of as a way to compensate for poor learning algorithms by performing a lot of extra computation. Fast algorithms such as decision trees are commonly used with ensembles (for example *Random Forest*), although slower algorithms can benefit from ensemble techniques as well.

11.2 Ensemble theory

An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. This flexibility can, in theory, enable them to over-fit the training data more than a single model would, but in practice, some ensemble techniques (especially bagging) tend to reduce problems related to over-fitting of the training data.

Empirically, ensembles tend to yield better results when there is a significant diversity among the models.[4][5] Many ensemble methods, therefore, seek to promote diversity among the models they combine.[6][7] Although perhaps non-intuitive, more random algorithms (like random decision trees) can be used to produce a stronger ensemble than very deliberate algorithms (like entropy-reducing decision trees).[8] Using a variety of strong learning algorithms, however, has been shown to be more effective than using techniques that attempt to *dumb-down* the models in order to promote diversity.[9]

11.3 Common types of ensembles

11.3.1 Bayes optimal classifier

The Bayes Optimal Classifier is a classification technique. It is an ensemble of all the hypotheses in the hypothesis space. On average, no other ensemble can outperform it.[10] Each hypothesis is given a vote proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis were true. To facilitate training data of finite size, the vote of each hypothesis is also multiplied by the prior probability of that hypothesis. The Bayes Optimal Classifier can be expressed with the following equation:

$$y = \text{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(T|h_i)P(h_i)$$

where y is the predicted class, C is the set of all possible classes, H is the hypothesis space, P refers to a *probability*, and T is the training data. As an ensemble, the Bayes Optimal Classifier represents a hypothesis that is not necessarily in H . The hypothesis represented by the Bayes Optimal Classifier, however, is the optimal hypothesis in *ensemble space* (the space of all possible ensembles consisting only of hypotheses in H).

Unfortunately, Bayes Optimal Classifier cannot be practically implemented for any but the most simple of problems. There are several reasons why the Bayes Optimal Classifier cannot be practically implemented:

1. Most interesting hypothesis spaces are too large to iterate over, as required by the argmax .
2. Many hypotheses yield only a predicted class, rather than a probability for each class as required by the term $P(c_j|h_i)$.
3. Computing an unbiased estimate of the probability of the training set given a hypothesis ($P(T|h_i)$) is non-trivial.
4. Estimating the prior probability for each hypothesis ($P(h_i)$) is rarely feasible.

11.3.2 Bootstrap aggregating (bagging)

Main article: Bootstrap aggregating

Bootstrap aggregating, often abbreviated as *bagging*, involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set. As an example, the random forest algorithm combines random decision trees with bagging to achieve very high classification accuracy.[11] An interesting application of bagging in unsupervised learning is provided here.[12][13]

11.3.3 Boosting

Main article: Boosting (meta-algorithm)

Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified. In some cases, boosting has been shown to yield better accuracy than bagging, but it also tends to be more likely to over-fit the training data. By far, the most common implementation of Boosting is Adaboost, although some newer algorithms are reported to achieve better results .

11.3.4 Bayesian model averaging

Bayesian model averaging (BMA) is an ensemble technique that seeks to approximate the Bayes Optimal Classifier by sampling hypotheses from the hypothesis space, and combining them using Bayes' law.[14] Unlike the Bayes optimal classifier, Bayesian model averaging can be practically implemented. Hypotheses are typically sampled using a Monte Carlo sampling technique such as MCMC. For example, Gibbs sampling may be used to draw hypotheses that are representative of the distribution $P(T|H)$. It has been shown that under certain circumstances, when hypotheses are drawn in this manner and averaged according to Bayes' law, this technique has an expected error that is bounded to be at most twice the expected error of the Bayes optimal classifier.[15] Despite the theoretical correctness of this technique, it has been found to promote over-fitting and to perform worse, empirically, compared to simpler ensemble techniques such as bagging;[16] however, these conclusions appear to be based on a misunderstanding of the purpose of Bayesian model averaging vs. model combination.[17]

Pseudo-code

function train_bayesian_model_averaging(T) z = -infinity For each model, m, in the ensemble: Train m, typically using a random subset of the training data, T. Let prior[m] be the prior probability that m is the generating hypothesis. Typically, uniform priors are used, so prior[m] = 1. Let x be the predictive accuracy (from 0 to 1) of m for predicting the labels in T. Use x to estimate log_likelihood[m]. Often, this is computed as log_likelihood[m] = |T| * (x * log(x) + (1 - x) * log(1 - x)), where |T| is the number of training patterns in T. z = max(z, log_likelihood[m]) For each model, m, in the ensemble: weight[m] = prior[m] * exp(log_likelihood[m] - z) Normalize all the model weights to sum to 1.

11.3.5 Bayesian model combination

Bayesian model combination (BMC) is an algorithmic correction to BMA. Instead of sampling each model in the ensemble individually, it samples from the space of possible ensembles (with model weightings drawn randomly from a Dirichlet distribution having uniform parameters). This modification overcomes the tendency of BMA to converge toward giving all of the weight to a single model. Although BMC is somewhat more computationally expensive than BMA, it tends to yield dramatically better results. The results from BMC have been shown to be better on average (with statistical significance) than BMA, and bagging.[18]

The use of Bayes' law to compute model weights necessitates computing the probability of the data given each model. Typically, none of the models in the ensemble are exactly the distribution from which the training data were generated, so all of them correctly receive a value close to zero for this term. This would work well if the ensemble were big enough to sample the entire model-space, but such is rarely possible. Consequently, each pattern in the training data will cause the ensemble weight to shift toward the model in the ensemble that is closest to the distribution of the training data. It essentially reduces to an unnecessarily complex method for doing model selection.

The possible weightings for an ensemble can be visualized as lying on a simplex. At each vertex of the simplex, all of the weight is given to a single model in the ensemble. BMA converges toward the vertex that is closest to the distribution of the training data. By contrast, BMC converges toward the point where this distribution projects onto the simplex. In other words, instead of selecting the one model that is closest to the generating distribution, it seeks the combination of models that is closest to the generating distribution.

The results from BMA can often be approximated by using cross-validation to select the best model from a bucket of models. Likewise, the results from BMC may be approximated by using cross-validation to select the best ensemble combination from a random sampling of possible weightings.

Pseudo-code

function train_bayesian_model_combination(T) For each model, m, in the ensemble: weight[m] = 0 sum_weight = 0 z = -infinity Let n be some number of weightings to sample. (100 might be a reasonable value. Smaller is faster. Bigger leads to more precise results.) for i from 0 to n - 1: For each model, m, in the ensemble: // draw from a uniform Dirichlet distribution v[m] = -log(random_uniform(0,1)) Normalize v to sum to 1 Let x be the predictive accuracy (from 0 to 1) of the entire ensemble, weighted according to v, for predicting the labels in T. Use x to estimate log_likelihood[i]. Often, this is computed as log_likelihood[i] = |T| * (x * log(x) + (1 - x) * log(1 - x)), where |T| is the number of training patterns in T. If log_likelihood[i] > z: // z is used to maintain numerical stability For each model, m, in the ensemble: weight[m] = weight[m] * exp(z - log_likelihood[i]) z = log_likelihood[i] w = exp(log_likelihood[i] - z) For each model, m, in the ensemble: weight[m] = weight[m] * sum_weight / (sum_weight + w) + w * v[m] sum_weight = sum_weight + w Normalize the model weights to sum to 1.

11.3.6 Bucket of models

A "bucket of models" is an ensemble in which a model selection algorithm is used to choose the best model for each problem. When tested with only one problem, a bucket of models can produce no better results than the best model in the set, but when evaluated across many problems, it will typically produce much better results, on average, than any model in the set.

The most common approach used for model-selection is cross-validation selection (sometimes called a "bake-off contest"). It is described with the following pseudo-code:

For each model m in the bucket: Do c times: (where 'c' is some constant) Randomly divide the training dataset into two datasets: A, and B. Train m with A Test m with B Select the model that obtains the highest average score

Cross-Validation Selection can be summed up as: "try them all with the training set, and pick the one that works best".[19]

Gating is a generalization of Cross-Validation Selection. It involves training another learning model to decide which of the models in the bucket is best-suited to solve the problem. Often, a perceptron is used for the gating model. It can be used to pick the "best" model, or it can be used to give a linear weight to the predictions from each model in the bucket.

When a bucket of models is used with a large set of problems, it may be desirable to avoid training some of the models that take a long time to train. Landmark learning is a meta-learning approach that seeks to solve this problem. It involves training only the fast (but imprecise) algorithms in the bucket, and then using the performance of these algorithms to help determine which slow (but accurate) algorithm is most likely to do best.[20]

11.3.7 Stacking

Stacking (sometimes called *stacked generalization*) involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. If an arbitrary combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described in this article, although in practice, a single-layer logistic regression model is often used as the combiner.

Stacking typically yields performance better than any single one of the trained models.[21] It has been successfully used on both supervised learning tasks (regression,[22] classification and distance learning [23]) and unsupervised learning (density estimation).[24] It has also been used to estimate bagging's error rate.[3][25] It has been reported to out-perform Bayesian model-averaging.[26] The two top-performers in the Netflix competition utilized *blending*, which may be considered to be a form of stacking.[27]

11.4 References

[1] Opitz, D.; Maclin, R. (1999). "Popular ensemble methods: An empirical study". *Journal of Artificial Intelligence Research* **11**: 169–198. doi:10.1613/jair.614.

[2] Polikar, R. (2006). "Ensemble based systems in decision making". *IEEE Circuits and Systems Magazine* **6** (3): 21–45. doi:10.1109/MCAS.2006.1688199.

[3] Rokach, L. (2010). "Ensemble-based classifiers". *Artificial Intelligence Review* **33** (1-2): 1–39. doi:10.1007/s10462-009-9124-7.

[4] Kuncheva, L. and Whitaker, C., Measures of diversity in classifier ensembles, *Machine Learning*, 51, pp. 181-207, 2003

[5] Sollich, P. and Krogh, A., *Learning with ensembles: How overfitting can be useful*, Advances in Neural Information Processing Systems, volume 8, pp. 190-196, 1996.

[6] Brown, G. and Wyatt, J. and Harris, R. and Yao, X., Diversity creation methods: a survey and categorisation., *Information Fusion*, 6(1), pp.5-20, 2005.

[7] *Accuracy and Diversity in Ensembles of Text Categorisers*. J. J. García Adeva, Ulises Cerviño, and R. Calvo, CLEI Journal, Vol. 8, No. 2, pp. 1 - 12, December 2005.

[8] Ho, T., Random Decision Forests, *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 278-282, 1995.

[9] Gashler, M. and Giraud-Carrier, C. and Martinez, T., *Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous*, The Seventh International Conference on Machine Learning and Applications, 2008, pp. 900-905., DOI 10.1109/ICMLA.2008.154

[10] Tom M. Mitchell, *Machine Learning*, 1997, pp. 175

[11] Breiman, L., Bagging Predictors, *Machine Learning*, 24(2), pp.123-140, 1996.

[12] Sahu, A., Runger, G., Apley, D., Image denoising with a multi-phase kernel principal component approach and an ensemble version, IEEE Applied Imagery Pattern Recognition Workshop, pp.1-7, 2011.

[13] Shinde, Amit, Anshuman Sahu, Daniel Apley, and George Runger. "Preimages for Variation Patterns from Kernel PCA and Bagging." IIE Transactions, Vol. 46, Iss. 5, 2014.

[14] Hoeting, J. A.; Madigan, D.; Raftery, A. E.; Volinsky, C. T. (1999). "Bayesian Model Averaging: A Tutorial". *Statistical Science* **14** (4): 382–401. doi:10.2307/2676803. JSTOR 2676803.

[15] David Haussler, Michael Kearns, and Robert E. Schapire. *Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension*. Machine Learning, 14:83–113, 1994

[16] Domingos, Pedro (2000). *Bayesian averaging of classifiers and the overfitting problem* (PDF). Proceedings of the 17th International Conference on Machine Learning (ICML). pp. 223—230.

[17] Minka, Thomas (2002), *Bayesian model averaging is not model combination* (PDF)

[18] Monteith, Kristine; Carroll, James; Seppi, Kevin; Martinez, Tony. (2011). *Turning Bayesian Model Averaging into Bayesian Model Combination* (PDF). Proceedings of the International Joint Conference on Neural Networks IJCNN'11. pp. 2657–2663.

[19] Bernard Zenko, *Is Combining Classifiers Better than Selecting the Best One*, Machine Learning, 2004, pp. 255-—273

[20] Bensusan, Hilan and Giraud-Carrier, Christophe G., Discovering Task Neighbourhoods Through Landmark Learning Performances, PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, Springer-Verlag, 2000, pages 325-—330

[21] Wolpert, D., *Stacked Generalization.*, Neural Networks, 5(2), pp. 241-259., 1992

[22] Breiman, L., *Stacked Regression*, Machine Learning, 24, 1996

[23] M. Ozay and F. T. Yarman Vural, *A New Fuzzy Stacked Generalization Technique and Analysis of its Performance*, 2012, arXiv:1204.0171

[24] Smyth, P. and Wolpert, D. H., *Linearly Combining Density Estimators via Stacking*, Machine Learning Journal, 36, 59-83, 1999

[25] Wolpert, D.H., and Macready, W.G., *An Efficient Method to Estimate Bagging's Generalization Error*, Machine Learning Journal, 35, 41-55, 1999

[26] Clarke, B., *Bayes model averaging and stacking when model approximation error cannot be ignored*, Journal of Machine Learning Research, pp 683-712, 2003

[27] Sill, J. and Takacs, G. and Mackey L. and Lin D., *Feature-Weighted Linear Stacking*, 2009, arXiv:0911.0460

11.5 Further reading

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

11.6 External links

- Ensemble learning at Scholarpedia, curated by Robi Polikar.
- The Waffles (machine learning) toolkit contains implementations of Bagging, Boosting, Bayesian Model Averaging, Bayesian Model Combination, Bucket-of-models, and other ensemble techniques

Chapter 12

k-nearest neighbors algorithm

In pattern recognition, the ***k*-Nearest Neighbors algorithm** (or ***k*-NN** for short) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

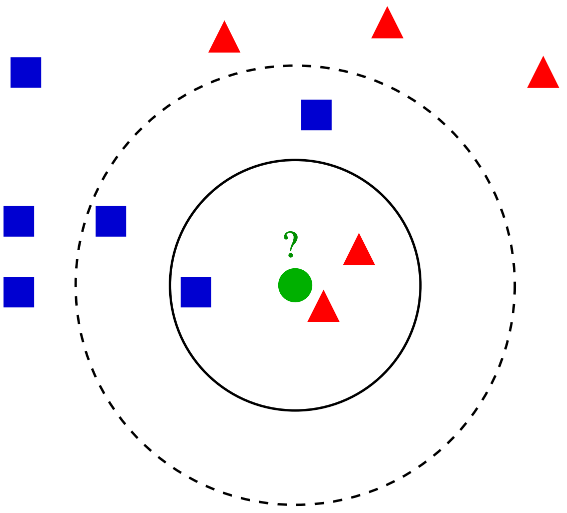
k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A shortcoming of the *k*-NN algorithm is that it is sensitive to the local structure of the data. The algorithm has nothing to do with and is not to be confused with *k*-means, another popular machine learning technique.

12.1 Algorithm



Example of k*-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If* k = 3 *(solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If* k = 5 *(dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).*

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or Hamming distance). In the context of gene expression microarray data, for example, *k*-NN has also been employed with correlation coefficients such as Pearson and Spearman.[3] Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number.[4] One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. K-NN can then be applied to the SOM.

12.2 Parameter selection

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification,[5] but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques (see hyperparameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when $k = 1$) is called the nearest neighbor algorithm.

The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling.[6] Another popular approach is to scale features by the mutual information of the training data with the training classes.

In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal k in this setting is via bootstrap method.[7]

12.3 Properties

k-NN is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel.[8] [9]

The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets. Using an appropriate nearest neighbor search algorithm makes k-NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed.

k-NN has some strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).[10] k-NN is guaranteed to approach the Bayes error rate for some value of k (where k increases as a function of the number of data points). Various improvements to k-NN are possible by using proximity graphs.[11]

12.4 Metric Learning

The K-nearest neighbor classification performance can often be significantly improved through (supervised) metric learning. Popular algorithms are Neighbourhood components analysis and Large margin nearest neighbor. Supervised metric learning algorithms use the label information to learn a new metric or pseudo-metric.

12.5 Feature extraction

When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying k-NN algorithm on the transformed data in feature space.

An example of a typical computer vision computation pipeline for face recognition using k-NN including feature extraction and dimension reduction pre-processing steps (usually implemented with OpenCV):

1. Haar face detection
2. Mean-shift tracking analysis
3. PCA or Fisher LDA projection into feature space, followed by k-NN classification

12.6 Dimension reduction

For high-dimensional data (e.g., with number of dimensions more than 10) dimension reduction is usually performed prior to applying the k-NN algorithm in order to avoid the effects of the curse of dimensionality. [12]

The curse of dimensionality in the k-NN context basically means that Euclidean distance is unhelpful in high dimensions because all vectors are almost equidistant to the search query vector (imagine multiple points lying more or less on a circle with the query point at the center; the distance from the query to all data points in the search space is almost the same).

Feature extraction and dimension reduction can be combined in one step using principal component analysis (PCA), linear discriminant analysis (LDA), or canonical correlation analysis (CCA) techniques as a pre-processing step, followed by clustering by k-NN on feature vectors in reduced-dimension space. In machine learning this process is also called low-dimensional embedding.[13]

For very-high-dimensional datasets (e.g. when performing a similarity search on live video streams, DNA data or high-dimensional time series) running a fast **approximate** k-NN search using locality sensitive hashing, "random projections",[14] "sketches" [15] or other high-dimensional similarity search techniques from VLDB toolbox might be the only feasible option.

12.7 Decision boundary

Nearest neighbor rules in effect implicitly compute the decision boundary. It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity.[16]

12.8 Data reduction

Data reduction is one of the most important problems for work with huge data sets. Usually, only some of the data points are needed for accurate classification. Those data are called the *prototypes* and can be found as follows:

1. Select the *class-outliers*, that is, training data that are classified incorrectly by k-NN (for a given k)

2. Separate the rest of the data into two sets: (i) the prototypes that are used for the classification decisions and (ii) the *absorbed points* that can be correctly classified by k-NN using prototypes. The absorbed points can then be removed from the training set.

12.8.1 Selection of class-outliers

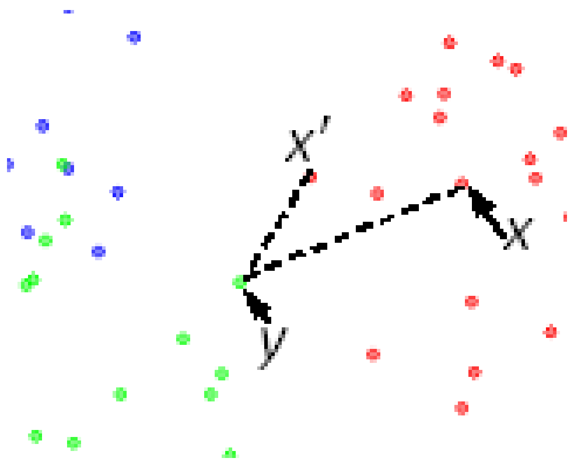
A training example surrounded by examples of other classes is called a class outlier. Causes of class outliers include:

- random error
- insufficient training examples of this class (an isolated example appears instead of a cluster)
- missing important features (the classes are separated in other dimensions which we do not know)
- too many training examples of other classes (unbalanced classes) that create a "hostile" background for the given small class

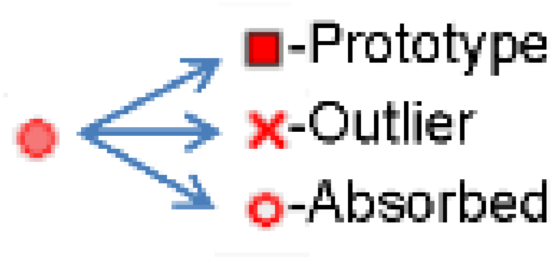
Class outliers with k-NN produce noise. They can be detected and separated for future analysis. Given two natural numbers, $k>r>0$, a training example is called a (k,r)NN class-outlier if its k nearest neighbors include more than r examples of other classes.

12.8.2 CNN for data reduction

Condensed nearest neighbor (CNN, the *Hart algorithm*) is an algorithm designed to reduce the data set for k-NN classification.[17] It selects the set of prototypes U from the training data, such that 1NN with U can classify the examples almost as accurately as 1NN does with the whole data set.



Calculation of the border ratio.



Three types of points: prototypes, class-outliers, and absorbed points.

Given a training set X, CNN works iteratively:

1. Scan all elements of X, looking for an element x whose nearest prototype from U has a different label than x.
2. Remove x from X and add it to U
3. Repeat the scan until no more prototypes are added to U.

Use U instead of X for classification. The examples that are not prototypes are called "absorbed" points.

It is efficient to scan the training examples in order of decreasing border ratio.[18] The border ratio of a training example x is defined as

$$a(x) = \|x'-y\| / \|x-y\|$$

where $\|x-y\|$ is the distance to the closest example y having a different color than x, and $\|x'-y\|$ is the distance from y to its closest example x' with the same label as x.

The border ratio is in the interval [0,1] because $\|x'-y\|$ never exceeds $\|x-y\|$. This ordering gives preference to the borders of the classes for inclusion in the set of prototypes$U$. A point of a different label than x is called external to x. The calculation of the border ratio is illustrated by the figure on the right. The data points are labeled by colors: the initial point is x and its label is red. External points are blue and green. The closest to x external point is y. The closest to y red point is x' . The border ratio $a(x)=\|x'-y\|/\|x-y\|$ is the attribute of the initial point x.

Below is an illustration of CNN in a series of figures. There are three classes (red, green and blue). Fig. 1: initially there are 60 points in each class. Fig. 2 shows the 1NN classification map: each pixel is classified by 1NN using all the data. Fig. 3 shows the 5NN classification map. White areas correspond to the unclassified regions, where 5NN voting is tied (for example, if there are two green, two red and one blue points among 5 nearest neighbors). Fig. 4 shows the reduced data set. The crosses are the class-outliers selected by the (3,2)NN rule (all the three nearest neighbors of these instances belong to other classes); the squares are the prototypes, and the empty circles are the absorbed points. The left bottom corner shows the numbers of the class-outliers, prototypes and absorbed points for all three classes. The number of prototypes varies from 15% to 20% for different classes in this example. Fig. 5 shows that the 1NN classification map with the prototypes is very similar to that with the initial data set. The figures were produced using the Mirkes applet.[18]

- CNN model reduction for k-NN classifiers
- Fig. 1. The dataset.
- Fig. 2. The 1NN classification map.
- Fig. 3. The 5NN classification map.
- Fig. 4. The CNN reduced dataset.
- Fig. 5. The 1NN classification map based on the CNN extracted prototypes.

12.9 *k*-NN regression

In k-NN regression, the k-NN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the k nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

1. Compute the Euclidean or Mahalanobis distance from the query example to the labeled examples.
2. Order the labeled examples by increasing distance.
3. Find a heuristically optimal number k of nearest neighbors, based on RMSE. This is done using cross validation.
4. Calculate an inverse distance weighted average with the k-nearest multivariate neighbors.

12.10 Validation of results

A confusion matrix or "matching matrix" is often used as a tool to validate the accuracy of k-NN classification. More robust statistical methods such as likelihood-ratio test can also be applied.

12.11 See also

- Instance-based learning
- Nearest neighbor search
- Statistical classification
- Cluster analysis
- Data mining
- Nearest centroid classifier
- Pattern recognition
- Curse of dimensionality
- Dimension reduction
- Principal Component Analysis
- Locality Sensitive Hashing
- MinHash
- Cluster hypothesis
- Closest pair of points problem

12.12 References

[1] Altman, N. S. (1992). “An introduction to kernel and nearest-neighbor nonparametric regression”. *The American Statistician* **46** (3): 175–185. doi:10.1080/00031305.1992.10475879.

[2] This scheme is a generalization of linear interpolation.

[3] Jaskowiak, P. A.; Campello, R. J. G. B. “Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data”. *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.208.993". Brazilian Symposium on Bioinformatics (BSB 2011). pp. 1–8. Retrieved 16 October 2014.*

[4] D. Coomans; D.L. Massart (1982). “Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-Nearest neighbour classification by using alternative voting rules”. *Analytica Chimica Acta* **136**: 15–27. doi:10.1016/S0003-2670(01)95359-0.

[5] Everitt, B. S., Landau, S., Leese, M. and Stahl, D. (2011) Miscellaneous Clustering Methods, in Cluster Analysis, 5th Edition, John Wiley & Sons, Ltd, Chichester, UK.

[6] Nigsch F, Bender A, van Buuren B, Tissen J, Nigsch E, Mitchell JB (2006). “Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization”. *Journal of Chemical Information and Modeling* **46** (6): 2412–2422. doi:10.1021/ci060149f. PMID 17125183.

[7] Hall P, Park BU, Samworth RJ (2008). “Choice of neighbor order in nearest-neighbor classification”. *Annals of Statistics* **36** (5): 2135–2152. doi:10.1214/07-AOS537.

[8] D. G. Terrell; D. W. Scott (1992). “Variable kernel density estimation”. *Annals of Statistics* **20** (3): 1236–1265. doi:10.1214/aos/1176348768.

[9] Mills, Peter. “Efficient statistical classification of satellite measurements”. *International Journal of Remote Sensing*.

[10] Cover TM, Hart PE (1967). “Nearest neighbor pattern classification”. *IEEE Transactions on Information Theory* **13** (1): 21–27. doi:10.1109/TIT.1967.1053964.

[11] Toussaint GT (April 2005). “Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining”. *International Journal of Computational Geometry and Applications* **15** (2): 101–150. doi:10.1142/S0218195905001622.

[12] Beyer, Kevin, et al.. 'When is “nearest neighbor” meaningful?. Database Theory—ICDT’99, 217-235|year 1999

[13] Shaw, Blake, and Tony Jebara. 'Structure preserving embedding. Proceedings of the 26th Annual International Conference on Machine Learning. ACM,2009

[14] Bingham, Ella, and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM | year 2001

[15] Shasha, D High Performance Discovery in Time Series.Berlin: Springer, 2004, ISBN 0-387-00857-8

[16] Bremner D, Demaine E, Erickson J, Iacono J, Langerman S, Morin P, Toussaint G (2005). “Output-sensitive algorithms for computing nearest-neighbor decision boundaries”. *Discrete and Computational Geometry* **33** (4): 593–604. doi:10.1007/s00454-004-1152-0.

[17] P. E. Hart, The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory 18 (1968) 515–516. doi: 10.1109/TIT.1968.1054155

[18] E. M. Mirkes, KNN and Potential Energy: applet. University of Leicester, 2011.

12.13 Further reading

- When Is “Nearest Neighbor” Meaningful?
- Belur V. Dasarathy, ed. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. ISBN 0-8186-8930-7.
- Shakhnarovish, Darrell, and Indyk, ed. (2005). *Nearest-Neighbor Methods in Learning and Vision*. MIT Press. ISBN 0-262-19547-X.
- Mäkelä H Pekkarinen A (2004-07-26). “Estimation of forest stand volumes by Landsat TM imagery and stand-level field-inventory data”. *Forest Ecology and Management* **196** (2–3): 245–255. doi:10.1016/j.foreco.2004.02.049.
- Fast k nearest neighbor search using GPU. In Proceedings of the CVPR Workshop on Computer Vision on GPU, Anchorage, Alaska, USA, June 2008. V. Garcia and E. Debreuve and M. Barlaud.
- Scholarpedia article on *k*-NN
- google-all-pairs-similarity-search

12.14 Text and image sources, contributors, and licenses

12.14.1 Text

ankan~enwiki, Kevin Baas, Rossami, Hike395, Jitse Niesen, Andrewman327, Taxman, Donarreiskoffer, Robbot, Jaredwf, Benwing, Gak, ZimZalaBim, Yelyos, Babbage, Henrygb, Jcoleff, Wile E. Heresiarch, Giftlite, BenFrantzDale, Fleminra, Alison, Duncharris, Jason Quinn, Ato, Utcursch, Pgan002, MarkSweep, Piotrus, Wurblzap~enwiki, Icairns, Urhixidur, Natrij, Discospinster, Rich Farmbrough, Pak21, Paul August, Bender235, Violetriga, Elwikipedista~enwiki, Gauge, MisterSheik, Spoon!, Perfecto, O18, Davidswelt, R. S. Shaw, Tobacman, Arcadian, NickSchweitzer, 99of9, Crust, Landroni, Storm Rider, Musiphil, Arthena, ABCD, Kotasik, Avenue, Snowolf, LFaraone, Forderud, Drummond, Oleg Alexandrov, Abanima, Tappancsa, Mindmatrix, BlaiseFEgan, Joerg Kurt Wegner, Lacurus~enwiki, Graham87, Qwertyus, Rjwilmsi, Vegaswikian, Matt Deres, TeaDrinker, Chobot, Manscher, FrankTobia, Wavelength, RussBot, Gaius Cornelius, Bug42, Afelton, Thiseye, Cruise, Moe Epsilon, Voidxor, Dggoldst, Arch o median, Arthur Rubin, Drallim, Anarch21, SolarMcPanel, SmackBot, NickyMcLean, Quazar777, Prodego, InverseHypercube, Jtneill, DanielPenfield, Evanreyes, Commander Keane bot, Ohnoitsjamie, Hraefen, Afa86, Markush, Amatulic, Feinstein, Oli Filth, John Reaves, Berland, Wolf87, Cybercobra, Semanticprecision, G716, Unco, Theblackgecko, Lambiam, Jonas August, Vjeet a, Nijdam, Beetstra, Emurph, Hu12, Pjrm, LAlawMedMBA, JoeBot, Chris53516, AlainD, Jsorens, Tawkerbot2, CmdrObot, JRavn, CBM, Anakata, Chrike, Harej bot, Thomasmeeks, Neelix, Cassmus, Bumbulski, MaxEnt, 137 0, Pedrolapinto, Mmmooonnnsssttteeerrr, Farshidforouz~enwiki, FrancoGG, Talgalili, Thijs!bot, Epbr123, Tolstoy the Cat, Jfaller, Whoooooknows, Natalie Erin, Woollymammoth, Mack2, JAnDbot, MER-C, Jeff560, Ph.eyes, Hectorlamadrid, Magioladitis, Tripbeetle, MastCell, Albmont, Baccyak4H, Ddr~enwiki, David Eppstein, Joostw, Apal~enwiki, Yonaa, R'n'B, Noyder, Charlesmartin14, Kawautar, Mbhiii, J.delanoy, Scythe of Death, Salih, TomyDuby, Jaxha, HyDeckar, Jewzip, MrPaul84, Copsi, Llorenzi, VolkovBot, Smarty07, Muzzamo, Mfreund~enwiki, Jsd115, Zhenqinli, Greswik, P1h3r1e3d13, Ricardo MC, Jhedengren, Petergans, Karthik Sarma, Rlendog, Zsniew, Paolo.dL, OKBot, Water and Land, Melcombe, Gpap.gpap, Tanvir Ahmmed, ClueBot, HairyFotr, Cp111, Rhubbarb, Dromedario~enwiki, Alexbot, Ecov, Kaspar.jan, Tokorode~enwiki, Skbkekas, Stephen Milborrow, Diaa abdelmoneim, Qwfp, Bigoperm, Sunsetsky, XLinkBot, Tofallis, W82~enwiki, Tayste, Addbot, RPHv, Fgnievinski, Doronp, MrOllie, Download, Forich, Zorrobot, Ettrig, Luckas-bot, Yobot, Sked123, It's Been Emotional, AnomieBOT, Rubinbot, IRP, Materialscientist, HanPritcher, Citation bot, Lixiaoxu, Sketchmoose, Flavio Guitian, Gtfjbl, Istrill, Mstangeland, Aa77zz, Imran.fanaswala, Fstonedahl, PhysicsJoe, Nickruiz, FrescoBot, X7q, Citation bot 1, AstaBOTh15, Boxplot, Pinethicket, Elockid, Kiefer.Wolfowitz, Rdecker02, Jonesey95, Stpasha, Oldrrb, Wotnow, Duoduoduo, PAC2, Wombathammer, Diannaa, RjwilmsiBot, Elitropia, John of Reading, Sugarfoot1001, Julienbarlan, Wikieconometrician, Bkearb, NGPriest, BartlebytheScrivener, Chewings72, Esaintpierre, Manipande, ClueBot NG, Mathstat, Frietjes, BlueScreenD, Helpful Pixie Bot, Grandwgy, Daonng, Mark Arsten, MyWikiNik, ChrisGualtieri, Illia Connell, Dansbecker, Dexbot, Hkoslik, Sa publishers, Ossifragus, Bha100710, Drvikas74, Melonkelon, Asif usa, Pandadai, Bryanrutherford0, Tertius51, Logan.dunbar, Monkbot, Bob nau, Moorshed, Velvel2, Whatfoxsays, 18trevor3695, Split97 and Anonymous: 372

- **Logistic regression** *Source:* http://en.wikipedia.org/wiki/Logistic%20regression?oldid=661581191 *Contributors:* Twanvl, Michael Hardy, Tomi, Den fjättrade ankan~enwiki, Benwing, Gak, Giftlite, BrendanH, YapaTi~enwiki, Dfrankow, Pgan002, Bolo1729, Qef, Rich Farmbrough, Mani1, Bender235, Mdf, O18, NickSchweitzer, CarrKnight, Kierano, Arthena, Velella, Oleg Alexandrov, LOL, BlaiseFEgan, Qwertyus, Rjwilmsi, Ground Zero, Sderose, Shaggyjacobs, YurikBot, Wavelength, Cancan101, Johndburger, Rodolfo Hermans, Jtneill, D nath1, Lassefolkersen, Aldaron, G716, Esrever, RomanSpa, Nutcracker, Cbuckley, Ionocube, Kenkleinman, Markjoseph125, David s graff, Jjoseph, Olberd, Requestion, Future Perfect at Sunrise, Kallerdis, Neoforma, LachlanA, Tomixdf, Mack2, JAnDbot, Every Creek Counts, Sanchom, Owenozier, Magioladitis, Baccyak4H, Nszilard, Mbhiii, Lilac Soul, Kpmiyapuram, Djjrjr, Ypetrachenko, Dvdpwiki, Squids and Chips, Ktalon, TXiKiBoT, Harrelfe, Antaltamas, Aaport, Jamelan, Synthebot, Dvandeventer, Anameofmyveryown, Prakash Nadkarni, BAnstedt, Junling, AlanUS, Melcombe, Sphilbrick, Denisarona, Alpapad, Statone, SchreiberBike, Cmelan, Aprock, Qwfp, XLinkBot, WikHead, Tayste, Addbot, Kurttg~enwiki, New Image Uploader 929, Luckas-bot, Yobot, Secondsminutes, AnomieBOT, Ciphers, Materialscientist, Xqbot, Gtfjbl, FrescoBot, X7q, Orubt, Chenopodiaceous, Albertzeyer, Duoduoduo, Diannaa, RjwilmsiBot, Grumpfel, Strano.m, Mudx77, EmausBot, RMGunton, Alexey.kudinkin, Zephyrus Tavvier, Kchowdhary, Sigma0 1, DASHBotAV, Vldscore, Kjalarr, Mhsmith0, Timflutre, Helpful Pixie Bot, Ngocminh.oss, BG19bot, Martha6981, DPL bot, BattyBot, Guziran99, Eflatmajor7th, AndrewSmithQueen's, Jey42, JTravelman, SFK2, Yongli Han, Merespiz, Tentinator, Pkalczynski, Yoboho, Hayes.rachel, E8xE8, Tertius51, Nyashinski, ThuyNgocTran, Monkbot, SantiLak, Kamerondeckerharris, Srp54, P.thesling, Alexander Craig Russell, Екатерина Конь, Velvel2, Anuragsodhi, Hughkf, PushTheButton108, LockemyCock and Anonymous: 181
- **Support vector machine** *Source:* http://en.wikipedia.org/wiki/Support%20vector%20machine?oldid=660729075 *Contributors:* The Anome, Gareth Owen, Enchanter, Ryguasu, Edward, Michael Hardy, Dmd~enwiki, Oliver Pereira, Kku, Zeno Gantner, Mark Foskey, Jll, Jimregan, Hike395, Barak~enwiki, Dysprosia, Pheon, Kgajos, Mpost89, Benwing, Altenmann, Wile E. Heresiarch, Tobias Bergemann, Giftlite, Sepreece, BenFrantzDale, Fropuff, Déjà Vu, Dfrankow, Neilc, Pgan002, Gene s, Urhixidur, Rich Farmbrough, Mathiasl26, Nowozin, Ylai, Andrejj, MisterSheik, Lycurgus, Alex Kosorukoff, Aaronbrick, Cyc~enwiki, Rajah, Diego Moya, Pzoot, Nvrmnd, Gortu, Boyd Steere, Alai, Stuartyeates, The Belgain, Ralf Mikut, Ruud Koot, Waldir, Joerg Kurt Wegner, Qwertyus, Michal.burda, Brighterorange, Mathbot, Vsion, Gurch, Sderose, Diza, Tedder, Chobot, Adoniscik, YurikBot, Ste1n, CambridgeBayWeather, Rsrikanth05, Pseudomonas, Seb35, Korny O'Near, Gareth Jones, Karipuf, Retardo, Bota47, SamuelRiv, Tribaal, Palmin, Thnidu, Digfarenough, CWenger, Sharat sc, Mebden, Amit man, Lordspaz, Otheus, SmackBot, RDBury, Jwmillerusa, Moeron, InverseHypercube, Vardhanw, Ealdent, Golwengaud, CommodiCast, Mcduff, Eskimbot, Vermorel, Mcld, Riedl, Srchvrs, Avb, MattOates, Memming, Mitar, Sadeq, Jonas August, Rijkbenik, Jrouquie, Adilraja, Dicklyon, Mattsachs, Hu12, Ojan, JoeBot, Atreys, Tawkerbot2, Lavaka, Harold f, Owen214, CmdrObot, FunPika, Ezrakilty, Innohead, Bumbulski, Anthonyhcole, Farzaneh, Carstensen, Gnfnrf, Thijs!bot, Drowne, Trevyn, Dkemper, Prolog, AnAj, Dougher, Wootery, Sanchom, Shashank4, BrotherE, Coffee2theorems, Tremilux, Qjqflash3, Americanhero, A3nm, Parunach, Jacktance~enwiki, Ledona delano, Andreas Mueller, Mschel, FelipeVargasRigo, David Callan, Freeboson, Nickvence, Polusfx, Singularitarian, Senu, Salih, JamesMayfield, Pradeepgk, Supportvector, STBotD, RJASE1, Satyr9, Svm slave, TXiKiBoT, Sanatio, Carcinus, Majeru, Semifinalist, Canavalia, Simonstr, Domination989, Kerveros 99, Quentin Pradet, AaronArvey, Caltas, Behind The Wall Of Sleep, Udirock, Savedthat, JeanSenellart, Melcombe, Tiny plastic Grey Knight, Wawe1811, Martarius, Sfan00 IMG, Grantbow, Peter.kese, Alexbot, Pot, DeltaQuad, MagnusPI, Chaosdruid, Djgulp3~enwiki, Qwfp, Asymptosis, Sunsetsky, Addbot, DOI bot, AndrewHZ, MrOllie, Lightbot, Peni, Legobot, Yobot, Legobot II, AnomieBOT, Erel Segal, Jim1138, Materialscientist, Citation bot, Salbang, Tripshall, Twri, LilHelpa, Megatang, Chuanren, RibotBOT, Hamidhaji, WaysToEscape, Alisaleh88, HB28205, Romainbrasselet, X7q, Elvenhack, Wikisteve316, Citation bot 1, RedBot, SpaceFlight89, Classifier1234, Zadroznyelkan, ACKiwi, RjwilmsiBot, Zhejiangustc, J36miles, EmausBot, Alfaisanomega, Nacopt, NikolasE~enwiki, Datakid1, Msayag, Stheodor, Colmenares jb, Manyu aditya, ZéroBot, Khaled.Boukharouba, Chire, Amit159, Tolly4bolly, Pintaio, Tahir512, ChengHsuanLi, Ecc81, DaChazTech, Aaaaa bbbbb1355, Liuyipei, Arafat.sultan, ClueBot NG, Kkddkkdd, MchLrn, Helpful Pixie Bot, Alisneaky, Elferdo, BG19bot, SciCompTeacher, Xlicolts613, Boyander, Bodormenta, Illia Connell, Elmackev, Mohraz2, Deepakiitmandi, Adrienbrunetwiki, Velvel2, CurtisZeng, Cjqed, ZhangJiaqiPKU and Anonymous: 340
- **Naive Bayes classifier** *Source:* http://en.wikipedia.org/wiki/Naive%20Bayes%20classifier?oldid=653270025 *Contributors:* The Anome, Awaterl, Olivier, Michael Hardy, Bewildebeast, Zeno Gantner, Karada, Cyp, Den fjättrade ankan~enwiki, Hike395, Njoshi3, WhisperToMe, Toreau, Phil Boswell, RedWolf, Bkell, Wile E. Heresiarch, Giftlite, Akella, JimD, Bovlb, Macrakis, Neilc, Pgan002, MarkSweep,

Gene s, Cagri, Anirvan, Trevor MacInnis, Thorwald, Splatty, Rich Farmbrough, Violetriga, Peterjoel, Smalljim, John Vandenberg, BlueNovember, Jason Davies, Caesura, Oleg Alexandrov, KKramer~enwiki, Btyner, Mandarax, Qwertyus, Rjwilmsi, Hgkamath, Johnnyw, Mathbot, Intgr, Sderose, YurikBot, Wavelength, PiAndWhippedCream, Cancan101, Bovineone, Arichnad, Karipuf, BOT-Superzerocool, Evryman, Johndburger, Mebden, XAVeRY, SmackBot, InverseHypercube, CommodiCast, Stimpy, ToddDeLuca, Gilliam, NickGarvey, Chris the speller, OrangeDog, PerVognsen, Can't sleep, clown will eat me, Memming, Mitar, Neshatian, Jklin, Ringger, WMod-NS, Tobym, Shorespirit, Mat1971, Dstanfor, Arauzo, Dantiston, Sytelus, Vera Rita~enwiki, Dkemper, Prolog, Ninjakannon, Jrennie, MSBOT, Coffee2theorems, Tremilux, Saurabh911, Robotman1974, David Eppstein, User A1, HebrewHammerTime, AllenDowney, Troos, AntiSpamBot, Newtman, STBotD, Mike V, RJASE1, VolkovBot, Maghnus, Anna Lincoln, Mbusux, Anders gorm, EverGreg, Fcady2007, Jojalozzo, Ddxc, Dchwalisz, AlanUS, Melcombe, Headlessplatter, Kotsiantis, Justin W Smith, Motmahp, Calimo, Dianegarey, Doobliebop, Alousybum, Sunsetsky, XLinkBot, Herlocker, Addbot, RPHv, Tsunanet, MrOllie, LaaknorBot, Yobot, TaBOT-zerem, Twexcom, AnomieBOT, Rubinbot, Smk65536, The Almighty Bob, Cantons-de-l'Est, گستهم, FrescoBot, X7q, Proffviktor, Svourdroculed, Rickyphyllis, Jonesey95, Geoffrey I Webb, Classifier1234, Mwojnars, Wingiii, Helwr, EmausBot, Orphan Wiki, Tommy2010, GarouDan, Joseagonzalez, ClueBot NG, Hofmic, NilsHaldenwang, Luoli2000, BG19bot, MusikAnimal, Chafe66, Kavishwar.wagholikar, Geduowenyang, Hipponix, Fcbarbi, Librawill, ChrisGualtieri, XMU zhangy, Alialamifard, CorvetteC6RVip, Jamesmcmahon0, Tonytonov, Jmagasin, ScienceRandomness, Qingyuanxingsi, Micpalmia, Sofia Koutsouveli, Yuchsiao, Mvdyck, Don neufeld, YoniSmolin, Rapanshi, Ananth.sankar.1963, Hmerzic and Anonymous: 181

- **Decision tree learning** *Source:* http://en.wikipedia.org/wiki/Decision%20tree%20learning?oldid=649067236 *Contributors:* Michael Hardy, TheEternalVortex, Greenrd, Maximus Rex, Tschild, Populus, Khalid hassani, Pgan002, Raand, Dan aka jack, Andreas Kaufmann, Discospinster, John Vandenberg, Giraffedata, Mdd, Equinoxe, VeXocide, Bushytails, GregorB, Qwertyus, Rjwilmsi, Gmelli, Salix alba, Vonkje, SmackBot, Diegotorquemada, Mcld, Riedl, Zven, Mitar, Krexer, Kuru, Beetstra, Courcelles, Ceran, Pgr94, Yaris678, Talgalili, A3RO, Nobar, Martinkunev, Destynova, Jessicapierce, Dobi~enwiki, User A1, Jalaska13, A m sheldon, Xs2me, Polyextremophile, Semifinalist, Extabgrad, Dodabe~enwiki, Foxj, Stephen Milborrow, Dank, Pichpich, Addbot, AndrewHZ, MrOllie, Download, Peni, Yobot, TestEditBot, AnomieBOT, Royote, Citation bot, FrescoBot, Hobsonlane, X7q, Kelos omos1, Orchidbox, Citation bot 1, Boxplot, Thinking of England, Janez Demsar, Mwojnars, Gzorg, Wik-dt, Chad.burrus, Bethnim, ZéroBot, Chire, Liorrokach, Pxtreme75, ClueBot NG, Psorakis, Aristitleism, Widr, Helpful Pixie Bot, BG19bot, QualitycontrolUS, BendelacBOT, Mightyteja, Djplaner, CitationCleanerBot, A923812, Sboosali, Douglas H Fisher, Zhang1989cn, JYBot, Lizhengui, Declaration1776, Jey42, Mgibby5, Pimgd, SiraRaven, Slash1986, Monkbot, HossPatrol and Anonymous: 88
- **Artificial neural network** *Source:* http://en.wikipedia.org/wiki/Artificial%20neural%20network?oldid=662321093 *Contributors:* Magnus Manske, Ed Poor, Iwnbap, PierreAbbat, Youandme, Susano, Hfastedge, Mrwojo, Michael Hardy, Erik Zachte, Oliver Pereira, Bobby D. Bryant, Zeno Gantner, Parmentier~enwiki, Delirium, Pieter Suurmond, (, Alfio, 168..., Ellywa, Ronz, Snoyes, Den fjättrade ankan~enwiki, Cgs, Glenn, Cyan, Hike395, Hashar, Novum, Charles Matthews, Guaka, Timwi, Reddi, Andrewman327, Munford, Furrykef, Bevo, Fvw, Raul654, Nyxos, Unknown, Pakcw, Robbot, Chopchopwhitey, Bkell, Hadal, Wikibot, Diberri, Xanzzibar, Wile E. Heresiarch, Connelly, Giftlite, Rs2, Markus Krötzsch, Spazzm, Seabhcan, BenFrantzDale, Zigger, Everyking, Rpyle731, Wikiwikifast, Foobar, Edrex, Jabowery, Wildt~enwiki, Wmahan, Neilc, Quadell, Beland, Lylum, Gene s, Sbledsoe, Mozzerati, Karl-Henner, Jmeppley, Asbestos, Fintor, AAAAA, Splatty, Rich Farmbrough, Pak21, NeuronExMachina, Michal Jurosz, Pjacobi, Mecanismo, Zarutian, Dbachmann, Bender235, ZeroOne, Violetriga, Mavhc, One-dimensional Tangent, Gyll, Stephane.magnenat, Mysteronald, .:Ajvol:., Fotinakis, Nk, Tritium6, JesseHogan, Mdd, Passw0rd, Zachlipton, Alansohn, Jhertel, Anthony Appleyard, Denoir, Arthena, Fritz Saalfeld, Sp00n17, Rickyp, Hu, Tyrell turing, Cburnett, Notjim, Drbreznjev, Forderud, Oleg Alexandrov, Mogigoma, Madmardigan53, Justinlebar, Olethros, Ylem, Dr.U, Gengiskanhg, Male1979, Bar0n, Waldir, Eslip17, Yoghurt, Ashmoo, Graham87, Qwertyus, Imersion, Grammarbot, Rjwilmsi, Jeema, Venullian, SpNeo, Intgr, Predictor, Kri, BradBeattie, Plarroy, Windharp, Mehran.asadi, Commander Nemet, Wavelength, Borgx, IanManka, Rsrikanth05, Philopedia, Ritchy, David R. Ingham, Grafen, Nrets, Exir Kamalabadi, Deodar~enwiki, Mosquitopsu, Jpbowen, Dennis!, JulesH, Moe Epsilon, Supten, DeadEyeArrow, Eclipsed, SamuelRiv, Tribaal, Chase me ladies, I'm the Cavalry, CWenger, Donhalcon, Banus, Shepard, John Broughton, A13ean, SmackBot, PinstripeMonkey, McGeddon, CommodiCast, Jfmiller28, Stimpy, Commander Keane bot, Feshmania, ToddDeLuca, Diegotorquemada, Patrickdepinguin, KYN, Gilliam, Bluebot, Oli Filth, Gardoma, Complexica, Nossac, Hongooi, Pdtl, Izhikevich, Trifon Triantafillidis, SeanAhern, Neshatian, Vernedj, Dankonikolic, Rory096, Sina2, SS2005, Kuru, Plison, Lakinekaki, Bjankuloski06en~enwiki, IronGargoyle, WMod-NS, Dicklyon, Citicat, StanfordProgrammer, Ojan, Chi3x10, Aeternus, CapitalR, Atreys, George100, Gveret Tered, Devourer09, SkyWalker, CmdrObot, Leonoel, CBM, Mcstrother, MarsRover, CX, Arauzo, Peterdjones, Josephorourke, Kozuch, ClydeC, NotQuiteEXPComplete, Irigi, Mbell, Oldiowl, Tolstoy the Cat, Headbomb, Mitchell.E.Timin, Davidhorman, Sbandrews, KrakatoaKatie, QuiteUnusual, Prolog, AnAj, LinaMishima, Whenning, Hamaryns, Daytona2, JAnDbot, MER-C, Dcooper, Extropian314, Magioladitis, VoABot II, Amitant, Jimjamjak, SSZ, Robotman1974, David Eppstein, User A1, Martynas Patasius, Pmbhagat, JaGa, Tuhinsubhrakonar, SoyYo, Nikoladie~enwiki, R'n'B, Maproom, K.menin, Gill110951, Tarotcards, Plasticup, Margareta, Paskari, Jamesontai, Kiran uvpce, Jamiejoseph, Error9312, Jlaramee, Jeff G., A4bot, Singleheart, Ebbedc, Lordvolton, Ask123, CanOfWorms, Mundhenk, Wikiisawesome, M karamanov, Enkya, Blumenkraft, Twikir, Mikemoral, Oldag07, Smsarmad, Flyer22, Janopus, Bwieliczko, Dhatfield, F.j.gaze, Mark Lewis Epstein, S2000magician, PuercoPop, Martarius, ClueBot, Ignacio Javier Igjav, Ahyeek, The Thing That Should Not Be, Fadesga, Zybler, Midiangr, Epsilon60198, Thomas Tvileren, Wduch, Excirial, Three-quarter-ten, Skbkekas, Chaosdruid, Aprock, Qwfp, Jean-claude perez, Achler, XLinkBot, AgnosticPreachersKid, BodhisattvaBot, Stickee, Cmr08, Porphyro, Fippy Darkpaw, Addbot, DOI bot, AndrewHZ, Thomblake, Techjerry, Looie496, MrOllie, Transmobilator, Jarble, Yobot, Blm19732008, Nguyengiap84~enwiki, SparkOfCreation, AnomieBOT, DemocraticLuntz, Tryptofish, Trevithj, Jim1138, Durran65, MockDuck, JonathanWilliford, Materialscientist, Citation bot, Eumolpo, Twri, NFD9001, Isheden, J04n, Omnipaedista, Mark Schierbecker, RibotBOT, RoodyBeep, Gunjan verma81, FrescoBot, X7q, Ömer Cengiz Çelebi, Outback the koala, Citation bot 1, Tylor.Sampson, Calmer Waters, Skyerise, Trappist the monk, Krassotkin, Cjlim, Fox Wilson, The Strategist, LilyKitty, Eparo, בן גרשון, Jfmantis, Mehdiabbasi, VernoWhitney, Wiknn, BertSeghers, DASHBot, EmausBot, Nacopt, Dzkd, Racerx11, Japs 88, GoingBatty, RaoInWiki, Roposeidon, Epsiloner, Stheodor, Benlansdell, Radshashi, K6ka, D'oh!, Thisisentchris87, Aavindraa, Chire, Glosser.ca, IGeMiNix, Donner60, Yoshua.Bengio, Shinosin, Venkatarun95, ChuckNorrisPwnedYou, Petrb, ClueBot NG, Raghith, Robiminer, Snotbot, Tideflat, Frietjes, Gms3591, Ryansandersuk, Widr, MerlIwBot, Helpful Pixie Bot, Trepier, BG19bot, Thwien, Adams7, Rahil2000, Michaelmalak, Compfreak7, Kirananils, Altaïr, J.Davis314, Attleboro, Pratyya Ghosh, JoshuSasori, Ferrarisailor, Eugenecheung, Mtschida, ChrisGualtieri, Dave2k6inthemix, Whebzy, APerson, JurgenNL, Oritnk, Stevebillings, Djfrost711, Sa publishers, ?, Mark viking, Markus.harz, Deeper Learning, Vinchaud20, Soueumxm, Toritris, Evolution and evolvability, Sboddhu, Sharva029, Paheld, Putting things straight, Rosario Berganza, Monkbot, Buggiehuggie, Santoshwriter, Likerhayter, Joma.huguet, Bclark401, Rahulpratapsingh06, Donkeychee, Michaelwine, Xsantostill, Jorge Guerra Pires, Wfwhitney, Loïc Bourgois and Anonymous: 487
- **Ensemble learning** *Source:* http://en.wikipedia.org/wiki/Ensemble%20learning?oldid=661760864 *Contributors:* The Anome, Greenrd, Violetriga, 3mta3, Jheald, Qwertyus, Vegaswikian, Wavelength, Crasshopper, ToddDeLuca, Littenberg, Mickeyg13, Mandra Oleka, Magioladitis, Destynova, Salih, EverGreg, Headlessplatter, Kotsiantis, Calimo, Skbkekas, Cowwaw, Qwfp, Sameer0s, Addbot, AndrewHZ,

12.14.2 Images

12.14.3 Content license