Machine Learning General Concepts

see more at http://ml.memect.com

Contents

Chapter 1

Machine learning

For the journal, see Machine Learning (journal).

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

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

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

1.1 Overview

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

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

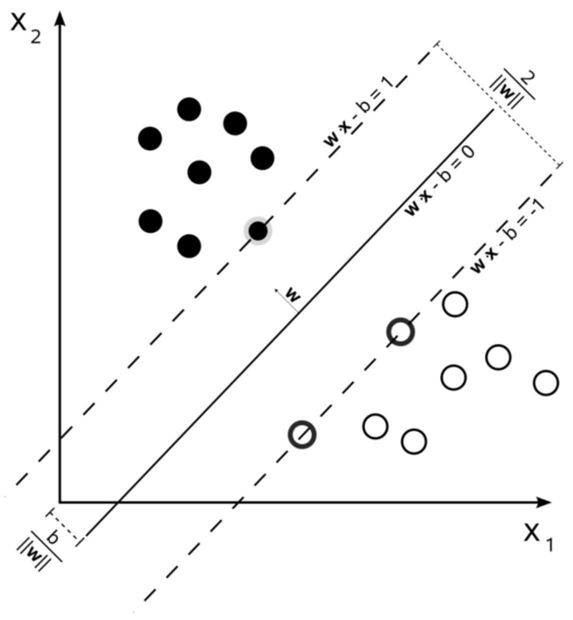
1.1.1 Types of problems and tasks

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

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

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

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



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

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

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

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

1.2 History and relationships to other fields

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

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

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

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

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

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

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

1.2.1 Relation to statistics

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

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

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

1.3 Theory

Main article: Computational learning theory

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

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

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

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

1.4 Approaches

Main article: List of machine learning algorithms

1.4.1 Decision tree learning

Main article: Decision tree learning

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

1.4.2 Association rule learning

Main article: Association rule learning

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

1.4.3 Artificial neural networks

Main article: Artificial neural network

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

1.4.4 Inductive logic programming

Main article: Inductive logic programming

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

1.4.5 Support vector machines

Main article: Support vector machines

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

1.4.6 Clustering

Main article: Cluster analysis

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

1.4.7 Bayesian networks

Main article: Bayesian network

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

1.4.8 Reinforcement learning

Main article: Reinforcement learning

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

1.4.9 Representation learning

Main article: Representation learning

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

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

1.4.10 Similarity and metric learning

Main article: Similarity learning

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

1.4.11 Sparse dictionary learning

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

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

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

1.4.12 Genetic algorithms

Main article: Genetic algorithm

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

1.5 Applications

Applications for machine learning include:

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

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

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

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

1.6 Software

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

1.6.1 Open-source software

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

1.6.2 Commercial software with open-source editions

- KNIME
- RapidMiner

1.6.3 Commercial software

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

1.7 Journals

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

1.8 Conferences

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

1.9 See also

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

1.10 References

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

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

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

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

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

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

[7] Phil Simon (March 18, 2013). *Too Big to Ignore: The Business Case for Big Data*. Wiley. p. 89. ISBN 978-1118638170.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[27]

[28]

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

1.11 Further reading

- Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012). *Foundations of Machine Learning*, The MIT Press. ISBN 9780262018258.
- Ian H. Witten and Eibe Frank (2011). *Data Mining: Practical machine learning tools and techniques* Morgan Kaufmann, 664pp., ISBN 978-0123748560.
- Sergios Theodoridis, Konstantinos Koutroumbas (2009) "Pattern Recognition", 4th Edition, Academic Press, ISBN 978-1-59749-272-0.
- Mierswa, Ingo and Wurst, Michael and Klinkenberg, Ralf and Scholz, Martin and Euler, Timm: *YALE: Rapid Prototyping for Complex Data Mining Tasks*, in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006.
- Bing Liu (2007), *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data.* Springer, ISBN 3-540-37881-2
- Toby Segaran (2007), *Programming Collective Intelligence*, O'Reilly, ISBN 0-596-52932-5
- Huang T.-M., Kecman V., Kopriva I. (2006), Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning, Springer-Verlag, Berlin, Heidelberg, 260 pp. 96 illus., Hardcover, ISBN 3-540-31681-7.
- Ethem Alpaydın (2004) *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, MIT Press, ISBN 0-262-01211-1
- MacKay, D.J.C. (2003). *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press. ISBN 0-521-64298-1.
- KECMAN Vojislav (2001), Learning and Soft Computing, Support Vector Machines, Neural Networks and Fuzzy Logic Models, The MIT Press, Cambridge, MA, 608 pp., 268 illus., ISBN 0-262-11255-8.
- Trevor Hastie, Robert Tibshirani and Jerome Friedman (2001). *The Elements of Statistical Learning*, Springer. ISBN 0-387-95284-5.
- Richard O. Duda, Peter E. Hart, David G. Stork (2001) *Pattern classification* (2nd edition), Wiley, New York, ISBN 0-471-05669-3.
- Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press. ISBN 0-19-853864-2.
- Ryszard S. Michalski, George Tecuci (1994), *Machine Learning: A Multistrategy Approach*, Volume IV, Morgan Kaufmann, ISBN 1-55860-251-8.
- Sholom Weiss and Casimir Kulikowski (1991). *Computer Systems That Learn*, Morgan Kaufmann. ISBN 1-55860-065-5.
- Yves Kodratoff, Ryszard S. Michalski (1990), *Machine Learning: An Artificial Intelligence Approach, Volume III*, Morgan Kaufmann, ISBN 1-55860-119-8.
- Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (1986), *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, ISBN 0-934613-00-1.
- Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (1983), *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, ISBN 0-935382-05-4.
- Vladimir Vapnik (1998). *Statistical Learning Theory*. Wiley-Interscience, ISBN 0-471-03003-1.
- Ray Solomonoff, *An Inductive Inference Machine*, IRE Convention Record, Section on Information Theory, Part 2, pp., 56-62, 1957.
- Ray Solomonoff, "An Inductive Inference Machine" A privately circulated report from the 1956 Dartmouth Summer Research Conference on AI.

1.12 External links

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

Chapter 2

Data mining

Not to be confused with analytics, information extraction, or data analysis.

Data mining (the analysis step of the “Knowledge Discovery in Databases” process, or KDD),[1] an interdisciplinary subfield of computer science,[2][3][4] is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems.[2] The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.[2] Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating.[2]

The term is a misnomer, because the goal is the extraction of patterns and knowledge from large amount of data, not the extraction of data itself.[5] It also is a buzzword[6] and is frequently applied to any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) as well as any application of computer decision support system, including artificial intelligence, machine learning, and business intelligence. The popular book “Data mining: Practical machine learning tools and techniques with Java”[7] (which covers mostly machine learning material) was originally to be named just “Practical machine learning”, and the term “data mining” was only added for marketing reasons.[8] Often the more general terms "(large scale) data analysis", or "analytics" – or when referring to actual methods, artificial intelligence and machine learning – are more appropriate.

The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection) and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation, nor result interpretation and reporting are part of the data mining step, but do belong to the overall KDD process as additional steps.

The related terms *data dredging*, *data fishing*, and *data snooping* refer to the use of data mining methods to sample parts of a larger population data set that are (or may be) too small for reliable statistical inferences to be made about the validity of any patterns discovered. These methods can, however, be used in creating new hypotheses to test against the larger data populations.

2.1 Etymology

In the 1960s, statisticians used terms like “Data Fishing” or “Data Dredging” to refer to what they considered the bad practice of analyzing data without an a-priori hypothesis. The term “Data Mining” appeared around 1990 in the database community. For a short time in 1980s, a phrase “database mining"™, was used, but since it was trademarked by HNC, a San Diego-based company, to pitch their Database Mining Workstation;[9] researchers consequently turned to “data mining”. Other terms used include Data Archaeology, Information Harvesting, Information Discovery, Knowledge Extraction, etc. Gregory Piatetsky-Shapiro coined the term “Knowledge Discovery in Databases” for the first workshop on the same topic (KDD-1989) and this term became more popular in AI and Machine Learning Community. However, the term data mining became more popular in the business and press communities.[10] Currently, Data Mining and Knowledge Discovery are used interchangeably. Since about 2007, “Predictive Analytics” and since 2011, “Data Science” terms were also used to describe this field.

2.2 Background

The manual extraction of patterns from data has occurred for centuries. Early methods of identifying patterns in data include Bayes' theorem (1700s) and regression analysis (1800s). The proliferation, ubiquity and increasing power of computer technology has dramatically increased data collection, storage, and manipulation ability. As data sets have grown in size and complexity, direct "hands-on" data analysis has increasingly been augmented with indirect, automated data processing, aided by other discoveries in computer science, such as neural networks, cluster analysis, genetic algorithms (1950s), decision trees and decision rules (1960s), and support vector machines (1990s). Data mining is the process of applying these methods with the intention of uncovering hidden patterns[11] in large data sets. It bridges the gap from applied statistics and artificial intelligence (which usually provide the mathematical background) to database management by exploiting the way data is stored and indexed in databases to execute the actual learning and discovery algorithms more efficiently, allowing such methods to be applied to ever larger data sets.

2.2.1 Research and evolution

The premier professional body in the field is the Association for Computing Machinery's (ACM) Special Interest Group (SIG) on Knowledge Discovery and Data Mining (SIGKDD).[12][13] Since 1989 this ACM SIG has hosted an annual international conference and published its proceedings,[14] and since 1999 it has published a biannual academic journal titled "SIGKDD Explorations".[15]

Computer science conferences on data mining include:

- CIKM Conference – ACM Conference on Information and Knowledge Management
- DMIN Conference – International Conference on Data Mining
- DMKD Conference – Research Issues on Data Mining and Knowledge Discovery
- ECDM Conference – European Conference on Data Mining
- ECML-PKDD Conference – European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases
- EDM Conference – International Conference on Educational Data Mining
- ICDM Conference – IEEE International Conference on Data Mining
- KDD Conference – ACM SIGKDD Conference on Knowledge Discovery and Data Mining
- MLDM Conference – Machine Learning and Data Mining in Pattern Recognition
- PAKDD Conference – The annual Pacific-Asia Conference on Knowledge Discovery and Data Mining
- PAW Conference – Predictive Analytics World
- SDM Conference – SIAM International Conference on Data Mining (SIAM)
- SSTD Symposium – Symposium on Spatial and Temporal Databases
- WSDM Conference – ACM Conference on Web Search and Data Mining

Data mining topics are also present on many data management/database conferences such as the ICDE Conference, SIGMOD Conference and International Conference on Very Large Data Bases

2.3 Process

The **Knowledge Discovery in Databases (KDD) process** is commonly defined with the stages:

(1) Selection

(2) Pre-processing

(3) Transformation

(4) *Data Mining*

(5) Interpretation/Evaluation.[1]

It exists, however, in many variations on this theme, such as the Cross Industry Standard Process for Data Mining (CRISP-DM) which defines six phases:

(1) Business Understanding

(2) Data Understanding

(3) Data Preparation

(4) Modeling

(5) Evaluation

(6) Deployment

or a simplified process such as (1) pre-processing, (2) data mining, and (3) results validation.

Polls conducted in 2002, 2004, and 2007 show that the CRISP-DM methodology is the leading methodology used by data miners.[16][17][18] The only other data mining standard named in these polls was SEMMA. However, 3-4 times as many people reported using CRISP-DM. Several teams of researchers have published reviews of data mining process models,[19][20] and Azevedo and Santos conducted a comparison of CRISP-DM and SEMMA in 2008.[21]

2.3.1 Pre-processing

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns actually present in the data, the target data set must be large enough to contain these patterns while remaining concise enough to be mined within an acceptable time limit. A common source for data is a data mart or data warehouse. Pre-processing is essential to analyze the multivariate data sets before data mining. The target set is then cleaned. Data cleaning removes the observations containing noise and those with missing data.

2.3.2 Data mining

Data mining involves six common classes of tasks:[1]

- Anomaly detection (Outlier/change/deviation detection) – The identification of unusual data records, that might be interesting or data errors that require further investigation.

- Association rule learning (Dependency modelling) – Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.

- Clustering – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.

- Classification – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".

- Regression – attempts to find a function which models the data with the least error.

- Summarization – providing a more compact representation of the data set, including visualization and report generation.

2.3.3 Results validation

Data mining can unintentionally be misused, and can then produce results which appear to be significant; but which do not actually predict future behavior and cannot be reproduced on a new sample of data and bear little use. Often this results from investigating too many hypotheses and not performing proper statistical hypothesis testing. A simple version of this problem in machine learning is known as overfitting, but the same problem can arise at different phases of the process and thus a train/test split - when applicable at all - may not be sufficient to prevent this from happening.

The final step of knowledge discovery from data is to verify that the patterns produced by the data mining algorithms occur in the wider data set. Not all patterns found by the data mining algorithms are necessarily valid. It is common for the data mining algorithms to find patterns in the training set which are not present in the general data set. This is called overfitting. To overcome this, the evaluation uses a test set of data on which the data mining algorithm was not trained. The learned patterns are applied to this test set, and the resulting output is compared to the desired output. For example, a data mining algorithm trying to distinguish "spam" from "legitimate" emails would be trained on a training set of sample e-mails. Once trained, the learned patterns would be applied to the test set of e-mails on which it had *not* been trained. The accuracy of the patterns can then be measured from how many e-mails they correctly classify. A number of statistical methods may be used to evaluate the algorithm, such as ROC curves.

If the learned patterns do not meet the desired standards, subsequently it is necessary to re-evaluate and change the pre-processing and data mining steps. If the learned patterns do meet the desired standards, then the final step is to interpret the learned patterns and turn them into knowledge.

2.4 Standards

There have been some efforts to define standards for the data mining process, for example the 1999 European Cross Industry Standard Process for Data Mining (CRISP-DM 1.0) and the 2004 Java Data Mining standard (JDM 1.0). Development on successors to these processes (CRISP-DM 2.0 and JDM 2.0) was active in 2006, but has stalled since. JDM 2.0 was withdrawn without reaching a final draft.

For exchanging the extracted models – in particular for use in predictive analytics – the key standard is the Predictive Model Markup Language (PMML), which is an XML-based language developed by the Data Mining Group (DMG) and supported as exchange format by many data mining applications. As the name suggests, it only covers prediction models, a particular data mining task of high importance to business applications. However, extensions to cover (for example) subspace clustering have been proposed independently of the DMG.[22]

2.5 Notable uses

See also: Category:Applied data mining.

2.5.1 Games

Since the early 1960s, with the availability of oracles for certain combinatorial games, also called tablebases (e.g. for 3x3-chess) with any beginning configuration, small-board dots-and-boxes, small-board-hex, and certain endgames in chess, dots-and-boxes, and hex; a new area for data mining has been opened. This is the extraction of human-usable strategies from these oracles. Current pattern recognition approaches do not seem to fully acquire the high level of abstraction required to be applied successfully. Instead, extensive experimentation with the tablebases – combined with an intensive study of tablebase-answers to well designed problems, and with knowledge of prior art (i.e., pre-tablebase knowledge) – is used to yield insightful patterns. Berlekamp (in dots-and-boxes, etc.) and John Nunn (in chess endgames) are notable examples of researchers doing this work, though they were not – and are not – involved in tablebase generation.

2.5.2 Business

In business, data mining is the analysis of historical business activities, stored as static data in data warehouse databases. The goal is to reveal hidden patterns and trends. Data mining software uses advanced pattern recognition algorithms to sift through large amounts of data to assist in discovering previously unknown strategic business information. Examples of what businesses use data mining for include performing market analysis to identify new product bundles, finding the root cause of manufacturing problems, to prevent customer attrition and acquire new customers, cross-selling to existing customers, and profiling customers with more accuracy.[23]

- In today's world raw data is being collected by companies at an exploding rate. For example, Walmart processes over 20 million point-of-sale transactions every day. This information is stored in a centralized database, but would be useless without some type of data mining software to analyze it. If Walmart analyzed their point-of-sale data with data mining techniques they would be able to determine sales trends, develop marketing campaigns, and more accurately predict customer loyalty.[24]

- Every time a credit card or a store loyalty card is being used, or a warranty card is being filled, data is being collected about the users behavior. Many people find the amount of information stored about us from companies, such as Google, Facebook, and Amazon, disturbing and are concerned about privacy. Although there is the potential for our personal data to be used in harmful, or unwanted, ways it is also being used to make our lives better. For example, Ford and Audi hope to one day collect information about customer driving patterns so they can recommend safer routes and warn drivers about dangerous road conditions.[25]

- Data mining in customer relationship management applications can contribute significantly to the bottom line. Rather than randomly contacting a prospect or customer through a call center or sending mail, a company can concentrate its efforts on prospects that are predicted to have a high likelihood of responding to an offer. More sophisticated methods may be used to optimize resources across campaigns so that one may predict to which channel and to which offer an individual is most likely to respond (across all potential offers). Additionally, sophisticated applications could be used to automate mailing. Once the results from data mining (potential prospect/customer and channel/offer) are determined, this “sophisticated application” can either automatically send an e-mail or a regular mail. Finally, in cases where many people will take an action without an offer, "uplift modeling" can be used to determine which people have the greatest increase in response if given an offer. Uplift modeling thereby enables marketers to focus mailings and offers on persuadable people, and not to send offers to people who will buy the product without an offer. Data clustering can also be used to automatically discover the segments or groups within a customer data set.

- Businesses employing data mining may see a return on investment, but also they recognize that the number of predictive models can quickly become very large. For example, rather than using one model to predict how many customers will churn, a business may choose to build a separate model for each region and customer type. In situations where a large number of models need to be maintained, some businesses turn to more automated data mining methodologies.

- Data mining can be helpful to human resources (HR) departments in identifying the characteristics of their most successful employees. Information obtained – such as universities attended by highly successful employees – can help HR focus recruiting efforts accordingly. Additionally, Strategic Enterprise Management applications help a company translate corporate-level goals, such as profit and margin share targets, into operational decisions, such as production plans and workforce levels.[26]

- Market basket analysis, relates to data-mining use in retail sales. If a clothing store records the purchases of customers, a data mining system could identify those customers who favor silk shirts over cotton ones. Although some explanations of relationships may be difficult, taking advantage of it is easier. The example deals with association rules within transaction-based data. Not all data are transaction based and logical, or inexact rules may also be present within a database.

- Market basket analysis has been used to identify the purchase patterns of the Alpha Consumer. Analyzing the data collected on this type of user has allowed companies to predict future buying trends and forecast supply demands.

- Data mining is a highly effective tool in the catalog marketing industry. Catalogers have a rich database of history of their customer transactions for millions of customers dating back a number of years. Data mining tools can identify patterns among customers and help identify the most likely customers to respond to upcoming mailing campaigns.

- Data mining for business applications can be integrated into a complex modeling and decision making process.[27] Reactive business intelligence (RBI) advocates a "holistic" approach that integrates data mining, modeling, and interactive visualization into an end-to-end discovery and continuous innovation process powered by human and automated learning.[28]

- In the area of decision making, the RBI approach has been used to mine knowledge that is progressively acquired from the decision maker, and then self-tune the decision method accordingly.[29] The relation between the quality of a data mining system and the amount of investment that the decision maker is willing to make was formalized by providing an economic perspective on the value of "extracted knowledge" in terms of its payoff to the organization[27] This decision-theoretic classification framework[27] was applied to a real-world semiconductor wafer manufacturing line, where decision rules for effectively monitoring and controlling the semiconductor wafer fabrication line were developed.[30]

- An example of data mining related to an integrated-circuit (IC) production line is described in the paper "Mining IC Test Data to Optimize VLSI Testing."[31] In this paper, the application of data mining and decision analysis to the problem of die-level functional testing is described. Experiments mentioned demonstrate the ability to apply a system of mining historical die-test data to create a probabilistic model of patterns of die failure. These patterns are then utilized to decide, in real time, which die to test next and when to stop testing. This system has been shown, based on experiments with historical test data, to have the potential to improve profits on mature IC products. Other examples[32][33] of the application of data mining methodologies in semiconductor manufacturing environments suggest that data mining methodologies may be particularly useful when data is scarce, and the various physical and chemical parameters that affect the process exhibit highly complex interactions. Another implication is that on-line monitoring of the semiconductor manufacturing process using data mining may be highly effective.

2.5.3 Science and engineering

In recent years, data mining has been used widely in the areas of science and engineering, such as bioinformatics, genetics, medicine, education and electrical power engineering.

- In the study of human genetics, sequence mining helps address the important goal of understanding the mapping relationship between the inter-individual variations in human DNA sequence and the variability in disease susceptibility. In simple terms, it aims to find out how the changes in an individual's DNA sequence affects the risks of developing common diseases such as cancer, which is of great importance to improving methods of diagnosing, preventing, and treating these diseases. One data mining method that is used to perform this task is known as multifactor dimensionality reduction.[34]

- In the area of electrical power engineering, data mining methods have been widely used for condition monitoring of high voltage electrical equipment. The purpose of condition monitoring is to obtain valuable information on, for example, the status of the insulation (or other important safety-related parameters). Data clustering techniques – such as the self-organizing map (SOM), have been applied to vibration monitoring and analysis of transformer on-load tap-changers (OLTCS). Using vibration monitoring, it can be observed that each tap change operation generates a signal that contains information about the condition of the tap changer contacts and the drive mechanisms. Obviously, different tap positions will generate different signals. However, there was considerable variability amongst normal condition signals for exactly the same tap position. SOM has been applied to detect abnormal conditions and to hypothesize about the nature of the abnormalities.[35]

- Data mining methods have been applied to dissolved gas analysis (DGA) in power transformers. DGA, as a diagnostics for power transformers, has been available for many years. Methods such as SOM has been applied to analyze generated data and to determine trends which are not obvious to the standard DGA ratio methods (such as Duval Triangle).[35]

- In educational research, where data mining has been used to study the factors leading students to choose to engage in behaviors which reduce their learning,[36] and to understand factors influencing university student retention.[37] A similar example of social application of data mining is its use in expertise finding systems, whereby descriptors of human expertise are extracted, normalized, and classified so as to facilitate the finding of experts, particularly in scientific and technical fields. In this way, data mining can facilitate institutional memory.

- Data mining methods of biomedical data facilitated by domain ontologies,[38] mining clinical trial data,[39] and traffic analysis using SOM.[40]

- In adverse drug reaction surveillance, the Uppsala Monitoring Centre has, since 1998, used data mining methods to routinely screen for reporting patterns indicative of emerging drug safety issues in the WHO global database of 4.6 million suspected adverse drug reaction incidents.[41] Recently, similar methodology has been developed to mine large collections of electronic health records for temporal patterns associating drug prescriptions to medical diagnoses.[42]

- Data mining has been applied to software artifacts within the realm of software engineering: Mining Software Repositories.

2.5.4 Human rights

Data mining of government records – particularly records of the justice system (i.e., courts, prisons) – enables the discovery of systemic human rights violations in connection to generation and publication of invalid or fraudulent legal records by various government agencies.[43][44]

2.5.5 Medical data mining

In 2011, the case of Sorrell v. IMS Health, Inc., decided by the Supreme Court of the United States, ruled that pharmacies may share information with outside companies. This practice was authorized under the 1st Amendment of the Constitution, protecting the "freedom of speech."[45] However, the passage of the Health Information Technology for Economic and Clinical Health Act (HITECH Act) helped to initiate the adoption of the electronic health record (EHR) and supporting technology in the United States.[46] The HITECH Act was signed into law on February 17, 2009 as part of the American Recovery and Reinvestment Act (ARRA) and helped to open the door to medical data mining.[47] Prior to the signing of this law, estimates of only 20% of United States-based physicians were utilizing electronic patient records.[46] Søren Brunak notes that "the patient record becomes as information-rich as possible" and thereby "maximizes the data mining opportunities."[46] Hence, electronic patient records further expands the possibilities regarding medical data mining thereby opening the door to a vast source of medical data analysis.

2.5.6 Spatial data mining

Spatial data mining is the application of data mining methods to spatial data. The end objective of spatial data mining is to find patterns in data with respect to geography. So far, data mining and Geographic Information Systems (GIS) have existed as two separate technologies, each with its own methods, traditions, and approaches to visualization and data analysis. Particularly, most contemporary GIS have only very basic spatial analysis functionality. The immense explosion in geographically referenced data occasioned by developments in IT, digital mapping, remote sensing, and the global diffusion of GIS emphasizes the importance of developing data-driven inductive approaches to geographical analysis and modeling.

Data mining offers great potential benefits for GIS-based applied decision-making. Recently, the task of integrating these two technologies has become of critical importance, especially as various public and private sector organizations possessing huge databases with thematic and geographically referenced data begin to realize the huge potential of the information contained therein. Among those organizations are:

- offices requiring analysis or dissemination of geo-referenced statistical data
- public health services searching for explanations of disease clustering
- environmental agencies assessing the impact of changing land-use patterns on climate change
- geo-marketing companies doing customer segmentation based on spatial location.

Challenges in Spatial mining: Geospatial data repositories tend to be very large. Moreover, existing GIS datasets are often splintered into feature and attribute components that are conventionally archived in hybrid data management systems. Algorithmic requirements differ substantially for relational (attribute) data management and for topological (feature) data management.[48] Related to this is the range and diversity of geographic data formats, which present unique challenges. The digital geographic data revolution is creating new types of data formats beyond the traditional "vector" and "raster" formats. Geographic data repositories increasingly include ill-structured data, such as imagery and geo-referenced multi-media.[49]

There are several critical research challenges in geographic knowledge discovery and data mining. Miller and Han[50] offer the following list of emerging research topics in the field:

- **Developing and supporting geographic data warehouses (GDW's)**: Spatial properties are often reduced to simple aspatial attributes in mainstream data warehouses. Creating an integrated GDW requires solving issues of spatial and temporal data interoperability – including differences in semantics, referencing systems, geometry, accuracy, and position.

- **Better spatio-temporal representations in geographic knowledge discovery**: Current geographic knowledge discovery (GKD) methods generally use very simple representations of geographic objects and spatial relationships. Geographic data mining methods should recognize more complex geographic objects (i.e., lines and polygons) and relationships (i.e., non-Euclidean distances, direction, connectivity, and interaction through attributed geographic space such as terrain). Furthermore, the time dimension needs to be more fully integrated into these geographic representations and relationships.

- **Geographic knowledge discovery using diverse data types**: GKD methods should be developed that can handle diverse data types beyond the traditional raster and vector models, including imagery and geo-referenced multimedia, as well as dynamic data types (video streams, animation).

2.5.7 Temporal data mining

Data may contain attributes generated and recorded at different times. In this case finding meaningful relationships in the data may require considering the temporal order of the attributes. A temporal relationship may indicate a causal relationship, or simply an association.

2.5.8 Sensor data mining

Wireless sensor networks can be used for facilitating the collection of data for spatial data mining for a variety of applications such as air pollution monitoring.[51] A characteristic of such networks is that nearby sensor nodes monitoring an environmental feature typically register similar values. This kind of data redundancy due to the spatial correlation between sensor observations inspires the techniques for in-network data aggregation and mining. By measuring the spatial correlation between data sampled by different sensors, a wide class of specialized algorithms can be developed to develop more efficient spatial data mining algorithms.[52]

2.5.9 Visual data mining

In the process of turning from analogical into digital, large data sets have been generated, collected, and stored discovering statistical patterns, trends and information which is hidden in data, in order to build predictive patterns. Studies suggest visual data mining is faster and much more intuitive than is traditional data mining.[53][54][55] See also Computer vision.

2.5.10 Music data mining

Data mining techniques, and in particular co-occurrence analysis, has been used to discover relevant similarities among music corpora (radio lists, CD databases) for purposes including classifying music into genres in a more objective manner.[56]

2.5.11 Surveillance

Data mining has been used by the U.S. government. Programs include the Total Information Awareness (TIA) program, Secure Flight (formerly known as Computer-Assisted Passenger Prescreening System (CAPPS II)), Analysis, Dissemination, Visualization, Insight, Semantic Enhancement (ADVISE),[57] and the Multi-state Anti-Terrorism Information Exchange (MATRIX).[58] These programs have been discontinued due to controversy over whether they violate the 4th Amendment to the United States Constitution, although many programs that were formed under them continue to be funded by different organizations or under different names.[59]

In the context of combating terrorism, two particularly plausible methods of data mining are "pattern mining" and "subject-based data mining".

2.5.12 Pattern mining

"Pattern mining" is a data mining method that involves finding existing patterns in data. In this context *patterns* often means association rules. The original motivation for searching association rules came from the desire to analyze supermarket transaction data, that is, to examine customer behavior in terms of the purchased products.

For example, an association rule "beer ⇒ potato chips (80%)" states that four out of five customers that bought beer also bought potato chips.

In the context of pattern mining as a tool to identify terrorist activity, the National Research Council provides the following definition: "Pattern-based data mining looks for patterns (including anomalous data patterns) that might be associated with terrorist activity — these patterns might be regarded as small signals in a large ocean of noise."[60][61][62] Pattern Mining includes new areas such a Music Information Retrieval (MIR) where patterns seen both in the temporal and non temporal domains are imported to classical knowledge discovery search methods.

2.5.13 Subject-based data mining

"Subject-based data mining" is a data mining method involving the search for associations between individuals in data. In the context of combating terrorism, the National Research Council provides the following definition: "Subject-based data mining uses an initiating individual or other datum that is considered, based on other information, to be of high interest, and the goal is to determine what other persons or financial transactions or movements, etc., are related to that initiating datum."[61]

2.5.14 Knowledge grid

Knowledge discovery "On the Grid" generally refers to conducting knowledge discovery in an open environment using grid computing concepts, allowing users to integrate data from various online data sources, as well make use of remote resources, for executing their data mining tasks. The earliest example was the Discovery Net,[63][64] developed at Imperial College London, which won the "Most Innovative Data-Intensive Application Award" at the ACM SC02 (Supercomputing 2002) conference and exhibition, based on a demonstration of a fully interactive distributed knowledge discovery application for a bioinformatics application. Other examples include work conducted by researchers at the University of Calabria, who developed a Knowledge Grid architecture for distributed knowledge discovery, based on grid computing.[65][66]

2.6 Privacy concerns and ethics

While the term "data mining" itself has no ethical implications, it is often associated with the mining of information in relation to peoples' behavior (ethical and otherwise).[67]

The ways in which data mining can be used can in some cases and contexts raise questions regarding privacy, legality, and ethics.[68] In particular, data mining government or commercial data sets for national security or law enforcement purposes, such as in the Total Information Awareness Program or in ADVISE, has raised privacy concerns.[69][70]

Data mining requires data preparation which can uncover information or patterns which may compromise confidentiality and privacy obligations. A common way for this to occur is through data aggregation. Data aggregation involves combining data together (possibly from various sources) in a way that facilitates analysis (but that also might make identification of private, individual-level data deducible or otherwise apparent).[71] This is not data mining *per se*, but a result of the preparation of data before – and for the purposes of – the analysis. The threat to an individual's privacy comes into play when the data, once compiled, cause the data miner, or anyone who has access to the newly compiled data set, to be able to identify specific individuals, especially when the data were originally anonymous.[72][73][74]

It is recommended that an individual is made aware of the following **before** data are collected:[71]

- the purpose of the data collection and any (known) data mining projects;
- how the data will be used;
- who will be able to mine the data and use the data and their derivatives;
- the status of security surrounding access to the data;
- how collected data can be updated.

Data may also be modified so as to *become* anonymous, so that individuals may not readily be identified.[71] However, even "de-identified"/"anonymized" data sets can potentially contain enough information to allow identification of individuals, as occurred when journalists were able to find several individuals based on a set of search histories that were inadvertently released by AOL.[75]

2.6.1 Situation in Europe

Europe has rather strong privacy laws, and efforts are underway to further strengthen the rights of the consumers. However, the U.S.-E.U. Safe Harbor Principles currently effectively expose European users to privacy exploitation by U.S. companies. As a consequence of Edward Snowden's Global surveillance disclosure, there has been increased discussion to revoke this agreement, as in particular the data will be fully exposed to the National Security Agency, and attempts to reach an agreement have failed.

2.6.2 Situation in the United States

In the United States, privacy concerns have been addressed by the US Congress via the passage of regulatory controls such as the Health Insurance Portability and Accountability Act (HIPAA). The HIPAA requires individuals to give their “informed consent” regarding information they provide and its intended present and future uses. According to an article in *Biotech Business Week*', "'*[i]n practice, HIPAA may not offer any greater protection than the longstanding regulations in the research arena,' says the AAHC. More importantly, the rule's goal of protection through informed consent is undermined by the complexity of consent forms that are required of patients and participants, which approach a level of incomprehensibility to average individuals.*"[76] *This underscores the necessity for data anonymity in data aggregation and mining practices.*

U.S. information privacy legislation such as HIPAA and the Family Educational Rights and Privacy Act (FERPA) applies only to the specific areas that each such law addresses. Use of data mining by the majority of businesses in the U.S. is not controlled by any legislation.

2.7 Copyright Law

2.7.1 Situation in Europe

Due to a lack of flexibilities in European copyright and database law, the mining of in-copyright works such as web mining without the permission of the copyright owner is not legal. Where a database is pure data in Europe there is likely to be no copyright, but database rights may exist so data mining becomes subject to regulations by the Database Directive. On the recommendation of the Hargreaves review this led to the UK government to amend its copyright law in 2014[77] to allow content mining as a limitation and exception. Only the second country in the world to do so after Japan, which introduced an exception in 2009 for data mining. However due to the restriction of the Copyright Directive, the UK exception only allows content mining for non-commercial purposes. UK copyright law also does not allow this provision to be overridden by contractual terms and conditions. The European Commission facilitated stakeholder discussion on text and data mining in 2013, under the title of Licences for Europe.[78] The focus on the solution to this legal issue being licences and not limitations and exceptions led to representatives of universities, researchers, libraries, civil society groups and open access publishers to leave the stakeholder dialogue in May 2013.[79]

2.7.2 Situation in the United States

By contrast to Europe, the flexible nature of US copyright law, and in particular fair use means that content mining in America, as well as other fair use countries such as Israel, Taiwan and South Korea is viewed as being legal. As content mining is transformative, that is it does not supplant the original work, it is viewed as being lawful under fair use. For example as part of the Google Book settlement the presiding judge on the case ruled that Google's digitisation project of in-copyright books was lawful, in part because of the transformative uses that the digitisation project displayed - one being text and data mining.[80]

2.8 Software

See also: Category:Data mining and machine learning software.

2.8.1 Free open-source data mining software and applications

- Carrot2: Text and search results clustering framework.
- Chemicalize.org: A chemical structure miner and web search engine.
- ELKI: A university research project with advanced cluster analysis and outlier detection methods written in the Java language.
- GATE: a natural language processing and language engineering tool.
- KNIME: The Konstanz Information Miner, a user friendly and comprehensive data analytics framework.
- ML-Flex: A software package that enables users to integrate with third-party machine-learning packages written in any programming language, execute classification analyses in parallel across multiple computing nodes, and produce HTML reports of classification results.
- MLPACK library: a collection of ready-to-use machine learning algorithms written in the C++ language.
- Massive Online Analysis (MOA): a real-time big data stream mining with concept drift tool in the Java programming language.
- NLTK (Natural Language Toolkit): A suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python language.
- OpenNN: Open neural networks library.
- Orange: A component-based data mining and machine learning software suite written in the Python language.
- R: A programming language and software environment for statistical computing, data mining, and graphics. It is part of the GNU Project.

- SCaViS: Java cross-platform data analysis framework developed at Argonne National Laboratory.
- SenticNet API: A semantic and affective resource for opinion mining and sentiment analysis.
- Tanagra: A visualisation-oriented data mining software, also for teaching.
- Torch: An open source deep learning library for the Lua programming language and scientific computing framework with wide support for machine learning algorithms.
- UIMA: The UIMA (Unstructured Information Management Architecture) is a component framework for analyzing unstructured content such as text, audio and video – originally developed by IBM.
- Weka: A suite of machine learning software applications written in the Java programming language.

2.8.2 Commercial data-mining software and applications

- Angoss KnowledgeSTUDIO: data mining tool provided by Angoss.
- Clarabridge: enterprise class text analytics solution.
- HP Vertica Analytics Platform: data mining software provided by HP.
- IBM SPSS Modeler: data mining software provided by IBM.
- KXEN Modeler: data mining tool provided by KXEN.
- Grapheme: data mining and visualization software provided by iChrome.
- LIONsolver: an integrated software application for data mining, business intelligence, and modeling that implements the Learning and Intelligent OptimizatioN (LION) approach.
- Microsoft Analysis Services: data mining software provided by Microsoft.
- NetOwl: suite of multilingual text and entity analytics products that enable data mining.
- Oracle Data Mining: data mining software by Oracle.
- RapidMiner: An environment for machine learning and data mining experiments.
- SAS Enterprise Miner: data mining software provided by the SAS Institute.
- STATISTICA Data Miner: data mining software provided by StatSoft.
- Qlucore Omics Explorer: data mining software provided by Qlucore.

2.8.3 Marketplace surveys

Several researchers and organizations have conducted reviews of data mining tools and surveys of data miners. These identify some of the strengths and weaknesses of the software packages. They also provide an overview of the behaviors, preferences and views of data miners. Some of these reports include:

- 2011 Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery[81]
- Rexer Analytics Data Miner Surveys (2007–2013)[82]
- Forrester Research 2010 Predictive Analytics and Data Mining Solutions report[83]
- Gartner 2008 “Magic Quadrant” report[84]
- Robert A. Nisbet’s 2006 Three Part Series of articles “Data Mining Tools: Which One is Best For CRM?"[85]
- Haughton et al.'s 2003 Review of Data Mining Software Packages in *The American Statistician*[86]
- Goebel & Gruenwald 1999 “A Survey of Data Mining a Knowledge Discovery Software Tools” in SIGKDD Explorations[87]

2.9 See also

Methods

- Anomaly/outlier/change detection
- Association rule learning
- Classification
- Cluster analysis
- Decision tree
- Factor analysis
- Genetic algorithms
- Intention mining
- Multilinear subspace learning
- Neural networks
- Regression analysis
- Sequence mining

- Structured data analysis
- Support vector machines
- Text mining
- Online analytical processing (OLAP)

Application domains

- Analytics
- Bioinformatics
- Business intelligence
- Data analysis
- Data warehouse
- Decision support system
- Drug discovery
- Exploratory data analysis
- Predictive analytics
- Web mining

Application examples

See also: Category:Applied data mining.

- Customer analytics
- Data mining in agriculture
- Data mining in meteorology
- Educational data mining
- National Security Agency
- Police-enforced ANPR in the UK
- Quantitative structure–activity relationship
- Surveillance / Mass surveillance (e.g., Stellar Wind)

Related topics

Data mining is about *analyzing* data; for information about extracting information out of data, see:

- Data integration
- Data transformation
- Electronic discovery
- Information extraction
- Information integration
- Named-entity recognition
- Profiling (information science)
- Web scraping

2.10 References

[1] Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic (1996). "From Data Mining to Knowledge Discovery in Databases" (PDF). Retrieved 17 December 2008.

[2] "Data Mining Curriculum". ACM SIGKDD. 2006-04-30. Retrieved 2014-01-27.

[3] Clifton, Christopher (2010). "Encyclopædia Britannica: Definition of Data Mining". Retrieved 2010-12-09.

[4] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2009). "The Elements of Statistical Learning: Data Mining, Inference, and Prediction". Retrieved 2012-08-07.

[5] Han, Jiawei; Kamber, Micheline (2001). *Data mining: concepts and techniques*. Morgan Kaufmann. p. 5. ISBN 9781558604896. Thus, data mining should habe been more appropriately named "knowledge mining from data," which is unfortunately somewhat long

[6] See e.g. OKAIRP 2005 Fall Conference, Arizona State University About.com: Datamining

[7] Witten, Ian H.; Frank, Eibe; Hall, Mark A. (30 January 2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3 ed.). Elsevier. ISBN 978-0-12-374856-0.

[8] Bouckaert, Remco R.; Frank, Eibe; Hall, Mark A.; Holmes, Geoffrey; Pfahringer, Bernhard; Reutemann, Peter; Witten, Ian H. (2010). "WEKA Experiences with a Java open-source project". *Journal of Machine Learning Research* **11**: 2533–2541. the original title, "Practical machine learning", was changed ... The term "data mining" was [added] primarily for marketing reasons.

[9] Mena, Jesús (2011). *Machine Learning Forensics for Law Enforcement, Security, and Intelligence*. Boca Raton, FL: CRC Press (Taylor & Francis Group). ISBN 978-1-4398-6069-4.

[10] Piatetsky-Shapiro, Gregory; Parker, Gary (2011). "Lesson: Data Mining, and Knowledge Discovery: An Introduction". *Introduction to Data Mining*. KD Nuggets. Retrieved 30 August 2012.

[11] Kantardzic, Mehmed (2003). *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons. ISBN 0-471-22852-4. OCLC 50055336.

[12] "Microsoft Academic Search: Top conferences in data mining". Microsoft Academic Search.

[13] "Google Scholar: Top publications - Data Mining & Analysis". Google Scholar.

[14] Proceedings, International Conferences on Knowledge Discovery and Data Mining, ACM, New York.

[15] SIGKDD Explorations, ACM, New York.

[16] Gregory Piatetsky-Shapiro (2002) *KDnuggets Methodology Poll*

[17] Gregory Piatetsky-Shapiro (2004) *KDnuggets Methodology Poll*

[18] Gregory Piatetsky-Shapiro (2007) *KDnuggets Methodology Poll*

[19] Óscar Marbán, Gonzalo Mariscal and Javier Segovia (2009); *A Data Mining & Knowledge Discovery Process Model*. In Data Mining and Knowledge Discovery in Real Life Applications, Book edited by: Julio Ponce and Adem Karahoca, ISBN 978-3-902613-53-0, pp. 438–453, February 2009, I-Tech, Vienna, Austria.

[20] Lukasz Kurgan and Petr Musilek (2006); *A survey of Knowledge Discovery and Data Mining process models*. The Knowledge Engineering Review. Volume 21 Issue 1, March 2006, pp 1–24, Cambridge University Press, New York, NY, USA doi:10.1017/S0269888906000737

[21] Azevedo, A. and Santos, M. F. KDD, SEMMA and CRISP-DM: a parallel overview. In Proceedings of the IADIS European Conference on Data Mining 2008, pp 182–185.

[22] Günnemann, Stephan; Kremer, Hardy; Seidl, Thomas (2011). "An extension of the PMML standard to subspace clustering models". *Proceedings of the 2011 workshop on Predictive markup language modeling - PMML '11*. p. 48. doi:10.1145/2023598.2023605. ISBN 9781450308373.

[23] O'Brien, J. A., & Marakas, G. M. (2011). Management Information Systems. New York, NY: McGraw-Hill/Irwin.

[24] Alexander, D. (n.d.). Data Mining. Retrieved from The University of Texas at Austin: College of Liberal Arts: http://www.laits.utexas.edu/~{}anorman/BUS.FOR/course.mat/Alex/

[25] Goss, S. (2013, April 10). Data-mining and our personal privacy. Retrieved from The Telegraph: http://www.macon.com/2013/04/10/2429775/data-mining-and-our-personal-privacy.html

[26] Monk, Ellen; Wagner, Bret (2006). *Concepts in Enterprise Resource Planning, Second Edition*. Boston, MA: Thomson Course Technology. ISBN 0-619-21663-8. OCLC 224465825.

[27] Elovici, Yuval; Braha, Dan (2003). "A Decision-Theoretic Approach to Data Mining" (PDF). *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans* **33** (1).

[28] Battiti, Roberto; and Brunato, Mauro; *Reactive Business Intelligence. From Data to Models to Insight*, Reactive Search Srl, Italy, February 2011. ISBN 978-88-905795-0-9.

[29] Battiti, Roberto; Passerini, Andrea (2010). "Brain-Computer Evolutionary Multi-Objective Optimization (BC-EMO): a genetic algorithm adapting to the decision maker" (PDF). *IEEE Transactions on Evolutionary Computation* **14** (15): 671–687. doi:10.1109/TEVC.2010.2058118.

[30] Braha, Dan; Elovici, Yuval; Last, Mark (2007). "Theory of actionable data mining with application to semiconductor manufacturing control" (PDF). *International Journal of Production Research* **45** (13).

[31] Fountain, Tony; Dietterich, Thomas; and Sudyka, Bill (2000); *Mining IC Test Data to Optimize VLSI Testing*, in Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM Press, pp. 18–25

[32] Braha, Dan; Shmilovici, Armin (2002). "Data Mining for Improving a Cleaning Process in the Semiconductor Industry" (PDF). *IEEE Transactions on Semiconductor Manufacturing* **15** (1).

[33] Braha, Dan; Shmilovici, Armin (2003). "On the Use of Decision Tree Induction for Discovery of Interactions in a Photolithographic Process" (PDF). *IEEE Transactions on Semiconductor Manufacturing* **16** (4).

[34] Zhu, Xingquan; Davidson, Ian (2007). *Knowledge Discovery and Data Mining: Challenges and Realities*. New York, NY: Hershey. p. 18. ISBN 978-1-59904-252-7.

[35] McGrail, Anthony J.; Gulski, Edward; Allan, David; Birtwhistle, David; Blackburn, Trevor R.; Groot, Edwin R. S. "Data Mining Techniques to Assess the Condition of High Voltage Electrical Plant". *CIGRÉ WG 15.11 of Study Committee 15*.

[36] Baker, Ryan S. J. d. "Is Gaming the System State-or-Trait? Educational Data Mining Through the Multi-Contextual Application of a Validated Behavioral Model". *Workshop on Data Mining for User Modeling 2007*.

[37] Superby Aguirre, Juan Francisco; Vandamme, Jean-Philippe; Meskens, Nadine. "Determination of factors influencing the achievement of the first-year university students using data mining methods". *Workshop on Educational Data Mining 2006*.

[38] Zhu, Xingquan; Davidson, Ian (2007). *Knowledge Discovery and Data Mining: Challenges and Realities*. New York, NY: Hershey. pp. 163–189. ISBN 978-1-59904-252-7.

[39] Zhu, Xingquan; Davidson, Ian (2007). *Knowledge Discovery and Data Mining: Challenges and Realities*. New York, NY: Hershey. pp. 31–48. ISBN 978-1-59904-252-7.

[40] Chen, Yudong; Zhang, Yi; Hu, Jianming; Li, Xiang (2006). "Traffic Data Analysis Using Kernel PCA and Self-Organizing Map". *IEEE Intelligent Vehicles Symposium*.

[41] Bate, Andrew; Lindquist, Marie; Edwards, I. Ralph; Olsson, Sten; Orre, Roland; Lansner, Anders; de Freitas, Rogelio Melhado (Jun 1998). "A Bayesian neural network method for adverse drug reaction signal generation" (PDF). *European Journal of Clinical Pharmacology* **54** (4): 315–21. doi:10.1007/s002280050466. PMID 9696956.

[42] Norén, G. Niklas; Bate, Andrew; Hopstadius, Johan; Star, Kristina; and Edwards, I. Ralph (2008); Temporal Pattern Discovery for Trends and Transient Effects: Its Application to Patient Records. *Proceedings of the Fourteenth International Conference on Knowledge Discovery and Data Mining (SIGKDD 2008), Las Vegas, NV*, pp. 963–971.

[43] Zernik, Joseph; Data Mining as a Civic Duty – Online Public Prisoners' Registration Systems, *International Journal on Social Media: Monitoring, Measurement, Mining*, 1: 84–96 (2010)

[44] Zernik, Joseph; Data Mining of Online Judicial Records of the Networked US Federal Courts, *International Journal on Social Media: Monitoring, Measurement, Mining*, 1:69–83 (2010)

[45] David G. Savage (2011-06-24). "Pharmaceutical industry: Supreme Court sides with pharmaceutical industry in two decisions". *Los Angeles Times*. Retrieved 2012-11-07.

[46] Analyzing Medical Data. (2012). *Communications of the ACM* 55(6), 13-15. doi:10.1145/2184319.2184324

[47] http://searchhealthit.techtarget.com/definition/HITECH-Act

[48] Healey, Richard G. (1991); *Database Management Systems*, in Maguire, David J.; Goodchild, Michael F.; and Rhind, David W., (eds.), *Geographic Information Systems: Principles and Applications*, London, GB: Longman

[49] Camara, Antonio S.; and Raper, Jonathan (eds.) (1999); *Spatial Multimedia and Virtual Reality*, London, GB: Taylor and Francis

[50] Miller, Harvey J.; and Han, Jiawei (eds.) (2001); *Geographic Data Mining and Knowledge Discovery*, London, GB: Taylor & Francis

[51] Ma, Y.; Richards, M.; Ghanem, M.; Guo, Y.; Hassard, J. (2008). "Air Pollution Monitoring and Mining Based on Sensor Grid in London". *Sensors* **8** (6): 3601. doi:10.3390/s8063601.

[52] Ma, Y.; Guo, Y.; Tian, X.; Ghanem, M. (2011). "Distributed Clustering-Based Aggregation Algorithm for Spatial Correlated Sensor Networks". *IEEE Sensors Journal* **11** (3): 641. doi:10.1109/JSEN.2010.2056916.

[53] Zhao, Kaidi; and Liu, Bing; Tirpark, Thomas M.; and Weimin, Xiao; *A Visual Data Mining Framework for Convenient Identification of Useful Knowledge*

[54] Keim, Daniel A.; *Information Visualization and Visual Data Mining*

[55] Burch, Michael; Diehl, Stephan; Weißgerber, Peter; *Visual Data Mining in Software Archives*

[56] Pachet, François; Westermann, Gert; and Laigre, Damien; *Musical Data Mining for Electronic Music Distribution*, Proceedings of the 1st WedelMusic Conference,Firenze, Italy, 2001, pp. 101–106.

[57] Government Accountability Office, *Data Mining: Early Attention to Privacy in Developing a Key DHS Program Could Reduce Risks*, GAO-07-293 (February 2007), Washington, DC

[58] Secure Flight Program report, MSNBC

[59] "Total/Terrorism Information Awareness (TIA): Is It Truly Dead?". *Electronic Frontier Foundation (official website)*. 2003. Retrieved 2009-03-15.

[60] Agrawal, Rakesh; Mannila, Heikki; Srikant, Ramakrishnan; Toivonen, Hannu; and Verkamo, A. Inkeri; *Fast discovery of association rules*, in *Advances in knowledge discovery and data mining*, MIT Press, 1996, pp. 307–328

[61] National Research Council, *Protecting Individual Privacy in the Struggle Against Terrorists: A Framework for Program Assessment*, Washington, DC: National Academies Press, 2008

[62] Haag, Stephen; Cummings, Maeve; Phillips, Amy (2006). *Management Information Systems for the information age*. Toronto: McGraw-Hill Ryerson. p. 28. ISBN 0-07-095569-7. OCLC 63194770.

[63] Ghanem, Moustafa; Guo, Yike; Rowe, Anthony; Wendel, Patrick (2002). "Grid-based knowledge discovery services for high throughput informatics". *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*. p. 416. doi:10.1109/HPDC.2002.1029946. ISBN 0-7695-1686-6.

[64] Ghanem, Moustafa; Curcin, Vasa; Wendel, Patrick; Guo, Yike (2009). "Building and Using Analytical Workflows in Discovery Net". *Data Mining Techniques in Grid Computing Environments*. p. 119. doi:10.1002/9780470699904.ch8. ISBN 9780470699904.

[65] Cannataro, Mario; Talia, Domenico (January 2003). "The Knowledge Grid: An Architecture for Distributed Knowledge Discovery" (PDF). *Communications of the ACM* **46** (1): 89–93. doi:10.1145/602421.602425. Retrieved 17 October 2011.

[66] Talia, Domenico; Trunfio, Paolo (July 2010). "How distributed data mining tasks can thrive as knowledge services" (PDF). *Communications of the ACM* **53** (7): 132–137. doi:10.1145/1785414.1785451. Retrieved 17 October 2011.

[67] Seltzer, William. "The Promise and Pitfalls of Data Mining: Ethical Issues" (PDF).

[68] Pitts, Chip (15 March 2007). "The End of Illegal Domestic Spying? Don't Count on It". *Washington Spectator*.

[69] Taipale, Kim A. (15 December 2003). "Data Mining and Domestic Security: Connecting the Dots to Make Sense of Data". *Columbia Science and Technology Law Review* **5** (2). OCLC 45263753. SSRN 546782.

[70] Resig, John; and Teredesai, Ankur (2004). "A Framework for Mining Instant Messaging Services". *Proceedings of the 2004 SIAM DM Conference*.

[71] *Think Before You Dig: Privacy Implications of Data Mining & Aggregation*, NASCIO Research Brief, September 2004

[72] Ohm, Paul. "Don't Build a Database of Ruin". Harvard Business Review.

[73] Darwin Bond-Graham, Iron Cagebook - The Logical End of Facebook's Patents, Counterpunch.org, 2013.12.03

[74] Darwin Bond-Graham, Inside the Tech industry's Startup Conference, Counterpunch.org, 2013.09.11

[75] *AOL search data identified individuals*, SecurityFocus, August 2006

[76] Biotech Business Week Editors (June 30, 2008); *BIOMEDICINE; HIPAA Privacy Rule Impedes Biomedical Research*, Biotech Business Week, retrieved 17 November 2009 from LexisNexis Academic

[77] UK Researchers Given Data Mining Right Under New UK Copyright Laws. *Out-Law.com*. Retrieved 14 November 2014

[78] "Licences for Europe - Structured Stakeholder Dialogue 2013". *European Commission*. Retrieved 14 November 2014.

[79] "Text and Data Mining:Its importance and the need for change in Europe". *Association of European Research Libraries*. Retrieved 14 November 2014.

[80] "Judge grants summary judgment in favor of Google Books — a fair use victory". *Lexology.com*. Antonelli Law Ltd. Retrieved 14 November 2014.

[81] Mikut, Ralf; Reischl, Markus (September–October 2011). "Data Mining Tools". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **1** (5): 431–445. doi:10.1002/widm.24. Retrieved October 21, 2011.

[82] Karl Rexer, Heather Allen, & Paul Gearan (2011); *Understanding Data Miners*, Analytics Magazine, May/June 2011 (INFORMS: Institute for Operations Research and the Management Sciences).

[83] Kobielus, James; *The Forrester Wave: Predictive Analytics and Data Mining Solutions, Q1 2010*, Forrester Research, 1 July 2008

[84] Herschel, Gareth; *Magic Quadrant for Customer Data-Mining Applications*, Gartner Inc., 1 July 2008

[85] Nisbet, Robert A. (2006); *Data Mining Tools: Which One is Best for CRM? Part 1*, Information Management Special Reports, January 2006

[86] Haughton, Dominique; Deichmann, Joel; Eshghi, Abdolreza; Sayek, Selin; Teebagy, Nicholas; and Topi, Heikki (2003); *A Review of Software Packages for Data Mining*, The American Statistician, Vol. 57, No. 4, pp. 290–309

[87] Goebel, Michael; Gruenwald, Le (1999); *A Survey of Data Mining and Knowledge Discovery Software Tools*, SIGKDD Explorations, Vol. 1, Issue 1, pp. 20–33

2.11 Further reading

- Cabena, Peter; Hadjnian, Pablo; Stadler, Rolf; Verhees, Jaap; and Zanasi, Alessandro (1997); *Discovering Data Mining: From Concept to Implementation*, Prentice Hall, ISBN 0-13-743980-6
- M.S. Chen, J. Han, P.S. Yu (1996) "Data mining: an overview from a database perspective". *Knowledge and data Engineering, IEEE Transactions* on 8 (6), 866-883
- Feldman, Ronen; and Sanger, James; *The Text Mining Handbook*, Cambridge University Press, ISBN 978-0-521-83657-9
- Guo, Yike; and Grossman, Robert (editors) (1999); *High Performance Data Mining: Scaling Algorithms, Applications and Systems*, Kluwer Academic Publishers
- Han, Jiawei, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.
- Hastie, Trevor, Tibshirani, Robert and Friedman, Jerome (2001); *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, ISBN 0-387-95284-5
- Liu, Bing (2007); *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*, Springer, ISBN 3-540-37881-2
- Murphy, Chris (16 May 2011). "Is Data Mining Free Speech?". *InformationWeek* (UMB): 12.
- Nisbet, Robert; Elder, John; Miner, Gary (2009); *Handbook of Statistical Analysis & Data Mining Applications*, Academic Press/Elsevier, ISBN 978-0-12-374765-5
- Poncelet, Pascal; Masseglia, Florent; and Teisseire, Maguelonne (editors) (October 2007); "Data Mining Patterns: New Methods and Applications", *Information Science Reference*, ISBN 978-1-59904-162-9
- Tan, Pang-Ning; Steinbach, Michael; and Kumar, Vipin (2005); *Introduction to Data Mining*, ISBN 0-321-32136-7
- Theodoridis, Sergios; and Koutroumbas, Konstantinos (2009); *Pattern Recognition*, 4th Edition, Academic Press, ISBN 978-1-59749-272-0
- Weiss, Sholom M.; and Indurkhya, Nitin (1998); *Predictive Data Mining*, Morgan Kaufmann
- Witten, Ian H.; Frank, Eibe; Hall, Mark A. (30 January 2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3 ed.). Elsevier. ISBN 978-0-12-374856-0. (See also Free Weka software)

- Ye, Nong (2003); *The Handbook of Data Mining*, Mahwah, NJ: Lawrence Erlbaum

2.12 External links

Chapter 3

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.

3.1 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*.

3.2 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.

3.3 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.

3.4 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.

3.5 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.

3.6 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.

3.7 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

3.8 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.

3.9 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

3.10 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

3.11 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.

3.12 External links

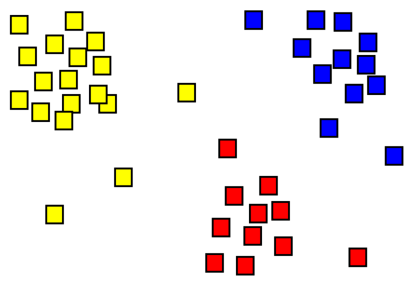
- Classifier showdown A practical comparison of classification algorithms.
- Statistical Pattern Recognition Toolbox for Matlab.
- TOOLDIAG Pattern recognition toolbox.
- Statistical classification software based on adaptive kernel density estimation.
- PAL Classification suite written in Java.
- kNN 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 4

Cluster analysis

For the supervised learning approach, see Statistical classification.

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.


The result of a cluster analysis shown as the coloring of the squares into three clusters.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem. The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It will often be necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

Besides the term *clustering*, there are a number of terms with similar meanings, including *automatic classification*, *numerical taxonomy*, *botryology* (from Greek βότρυς "grape") and *typological analysis*. The subtle differences are often in the usage of the results: while in data mining, the resulting groups are the matter of interest, in automatic classification the resulting discriminative power is of interest. This often leads to misunderstandings between researchers coming from the fields of data mining and machine learning, since they use the same terms and often the same algorithms, but have different goals.

Cluster analysis was originated in anthropology by Driver and Kroeber in 1932 and introduced to psychology by Zubin in 1938 and Robert Tryon in 1939[1][2] and famously used by Cattell beginning in 1943[3] for trait theory classification in personality psychology.

4.1 Definition

According to Vladimir Estivill-Castro, the notion of a "cluster" cannot be precisely defined, which is one of the reasons why there are so many clustering algorithms.[4] There is a common denominator: a group of data objects. However, different researchers employ different cluster models, and for each of these cluster models again different algorithms can be given. The notion of a cluster, as found by different algorithms, varies significantly in its properties. Understanding these "cluster models" is key to understanding the differences between the various algorithms. Typical cluster models include:

- Connectivity models: for example hierarchical clustering builds models based on distance connectivity.
- Centroid models: for example the k-means algorithm represents each cluster by a single mean vector.
- Distribution models: clusters are modeled using statistical distributions, such as multivariate normal distributions used by the Expectation-maximization algorithm.

- Density models: for example DBSCAN and OPTICS defines clusters as connected dense regions in the data space.
- Subspace models: in Biclustering (also known as Co-clustering or two-mode-clustering), clusters are modeled with both cluster members and relevant attributes.
- Group models: some algorithms do not provide a refined model for their results and just provide the grouping information.
- Graph-based models: a clique, i.e., a subset of nodes in a graph such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster. Relaxations of the complete connectivity requirement (a fraction of the edges can be missing) are known as quasi-cliques.

A “clustering” is essentially a set of such clusters, usually containing all objects in the data set. Additionally, it may specify the relationship of the clusters to each other, for example a hierarchy of clusters embedded in each other. Clusterings can be roughly distinguished as:

- hard clustering: each object belongs to a cluster or not
- soft clustering (also: fuzzy clustering): each object belongs to each cluster to a certain degree (e.g. a likelihood of belonging to the cluster)

There are also finer distinctions possible, for example:

- strict partitioning clustering: here each object belongs to exactly one cluster
- strict partitioning clustering with outliers: objects can also belong to no cluster, and are considered outliers.
- overlapping clustering (also: alternative clustering, multi-view clustering): while usually a hard clustering, objects may belong to more than one cluster.
- hierarchical clustering: objects that belong to a child cluster also belong to the parent cluster
- subspace clustering: while an overlapping clustering, within a uniquely defined subspace, clusters are not expected to overlap.

4.2 Algorithms

Main category: Data clustering algorithms

Clustering algorithms can be categorized based on their cluster model, as listed above. The following overview will only list the most prominent examples of clustering algorithms, as there are possibly over 100 published clustering algorithms. Not all provide models for their clusters and can thus not easily be categorized. An overview of algorithms explained in Wikipedia can be found in the list of statistics algorithms.

There is no objectively “correct” clustering algorithm, but as it was noted, “clustering is in the eye of the beholder.”[4] The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another. It should be noted that an algorithm that is designed for one kind of model has no chance on a data set that contains a radically different kind of model.[4] For example, k-means cannot find non-convex clusters.[4]

4.2.1 Connectivity based clustering (hierarchical clustering)

Main article: Hierarchical clustering

Connectivity based clustering, also known as *hierarchical clustering*, is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect “objects” to form “clusters” based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster. At different distances, different clusters will form, which can be represented using a dendrogram, which explains where the common name “hierarchical clustering” comes from: these algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances. In a dendrogram, the y-axis marks the distance at which the clusters merge, while the objects are placed along the x-axis such that the clusters don't mix.

Connectivity based clustering is a whole family of methods that differ by the way distances are computed. Apart from the usual choice of distance functions, the user also needs to decide on the linkage criterion (since a cluster consists of multiple objects, there are multiple candidates to compute the distance to) to use. Popular choices are known as single-linkage clustering (the minimum of object distances), complete linkage clustering (the maximum of object distances) or UPGMA (“Unweighted Pair Group Method with Arithmetic Mean”, also known as average linkage clustering). Furthermore, hierarchical clustering can be agglomerative (starting with single elements and aggregating them into clusters) or divisive (starting with the complete data set and dividing it into partitions).

These methods will not produce a unique partitioning of the data set, but a hierarchy from which the user still needs to choose appropriate clusters. They are not very robust towards outliers, which will either show up as additional clusters or even cause other clusters to merge (known as "chaining phenomenon", in particular with single-linkage clustering). In the general case, the complexity is $\mathcal{O}(n^3)$, which makes them too slow for large data sets. For some special cases, optimal efficient methods (of complexity $\mathcal{O}(n^2)$) are known: SLINK[5] for single-linkage and CLINK[6] for complete-linkage clustering. In the data mining community these methods are recognized as a theoretical foundation of cluster analysis, but often considered obsolete. They did however provide inspiration for many later methods such as density based clustering.

- Linkage clustering examples
- Single-linkage on Gaussian data. At 35 clusters, the biggest cluster starts fragmenting into smaller parts, while before it was still connected to the second largest due to the single-link effect.
- Single-linkage on density-based clusters. 20 clusters extracted, most of which contain single elements, since linkage clustering does not have a notion of "noise".

4.2.2 Centroid-based clustering

Main article: k-means clustering

In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, *k*-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

The optimization problem itself is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximative method is Lloyd's algorithm,[7] often actually referred to as "*k-means algorithm*". It does however only find a local optimum, and is commonly run multiple times with different random initializations. Variations of k-means often include such optimizations as choosing the best of multiple runs, but also restricting the centroids to members of the data set (k-medoids), choosing medians (k-medians clustering), choosing the initial centers less randomly (K-means++) or allowing a fuzzy cluster assignment (Fuzzy c-means).

Most k-means-type algorithms require the number of clusters - k - to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. This often leads to incorrectly cut borders in between of clusters (which is not surprising, as the algorithm optimized cluster centers, not cluster borders).

K-means has a number of interesting theoretical properties. On the one hand, it partitions the data space into a structure known as a Voronoi diagram. On the other hand, it is conceptually close to nearest neighbor classification, and as such is popular in machine learning. Third, it can be seen as a variation of model based classification, and Lloyd's algorithm as a variation of the Expectation-maximization algorithm for this model discussed below.

- k-Means clustering examples
- K-means separates data into Voronoi-cells, which assumes equal-sized clusters (not adequate here)
- K-means cannot represent density-based clusters

4.2.3 Distribution-based clustering

The clustering model most closely related to statistics is based on distribution models. Clusters can then easily be defined as objects belonging most likely to the same distribution. A convenient property of this approach is that this closely resembles the way artificial data sets are generated: by sampling random objects from a distribution.

While the theoretical foundation of these methods is excellent, they suffer from one key problem known as overfitting, unless constraints are put on the model complexity. A more complex model will usually be able to explain the data better, which makes choosing the appropriate model complexity inherently difficult.

One prominent method is known as Gaussian mixture models (using the expectation-maximization algorithm). Here, the data set is usually modelled with a fixed (to avoid overfitting) number of Gaussian distributions that are initialized randomly and whose parameters are iteratively optimized to fit better to the data set. This will converge to a local optimum, so multiple runs may produce different results. In order to obtain a hard clustering, objects are often then assigned to the Gaussian distribution they most likely belong to; for soft clusterings, this is not necessary.

Distribution-based clustering produces complex models for clusters that can capture correlation and dependence between attributes. However, these algorithms put an extra burden on the user: for many real data sets, there may be no concisely defined mathematical model (e.g. assuming Gaussian distributions is a rather strong assumption on the data).

- Expectation-Maximization (EM) clustering examples
- On Gaussian-distributed data, EM works well, since it uses Gaussians for modelling clusters

- Density-based clusters cannot be modeled using Gaussian distributions

4.2.4 Density-based clustering

In density-based clustering,[8] clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points.

The most popular[9] density based clustering method is DBSCAN.[10] In contrast to many newer methods, it features a well-defined cluster model called "density-reachability". Similar to linkage based clustering, it is based on connecting points within certain distance thresholds. However, it only connects points that satisfy a density criterion, in the original variant defined as a minimum number of other objects within this radius. A cluster consists of all density-connected objects (which can form a cluster of an arbitrary shape, in contrast to many other methods) plus all objects that are within these objects' range. Another interesting property of DBSCAN is that its complexity is fairly low - it requires a linear number of range queries on the database - and that it will discover essentially the same results (it is deterministic for core and noise points, but not for border points) in each run, therefore there is no need to run it multiple times. OPTICS[11] is a generalization of DBSCAN that removes the need to choose an appropriate value for the range parameter ε , and produces a hierarchical result related to that of linkage clustering. DeLi-Clu,[12] Density-Link-Clustering combines ideas from single-linkage clustering and OPTICS, eliminating the ε parameter entirely and offering performance improvements over OPTICS by using an R-tree index.

The key drawback of DBSCAN and OPTICS is that they expect some kind of density drop to detect cluster borders. Moreover, they cannot detect intrinsic cluster structures which are prevalent in the majority of real life data. A variation of DBSCAN, EnDBSCAN,[13] efficiently detects such kinds of structures. On data sets with, for example, overlapping Gaussian distributions - a common use case in artificial data - the cluster borders produced by these algorithms will often look arbitrary, because the cluster density decreases continuously. On a data set consisting of mixtures of Gaussians, these algorithms are nearly always outperformed by methods such as EM clustering that are able to precisely model this kind of data.

Mean-shift is a clustering approach where each object is moved to the densest area in its vicinity, based on kernel density estimation. Eventually, objects converge to local maxima of density. Similar to k-means clustering, these "density attractors" can serve as representatives for the data set, but mean-shift can detect arbitrary-shaped clusters similar to DBSCAN. Due to the expensive iterative procedure and density estimation, mean-shift is usually slower than DBSCAN or k-Means.

- Density-based clustering examples
- Density-based clustering with DBSCAN.
- DBSCAN assumes clusters of similar density, and may have problems separating nearby clusters
- OPTICS is a DBSCAN variant that handles different densities much better

4.2.5 Recent developments

In recent years considerable effort has been put into improving the performance of existing algorithms.[14][15] Among them are *CLARANS* (Ng and Han, 1994),[16] and *BIRCH* (Zhang et al., 1996).[17] With the recent need to process larger and larger data sets (also known as big data), the willingness to trade semantic meaning of the generated clusters for performance has been increasing. This led to the development of pre-clustering methods such as canopy clustering, which can process huge data sets efficiently, but the resulting "clusters" are merely a rough pre-partitioning of the data set to then analyze the partitions with existing slower methods such as k-means clustering. Various other approaches to clustering have been tried such as seed based clustering.[18]

For high-dimensional data, many of the existing methods fail due to the curse of dimensionality, which renders particular distance functions problematic in high-dimensional spaces. This led to new clustering algorithms for high-dimensional data that focus on subspace clustering (where only some attributes are used, and cluster models include the relevant attributes for the cluster) and correlation clustering that also looks for arbitrary rotated ("correlated") subspace clusters that can be modeled by giving a correlation of their attributes. Examples for such clustering algorithms are CLIQUE[19] and SUBCLU.[20]

Ideas from density-based clustering methods (in particular the DBSCAN/OPTICS family of algorithms) have been adopted to subspace clustering (HiSC,[21] hierarchical subspace clustering and DiSH[22]) and correlation clustering (HiCO,[23] hierarchical correlation clustering, 4C[24] using "correlation connectivity" and ERiC[25] exploring hierarchical density-based correlation clusters).

Several different clustering systems based on mutual information have been proposed. One is Marina Meilă's *variation of information* metric;[26] another provides hierarchical clustering.[27] Using genetic algorithms, a wide range of different fit-functions can be optimized, including mutual information.[28] Also message passing algorithms, a recent development in Computer Science and Statistical Physics, has led to the creation of new types of clustering algorithms.[29]

4.2.6 Other methods

- Basic sequential algorithmic scheme (BSAS)

4.3 Evaluation and assessment

Evaluation of clustering results sometimes is referred to as cluster validation.

There have been several suggestions for a measure of similarity between two clusterings. Such a measure can be used to compare how well different data clustering algorithms perform on a set of data. These measures are usually tied to the type of criterion being considered in assessing the quality of a clustering method.

4.3.1 Internal evaluation

When a clustering result is evaluated based on the data that was clustered itself, this is called internal evaluation. These methods usually assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters. One drawback of using internal criteria in cluster evaluation is that high scores on an internal measure do not necessarily result in effective information retrieval applications.[30] Additionally, this evaluation is biased towards algorithms that use the same cluster model. For example k-Means clustering naturally optimizes object distances, and a distance-based internal criterion will likely overrate the resulting clustering.

Therefore, the internal evaluation measures are best suited to get some insight into situations where one algorithm performs better than another, but this shall not imply that one algorithm produces more valid results than another.[4] Validity as measured by such an index depends on the claim that this kind of structure exists in the data set. An algorithm designed for some kind of models has no chance if the data set contains a radically different set of models, or if the evaluation measures a radically different criterion.[4] For example, k-means clustering can only find convex clusters, and many evaluation indexes assume convex clusters. On a data set with non-convex clusters neither the use of k-means, nor of an evaluation criterion that assumes convexity, is sound.

The following methods can be used to assess the quality of clustering algorithms based on internal criterion:

- **Davies–Bouldin index**

  The Davies–Bouldin index can be calculated by the following formula:

  $DB = \frac{1}{n}\sum_{i=1}^{n}\max_{j\neq i}\left(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\right)$

  where n is the number of clusters, c_x is the centroid of cluster x , σ_x is the average distance of all elements in cluster x to centroid c_x , and $d(c_i, c_j)$ is the distance between centroids c_i and c_j . Since algorithms that produce clusters with low intra-cluster distances (high intra-cluster similarity) and high inter-cluster distances (low inter-cluster similarity) will have a low Davies–Bouldin index, the clustering algorithm that produces a collection of clusters with the smallest Davies–Bouldin index is considered the best algorithm based on this criterion.

- **Dunn index**

  The Dunn index aims to identify dense and well-separated clusters. It is defined as the ratio between the minimal inter-cluster distance to maximal intra-cluster distance. For each cluster partition, the Dunn index can be calculated by the following formula:[31]

  $D = \frac{\min_{1\leq i<j\leq n} d(i,j)}{\max_{1\leq k\leq n} d'(k)}$,

  where *d*(*i*,*j*) represents the distance between clusters *i* and *j*, and *d* '(*k*) measures the intra-cluster distance of cluster *k*. The inter-cluster distance *d*(*i*,*j*) between two clusters may be any number of distance measures, such as the distance between the centroids of the clusters. Similarly, the intra-cluster distance *d* '(*k*) may be measured in a variety ways, such as the maximal distance between any pair of elements in cluster *k*. Since internal criterion seek clusters with high intra-cluster similarity and low inter-cluster similarity, algorithms that produce clusters with high Dunn index are more desirable.

- Silhouette coefficient

  The silhouette coefficient contrasts the average distance to elements in the same cluster with the average distance to elements in other clusters. Objects with a high silhouette value are considered well clustered, objects with a low value may be outliers. This index works well with k-means clustering, and is also used to determine the optimal number of clusters.

4.3.2 External evaluation

In external evaluation, clustering results are evaluated based on data that was not used for clustering, such as known class labels and external benchmarks. Such benchmarks consist of a set of pre-classified items, and these sets are often created by human (experts). Thus, the benchmark sets can be thought of as a gold standard for evaluation. These types of evaluation methods measure how close the clustering is to the predetermined benchmark classes. However, it has recently been discussed whether this is adequate for real data, or only on synthetic data sets with a factual ground truth, since classes can contain internal structure, the attributes present may not allow separation of clusters or the classes may contain anomalies.[32] Additionally, from a knowledge discovery point of view, the reproduction of known knowledge may not necessarily be the intended result.[32]

A number of measures are adapted from variants used to evaluate classification tasks. In place of counting the number of times a class was correctly assigned to a single data point (known as true positives), such *pair counting* metrics assess whether each pair of data points that is truly in the same cluster is predicted to be in the same cluster.

Some of the measures of quality of a cluster algorithm using external criterion include:

- **Rand measure** (William M. Rand 1971)[33]

  The Rand index computes how similar the clusters (returned by the clustering algorithm) are to the benchmark classifications. One can also view the Rand index as a measure of the percentage of correct decisions made by the algorithm. It can be computed using the following formula:

  $RI = \frac{TP+TN}{TP+FP+FN+TN}$

  where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. One issue with the Rand index is that false positives and false negatives are equally weighted. This may be an undesirable characteristic for some clustering applications. The F-measure addresses this concern, as does the chance-corrected adjusted Rand index.

- **F-measure**

  The F-measure can be used to balance the contribution of false negatives by weighting recall through a parameter $\beta \geq 0$. Let precision and recall be defined as follows:

  $P = \frac{TP}{TP+FP}$

  $R = \frac{TP}{TP+FN}$

  where P is the precision rate and R is the recall rate. We can calculate the F-measure by using the following formula:[30]

  $F_\beta = \frac{(\beta^2+1)\cdot P \cdot R}{\beta^2 \cdot P + R}$

  Notice that when $\beta = 0$, $F_0 = P$. In other words, recall has no impact on the F-measure when $\beta = 0$, and increasing β allocates an increasing amount of weight to recall in the final F-measure.

- **Jaccard index**

  The Jaccard index is used to quantify the similarity between two datasets. The Jaccard index takes on a value between 0 and 1. An index of 1 means that the two dataset are identical, and an index of 0 indicates that the datasets have no common elements. The Jaccard index is defined by the following formula:

  $J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP+FP+FN}$

  This is simply the number of unique elements common to both sets divided by the total number of unique elements in both sets.

- **Fowlkes–Mallows index** (E. B. Fowlkes & C. L. Mallows 1983)[34]

  The Fowlkes-Mallows index computes the similarity between the clusters returned by the clustering algorithm and the benchmark classifications. The higher the value of the Fowlkes-Mallows index the more similar the clusters and the benchmark classifications are. It can be computed using the following formula:

  $FM = \sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}$

  where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. The FM index is the geometric mean of the precision and recall P and R , while the F-measure is their harmonic mean.[35] Moreover, precision and recall are also known as Wallace's indices B^I and B^{II} .[36]

- The **Mutual Information** is an information theoretic measure of how much information is shared between a clustering and a ground-truth classification that can detect a non-linear similarity between two clusterings. Adjusted mutual information is the corrected-for-chance variant of this that has a reduced bias for varying cluster numbers.

- **Confusion matrix**

  A confusion matrix can be used to quickly visualize the results of a classification (or clustering) algorithm. It shows how different a cluster is from the gold standard cluster.

4.4 Applications

4.5 See also

4.5.1 Specialized types of cluster analysis

Others

Social science

Computer science

World wide web

Business and marketing

Medicine

Biology, computational biology and bioinformatics

Plant and animal ecology cluster analysis is used to describe and to make spatial and temporal comparisons of communities (assemblages) of organisms in heterogeneous environments; it is also used in plant systematics to generate artificial phylogenies or clusters of organisms (individuals) at the species, genus or higher level that share a number of attributes

Transcriptomics clustering is used to build groups of genes with related expression patterns (also known as coexpressed genes). Often such groups contain functionally related proteins, such as enzymes for a specific pathway, or genes that are co-regulated. High throughput experiments using expressed sequence tags (ESTs) or DNA microarrays can be a powerful tool for genome annotation, a general aspect of genomics.

Medical imaging

Sequence analysis clustering is used to group homologous sequences into gene families. This is a very important concept in bioinformatics, and evolutionary biology in general. See evolution by gene duplication.

High-throughput genotyping platforms clustering algorithms are used to automatically assign genotypes.

Market research

Human genetic clustering The similarity of genetic data is used in clustering to infer population structures.

On PET scans, cluster analysis can be used to differentiate between different types of tissue and blood in a three-dimensional image. In this application, actual position does not matter, but the voxel intensity is considered as a vector, with a dimension for each image that was taken over time. This technique allows, for example, accurate measurement of the rate a radioactive tracer is delivered to the area of interest, without a separate sampling of arterial blood, an intrusive technique that is most common today.

Social network analysis

Software evolution

Analysis of antimicrobial activity Cluster analysis can be used to analyse patterns of antibiotic resistance, to classify antimicrobial compounds according to their mechanism of action, to classify antibiotics according to their antibacterial activity.

IMRT segmentation Clustering can be used to divide a fluence map into distinct regions for conversion into deliverable fields in MLC-based Radiation Therapy.

Crime analysis

Clustering high-dimensional data

- Conceptual clustering
- Consensus clustering
- Constrained clustering
- Data stream clustering
- Sequence clustering
- Spectral clustering

4.5.2 Techniques used in cluster analysis

- Artificial neural network (ANN)
- Nearest neighbor search
- Neighbourhood components analysis
- Latent class analysis

4.5.3 Data projection and preprocessing

- Dimension reduction
- Principal component analysis
- Multidimensional scaling

4.5.4 Other

- Cluster-weighted modeling
- Curse of dimensionality
- Determining the number of clusters in a data set
- Parallel coordinates
- Structured data analysis

4.6 References

[1] Bailey, Ken (1994). "Numerical Taxonomy and Cluster Analysis". *Typologies and Taxonomies*. p. 34. ISBN 9780803952591.

[2] Tryon, Robert C. (1939). *Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality*. Edwards Brothers.

[3] Cattell, R. B. (1943). "The description of personality: Basic traits resolved into clusters". *Journal of Abnormal and Social Psychology* **38**: 476–506. doi:10.1037/h0054116.

[4] Estivill-Castro, Vladimir (20 June 2002). "Why so many clustering algorithms — A Position Paper". *ACM SIGKDD Explorations Newsletter* **4** (1): 65–75. doi:10.1145/568574.568575.

[5] Sibson, R. (1973). "SLINK: an optimally efficient algorithm for the single-link cluster method" (PDF). *The Computer Journal* (British Computer Society) **16** (1): 30–34. doi:10.1093/comjnl/16.1.30.

[6] Defays, D. (1977). "An efficient algorithm for a complete link method". *The Computer Journal* (British Computer Society) **20** (4): 364–366. doi:10.1093/comjnl/20.4.364.

[7] Lloyd, S. (1982). "Least squares quantization in PCM". *IEEE Transactions on Information Theory* **28** (2): 129–137. doi:10.1109/TIT.1982.1056489.

[8] Kriegel, Hans-Peter; Kröger, Peer; Sander, Jörg; Zimek, Arthur (2011). "Density-based Clustering". *WIREs Data Mining and Knowledge Discovery* **1** (3): 231–240. doi:10.1002/widm.30.

[9] Microsoft academic search: most cited data mining articles: DBSCAN is on rank 24, when accessed on: 4/18/2010

[10] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press. pp. 226–231. ISBN 1-57735-004-9. CiteSeerX: 10.1.1.71.1980.

[11] Ankerst, Mihael; Breunig, Markus M.; Kriegel, Hans-Peter; Sander, Jörg (1999). "OPTICS: Ordering Points To Identify the Clustering Structure". *ACM SIGMOD international conference on Management of data*. ACM Press. pp. 49–60. CiteSeerX: 10.1.1.129.6542.

[12] Achtert, E.; Böhm, C.; Kröger, P. (2006). "DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking". *LNCS: Advances in Knowledge Discovery and Data Mining*. Lecture Notes in Computer Science **3918**: 119–128. doi:10.1007/11731139_16. ISBN 978-3-540-33206-0.

[13] Roy, S.; Bhattacharyya, D. K. (2005). "An Approach to find Embedded Clusters Using Density Based Techniques". *LNCS Vol.3816*. Springer Verlag. pp. 523–535.

[14] Sculley, D. (2010). *Web-scale k-means clustering*. Proc. 19th WWW.

[15] Huang, Z. (1998). "Extensions to the *k*-means algorithm for clustering large data sets with categorical values". *Data Mining and Knowledge Discovery* **2**: 283–304.

[16] R. Ng and J. Han. "Efficient and effective clustering method for spatial data mining". In: Proceedings of the 20th VLDB Conference, pages 144-155, Santiago, Chile, 1994.

[17] Tian Zhang, Raghu Ramakrishnan, Miron Livny. "An Efficient Data Clustering Method for Very Large Databases." In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 103–114.

[18] Can, F.; Ozkarahan, E. A. (1990). "Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases". *ACM Transactions on Database Systems* **15** (4): 483. doi:10.1145/99935.99938.

[19] Agrawal, R.; Gehrke, J.; Gunopulos, D.; Raghavan, P. (2005). "Automatic Subspace Clustering of High Dimensional Data". *Data Mining and Knowledge Discovery* **11**: 5. doi:10.1007/s10618-005-1396-1.

[20] Karin Kailing, Hans-Peter Kriegel and Peer Kröger. *Density-Connected Subspace Clustering for High-Dimensional Data*. In: *Proc. SIAM Int. Conf. on Data Mining (SDM'04)*, pp. 246-257, 2004.

[21] Achtert, E.; Böhm, C.; Kriegel, H. P.; Kröger, P.; Müller-Gorman, I.; Zimek, A. (2006). "Finding Hierarchies of Subspace Clusters". *LNCS: Knowledge Discovery in Databases: PKDD 2006*. Lecture Notes in Computer Science **4213**: 446–453. doi:10.1007/11871637_42. ISBN 978-3-540-45374-1.

[22] Achtert, E.; Böhm, C.; Kriegel, H. P.; Kröger, P.; Müller-Gorman, I.; Zimek, A. (2007). "Detection and Visualization of Subspace Cluster Hierarchies". *LNCS: Advances in Databases: Concepts, Systems and Applications*. Lecture Notes in Computer Science **4443**: 152–163. doi:10.1007/978-3-540-71703-4_15. ISBN 978-3-540-71702-7.

[23] Achtert, E.; Böhm, C.; Kröger, P.; Zimek, A. (2006). "Mining Hierarchies of Correlation Clusters". *Proc. 18th International Conference on Scientific and Statistical Database Management (SSDBM)*: 119–128. doi:10.1109/SSDBM.2006.35. ISBN 0-7695-2590-3.

[24] Böhm, C.; Kailing, K.; Kröger, P.; Zimek, A. (2004). "Computing Clusters of Correlation Connected objects". *Proceedings of the 2004 ACM SIGMOD international conference on Management of data - SIGMOD '04*. p. 455. doi:10.1145/1007568.1007620. ISBN 1581138598.

[25] Achtert, E.; Bohm, C.; Kriegel, H. P.; Kröger, P.; Zimek, A. (2007). "On Exploring Complex Relationships of Correlation Clusters". *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*. p. 7. doi:10.1109/SSDBM.2007.21. ISBN 0-7695-2868-6.

[26] Meilă, Marina (2003). "Comparing Clusterings by the Variation of Information". *Learning Theory and Kernel Machines*. Lecture Notes in Computer Science **2777**: 173–187. doi:10.1007/978-3-540-45167-9_14. ISBN 978-3-540-40720-1.

[27] Kraskov, Alexander; Stögbauer, Harald; Andrzejak, Ralph G.; Grassberger, Peter (1 December 2003) [28 November 2003]. "Hierarchical Clustering Based on Mutual Information". arXiv:q-bio/0311039.

[28] Auffarth, B. (July 18–23, 2010). "Clustering by a Genetic Algorithm with Biased Mutation Operator". *WCCI CEC* (IEEE). CiteSeerX: 10.1.1.170.869.

[29] Frey, B. J.; Dueck, D. (2007). "Clustering by Passing Messages Between Data Points". *Science* **315** (5814): 972–976. doi:10.1126/science.1136800. PMID 17218491.

[30] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. *Introduction to Information Retrieval*. Cambridge University Press. ISBN 978-0-521-86571-5.

[31] Dunn, J. (1974). "Well separated clusters and optimal fuzzy partitions". *Journal of Cybernetics* **4**: 95–104. doi:10.1080/01969727408546059.

[32] Färber, Ines; Günnemann, Stephan; Kriegel, Hans-Peter; Kröger, Peer; Müller, Emmanuel; Schubert, Erich; Seidl, Thomas; Zimek, Arthur (2010). "On Using Class-Labels in Evaluation of Clusterings" (PDF). In Fern, Xiaoli Z.; Davidson, Ian; Dy, Jennifer. *MultiClust: Discovering, Summarizing, and Using Multiple Clusterings*. ACM SIGKDD.

[33] Rand, W. M. (1971). "Objective criteria for the evaluation of clustering methods". *Journal of the American Statistical Association* (American Statistical Association) **66** (336): 846–850. doi:10.2307/2284239. JSTOR 2284239.

[34] E. B. Fowlkes & C. L. Mallows (1983), "A Method for Comparing Two Hierarchical Clusterings", Journal of the American Statistical Association 78, 553–569.

[35] L. Hubert et P. Arabie. Comparing partitions. J. of Classification, 2(1), 1985.

[36] D. L. Wallace. Comment. Journal of the American Statistical Association, 78 :569– 579, 1983.

[37] Bewley, A. et al. "Real-time volume estimation of a dragline payload". *IEEE International Conference on Robotics and Automation* **2011**: 1571–1576.

[38] Basak, S.C.; Magnuson, V.R.; Niemi, C.J.; Regal, R.R. "Determining Structural Similarity of Chemicals Using Graph Theoretic Indices". *Discr. Appl. Math., 19* **1988**: 17–44.

[39] Huth, R. et al. (2008). "Classifications of Atmospheric Circulation Patterns: Recent Advances and Applications". *Ann. N.Y. Acad. Sci.* **1146**: 105–152.

4.7 External links

- Data Mining at DMOZ

Chapter 5

Anomaly detection

In data mining, **anomaly detection** (or **outlier detection**) is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset.[1] Typically the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or finding errors in text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions.[2]

In particular in the context of abuse and network intrusion detection, the interesting objects are often not *rare* objects, but unexpected *bursts* in activity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular unsupervised methods) will fail on such data, unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro clusters formed by these patterns.[3]

Three broad categories of anomaly detection techniques exist. **Unsupervised anomaly detection** techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set. **Supervised anomaly detection** techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection). **Semi-supervised anomaly detection** techniques construct a model representing normal behavior from a given *normal* training data set, and then testing the likelihood of a test instance to be generated by the learnt model.

5.1 Applications

Anomaly detection is applicable in a variety of domains, such as intrusion detection, fraud detection, fault detection, system health monitoring, event detection in sensor networks, and detecting Eco-system disturbances. It is often used in preprocessing to remove anomalous data from the dataset. In supervised learning, removing the anomalous data from the dataset often results in a statistically significant increase in accuracy.[4][5]

5.2 Popular techniques

Several anomaly detection techniques have been proposed in literature. Some of the popular techniques are:

- Density-based techniques (k-nearest neighbor,[6][7][8] local outlier factor,[9] and many more variations of this concept[10]).
- Subspace-[11] and correlation-based [12] outlier detection for high-dimensional data.[13]
- One class support vector machines.[14]
- Replicator neural networks.
- Cluster analysis based outlier detection.[15]
- Deviations from association rules and frequent itemsets.
- Fuzzy logic based outlier detection.
- Ensemble techniques, using feature bagging,[16][17] score normalization[18][19] and different sources of diversity.[20][21]

5.3 Application to data security

Anomaly detection was proposed for Intrusion detection systems (IDS) by Dorothy Denning in 1986.[22] Anomaly detection for IDS is normally accomplished with thresholds and statistics, but can also be done with Soft computing, and inductive learning.[23] Types of statistics proposed by 1999 included profiles of users, workstations, networks, remote hosts, groups of users, and programs based on frequencies, means, variances, covariances, and standard deviations.[24] The counterpart of anomaly detection in intrusion detection is misuse detection.

5.4 Software

- ELKI is an open-source Java data mining toolkit that contains several anomaly detection algorithms, as well as index acceleration for them.

5.5 See also

- Outliers in statistics
- Change detection
- Novelty detection

5.6 References

[1] Chandola, V.; Banerjee, A.; Kumar, V. (2009). "Anomaly detection: A survey" (PDF). *ACM Computing Surveys* **41** (3): 1. doi:10.1145/1541880.1541882.

[2] Hodge, V. J.; Austin, J. (2004). "A Survey of Outlier Detection Methodologies" (PDF). *Artificial Intelligence Review* **22** (2): 85. doi:10.1007/s10462-004-4304-y.

[3] Dokas, Paul; Ertoz, Levent; Kumar, Vipin; Lazarevic, Aleksandar; Srivastava, Jaideep; Tan, Pang-Ning (2002). "Data mining for network intrusion detection" (PDF). *Proceedings NSF Workshop on Next Generation Data Mining*.

[4] Tomek, Ivan (1976). "An Experiment with the Edited Nearest-Neighbor Rule". *IEEE Transactions on Systems, Man, and Cybernetics* **6** (6): 448. doi:10.1109/TSMC.1976.4309523.

[5] Smith, M. R.; Martinez, T. (2011). "Improving classification accuracy by identifying and removing instances that should be misclassified". *The 2011 International Joint Conference on Neural Networks* (PDF). p. 2690. doi:10.1109/IJCNN.2011.6033571. ISBN 978-1-4244-9635-8.

[6] Knorr, E. M.; Ng, R. T.; Tucakov, V. (2000). "Distance-based outliers: Algorithms and applications". *The VLDB Journal the International Journal on Very Large Data Bases* **8** (3–4): 237. doi:10.1007/s007780050006.

[7] Ramaswamy, S.; Rastogi, R.; Shim, K. (2000). *Efficient algorithms for mining outliers from large data sets*. Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00. p. 427. doi:10.1145/342009.335437. ISBN 1581132174.

[8] Angiulli, F.; Pizzuti, C. (2002). *Fast Outlier Detection in High Dimensional Spaces*. Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science **2431**. p. 15. doi:10.1007/3-540-45681-3_2. ISBN 978-3-540-44037-6.

[9] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; Sander, J. (2000). *LOF: Identifying Density-based Local Outliers* (PDF). *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD: 93–104. doi:10.1145/335191.335388. ISBN 1-58113-217-4.

[10] Schubert, E.; Zimek, A.; Kriegel, H. -P. (2012). "Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection". *Data Mining and Knowledge Discovery*. doi:10.1007/s10618-012-0300-z.

[11] Kriegel, H. P.; Kröger, P.; Schubert, E.; Zimek, A. (2009). *Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data*. Advances in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science **5476**. p. 831. doi:10.1007/978-3-642-01307-2_86. ISBN 978-3-642-01306-5.

[12] Kriegel, H. P.; Kroger, P.; Schubert, E.; Zimek, A. (2012). *Outlier Detection in Arbitrarily Oriented Subspaces*. 2012 IEEE 12th International Conference on Data Mining. p. 379. doi:10.1109/ICDM.2012.21. ISBN 978-1-4673-4649-8.

[13] Zimek, A.; Schubert, E.; Kriegel, H.-P. (2012). "A survey on unsupervised outlier detection in high-dimensional numerical data". *Statistical Analysis and Data Mining* **5** (5): 363–387. doi:10.1002/sam.11161.

[14] Schölkopf, B.; Platt, J. C.; Shawe-Taylor, J.; Smola, A. J.; Williamson, R. C. (2001). "Estimating the Support of a High-Dimensional Distribution". *Neural Computation* **13** (7): 1443. doi:10.1162/089976601750264965.

[15] He, Z.; Xu, X.; Deng, S. (2003). "Discovering cluster-based local outliers". *Pattern Recognition Letters* **24** (9–10): 1641. doi:10.1016/S0167-8655(03)00003-5.

[16] Lazarevic, A.; Kumar, V. (2005). "Feature bagging for outlier detection". *Proc. 11th ACM SIGKDD international conference on Knowledge Discovery in Data Mining*: 157–166. doi:10.1145/1081870.1081891.

[17] Nguyen, H. V.; Ang, H. H.; Gopalkrishnan, V. (2010). *Mining Outliers with Ensemble of Heterogeneous Detectors on Random Subspaces*. Database Systems for Advanced Applications. Lecture Notes in Computer Science **5981**. p. 368. doi:10.1007/978-3-642-12026-8_29. ISBN 978-3-642-12025-1.

[18] Kriegel, H. P.; Kröger, P.; Schubert, E.; Zimek, A. (2011). *Interpreting and Unifying Outlier Scores* (PDF). Proceedings of the 2011 SIAM International Conference on Data Mining. pp. 13–24. doi:10.1137/1.9781611972818.2. ISBN 978-0-89871-992-5.

[19] Schubert, E.; Wojdanowski, R.; Zimek, A.; Kriegel, H. P. (2012). *On Evaluation of Outlier Rankings and Outlier Scores* (PDF). Proceedings of the 2012 SIAM International Conference on Data Mining. pp. 1047–1058. doi:10.1137/1.9781611972825.90. ISBN 978-1-61197-232-0.

[20] Zimek, A.; Campello, R. J. G. B.; Sander, J. R. (2014). "Ensembles for unsupervised outlier detection". *ACM SIGKDD Explorations Newsletter* **15**: 11. doi:10.1145/2594473.2594476.

[21] Zimek, A.; Campello, R. J. G. B.; Sander, J. R. (2014). *Data perturbation for outlier detection ensembles*. Proceedings of the 26th International Conference on Scientific and Statistical Database Management - SSDBM '14. p. 1. doi:10.1145/2618243.2618257. ISBN 9781450327220.

[22] Denning, D. E. (1987). "An Intrusion-Detection Model" (PDF). *IEEE Transactions on Software Engineering* (2): 222. doi:10.1109/TSE.1987.232894. CiteSeerX: 10.1.1.102.5127.

[23] Teng, H. S.; Chen, K.; Lu, S. C. (1990). "Adaptive real-time anomaly detection using inductively generated sequential patterns" (PDF). *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*: 278–284. doi:10.1109/RISP.1990.63857. ISBN 0-8186-2060-9.

[24] Jones, Anita K.; Sielken, Robert S. (1999). "Computer System Intrusion Detection: A Survey". *Technical Report, Department of Computer Science, University of Virginia, Charlottesville, VA*. CiteSeerX: 10.1.1.24.7802.

Chapter 6

Association rule learning

Association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness.[1] Based on the concept of strong rules, Rakesh Agrawal et al.[2] introduced association rules for discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets. For example, the rule $\{\mathrm{onions}, \mathrm{potatoes}\} \Rightarrow \{\mathrm{burger}\}$ found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as, e.g., promotional pricing or product placements. In addition to the above example from market basket analysis association rules are employed today in many application areas including Web usage mining, intrusion detection, Continuous production, and bioinformatics. In contrast with sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

6.1 Definition

Following the original definition by Agrawal et al.[2] the problem of association rule mining is defined as: Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called *items*. Let $D = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the *database*. Each transaction in D has a unique transaction ID and contains a subset of the items in I . A *rule* is defined as an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$. The sets of items (for short *itemsets*) X and Y are called *antecedent* (left-hand-side or LHS) and *consequent* (right-hand-side or RHS) of the rule respectively.

To illustrate the concepts, we use a small example from the supermarket domain. The set of items is $I = \{\mathrm{milk}, \mathrm{bread}, \mathrm{butter}, \mathrm{beer}, \mathrm{diapers}\}$ and in the table to the right is shown a small database containing the items (1 codes presence and 0 codes absence of an item in a transaction). An example rule for the supermarket could be $\{\mathrm{butter}, \mathrm{bread}\} \Rightarrow \{\mathrm{milk}\}$ meaning that if butter and bread are bought, customers also buy milk.

Note: this example is extremely small. In practical applications, a rule needs a support of several hundred transactions before it can be considered statistically significant, and datasets often contain thousands or millions of transactions.

6.2 Useful Concepts

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best-known constraints are minimum thresholds on support and confidence.

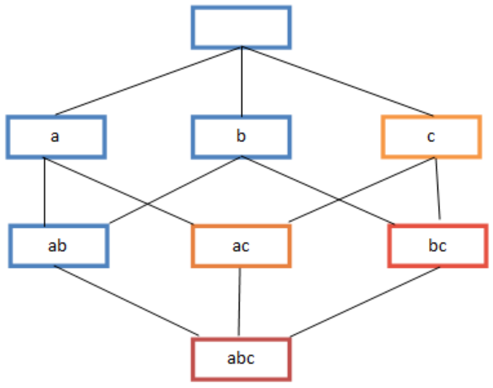
- The *support* $\mathrm{supp}(X)$ of an itemset X is defined as the proportion of transactions in the database which contain the itemset. In the example database, the itemset $\{\mathrm{milk}, \mathrm{bread}, \mathrm{butter}\}$ has a support of $1/5 = 0.2$ since it occurs in 20% of all transactions (1 out of 5 transactions). The argument of supp() is a set of preconditions, and thus becomes more restrictive as it grows (instead of more inclusive).

- The *confidence* of a rule is defined as $\mathrm{conf}(X \Rightarrow Y) = \mathrm{supp}(X \cup Y)/\mathrm{supp}(X)$. For example, the rule $\{\mathrm{butter}, \mathrm{bread}\} \Rightarrow \{\mathrm{milk}\}$ has a confidence of $0.2/0.2 = 1.0$ in the database, which means that for 100% of the transactions containing butter and bread the rule is correct (100% of the times a customer buys butter and bread, milk is bought as well). Note that $supp(X \cup Y)$ means the support of the union of the items in X and Y. This is somewhat confusing since we normally think in terms of probabilities of events and not sets of items. We can rewrite $supp(X \cup Y)$ as the joint probability $P(E_X \cap E_Y)$, where E_X and E_Y are the events that a transaction contains itemset X or Y , respectively.[3] Thus confidence can be interpreted as an estimate of the conditional probability $P(E_Y | E_X)$, the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.[4]

- The *lift* of a rule is defined as $\mathrm{lift}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X) \times \mathrm{supp}(Y)}$ or the ratio of the observed support to that expected if X and Y were independent. The rule $\{\mathrm{milk}, \mathrm{bread}\} \Rightarrow \{\mathrm{butter}\}$ has a lift of $\frac{0.2}{0.4 \times 0.4} = 1.25$.

- The *conviction* of a rule is defined as $\mathrm{conv}(X \Rightarrow Y) = \frac{1 - \mathrm{supp}(Y)}{1 - \mathrm{conf}(X \Rightarrow Y)}$. The rule $\{\mathrm{milk}, \mathrm{bread}\} \Rightarrow \{\mathrm{butter}\}$ has a conviction of $\frac{1-0.4}{1-.5} = 1.2$, and can be interpreted as the ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions. In this example, the conviction value of 1.2 shows that the rule $\{\mathrm{milk}, \mathrm{bread}\} \Rightarrow \{\mathrm{butter}\}$ would be incorrect 20% more often (1.2 times as often) if the association between X and Y was purely random chance.

6.3 Process



Frequent itemset lattice, where the color of the box indicates how many transactions contain the combination of items. Note that lower levels of the lattice can contain at most the minimum number of their parents' items; e.g. {ac} can have only at most $min(a, c)$ items. This is called the downward-closure property.*[2]*

Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Association rule generation is usually split up into two separate steps:

1. First, minimum support is applied to find all *frequent itemsets* in a database.
2. Second, these frequent itemsets and the minimum confidence constraint are used to form rules.

While the second step is straightforward, the first step needs more attention.

Finding all frequent itemsets in a database is difficult since it involves searching all possible itemsets (item combinations). The set of possible itemsets is the power set over I and has size $2^n - 1$ (excluding the empty set which is not a valid itemset). Although the size of the powerset grows exponentially in the number of items n in I , efficient search is possible using the ***downward-closure property*** of support[2][5] (also called *anti-monotonicity*[6]) which guarantees that for a frequent itemset, all its subsets are also frequent and thus for an infrequent itemset, all its supersets must also be infrequent. Exploiting this property, efficient algorithms (e.g., Apriori[7] and Eclat[8]) can find all frequent itemsets.

6.4 History

The concept of association rules was popularised particularly due to the 1993 article of Agrawal et al.,[2] which has acquired more than 6000 citations according to Google Scholar, as of March 2008, and is thus one of the most cited papers in the Data Mining field. However, it is possible that what is now called "association rules" is similar to what appears in the 1966 paper[9] on GUHA, a general data mining method developed by Petr Hájek et al.[10]

6.5 Alternative measures of interestingness

In addition to confidence, other measures of *interestingness* for rules have been proposed. Some popular measures are:

- All-confidence[11]
- Collective strength[12]
- Conviction[13]
- Leverage[14]
- Lift (originally called interest)[15]

A definition of these measures can be found here. Several more measures are presented and compared by Tan et al.[16] Looking for techniques that can model what the user has known (and using these models as interestingness measures) is currently an active research trend under the name of "Subjective Interestingness."

6.6 Statistically sound associations

One limitation of the standard approach to discovering associations is that by searching massive numbers of possible associations to look for collections of items that appear to be associated, there is a large risk of finding many spurious associations. These are collections of items that co-occur with unexpected frequency in the data, but only do so by chance. For example, suppose we are considering a collection of 10,000 items and looking for rules containing two items in the left-hand-side and 1 item in the right-hand-side. There are approximately 1,000,000,000,000 such rules. If we apply a statistical test for independence with a significance level of 0.05 it means there is only a 5% chance of accepting a rule if there is no association. If we assume there are no associations, we should nonetheless expect to find 50,000,000,000 rules. Statistically sound association discovery[17][18] controls this risk, in most cases reducing the risk of finding *any* spurious associations to a user-specified significance level.

6.7 Algorithms

Many algorithms for generating association rules were presented over time.

Some well known algorithms are Apriori, Eclat and FP-Growth, but they only do half the job, since they are algorithms for mining frequent itemsets. Another step needs to be done after to generate rules from frequent itemsets found in a database.

6.7.1 Apriori algorithm

Main article: Apriori algorithm

Apriori[7] is the best-known algorithm to mine association rules. It uses a breadth-first search strategy to count the support of itemsets and uses a candidate generation function which exploits the downward closure property of support.

6.7.2 Eclat algorithm

Eclat[8] (alt. ECLAT, stands for Equivalence Class Transformation) is a depth-first search algorithm using set intersection. It is a naturally elegant algorithm suitable for both sequential as well as parallel execution with locality enhancing properties. It was first introduced by Zaki, Parthasarathy, Li and Ogihara in a series of papers written in 1997.

Mohammed Javeed Zaki, Srinivasan Parthasarathy, Wei Li: A Localized Algorithm for Parallel Association Mining. SPAA 1997: 321-330

Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li: Parallel Algorithms for Discovery of Association Rules. Data Min. Knowl. Discov. 1(4): 343-373 (1997)

6.7.3 FP-growth algorithm

FP stands for frequent pattern.

In the first pass, the algorithm counts occurrence of items (attribute-value pairs) in the dataset, and stores them to 'header table'. In the second pass, it builds the FP-tree structure by inserting instances. Items in each instance have to be sorted by descending order of their frequency in the dataset, so that the tree can be processed quickly. Items in each instance that do not meet minimum coverage threshold are discarded. If many instances share most frequent items, FP-tree provides high compression close to tree root.

Recursive processing of this compressed version of main dataset grows large item sets directly, instead of generating candidate items and testing them against the entire database. Growth starts from the bottom of the header table (having longest branches), by finding all instances matching given condition. New tree is created, with counts projected from the original tree corresponding to the set of instances that are conditional on the attribute, with each node getting sum of its children counts. Recursive growth ends when no individual items conditional on the attribute meet minimum support threshold, and processing continues on the remaining header items of the original FP-tree.

Once the recursive process has completed, all large item sets with minimum coverage have been found, and association rule creation begins.[19]

6.7.4 Others

AprioriDP

AprioriDP[20] utilizes Dynamic Programming in Frequent itemset mining. The working principle is to eliminate the candidate generation like FP-tree, but it stores support count in specialized data structure instead of tree.

Context Based Association Rule Mining Algorithm

Main article: Context Based Association Rules

CBPNARM is the newly developed algorithm which is developed in 2013 to mine association rules on the basis of context. It uses context variable on the basis of which the support of an itemset is changed on the basis of which the rules are finally populated to the rule set.

Node-set-based algorithms

FIN,[21] PrePost [22] and PPV [23] are three algorithms based on node sets. They use nodes in a coding FP-tree to represent itemsets, and employ a depth-first search strategy to discovery frequent itemsets using “intersection” of node sets.

GUHA procedure ASSOC

GUHA is a general method for exploratory data analysis that has theoretical foundations in observational calculi.[24]

The ASSOC procedure[25] is a GUHA method which mines for generalized association rules using fast bitstrings operations. The association rules mined by this method are more general than those output by apriori, for example “items” can be connected both with conjunction and disjunctions and the relation between antecedent and consequent of the rule is not restricted to setting minimum support and confidence as in apriori: an arbitrary combination of supported interest measures can be used.

OPUS search

OPUS is an efficient algorithm for rule discovery that, in contrast to most alternatives, does not require either monotone or anti-monotone constraints such as minimum support.[26] Initially used to find rules for a fixed consequent[26][27] it has subsequently been extended to find rules with any item as a consequent.[28] OPUS search is the core technology in the popular Magnum Opus association discovery system.

6.8 Lore

A famous story about association rule mining is the “beer and diaper” story. A purported survey of behavior of supermarket shoppers discovered that customers (presumably young men) who buy diapers tend also to buy beer. This anecdote became popular as an example of how unexpected association rules might be found from everyday data. There are varying opinions as to how much of the story is true.[29] Daniel Powers says:[29]

> In 1992, Thomas Blischok, manager of a retail consulting group at Teradata, and his staff prepared an analysis of 1.2 million market baskets from about 25 Osco Drug stores. Database queries were developed to identify affinities. The analysis “did discover that between 5:00 and 7:00 p.m. that consumers bought beer and diapers”. Osco managers did NOT exploit the beer and diapers relationship by moving the products closer together on the shelves.

6.9 Other types of association mining

Multi-Relation Association Rules: Multi-Relation Association Rules (MRAR) is a new class of association rules which in contrast to primitive, simple and even multi-relational association rules (that are usually extracted from multi-relational databases), each rule item consists of one entity but several relations. These relations indicate indirect relationship between the entities. Consider the following MRAR where the first item consists of three relations *live in*, *nearby* and *humid*: “Those who *live in* a place which is *near by* a city with *humid* climate type and also are *younger* than 20 -> their *health condition* is good”. Such association rules are extractable from RDBMS data or semantic web data.[30]

Context Based Association Rules is a form of association rule. **Context Based Association Rules** claims more accuracy in association rule mining by considering a hidden variable named context variable which changes the final set of association rules depending upon the value of context variables. For example the baskets orientation in market basket analysis reflects an odd pattern in the early days of month.This might be because of abnormal context i.e. salary is drawn at the start of the month [31]

Contrast set learning is a form of associative learning. **Contrast set learners** use rules that differ meaningfully in their distribution across subsets.[32][33]

Weighted class learning is another form of associative learning in which weight may be assigned to classes to give focus to a particular issue of concern for the consumer of the data mining results.

High-order pattern discovery facilitate the capture of high-order (polythetic) patterns or event associations that are intrinsic to complex real-world data. [34]

K-optimal pattern discovery provides an alternative to the standard approach to association rule learning that requires that each pattern appear frequently in the data.

Approximate Frequent Itemset mining is a relaxed version of Frequent Itemset mining that allows some of the items in some of the rows to be 0.[35]

Generalized Association Rules hierarchical taxonomy (concept hierarchy)

Quantitative Association Rules categorical and quantitative data [36]

Interval Data Association Rules e.g. partition the age into 5-year-increment ranged

Maximal Association Rules

Sequential pattern mining discovers subsequences that are common to more than minsup sequences in a sequence database, where minsup is set by the user. A sequence is an ordered list of transactions.[37]

Sequential Rules discovering relationships between items while considering the time ordering. It is generally applied on a sequence database. For example, a sequential rule found in database of sequences of customer transactions can be that customers who bought a computer and CD-Roms, later bought a webcam, with a given confidence and support.

Warmr is shipped as part of the ACE data mining suite. It allows association rule learning for first order relational rules.[38]

6.10 See also

- Sequence mining
- Production system

6.11 References

[1] Piatetsky-Shapiro, Gregory (1991), *Discovery, analysis, and presentation of strong rules*, in Piatetsky-Shapiro, Gregory; and Frawley, William J.; eds., *Knowledge Discovery in Databases*, AAAI/MIT Press, Cambridge, MA.

[2] Agrawal, R.; Imieliński, T.; Swami, A. (1993). "Mining association rules between sets of items in large databases". *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*. p. 207. doi:10.1145/170035.170072. ISBN 0897915925.

[3] Michael Hahsler (2015). A Probabilistic Comparison of Commonly Used Interest Measures for Association Rules. http://michael.hahsler.net/research/association_rules/measures.html

[4] Hipp, J.; Güntzer, U.; Nakhaeizadeh, G. (2000). "Algorithms for association rule mining --- a general survey and comparison". *ACM SIGKDD Explorations Newsletter* **2**: 58. doi:10.1145/360402.360421.

[5] Tan, Pang-Ning; Michael, Steinbach; Kumar, Vipin (2005). "Chapter 6. Association Analysis: Basic Concepts and Algorithms" (PDF). *Introduction to Data Mining*. Addison-Wesley. ISBN 0-321-32136-7.

[6] Pei, Jian; Han, Jiawei; and Lakshmanan, Laks V. S.; *Mining frequent itemsets with convertible constraints*, in *Proceedings of the 17th International Conference on Data Engineering, April 2–6, 2001, Heidelberg, Germany*, 2001, pages 433-442

[7] Agrawal, Rakesh; and Srikant, Ramakrishnan; *Fast algorithms for mining association rules in large databases*, in Bocca, Jorge B.; Jarke, Matthias; and Zaniolo, Carlo; editors, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Santiago, Chile, September 1994*, pages 487-499

[8] Zaki, M. J. (2000). "Scalable algorithms for association mining". *IEEE Transactions on Knowledge and Data Engineering* **12** (3): 372–390. doi:10.1109/69.846291.

[9] Hájek, Petr; Havel, Ivan; Chytil, Metoděj; *The GUHA method of automatic hypotheses determination*, Computing 1 (1966) 293-308

[10] Hájek, Petr; Feglar, Tomas; Rauch, Jan; and Coufal, David; *The GUHA method, data preprocessing and mining*, Database Support for Data Mining Applications, Springer, 2004, ISBN 978-3-540-22479-2

[11] Omiecinski, Edward R.; *Alternative interest measures for mining associations in databases*, IEEE Transactions on Knowledge and Data Engineering, 15(1):57-69, Jan/Feb 2003

[12] Aggarwal, Charu C.; and Yu, Philip S.; *A new framework for itemset generation*, in *PODS 98, Symposium on Principles of Database Systems, Seattle, WA, USA, 1998*, pages 18-24

[13] Brin, Sergey; Motwani, Rajeev; Ullman, Jeffrey D.; and Tsur, Shalom; *Dynamic itemset counting and implication rules for market basket data*, in *SIGMOD 1997, Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1997), Tucson, Arizona, USA, May 1997*, pp. 255-264

[14] Piatetsky-Shapiro, Gregory; *Discovery, analysis, and presentation of strong rules*, Knowledge Discovery in Databases, 1991, pp. 229-248

[15] Brin, Sergey; Motwani, Rajeev; Ullman, Jeffrey D.; and Tsur, Shalom; *Dynamic itemset counting and implication rules for market basket data*, in *SIGMOD 1997, Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1997), Tucson, Arizona, USA, May 1997*, pp. 265-276

[16] Tan, Pang-Ning; Kumar, Vipin; and Srivastava, Jaideep; *Selecting the right objective measure for association analysis*, Information Systems, 29(4):293-313, 2004

[17] Webb, Geoffrey I. (2007); *Discovering Significant Patterns*, Machine Learning 68(1), Netherlands: Springer, pp. 1-33 online access

[18] Gionis, Aristides; Mannila, Heikki; Mielikäinen, Taneli; and Tsaparas, Panayiotis; *Assessing Data Mining Results via Swap Randomization*, ACM Transactions on Knowledge Discovery from Data (TKDD), Volume 1, Issue 3 (December 2007), Article No. 14

[19] Witten, Frank, Hall: Data mining practical machine learning tools and techniques, 3rd edition

[20] D. Bhalodiya, K. M. Patel and C. Patel. An Efficient way to Find Frequent Pattern with Dynamic Programming Approach . NIRMA UNIVERSITY INTERNATIONAL CONFERENCE ON ENGINEERING, NUiCONE-2013, 28-30 NOVEMBER, 2013.

[21] Z. H. Deng and S. L. Lv. Fast mining frequent itemsets using Nodesets.. Expert Systems with Applications, 41(10): 4505–4512, 2014.

[22] Z. H. Deng, Z. Wang,and J. Jiang. A New Algorithm for Fast Mining Frequent Itemsets Using N-Lists . SCIENCE CHINA Information Sciences, 55 (9): 2008 - 2030, 2012.

[23] Z. H. Deng and Z. Wang. A New Fast Vertical Method for Mining Frequent Patterns . International Journal of Computational Intelligence Systems, 3(6): 733 - 744, 2010.

[24] Rauch, Jan; *Logical calculi for knowledge discovery in databases*, in *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, Springer, 1997, pp. 47-57

[25] Hájek, Petr; Havránek, Tomáš (1978). *Mechanizing Hypothesis Formation: Mathematical Foundations for a General Theory*. Springer-Verlag. ISBN 3-540-08738-9.

[26] Webb, Geoffrey I. (1995); *OPUS: An Efficient Admissible Algorithm for Unordered Search*, Journal of Artificial Intelligence Research 3, Menlo Park, CA: AAAI Press, pp. 431-465 online access

[27] Bayardo, Roberto J., Jr.; Agrawal, Rakesh; Gunopulos, Dimitrios (2000). "Constraint-based rule mining in large, dense databases". *Data Mining and Knowledge Discovery* **4** (2): 217–240. doi:10.1023/A:1009895914772.

[28] Webb, Geoffrey I. (2000); *Efficient Search for Association Rules*, in Ramakrishnan, Raghu; and Stolfo, Sal; eds.; *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000), Boston, MA*, New York, NY: The Association for Computing Machinery, pp. 99-107 online access

[29] http://www.dssresources.com/newsletters/66.php

[30] Ramezani, Reza, Mohamad Saraee, and Mohammad Ali Nematbakhsh; *MRAR: Mining Multi-Relation Association Rules*, Journal of Computing and Security, 1, no. 2 (2014)

[31] Shaheen, M; Shahbaz, M; and Guergachi, A; *Context Based Positive and Negative Spatio Temporal Association Rule Mining*, Elsevier Knowledge-Based Systems, Jan 2013, pp. 261-273

[32] GI Webb and S. Butler and D. Newlands (2003). *On Detecting Differences Between Groups*. KDD'03 Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[33] Menzies, Tim; and Hu, Ying; *Data Mining for Very Busy People*, IEEE Computer, October 2003, pp. 18-25

[34] Wong, Andrew K.C.; Wang, Yang (1997). "High-order pattern discovery from discrete-valued data". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*: 877–893.

[35] Jinze Liu, Susan Paulsen, Xing Sun, Wei Wang, Andrew Nobel, J. P. (2006). Mining approximate frequent itemsets in the presence of noise: Algorithm and analysis. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.3805

[36] Salleb-Aouissi, Ansaf; Vrain, Christel; Nortet, Cyril (2007). "QuantMiner: A Genetic Algorithm for Mining Quantitative Association Rules". *International Joint Conference on Artificial Intelligence (IJCAI)*: 1035–1040.

[37] Zaki, Mohammed J. (2001); *SPADE: An Efficient Algorithm for Mining Frequent Sequences*, Machine Learning Journal, 42, pp. 31–60

[38] "Warmr: a data mining tool for chemical data.". *J Comput Aided Mol Des* **15** (2): 173–81. Feb 2001. PMID 11272703.

6.12 External links

6.12.1 Bibliographies

- Extensive Bibliography on Association Rules by J.M. Luna
- Annotated Bibliography on Association Rules by M. Hahsler
- Statsoft Electronic Statistics Textbook: Association Rules by Dell Software

6.12.2 Implementations

Open-Source data-mining suites

- Christian Borgelt's implementations of Apriori, FP-Growth and Eclat written in C with Python bindings.
- ELKI includes Java implementations of Apriori, Eclat and FPGrowth.
- Orange module orngAssoc.
- R package arules for mining association rules and frequent itemsets.
- SPMF offers many open-source implementations for association rule mining, itemset mining and sequential pattern mining.
- Weka, a collection of machine learning algorithms for data mining tasks written in Java

Academic example code

- ARtool, GPL Java association rule mining application with GUI, offering implementations of multiple algorithms for discovery of frequent patterns and extraction of association rules (includes Apriori and FPgrowth, last updated 2002)
- Bart Goethals' frequent pattern mining implementations
- Ferda Dataminer, an extensible visual data mining platform, implements GUHA procedures ASSOC and features multirelational data mining
- Frequent Itemset Mining Implementations Repository (FIMI)
- Java implementations of association rule mining algorithms by KDIS

- Ruby implementation (AI4R)
- Zaki, Mohammed J.; Data Mining Software

Commercial offers

- KNIME, an open source workflow oriented data preprocessing and analysis platform
- KXEN, a commercial Data Mining software
- LISp Miner, mines for generalized (GUHA) association rules (uses bitstrings, not apriori algorithm)
- Magnum Opus, a system for statistically sound association discovery
- RapidMiner, a Java data mining software suite
- STATISTICA, commercial statistics software with an Association Rules module

Chapter 7

Reinforcement learning

For reinforcement learning in psychology, see Reinforcement.

Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take *actions* in an *environment* so as to maximize some notion of cumulative *reward*. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics, and genetic algorithms. In the operations research and control literature, the field where reinforcement learning methods are studied is called *approximate dynamic programming*. The problem has been studied in the theory of optimal control, though most studies are concerned with the existence of optimal solutions and their characterization, and not with the learning or approximation aspects. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality.

In machine learning, the environment is typically formulated as a Markov decision process (MDP) as many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical techniques and reinforcement learning algorithms is that the latter do not need knowledge about the MDP and they target large MDPs where exact methods become infeasible.

Reinforcement learning differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The exploration vs. exploitation trade-off in reinforcement learning has been most thoroughly studied through the multi-armed bandit problem and in finite MDPs.

7.1 Introduction

The basic reinforcement learning model consists of:

1. a set of environment states S ;
2. a set of actions A ;
3. rules of transitioning between states;
4. rules that determine the *scalar immediate reward* of a transition; and
5. rules that describe what the agent observes.

The rules are often stochastic. The observation typically involves the scalar immediate reward associated with the last transition. In many works, the agent is also assumed to observe the current environmental state, in which case we talk about *full observability*, whereas in the opposing case we talk about *partial observability*. Sometimes the set of actions available to the agent is restricted (e.g., you cannot spend more money than what you possess).

A reinforcement learning agent interacts with its environment in discrete time steps. At each time t , the agent receives an observation o_t , which typically includes the reward r_t . It then chooses an action a_t from the set of actions available, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the *transition* (s_t, a_t, s_{t+1}) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible. The agent can choose any action as a function of the history and it can even randomize its action selection.

When the agent's performance is compared to that of an agent which acts optimally from the beginning, the difference in performance gives rise to the notion of *regret*. Note that in order to act near optimally, the agent must reason about the long term consequences of its actions: In order to maximize my future income I had better go to school now, although the immediate monetary reward associated with this might be negative.

Thus, reinforcement learning is particularly well suited to problems which include a long-term versus short-term reward trade-off. It has been applied successfully to various problems, including robot control, elevator scheduling, telecommunications, backgammon and checkers (Sutton and Barto 1998, Chapter 11).

Two components make reinforcement learning powerful: The use of samples to optimize performance and the use of function approximation to deal with large environments. Thanks to these two key components, reinforcement learning can be used in large environments in any of the following situations:

- A model of the environment is known, but an analytic solution is not available;
- Only a simulation model of the environment is given (the subject of simulation-based optimization);[1]
- The only way to collect information about the environment is by interacting with it.

The first two of these problems could be considered planning problems (since some form of the model is available), while the last one could be considered as a genuine learning problem. However, under a reinforcement learning methodology both planning problems would be converted to machine learning problems.

7.2 Exploration

The reinforcement learning problem as described requires clever exploration mechanisms. Randomly selecting actions, without reference to an estimated probability distribution, is known to give rise to very poor performance. The case of (small) finite MDPs is relatively well understood by now. However, due to the lack of algorithms that would provably scale well with the number of states (or scale to problems with infinite state spaces), in practice people resort to simple exploration methods. One such method is ϵ-greedy, when the agent chooses the action that it believes has the best long-term effect with probability $1-\epsilon$, and it chooses an action uniformly at random, otherwise. Here, $0 < \epsilon < 1$ is a tuning parameter, which is sometimes changed, either according to a fixed schedule (making the agent explore less as time goes by), or adaptively based on some heuristics (Tokic & Palm, 2011).

7.3 Algorithms for control learning

Even if the issue of exploration is disregarded and even if the state was observable (which we assume from now on), the problem remains to find out which actions are good based on past experience.

7.3.1 Criterion of optimality

For simplicity, assume for a moment that the problem studied is *episodic*, an episode ending when some *terminal state* is reached. Assume further that no matter what course of actions the agent takes, termination is inevitable. Under some additional mild regularity conditions the expectation of the total reward is then well-defined, for *any* policy and any initial distribution over the states. Here, a policy refers to a mapping that assigns some probability distribution over the actions to all possible histories.

Given a fixed initial distribution μ, we can thus assign the expected return ρ^π to policy π:

$$\rho^\pi = E[R|\pi],$$

where the random variable R denotes the *return* and is defined by

$$R = \sum_{t=0}^{N-1} r_{t+1},$$

where r_{t+1} is the reward received after the t-th transition, the initial state is sampled at random from μ and actions are selected by policy π. Here, N denotes the (random) time when a terminal state is reached, i.e., the time when the episode terminates.

In the case of non-episodic problems the return is often *discounted*,

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1},$$

giving rise to the total expected discounted reward criterion. Here $0 \le \gamma \le 1$ is the so-called *discount-factor*. Since the undiscounted return is a special case of the discounted return, from now on we will assume discounting. Although this looks innocent enough, discounting is in fact problematic if one cares about online performance. This is because discounting makes the initial time steps more important. Since a learning agent is likely to make mistakes during the first few steps after its "life" starts, no uninformed learning algorithm can achieve near-optimal performance under discounting even if the class of environments is restricted to that of finite MDPs. (This does not mean though that, given enough time, a learning agent cannot figure how to act near-optimally, if time was restarted.)

The problem then is to specify an algorithm that can be used to find a policy with maximum expected return. From the theory of MDPs it is known that, without loss of generality, the search can be restricted to the set of the so-called *stationary* policies. A policy is called stationary if the action-distribution returned by it depends only on the last state visited (which is part of the observation history of the agent, by our simplifying assumption). In fact, the search can be further restricted to *deterministic* stationary policies. A deterministic stationary policy is one which deterministically selects actions based on the current state. Since any such policy can be identified with a mapping from the set of states to the set of actions, these policies can be identified with such mappings with no loss of generality.

7.3.2 Brute force

The brute force approach entails the following two steps:

1. For each possible policy, sample returns while following it
2. Choose the policy with the largest expected return

One problem with this is that the number of policies can be extremely large, or even infinite. Another is that variance of the returns might be large, in which case a large number of samples will be required to accurately estimate the return of each policy.

These problems can be ameliorated if we assume some structure and perhaps allow samples generated from one policy to influence the estimates made for another. The two main approaches for achieving this are value function estimation and direct policy search.

7.3.3 Value function approaches

Value function approaches attempt to find a policy that maximizes the return by maintaining a set of estimates of expected returns for some policy (usually either the "current" or the optimal one).

These methods rely on the theory of MDPs, where optimality is defined in a sense which is stronger than the above one: A policy is called optimal if it achieves the best expected return from *any* initial state (i.e., initial distributions play no role in this definition). Again, one can always find an optimal policy amongst stationary policies.

To define optimality in a formal manner, define the value of a policy π by

$$V^\pi(s) = E[R|s,\pi],$$

where R stands for the random return associated with following π from the initial state s . Define $V^*(s)$ as the maximum possible value of $V^\pi(s)$, where π is allowed to change:

$$V^*(s) = \sup_\pi V^\pi(s).$$

A policy which achieves these *optimal values* in *each* state is called *optimal*. Clearly, a policy optimal in this strong sense is also optimal in the sense that it maximizes the expected return ρ^π , since $\rho^\pi = E[V^\pi(S)]$, where S is a state randomly sampled from the distribution μ .

Although state-values suffice to define optimality, it will prove to be useful to define action-values. Given a state s , an action a and a policy π , the action-value of the pair (s,a) under π is defined by

$$Q^\pi(s,a) = E[R|s,a,\pi],$$

where, now, R stands for the random return associated with first taking action a in state s and following π , thereafter.

It is well-known from the theory of MDPs that if someone gives us Q for an optimal policy, we can always choose optimal actions (and thus act optimally) by simply choosing the action with the highest value at each state. The *action-value function* of such an optimal policy is called the *optimal action-value function* and is denoted by Q^* . In summary, the knowledge of the optimal action-value function *alone* suffices to know how to act optimally.

Assuming full knowledge of the MDP, there are two basic approaches to compute the optimal action-value function, value iteration and policy iteration. Both algorithms compute a sequence of functions Q_k ($k = 0,1,2,\ldots,$) which converge to Q^* . Computing these functions involves computing expectations over the whole state-space, which is impractical for all, but the smallest (finite) MDPs, never mind the case when the MDP is unknown. In reinforcement learning methods the expectations are approximated by averaging over samples and one uses function approximation techniques to cope with the need to represent value functions over large state-action spaces.

Monte Carlo methods

The simplest Monte Carlo methods can be used in an algorithm that mimics policy iteration. Policy iteration consists of two steps: *policy evaluation* and *policy improvement*. The Monte Carlo methods are used in the policy evaluation step. In this step, given a stationary, deterministic policy π , the goal is to compute the function values $Q^\pi(s,a)$ (or a good approximation to them) for all state-action pairs (s,a) . Assume (for simplicity) that the MDP is finite and in fact a table representing the action-values fits into the memory. Further, assume that the problem is episodic and after each episode a new one starts from some random initial state. Then, the estimate of the value of a given state-action pair (s,a) can be computed by simply averaging the sampled returns which originated from (s, a) over time. Given enough time, this procedure can thus construct a precise estimate Q of the action-value function Q^{π} . This finishes the description of the policy evaluation step. In the policy improvement step, as it is done in the standard policy iteration algorithm, the next policy is obtained by computing a *greedy* policy with respect to Q : Given a state s , this new policy returns an action that maximizes $Q(s, \cdot)$. In practice one often avoids computing and storing the new policy, but uses lazy evaluation to defer the computation of the maximizing actions to when they are actually needed.

A few problems with this procedure are as follows:

- The procedure may waste too much time on evaluating a suboptimal policy;
- It uses samples inefficiently in that a long trajectory is used to improve the estimate only of the *single* state-action pair that started the trajectory;
- When the returns along the trajectories have *high variance*, convergence will be slow;
- It works in *episodic problems only*;
- It works in *small, finite MDPs only*.

Temporal difference methods

The first issue is easily corrected by allowing the procedure to change the policy (at all, or at some states) before the values settle. However good this sounds, this may be dangerous as this might prevent convergence. Still, most current algorithms implement this idea, giving rise to the class of *generalized policy iteration* algorithm. We note in passing that actor critic methods belong to this category.

The second issue can be corrected within the algorithm by allowing trajectories to contribute to any state-action pair in them. This may also help to some extent with the third problem, although a better solution when returns have high variance is to use Sutton's temporal difference (TD) methods which are based on the recursive Bellman equation. Note that the computation in TD methods can be incremental (when after each transition the memory is changed and the transition is thrown away), or batch (when the transitions are collected and then the estimates are computed once based on a large number of transitions). Batch methods, a prime example of which is the least-squares temporal difference method due to Bradtke and Barto (1996), may use the information in the samples better, whereas incremental methods are the only choice when batch methods become infeasible due to their high computational or memory complexity. In addition, there exist methods that try to unify the advantages of the two approaches. Methods based on temporal differences also overcome the second but last issue.

In order to address the last issue mentioned in the previous section, *function approximation methods* are used. In *linear function approximation* one starts with a mapping ϕ that assigns a finite-dimensional vector to each state-action pair. Then, the action values of a state-action pair (s, a) are obtained by linearly combining the components of $\phi(s, a)$ with some *weights* θ :

$$Q(s, a) = \sum_{i=1}^{d} \theta_i \phi_i(s, a)$$

The algorithms then adjust the weights, instead of adjusting the values associated with the individual state-action pairs. However, linear function approximation is not the only choice. More recently, methods based on ideas from nonparametric statistics (which can be seen to construct their own features) have been explored.

So far, the discussion was restricted to how policy iteration can be used as a basis of the designing reinforcement learning algorithms. Equally importantly, value iteration can also be used as a starting point, giving rise to the Q-Learning algorithm (Watkins 1989) and its many variants.

The problem with methods that use action-values is that they may need highly precise estimates of the competing action values, which can be hard to obtain when the returns are noisy. Though this problem is mitigated to some extent by temporal difference methods and if one uses the so-called compatible function approximation method, more work remains to be done to increase generality and efficiency. Another problem specific to temporal difference methods comes from their reliance on the recursive Bellman equation. Most temporal difference methods have a so-called λ parameter $(0 \leq \lambda \leq 1)$ that allows one to continuously interpolate between Monte-Carlo methods (which do not rely on the Bellman equations) and the basic temporal difference methods (which rely entirely on the Bellman equations), which can thus be effective in palliating this issue.

7.3.4 Direct policy search

An alternative method to find a good policy is to search directly in (some subset) of the policy space, in which case the problem becomes an instance of stochastic optimization. The two approaches available are gradient-based and gradient-free methods.

Gradient-based methods (giving rise to the so-called *policy gradient methods*) start with a mapping from a finite-dimensional (parameter) space to the space of policies: given the parameter vector θ , let π_θ denote the policy associated to θ . Define the performance function by

$$\rho(\theta) = \rho^{\pi_\theta}.$$

Under mild conditions this function will be differentiable as a function of the parameter vector θ . If the gradient of ρ was known, one could use gradient ascent. Since an analytic expression for the gradient is not available, one must rely on a noisy estimate. Such an estimate can be constructed in many ways, giving rise to algorithms like Williams' REINFORCE method (which is also known as the likelihood ratio method in the simulation-based optimization literature). Policy gradient methods have received a lot of attention in the last couple of years (e.g., Peters et al. (2003)), but they remain an active field. An overview of policy search methods in the context of robotics has been given by Deisenroth, Neumann and Peters.[2] The issue with many of these methods is that they may get stuck in local optima (as they are based on local search).

A large class of methods avoids relying on gradient information. These include simulated annealing, cross-entropy search or methods of evolutionary computation. Many gradient-free methods can achieve (in theory and in the limit) a global optimum. In a number of cases they have indeed demonstrated remarkable performance.

The issue with policy search methods is that they may converge slowly if the information based on which they act is noisy. For example, this happens when in episodic problems the trajectories are long and the variance of the returns is large. As argued beforehand, value-function based methods that rely on temporal differences might help in this case. In recent years, several actor-critic algorithms have been proposed following this idea and were demonstrated to perform well in various problems.

7.4 Theory

The theory for small, finite MDPs is quite mature. Both the asymptotic and finite-sample behavior of most algorithms is well-understood. As mentioned beforehand, algorithms with provably good online performance (addressing the exploration issue) are known. The theory of large MDPs needs more work. Efficient exploration is largely untouched (except for the case of bandit problems). Although finite-time performance bounds appeared for many algorithms in the recent years, these bounds are expected to be rather loose and thus more work is needed to better understand the relative advantages, as well as the limitations of these algorithms. For incremental algorithm asymptotic convergence issues have been settled. Recently, new incremental, temporal-difference-based algorithms have appeared which converge under a much wider set of conditions than was previously possible (for example, when used with arbitrary, smooth function approximation).

7.5 Current research

Current research topics include: adaptive methods which work with fewer (or no) parameters under a large number of conditions, addressing the exploration problem in large MDPs, large-scale empirical evaluations, learning and acting under partial information (e.g., using Predictive State Representation), modular and hierarchical reinforcement learning, improving existing value-function and policy search methods, algorithms that work well with large (or continuous) action spaces, transfer learning, lifelong learning, efficient sample-based planning (e.g., based on Monte-Carlo tree search). Multiagent or Distributed Reinforcement Learning is also a topic of interest in current research. There is also a growing interest in real life applications of reinforcement learning. Successes of reinforcement learning are collected on here and here.

Reinforcement learning algorithms such as TD learning are also being investigated as a model for Dopamine-based learning in the brain. In this model, the dopaminergic projections from the substantia nigra to the basal ganglia function as the prediction error. Reinforcement learning has also been used as a part of the model for human skill learning, especially in relation to the interaction between implicit and explicit learning in skill acquisition (the first publication on this application was in 1995-1996, and there have been many follow-up studies). See http://webdocs.cs.ualberta.ca/~{ }sutton/RL-FAQ.html#behaviorism for further details of these research areas above.

7.6 Literature

7.6.1 Conferences, journals

Most reinforcement learning papers are published at the major machine learning and AI conferences (ICML, NIPS, AAAI, IJCAI, UAI, AI and Statistics) and journals (JAIR, JMLR, Machine learning journal, IEEE T-CIAIG). Some theory papers are published at COLT and ALT. However, many papers appear in robotics conferences (IROS, ICRA) and the "agent" conference AAMAS. Operations researchers publish their papers at the INFORMS conference and, for example, in the Operation Research, and the Mathematics of Operations Research journals. Control researchers publish their papers at the CDC and ACC conferences, or, e.g., in the journals IEEE Transactions on Automatic Control, or Automatica, although applied works tend to be published in more specialized journals. The Winter Simulation Conference also publishes many relevant papers. Other than this, papers also published in the major conferences of the neural networks, fuzzy, and evolutionary computation communities. The annual IEEE symposium titled Approximate Dynamic Programming and Reinforcement Learning (ADPRL) and the biannual European Workshop on Reinforcement Learning (EWRL) are two regularly held meetings where RL researchers meet.

7.7 See also

- Temporal difference learning
- Q-learning
- SARSA
- Fictitious play
- Learning classifier system
- Optimal control
- Dynamic treatment regimes
- Error-driven learning
- Multi-agent system
- Distributed artificial intelligence

7.8 Implementations

- RL-Glue provides a standard interface that allows you to connect agents, environments, and experiment programs together, even if they are written in different languages.
- Maja Machine Learning Framework The Maja Machine Learning Framework (MMLF) is a general framework for problems in the domain of Reinforcement Learning (RL) written in python.
- Software Tools for Reinforcement Learning (Matlab and Python)
- PyBrain(Python)
- TeachingBox is a Java reinforcement learning framework supporting many features like RBF networks, gradient descent learning methods, ...
- C++ and Python implementations for some well known reinforcement learning algorithms with source.
- Orange, a free data mining software suite, module orngReinforcement
- Policy Gradient Toolbox provides a package for learning about policy gradient approaches.
- BURLAP is an open source Java library that provides a wide range of single and multi-agent learning and planning methods.

7.9 References

- Sutton, Richard S. (1984). *Temporal Credit Assignment in Reinforcement Learning* (PhD thesis). University of Massachusetts, Amherst, MA.
- Williams, Ronald J. (1987). “A class of gradient-estimating algorithms for reinforcement learning in neural networks”. *Proceedings of the IEEE First International Conference on Neural Networks.*
- Sutton, Richard S. (1988). “Learning to predict by the method of temporal differences”. *Machine Learning* (Springer) **3**: 9–44. doi:10.1007/BF00115009.
- Watkins, Christopher J.C.H. (1989). *Learning from Delayed Rewards* (PDF) (PhD thesis). King’s College, Cambridge, UK.
- Bradtke, Steven J.; Andrew G. Barto (1996). “Learning to predict by the method of temporal differences”. *Machine Learning* (Springer) **22**: 33–57. doi:10.1023/A:1018056104778.
- Bertsekas, Dimitri P.; John Tsitsiklis (1996). *Neuro-Dynamic Programming*. Nashua, NH: Athena Scientific. ISBN 1-886529-10-8.
- Kaelbling, Leslie P.; Michael L. Littman; Andrew W. Moore (1996). “Reinforcement Learning: A Survey”. *Journal of Artificial Intelligence Research* **4**: 237–285.
- Sutton, Richard S.; Barto, Andrew G. (1998). *Reinforcement Learning: An Introduction*. MIT Press. ISBN 0-262-19398-1.
- Peters, Jan; Sethu Vijayakumar; Stefan Schaal (2003). “Reinforcement Learning for Humanoid Robotics” (PDF). *IEEE-RAS International Conference on Humanoid Robots.*
- Powell, Warren (2007). *Approximate dynamic programming: solving the curses of dimensionality*. Wiley-Interscience. ISBN 0-470-17155-3.
- Auer, Peter; Thomas Jaksch; Ronald Ortner (2010). “Near-optimal regret bounds for reinforcement learning”. *Journal of Machine Learning Research* **11**: 1563–1600.
- Szita, Istvan; Csaba Szepesvari (2010). “Model-based Reinforcement Learning with Nearly Tight Exploration Complexity Bounds” (PDF). *ICML 2010*. Omnipress. pp. 1031–1038.

- Bertsekas, Dimitri P. (August 2010). “Chapter 6 (online): Approximate Dynamic Programming”. *Dynamic Programming and Optimal Control* (PDF) **II** (3 ed.).

- Busoniu, Lucian; Robert Babuska ; Bart De Schutter ; Damien Ernst (2010). *Reinforcement Learning and Dynamic Programming using Function Approximators*. Taylor & Francis CRC Press. ISBN 978-1-4398-2108-4.

- Tokic, Michel; Günther Palm ; (2011). “Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax”. *KI 2011: Advances in Artificial Intelligence* (PDF). Lecture Notes in Computer Science **7006**. Springer Berlin / Heidelberg. pp. 335–346.

- Röttger, Michael C.; Andreas W. Liehr (2009). “Control task for Reinforcement Learning with known optimal solution for discrete and continuous actions”. *Journal of Intelligent Learning Systems and Applications* **1**: 26–39. doi:10.4236/jilsa.2009.11002.

- Deisenroth, Marc Peter; Gerhard Neumann; Jan Peters (2013). *A Survey on Policy Search for Robotics*. Foundations and Trends in Robotics **2**. NOW Publishers. pp. 1–142.

[1] Gosavi, Abhijit (2003). *Simulation-based Optimization: Parametric Optimization Techniques and Reinforcement*. Springer. ISBN 1-4020-7454-9.

[2] Deisenroth, Marc Peter; Neumann, Gerhard; Peters, Jan (2013). *A Survey on Policy Search for Robotics*. NOW Publishers. pp. 1–142. ISBN 978-1-60198-702-0.

7.10 External links

- Website for *Reinforcement Learning: An Introduction* (1998), by Rich Sutton and Andrew Barto, MIT Press, including a link to an html version of the book.
- Reinforcement Learning Repository
- Reinforcement Learning and Artificial Intelligence (RLAI, Rich Sutton's lab at the University of Alberta)
- Autonomous Learning Laboratory (ALL, Andrew Barto's lab at the University of Massachusetts Amherst)
- RL-Glue
- Software Tools for Reinforcement Learning (Matlab and Python)
- The Reinforcement Learning Toolbox from the (Graz University of Technology)
- Hybrid reinforcement learning
- Piqle: a Generic Java Platform for Reinforcement Learning
- A Short Introduction To Some Reinforcement Learning Algorithms
- Reinforcement Learning applied to Tic-Tac-Toe Game
- Scholarpedia Reinforcement Learning
- Scholarpedia Temporal Difference Learning
- Stanford Reinforcement Learning Course
- Real-world reinforcement learning experiments at Delft University of Technology
- Reinforcement Learning Tools for Matlab
- Stanford University Andrew Ng Lecture on Reinforcement Learning

Chapter 8

Structured prediction

Structured prediction or **structured (output) learning** is an umbrella term for supervised machine learning techniques that involve predicting structured objects, rather than scalar discrete or real values.[1]

For example, the problem of translating a natural language sentence into a syntactic representation such as a parse tree can be seen as a structured prediction problem in which the structured output domain is the set of all possible parse trees.

Probabilistic graphical models form a large class of structured prediction models. In particular, Bayesian networks and random fields are popularly used to solve structured prediction problems in a wide variety of application domains including bioinformatics, natural language processing, speech recognition, and computer vision. Other algorithms and models for structured prediction include inductive logic programming, structured SVMs, Markov logic networks and constrained conditional models.

Similar to commonly used supervised learning techniques, structured prediction models are typically trained by means of observed data in which the true prediction value is used to adjust model parameters. Due to the complexity of the model and the interrelations of predicted variables the process of prediction using a trained model and of training itself is often computationally infeasible and approximate inference and learning methods are used.

8.1 Example: sequence tagging

Sequence tagging is a class of problems prevalent in natural language processing, where input data are often sequences (e.g. sentences of text). The sequence tagging problem appears in several guises, e.g. part-of-speech tagging and named entity recognition. In POS tagging, each word in a sequence must receive a "tag" (class label) that expresses its "type" of word:

This	DT
is	VBZ
a	DT
tagged	JJ
sentence	NN
.	.

The main challenge in this problem is to resolve ambiguity: the word "sentence" can also be a verb in English, and so can "tagged".

While this problem can be solved by simply performing classification of individual tokens, that approach does not take into account the empirical fact that tags do not occur independently; instead, each tag displays a strong conditional dependence on the tag of the previous word. This fact can be exploited in a sequence model such as a hidden Markov model or conditional random field[2] that predicts the entire tag sequence for a sentence, rather than just individual tags, by means of the Viterbi algorithm.

8.2 Structured perceptron

One of the easiest ways to understand algorithms for general structured prediction is the structured perceptron of Collins.[3] This algorithm combines the venerable perceptron algorithm for learning linear classifiers with an inference algorithm (classically the Viterbi algorithm when used on sequence data) and can be described abstractly as follows. First define a "joint feature function" $\Phi(\mathbf{x}, \mathbf{y})$ that maps a training sample $\mathbf{x}$ and a candidate prediction $\mathbf{y}$ to a vector of length n ($\mathbf{x}$ and $\mathbf{y}$ may have any structure; n is problem-dependent, but must be fixed for each model). Let GEN be a function that generates candidate predictions. Then:

```
Let $\mathbf{w}$ be a weight vector of length $n$
For a pre-determined number of iterations:
    For each sample $\mathbf{x}$ in the training set with true output $\mathbf{t}$:
        Make a prediction $\hat{\mathbf{y}} = \arg\max \{\mathbf{y} \in \mathrm{GEN}(\mathbf{x})\} (\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}))$
    Update $\mathbf{w}$, from $\hat{\mathbf{y}}$ to $\mathbf{t}$: $\mathbf{w}=\mathbf{w}+c(-\Phi(\mathbf{x}, \hat{\mathbf{y}})+ \Phi(\mathbf{x}, \mathbf{t}))$, $c$ is learning rate
```

In practice, finding the argmax over GEN($\mathbf{x}$) will be done using an algorithm such as Viterbi or max-sum, rather than an exhaustive search through an exponentially large set of candidates.

The idea of learning is similar to multiclass perceptron.

8.3 See also

- Conditional random field
- Structured support vector machine
- Recurrent neural network, in particular Elman networks (SRNs)

8.4 References

[1] Gökhan BakIr, Ben Taskar, Thomas Hofmann, Bernhard Schölkopf, Alex Smola and SVN Vishwanathan (2007), Predicting Structured Data, MIT Press.

[2] Lafferty, J., McCallum, A., Pereira, F. (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data" (PDF). *Proc. 18th International Conf. on Machine Learning*. pp. 282–289.

[3] Collins, Michael (2002). *Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms* (PDF). Proc. EMNLP **10**.

- Noah Smith, Linguistic Structure Prediction, 2011.

8.5 External links

- Implementation of Collins structured perceptron

Chapter 9

Feature learning

Feature learning or **representation learning**[1] is a set of techniques that learn a transformation of raw data input to a representation that can be effectively exploited in machine learning tasks.

Feature learning is motivated by the fact that machine learning tasks such as classification often require input that is mathematically and computationally convenient to process. However, real-world data such as images, video, and sensor measurement is usually complex, redundant, and highly variable. Thus, it is necessary to discover useful features or representations from raw data. Traditional hand-crafted features often require expensive human labor and often rely on expert knowledge. Also, they normally do not generalize well. This motivates the design of efficient feature learning techniques.

Feature learning can be divided into two categories: supervised and unsupervised feature learning.

- In supervised feature learning, features are learned with labeled input data. Examples include neural networks, multilayer perceptron, and (supervised) dictionary learning.

- In unsupervised feature learning, features are learned with unlabeled input data. Examples include dictionary learning, independent component analysis, autoencoders, matrix factorization,[2] and various forms of clustering.[3][4][5]

9.1 Supervised feature learning

Supervised feature learning is to learn features from labeled data. Several approaches are introduced in the following.

9.1.1 Supervised dictionary learning

Dictionary learning is to learn a set (dictionary) of representative elements from the input data such that each data point can be represented as a weighted sum of the representative elements. The dictionary elements and the weights may be found by minimizing the average representation error (over the input data), together with a *L1* regularization on the weights to enable sparsity (i.e., the representation of each data point has only a few nonzero weights).

Supervised dictionary learning exploits both the structure underlying the input data and the labels for optimizing the dictionary elements. For example, a supervised dictionary learning technique was proposed by Mairal et al. in 2009.[6] The authors apply dictionary learning on classification problems by jointly optimizing the dictionary elements, weights for representing data points, and parameters of the classifier based on the input data. In particular, a minimization problem is formulated, where the objective function consists of the classification error, the representation error, an *L1* regularization on the representing weights for each data point (to enable sparse representation of data), and an *L2* regularization on the parameters of the classifier.

9.1.2 Neural networks

Neural networks are used to illustrate a family of learning algorithms via a “network” consisting of multiple layers of inter-connected nodes. It is inspired by the nervous system, where the nodes are viewed as neurons and edges are viewed as synapse. Each edge has an associated weight, and the network defines computational rules that passes input data from the input layer to the output layer. A network function associated with a neural network characterizes the relationship between input and output layers, which is parameterized by the weights. With appropriately defined network functions, various learning tasks can be performed by minimizing a cost function over the network function (weights).

Multilayer neural networks can be used to perform feature learning, since they learn a representation of their input at the hidden layer(s) which is subsequently used for classification or regression at the output layer.

9.2 Unsupervised feature learning

Unsupervised feature learning is to learn features from unlabeled data. The goal of unsupervised feature learning is often to discover low-dimensional features that captures some structure underlying the high-dimensional input data. When the feature learning is performed in an unsupervised way, it enables a form of semisupervised learning where first, features are learned from an unlabeled dataset, which are then employed to improve performance in a supervised setting with labeled data.[7][8] Several approaches are introduced in the following.

9.2.1 *K*-means clustering

K-means clustering is an approach for vector quantization. In particular, given a set of *n* vectors, *k*-means clustering groups them into k clusters (i.e., subsets) in such a way that each vector belongs to the cluster with the closest mean. The problem is computationally NP-hard, and suboptimal greedy algorithms have been developed for *k*-means clustering.

In feature learning, *k*-means clustering can be used to group an unlabeled set of inputs into *k* clusters, and then use the centroids of these clusters to produce features. These features can be produced in several ways. The simplest way is to add *k* binary features to each sample, where each feature *j* has value one iff the *j*th centroid learned by *k*-means is the closest to the sample under consideration.[3] It is also possible to use the distances to the clusters as features, perhaps after transforming them through a radial basis function (a technique that has used to train RBF networks[9]). Coates and Ng note that certain variants of *k*-means behave similarly to sparse coding algorithms.[10]

In a comparative evaluation of unsupervised feature learning methods, Coates, Lee and Ng found that *k*-means clustering with an appropriate transformation outperforms the more recently invented auto-encoders and RBMs on an image classification task.[3] *K*-means has also been shown to improve performance in the domain of NLP, specifically for named-entity recognition;[11] there, it competes with Brown clustering, as well as with distributed word representations (also known as neural word embeddings).[8]

9.2.2 Principal component analysis

Principal component analysis (PCA) is often used for dimension reduction. Given a unlabeled set of *n* input data vectors, PCA generates *p* (which is much smaller than the dimension of the input data) right singular vectors corresponding to the *p* largest singular values of the data matrix, where the *k*th row of the data matrix is the *k*th input data vector shifted by the sample mean of the input (i.e., subtracting the sample mean from the data vector). Equivalently, these singular vectors are the eigenvectors corresponding to the *p* largest eigenvalues of the sample covariance matrix of the input vectors. These *p* singular vectors are the feature vectors learned from the input data, and they represent directions along which the data has the largest variations.

PCA is a linear feature learning approach since the *p* singular vectors are linear functions of the data matrix. The singular vectors can be generated via a simple algorithm with *p* iterations. In the *i*th iteration, the projection of the data matrix on the *(i-1)*th eigenvector is subtracted, and the *i*th singular vector is found as the right singular vector corresponding to the largest singular of the residual data matrix.

PCA has several limitations. First, it assumes that the directions with large variance are of most interest, which may not be the case in many applications. PCA only relies on orthogonal transformations of the original data, and it only exploits the first- and second-order moments of the data, which may not well characterize the distribution of the data. Furthermore, PCA can effectively reduce dimension only when the input data vectors are correlated (which results in a few dominant eigenvalues).

9.2.3 Local linear embedding

Local linear embedding (LLE) is a nonlinear unsupervised learning approach for generating low-dimensional neighbor-preserving representations from (unlabeled) high-dimension input. The approach was proposed by Sam T. Roweis and Lawrence K. Saul in 2000.[12][13]

The general idea of LLE is to reconstruct the original high-dimensional data using lower-dimensional points while maintaining some geometric properties of the neighborhoods in the original data set. LLE consists of two major steps. The first step is for "neighbor-preserving," where each input data point *Xi* is reconstructed as a weighted sum of *K* nearest neighboring data points, and the optimal weights are found by minimizing the average squared reconstruction error (i.e., difference between a point and its reconstruction) under the constraint that the weights associated to each point sum up to one. The second step is for "dimension reduction," by looking for vectors in a lower-dimensional space that minimizes the representation error using the optimized weights in the first step. Note that in the first step, the weights are optimized with data being fixed, which can be solved as a least squares problem; while in the second step, lower-dimensional points are optimized with the weights being fixed, which can be solved via sparse eigenvalue decomposition.

The reconstruction weights obtained in the first step captures the "intrinsic geometric properties" of a neighborhood in the input data.[13] It is assumed that original data lie on a smooth lower-dimensional manifold, and the "intrinsic geometric properties" captured by the weights of the original data are expected also on the manifold. This is why the same weights are used in the second step of LLE. Compared with PCA, LLE is more powerful in exploiting the underlying structure of data.

9.2.4 Independent component analysis

Independent component analysis (ICA) is technique for learning a representation of data using a weighted sum of independent non-Gaussian components.[14] The assumption of non-Gaussian is imposed since the weights cannot be uniquely determined when all the components follow Gaussian distribution.

9.2.5 Unsupervised dictionary learning

Different from supervised dictionary learning, unsupervised dictionary learning does not utilize the labels of the data and only exploits the structure underlying the data for optimizing the dictionary elements. An example of unsupervised dictionary learning is sparse coding, which aims to learn basis functions (dictionary elements) for data representation from unlabeled input data. Sparse coding can be applied to learn overcomplete dictionary, where the number of dictionary elements is larger than the dimension of the input data.[15] Aharon et al. proposed an algorithm known as K-SVD for learning from unlabeled input data a dictionary of elements that enables sparse representation of the data.[16]

9.3 Multilayer/Deep architectures

The hierarchical architecture of the neural system inspires deep learning architectures for feature learning by stacking multiple layers of simple learning blocks.[17] These architectures are often designed based on the assumption of distributed representation: observed data is generated by the interactions of many different factors on multiple levels. In a deep learning architecture, the output of each intermediate layer can be viewed as a representation of the original input data. Each level uses the representation produced by previous level as input, and produces new representations as output, which is then fed to higher levels. The input of bottom layer is the raw data, and the output of the final layer is the final low-dimensional feature or representation.

9.3.1 Restricted Boltzmann machine

Restricted Boltzmann machines (RBMs) are often used as a building block for multilayer learning architectures.[3][18] An RBM can be represented by an undirected bipartite graph consisting of a group of binary hidden variables, a group of visible variables, and edges connecting the hidden and visible nodes. It is a special case of the more general Boltzmann machines with the constraint of no intra-node connections. Each edge in an RBM is associated with a weight. The weights together with the connections define an energy function, based on which a joint distribution of visible and hidden nodes can be devised. Based on the topology of the RBM, the hidden (visible) variables are independent conditioned on the visible (hidden) variables. Such conditional independence facilitates computations on RBM.

An RBM can be viewed as a single layer architecture for unsupervised feature learning. In particular, the visible variables correspond to input data, and the hidden variables correspond to feature detectors. The weights can be trained by maximizing the probability of visible variables using the contrastive divergence (CD) algorithm by Geoffrey Hinton.[18]

In general, the training of RBM by solving the above maximization problem tends to result in non-sparse representations. The sparse RBM, [19] a modification of the RBM, was proposed to enable sparse representations. The idea is to add a regularization term in the objective function of data likelihood, which penalizes the deviation of the expected hidden variables from a small constant p .

9.3.2 Autoencoder

An autoencoder consisting of encoder and decoder is a paradigm for deep learning architectures. An example is provided by Hinton and Salakhutdinov[18] where the encoder uses raw data (e.g., image) as input and produces feature or representation as output, and the decoder uses the extracted feature from the encoder as input and reconstructs the original input raw data as output. The encoder and decoder are constructed by stacking multiple layers of RBMs. The parameters involved in the architecture are trained in a greedy layer-by-layer manner: after one layer of feature detectors is learned, they are fed to upper layers as visible variables for training the corresponding RBM. The process can be repeated until some stopping criteria is satisfied.

9.4 See also

- Basis function
- Deep learning
- Feature detection (computer vision)
- Feature extraction
- Kernel trick
- Vector quantization

9.5 References

[1] Y. Bengio; A. Courville; P. Vincent (2013). “Representation Learning: A Review and New Perspectives”. *IEEE Trans. PAMI, special issue Learning Deep Architectures*.

[2] Nathan Srebro; Jason D. M. Rennie; Tommi S. Jaakkola (2004). *Maximum-Margin Matrix Factorization*. NIPS.

[3] Coates, Adam; Lee, Honglak; Ng, Andrew Y. (2011). *An analysis of single-layer networks in unsupervised feature learning* (PDF). Int'l Conf. on AI and Statistics (AISTATS).

[4] Csurka, Gabriella; Dance, Christopher C.; Fan, Lixin; Willamowski, Jutta; Bray, Cédric (2004). *Visual categorization with bags of keypoints* (PDF). ECCV Workshop on Statistical Learning in Computer Vision.

[5] Daniel Jurafsky; James H. Martin (2009). *Speech and Language Processing*. Pearson Education International. pp. 145–146.

[6] Mairal, Julien; Bach, Francis; Ponce, Jean; Sapiro, Guillermo; Zisserman, Andrew (2009). “Supervised Dictionary Learning”. *Advances in neural information processing systems*.

[7] Percy Liang (2005). *Semi-Supervised Learning for Natural Language* (PDF) (M. Eng.). MIT. pp. 44–52.

[8] Joseph Turian; Lev Ratinov; Yoshua Bengio (2010). *Word representations: a simple and general method for semi-supervised learning* (PDF). Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.

[9] Schwenker, Friedhelm; Kestler, Hans A.; Palm, Günther (2001). “Three learning phases for radial-basis-function networks”. *Neural Networks* **14**: 439–458. doi:10.1016/s0893-6080(01)00027-2. CiteSeerX: 10.1.1.109.312.

[10] Coates, Adam; Ng, Andrew Y. (2012). “Learning feature representations with k-means”. In G. Montavon, G. B. Orr and K.-R. Müller. *Neural Networks: Tricks of the Trade*. Springer.

[11] Dekang Lin; Xiaoyun Wu (2009). *Phrase clustering for discriminative learning* (PDF). Proc. J. Conf. of the ACL and 4th Int'l J. Conf. on Natural Language Processing of the AFNLP. pp. 1030–1038.

[12] Roweis, Sam T; Saul, Lawrence K (2000). “Nonlinear Dimensionality Reduction by Locally Linear Embedding”. *Science, New Series* **290** (5500): 2323–2326. doi:10.1126/science.290.5500.2323.

[13] Saul, Lawrence K; Roweis, Sam T (2000). “An Introduction to Locally Linear Embedding”.

[14] Hyvärinen, Aapo; Oja, Erkki (2000). “Independent Component Analysis: Algorithms and Applications”. *Neural networks* (4): 411–430.

[15] Lee, Honglak; Battle, Alexis; Raina, Rajat; Ng, Andrew Y (2007). “Efficient sparse coding algorithms”. *Ad- vances in neural information processing systems*.

[16] Aharon, Michal; Elad, Michael; Bruckstein, Alfred (2006). “K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation”. *IEEE Trans. Signal Process.* **54** (11): 4311–4322. doi:10.1109/TSP.2006.881199.

[17] Bengio, Yoshua (2009). “Learning Deep Architectures for AI”. *Foundations and Trends® in Machine Learning* **2** (1): 1–127. doi:10.1561/2200000006.

[18] Hinton, G. E.; Salakhutdinov, R. R. (2006). “Reducing the Dimensionality of Data with Neural Networks” (PDF). *Science* **313** (5786): 504–507. doi:10.1126/science.1127647. PMID 16873662.

[19] Lee, Honglak; Ekanadham, Chaitanya; Andrew, Ng (2008). “Sparse deep belief net model for visual area V2”. *Advances in neural information processing systems*.

Chapter 10

Online machine learning

Online machine learning is used in the case where the data becomes available in a sequential fashion, in order to determine a mapping from the dataset to the corresponding labels. The key difference between online learning and batch learning (or "offline" learning) techniques, is that in online learning the mapping is updated after the arrival of every new datapoint in a scalable fashion, whereas batch techniques are used when one has access to the entire training dataset at once. Online learning could be used in the case of a process occurring in time, for example the value of a stock given its history and other external factors, in which case the mapping updates as time goes on and we get more and more samples.

Ideally in online learning, the memory needed to store the function remains constant even with added datapoints, since the solution computed at one step is updated when a new datapoint becomes available, after which that datapoint can then be discarded. For many formulations, for example nonlinear kernel methods, true online learning is not possible, though a form of hybrid online learning with recursive algorithms can be used. In this case, the space requirements are no longer guaranteed to be constant since it requires storing all previous datapoints, but the solution may take less time to compute with the addition of a new datapoint, as compared to batch learning techniques.

As in all machine learning problems, the goal of the algorithm is to minimize some performance criteria using a loss function. For example, with stock market prediction the algorithm may attempt to minimize the mean squared error between the predicted and true value of a stock. Another popular performance criterion is to minimize the number of mistakes when dealing with classification problems. In addition to applications of a sequential nature, online learning algorithms are also relevant in applications with huge amounts of data such that traditional learning approaches that use the entire data set in aggregate are computationally infeasible.

10.1 A prototypical online supervised learning algorithm

In the setting of supervised learning, or learning from examples, we are interested in learning a function $f : X \to Y$, where X is thought of as a space of inputs and Y as a space of outputs, that predicts well on instances that are drawn from a joint probability distribution $p(x, y)$ on $X \times Y$. In this setting, we are given a loss function $V : Y \times Y \to \mathbb{R}$, such that $V(f(x), y)$ measures the difference between the predicted value $f(x)$ and the true value y . The ideal goal is to select a function $f \in \mathcal{H}$, where $\mathcal{H}$ is a space of functions called a hypothesis space, so as to minimize the expected risk:

$$I[f] = \mathbb{E}[V(f(x), y)] = \int V(f(x), y)\, dp(x, y) \ .$$

In reality, the learner never knows the true distribution $p(x, y)$ over instances. Instead, the learner usually has access to a training set of examples $(x_1, y_1), \ldots, (x_n, y_n)$ that are assumed to have been drawn i.i.d. from the true distribution $p(x, y)$. A common paradigm in this situation is to estimate a function $\hat{f}$ through empirical risk minimization or regularized empirical risk minimization (usually Tikhonov regularization). The choice of loss function here gives rise to several well-known learning algorithms such as regularized least squares and support vector machines.

The above paradigm is not well-suited to the online learning setting though, as it requires complete a priori knowledge of the entire training set. In the pure online learning approach, the learning algorithm should update a sequence of functions $f_1, f_2, \ldots$ in a way such that the function f_{t+1} depends only on the previous function f_t and the next data point (x_t, y_t) . This approach has low memory requirements in the sense that it only requires storage of a representation of the current function f_t and the next data point (x_t, y_t) . A related approach that has larger memory requirements allows f_{t+1} to depend on f_t and all previous data points $(x_1, y_1), \ldots, (x_t, y_t)$. We focus solely on the former approach here, and we consider both the case where the data is coming from an infinite stream $(x_1, y_1), (x_2, y_2), \ldots$ and the case where the data is coming from a finite training set $(x_1, y_1), \ldots, (x_n, y_n)$, in which case the online learning algorithm may make multiple passes through the data.

10.1.1 The algorithm and its interpretations

Here we outline a prototypical online learning algorithm in the supervised learning setting and we discuss several interpretations of this algorithm. For simplicity, consider the case where $X = \mathbb{R}^d$, $Y \subseteq \mathbb{R}$, and $\mathcal{H} = \{\langle w, \cdot \rangle : w \in \mathbb{R}^d\}$ is the set of all linear functionals from X into $\mathbb{R}$, i.e. we are working with a linear kernel and functions $f \in \mathcal{H}$ can be identified with vectors $w \in \mathbb{R}^d$. Furthermore, assume that $V(\cdot, \cdot)$ is a convex, differentiable loss function. An online learning algorithm satisfying the low memory property discussed above consists of the following iteration:

$$w_{t+1} \leftarrow w_t - \gamma_t \nabla V(\langle w_t, x_t \rangle, y_t) ,$$

where $w_1 \leftarrow 0$, $\nabla V(\langle w_t, x_t \rangle, y_t)$ is the gradient of the loss for the next data point (x_t, y_t) evaluated at the current linear functional w_t , and $\gamma_t > 0$ is a step-size parameter. In the case of an infinite stream of data, one can run this iteration, in principle, forever, and in the case of a finite but large set of data, one can consider a single pass or multiple passes (epochs) through the data.

Interestingly enough, the above simple iterative online learning algorithm has three distinct interpretations, each of which has distinct implications about the predictive quality of the sequence of functions $w_1, w_2, \ldots$. The first interpretation considers the above iteration as an instance of the stochastic gradient descent method applied to the problem of minimizing the expected risk $I[w]$ defined above.[1] Indeed, in the case of an infinite stream of data, since the examples $(x_1, y_1), (x_2, y_2), \ldots$ are assumed to be drawn i.i.d. from the distribution $p(x, y)$, the sequence of gradients of $V(\cdot, \cdot)$ in the above iteration are an i.i.d. sample of stochastic estimates of the gradient of the expected risk $I[w]$ and therefore one can apply complexity results for the stochastic gradient descent method to bound the deviation $I[w_t] - I[w^*]$, where w^* is the minimizer of $I[w]$.[2] This interpretation is also valid in the case of a finite training set; although with multiple passes through the data the gradients are no longer independent, still complexity results can be obtained in special cases.

The second interpretation applies to the case of a finite training set and considers the above recursion as an instance of the incremental gradient descent method[3] to minimize the empirical risk:

$$I_n[w] = \frac{1}{n} \sum_{i=1}^{n} V(\langle w, x_i \rangle, y_i) .$$

Since the gradients of $V(\cdot, \cdot)$ in the above iteration are also stochastic estimates of the gradient of $I_n[w]$, this interpretation is also related to the stochastic gradient descent method, but applied to minimize the empirical risk as opposed to the expected risk. Since this interpretation concerns the empirical risk and not the expected risk, multiple passes through the data are readily allowed and actually lead to tighter bounds on the deviations $I_n[w_t] - I_n[w_n^*]$, where w_n^* is the minimizer of $I_n[w]$.

The third interpretation of the above recursion is distinctly different from the first two and concerns the case of sequential trials discussed above, where the data are potentially not i.i.d. and can perhaps be selected in an adversarial manner. At each step of this process, the learner is given an input x_t and makes a prediction based on the current linear function w_t . Only after making this prediction does the learner see the true label y_t , at which point the learner is allowed to update w_t to w_{t+1} . Since we are not making any distributional assumptions about the data, the goal here is to perform as well as if we could view the entire sequence of examples ahead of time; that is, we would like the sequence of functions $w_1, w_2, \ldots$ to have low regret relative to any vector w^* :

$$R_T(w^*) = \sum_{t=1}^{T} V(\langle w_t, x_t \rangle, y_t) - \sum_{t=1}^{T} V(\langle w^*, x_t \rangle, y_t) .$$

In this setting, the above recursion can be considered as an instance of the online gradient descent method for which there are complexity bounds that guarantee $O(\sqrt{T})$ regret.[4]

It should be noted that although the three interpretations of this algorithm yield complexity bounds in three distinct settings, each bound depends on the choice of step-size sequence $\{\gamma_t\}$ in a different way, and thus we cannot simultaneously apply the consequences of all three interpretations; we must instead select the step-size sequence in a way that is tailored for the interpretation that is most relevant. Furthermore, the above algorithm and these interpretations can be extended to the case of a nonlinear kernel by simply considering X to be the feature space associated with the kernel. Although in this case the memory requirements at each iteration are no longer $O(d)$, but are rather on the order of the number of data points considered so far.

10.2 Example: Complexity in the Case of Linear Least Squares

10.2.1 Batch Learning

Let us consider the setting of supervised learning with the square loss function $V(\langle w, x_i\rangle, y_i) = (x_i^T w - y_i)^2$, ($x_i \in \mathbb{R}^d$, $w_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$). The solution after the arrival of every datapoint $\{x_i, y_i\}$ is given by $w^* = (X^T X)^{-1} X^T Y$ where X and Y is built from the i data points, with X being i -by- d and Y being i -by- 1 . The solution of linear least squares problem is roughly $O(id^2)$.

If we have n total points in the dataset and we have to recompute the solution after the arrival of every datapoint $i = 1, \ldots, n$, we have a total complexity $O(n^2 d^2)$. Here we assume that the matrix $X^T X$ is invertible, otherwise we can proceed in a similar fashion with Tikhonov regularization.

10.2.2 Online Learning

The recursive least squares algorithm considers an online approach to the least squares problem. It can be shown that for suitable initializations of $w_0 \in \mathbb{R}^d$ and $\Gamma_0 \in \mathbb{R}^{dxd}$, the solution of the linear least squares problem given in the previous section can be computed by the following iteration:

$$\Gamma_i = \Gamma_{i-1} - \frac{\Gamma_{i-1} x_i x_i^T \Gamma_{i-1}}{1 + x_i^T \Gamma_{i-1} x_i}$$

$$w_i = w_{i-1} - \Gamma_i x_i (x_i^T w_{i-1} - y_i)$$

For the proof, see RLS.

The complexity for n steps of this algorithm is $O(nd^2)$, which is an order of magnitude faster than the corresponding batch learning complexity. The storage requirements at every step i here are constant at $O(d^2)$, i.e. that of storing the matrix Γ_i .

Stochastic Gradient Descent

If we now replace $w_i = w_{i-1} - \Gamma_i x_n (x_i^T w_{i-1} - y_i)$ by $w_i = w_{i-1} - \gamma_i x_i (x_i^T w_{i-1} - y_i)$ (i.e. replacing $\Gamma_i \in \mathbb{R}^{d \times d}$ by $\gamma_i \in \mathbb{R}$), we have a stochastic gradient descent algorithm. In this case, the complexity for n steps of this algorithm reduces to $O(nd)$. The storage requirements at every step i are constant at $O(d)$.

However, the stepsize γ_i needs to be chosen carefully to solve the expected risk minimization problem, as detailed above.

10.3 Books with substantial treatment of online machine learning

- *Algorithmic Learning in a Random World* by Vladimir Vovk, Alex Gammerman, and Glenn Shafer. Published by Springer Science+Business Media, Inc. 2005 ISBN 0-387-00152-2

- *Prediction, learning, and games* by Nicolò Cesa-Bianchi and Gábor Lugosi. Cambridge University Press, 2006 ISBN 0-521-84108-9

10.4 See also

- Hierarchical temporal memory
- k-nearest neighbor algorithm
- Lazy learning
- Learning Vector Quantization
- Offline learning, the opposite model
- Online algorithm
- Streaming Algorithm
- Perceptron
- Stochastic gradient descent
- Supervised learning

10.5 References

[1] Bottou, Léon (1998). "Online Algorithms and Stochastic Approximations". *Online Learning and Neural Networks*. Cambridge University Press. ISBN 978-0-521-65263-6

[2] *Stochastic Approximation Algorithms and Applications*, Harold J. Kushner and G. George Yin, New York: Springer-Verlag, 1997. ISBN 0-387-94916-X; 2nd ed., titled *Stochastic Approximation and Recursive Algorithms and Applications*, 2003, ISBN 0-387-00894-2.

[3] Bertsekas, D. P. (2011). Incremental gradient, subgradient, and proximal methods for convex optimization: a survey. Optimization for Machine Learning, 85.

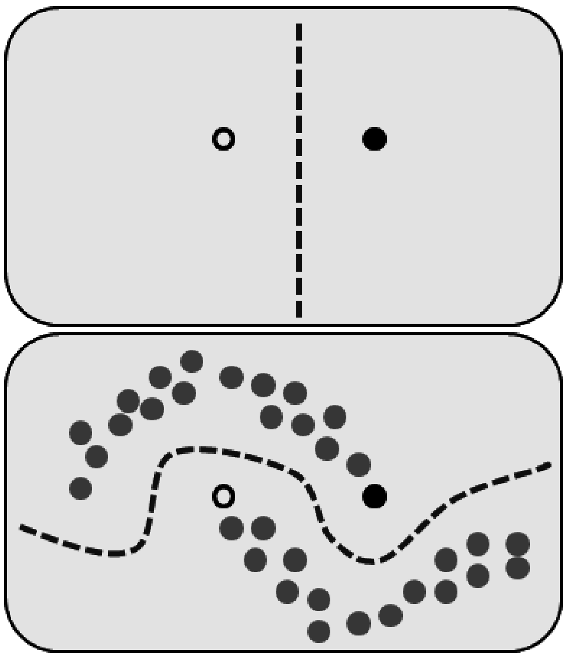
[4] Shalev-Shwartz, S. (2011). Online learning and online convex optimization. Foundations and Trends in Machine Learning, 4(2), 107-194.

10.6 External links

- http://onlineprediction.net/, Wiki for On-Line Prediction.

Chapter 11

Semi-supervised learning



An example of the influence of unlabeled data in semi-supervised learning. The top panel shows a decision boundary we might adopt after seeing only one positive (white circle) and one negative (black circle) example. The bottom panel shows a decision boundary we might adopt if, in addition to the two labeled examples, we were given a collection of unlabeled data (gray circles). This could be viewed as performing clustering and then labeling the clusters with the labeled data, pushing the decision boundary away from high-density regions, or learning an underlying one-dimensional manifold where the data reside.

Semi-supervised learning is a class of supervised learning tasks and techniques that also make use of unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent (e.g. to transcribe an audio segment) or a physical experiment (e.g. determining the 3D structure of a protein or determining whether there is oil at a particular location). The cost associated with the labeling process thus may render a fully labeled training set infeasible, whereas acquisition of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value. Semi-supervised learning is also of theoretical interest in machine learning and as a model for human learning.

As in the supervised learning framework, we are given a set of l independently identically distributed examples $x_1, \ldots, x_l \in X$ with corresponding labels $y_1, \ldots, y_l \in Y$. Additionally, we are given u unlabeled examples $x_{l+1}, \ldots, x_{l+u} \in X$. Semi-supervised learning attempts to make use of this combined information to surpass the classification performance that could be obtained either by discarding the unlabeled data and doing supervised learning or by discarding the labels and doing unsupervised learning.

Semi-supervised learning may refer to either transductive learning or inductive learning. The goal of transductive learning is to infer the correct labels for the given unlabeled data $x_{l+1}, \ldots, x_{l+u}$ only. The goal of inductive learning is to infer the correct mapping from X to Y .

Intuitively, we can think of the learning problem as an exam and labeled data as the few example problems that the teacher solved in class. The teacher also provides a set of unsolved problems. In the transductive setting, these unsolved problems are a take-home exam and you want to do well on them in particular. In the inductive setting, these are practice problems of the sort you will encounter on the in-class exam.

It is unnecessary (and, according to Vapnik's principle, imprudent) to perform transductive learning by way of inferring a classification rule over the entire input space; however, in practice, algorithms formally designed for transduction or induction are often used interchangeably.

11.1 Assumptions used in semi-supervised learning

In order to make any use of unlabeled data, we must assume some structure to the underlying distribution of data. Semi-supervised learning algorithms make use of at least one of the following assumptions. [1]

11.1.1 Smoothness assumption

Points which are close to each other are more likely to share a label. This is also generally assumed in supervised learning and yields a preference for geometrically simple decision boundaries. In the case of semi-supervised learning, the smoothness assumption additionally yields a preference for decision boundaries in low-density regions, so that there are fewer points close to each other but in different classes.

11.1.2 Cluster assumption

The data tend to form discrete clusters, and points in the same cluster are more likely to share a label (although data sharing a label may be spread across multiple clusters). This is a special case of the smoothness assumption and gives rise to feature learning with clustering algorithms.

11.1.3 Manifold assumption

The data lie approximately on a manifold of much lower dimension than the input space. In this case we can attempt to learn the manifold using both the labeled and unlabeled data to avoid the curse of dimensionality. Then learning can proceed using distances and densities defined on the manifold.

The manifold assumption is practical when high-dimensional data are being generated by some process that may be hard to model directly, but which only has a few degrees of freedom. For instance, human voice is controlled by a few vocal folds,[2] and images of various facial expressions are controlled by a few muscles. We would like in these cases to use distances and smoothness in the natural space of the generating problem, rather than in the space of all possible acoustic waves or images respectively.

11.2 History

The heuristic approach of *self-training* (also known as *self-learning* or *self-labeling*) is historically the oldest approach to semi-supervised learning,[1] with examples of applications starting in the 1960s (see for instance Scudder (1965)[3]).

The transductive learning framework was formally introduced by Vladimir Vapnik in the 1970s.[4] Interest in inductive learning using generative models also began in the 1970s. A *probably approximately correct* learning bound for semi-supervised learning of a Gaussian mixture was demonstrated by Ratsaby and Venkatesh in 1995 [5]

Semi-supervised learning has recently become more popular and practically relevant due to the variety of problems for which vast quantities of unlabeled data are available—e.g. text on websites, protein sequences, or images. For a review of recent work see a survey article by Zhu (2008).[6]

11.3 Methods for semi-supervised learning

11.3.1 Generative models

Generative approaches to statistical learning first seek to estimate $p(x|y)$, the distribution of data points belonging to each class. The probability $p(y|x)$ that a given point x has label y is then proportional to $p(x|y)p(y)$ by Bayes' rule. Semi-supervised learning with generative models can be viewed either as an extension of supervised learning (classification plus information about $p(x)$) or as an extension of unsupervised learning (clustering plus some labels).

Generative models assume that the distributions take some particular form $p(x|y,\theta)$ parameterized by the vector θ . If these assumptions are incorrect, the unlabeled data may actually decrease the accuracy of the solution relative to what would have been obtained from labeled data alone. [7] However, if the assumptions are correct, then the unlabeled data necessarily improves performance.[5]

The unlabeled data are distributed according to a mixture of individual-class distributions. In order to learn the mixture distribution from the unlabeled data, it must be identifiable, that is, different parameters must yield different summed distributions. Gaussian mixture distributions are identifiable and commonly used for generative models.

The parameterized joint distribution can be written as $p(x,y|\theta) = p(y|\theta)p(x|y,\theta)$ by using the Chain rule. Each parameter vector θ is associated with a decision function $f_\theta(x) = \underset{y}{\operatorname{argmax}}\ p(y|x,\theta)$. The parameter is then chosen based on fit to both the labeled and unlabeled data, weighted by λ :

$$\underset{\Theta}{\operatorname{argmax}} \left(\log p(\{x_i, y_i\}_{i=1}^{l}|\theta) + \lambda \log p(\{x_i\}_{i=l+1}^{l+u}|\theta) \right)$$

[8]

11.3.2 Low-density separation

Another major class of methods attempts to place boundaries in regions where there are few data points (labeled or unlabeled). One of the most commonly used algorithms is the transductive support vector machine, or TSVM (which, despite its name, may be used for inductive learning as well). Whereas support vector machines for supervised learning seek a decision boundary with maximal margin over the labeled data, the goal of TSVM is a labeling of the unlabeled data such that the decision boundary has maximal margin over all of the data. In addition to the standard hinge loss $(1-yf(x))_+$ for labeled data, a loss function $(1-|f(x)|)_+$ is introduced over the unlabeled data by letting $y = \operatorname{sign} f(x)$. TSVM then selects $f^*(x) = h^*(x) + b$ from a reproducing kernel Hilbert space $\mathcal{H}$ by minimizing the regularized empirical risk:

$$f^* = \underset{f}{\operatorname{argmin}} \left(\sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 ||h||^2_{\mathcal{H}} + \lambda_2 \sum_{i=l+1}^{l+u} (1 - |f(x_i)|)_+ \right)$$

An exact solution is intractable due to the non-convex term $(1-|f(x)|)_+$, so research has focused on finding useful approximations.[8]

Other approaches that implement low-density separation include Gaussian process models, information regularization, and entropy minimization (of which TSVM is a special case).

11.3.3 Graph-based methods

Graph-based methods for semi-supervised learning use a graph representation of the data, with a node for each labeled and unlabeled example. The graph may be constructed using domain knowledge or similarity of examples; two common methods are to connect each data point to its k nearest neighbors or to examples within some distance ϵ . The weight W_{ij} of an edge between x_i and x_j is then set to $e^{\frac{-||x_i - x_j||^2}{\epsilon}}$.

Within the framework of *manifold regularization*, [9] [10] the graph serves as a proxy for the manifold. A term is added to the standard Tikhonov regularization problem to enforce smoothness of the solution relative to the manifold (in the intrinsic space of the problem) as well as relative to the ambient input space. The minimization problem becomes

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \left(\frac{1}{l} \sum_{i=1}^{l} V(f(x_i), y_i) + \lambda_A ||f||^2_{\mathcal{H}} + \lambda_I \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} f(x)||^2 dp(x) \right)$$
[8]

where $\mathcal{H}$ is a reproducing kernel Hilbert space and $\mathcal{M}$ is the manifold on which the data lie. The regularization parameters λ_A and λ_I control smoothness in the ambient and intrinsic spaces respectively. The graph is used to approximate the intrinsic regularization term. Defining the graph Laplacian $L = D - W$ where $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$ and $\mathbf{f}$ the vector $[f(x_1) \dots f(x_{l+u})]$, we have

$$\mathbf{f}^T L \mathbf{f} = \sum_{i,j=1}^{l+u} W_{ij}(f_i - f_j)^2 \approx \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} f(x)||^2 dp(x)$$

The Laplacian can also be used to extend the supervised learning algorithms: regularized least squares and support vector machines (SVM) to semi-supervised versions Laplacian regularized least squares and Laplacian SVM.

11.3.4 Heuristic approaches

Some methods for semi-supervised learning are not intrinsically geared to learning from both unlabeled and labeled data, but instead make use of unlabeled data within a supervised learning framework. For instance, the labeled and unlabeled examples $x_1, \dots, x_{l+u}$ may inform a choice of representation, distance metric, or kernel for the data in an unsupervised first step. Then supervised learning proceeds from only the labeled examples.

Self-training is a wrapper method for semi-supervised learning. First a supervised learning algorithm is used to select a classifier based on the labeled data only. This classifier is then applied to the unlabeled data to generate more labeled examples as input for another supervised learning problem. Generally only the labels the classifier is most confident of are added at each step.

Co-training is an extension of self-training in which multiple classifiers are trained on different (ideally disjoint) sets of features and generate labeled examples for one another.

11.4 Semi-supervised learning in human cognition

Human responses to formal semi-supervised learning problems have yielded varying conclusions about the degree of influence of the unlabeled data (for a summary see [11]). More natural learning problems may also be viewed as instances of semi-supervised learning. Much of human concept learning involves a small amount of direct instruction (e.g. parental labeling of objects during childhood) combined with large amounts of unlabeled experience (e.g. observation of objects without naming or counting them, or at least without feedback).

Human infants are sensitive to the structure of unlabeled natural categories such as images of dogs and cats or male and female faces.[12] More recent work has shown that infants and children take into account not only the unlabeled examples available, but the sampling process from which labeled examples arise.[13][14]

11.5 See also

- PU learning

11.6 References

[1] Chapelle, Olivier; Schölkopf, Bernhard; Zien, Alexander (2006). *Semi-supervised learning*. Cambridge, Mass.: MIT Press. ISBN 978-0-262-03358-9.

[2] Stevens, K.N.(2000), Acoustic Phonetics, MIT Press, ISBN 0-262-69250-3, 978-0-262-69250-2

[3] Scudder, H.J. Probability of Error of Some Adaptive Pattern-Recognition Machines. IEEE Transaction on Information Theory, 11:363–371 (1965). Cited in Chapelle et al. 2006, page 3.

[4] Vapnik, V. and Chervonenkis, A. Theory of Pattern Recognition [in Russian]. Nauka, Moscow (1974). Cited in Chapelle et al. 2006, page 3.

[5] Ratsaby, J. and Venkatesh, S. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 412-417 (1995). Cited in Chapelle et al. 2006, page 4.

[6] Zhu, Xiaojin. Semi-supervised learning literature survey. Computer Sciences, University of Wisconsin-Madison (2008).

[7] Cozman, F. and Cohen, I. Risks of semi-supervised learning: how unlabeled data can degrade performance of generative classifiers. In: Chapelle et al. (2006).

[8] Zhu, Xiaojin. Semi-Supervised Learning University of Wisconsin-Madison.

[9] M. Belkin, P. Niyogi. Semi-supervised Leifolds. Machine Learning, 56, Special Issue on Clustering, 209-239, 2004.

[10] M. Belkin, P. Niyogi, V. Sindhwani. On Manifold Regularization. AISTATS 2005.

[11] Zhu, Xiaojin; Goldberg, Andrew B. (2009). *Introduction to semi-supervised learning*. Morgan & Claypool. ISBN 9781598295481.

[12] Younger, B. A. and Fearing, D. D. (1999), Parsing Items into Separate Categories: Developmental Change in Infant Categorization. Child Development, 70: 291–303.

[13] Xu, F. and Tenenbaum, J. B. (2007), Sensitivity to sampling in Bayesian word learning. Developmental Science, 10: 288–297.

[14] Gweon, H., Tenenbaum J.B., and Schulz L.E (2010), Infants consider both the sample and the sampling process in inductive generalization. Proc Natl Acad Sci U S A., 107(20):9066-71.

11.7 External links

- A freely available MATLAB implementation of the graph-based semi-supervised algorithms Laplacian support vector machines and Laplacian regularized least squares.

Chapter 12

Grammar induction

Grammar induction, also known as **grammatical inference** or syntactic pattern recognition, refers to the process in machine learning of learning a formal grammar (usually as a collection of *re-write rules* or *productions* or alternatively as a finite state machine or automaton of some kind) from a set of observations, thus constructing a model which accounts for the characteristics of the observed objects. More generally, grammatical inference is that branch of machine learning where the instance space consists of discrete combinatorial objects such as strings, trees and graphs.

There is now a rich literature on learning different types of grammar and automata, under various different learning models and using various different methodologies.

12.1 Grammar Classes

Grammatical inference has often been very focused on the problem of learning finite state machines of various types (see the article Induction of regular languages for details on these approaches), since there have been efficient algorithms for this problem since the 1980s.

More recently these approaches have been extended to the problem of inference of context-free grammars and richer formalisms, such as multiple context-free grammars and parallel multiple context-free grammars. Other classes of grammars for which grammatical inference has been studied are contextual grammars, and pattern languages.

12.2 Learning Models

The simplest form of learning is where the learning algorithm merely receives a set of examples drawn from the language in question, but other learning models have been studied. One frequently studied alternative is the case where the learner can ask membership queries as in the exact query learning model or minimally adequate teacher model introduced by Angluin.

12.3 Methodologies

There are a wide variety of methods for grammatical inference. Two of the classic sources are Fu (1977) and Fu (1982). Duda, Hart & Stork (2001) also devote a brief section to the problem, and cite a number of references. The basic trial-and-error method they present is discussed below. For approaches to infer subclasses of regular languages in particular, see *Induction of regular languages*. A more recent textbook is de la Higuera (2010) [1] which covers the theory of grammatical inference of regular languages and finite state automata. D'Ulizia, Ferri and Grifoni [2] provide a survey that explores grammatical inference methods for natural languages.

12.3.1 Grammatical inference by trial-and-error

The method proposed in Section 8.7 of Duda, Hart & Stork (2001) suggests successively guessing grammar rules (productions) and testing them against positive and negative observations. The rule set is expanded so as to be able to generate each positive example, but if a given rule set also generates a negative example, it must be discarded. This particular approach can be characterized as "hypothesis testing" and bears some similarity to Mitchel's version space algorithm. The Duda, Hart & Stork (2001) text provide a simple example which nicely illustrates the process, but the feasibility of such an unguided trial-and-error approach for more substantial problems is dubious.

12.3.2 Grammatical inference by genetic algorithms

Grammatical Induction using evolutionary algorithms is the process of evolving a representation of the grammar of a target language through some evolutionary process. Formal grammars can easily be represented as a tree structure of production rules that can be subjected to evolutionary operators. Algorithms of this sort stem from the genetic programming paradigm pioneered by John Koza. Other early work on simple formal languages used the binary string representation of genetic algorithms, but the inherently hierarchical structure of grammars couched in the EBNF language made trees a more flexible approach.

Koza represented Lisp programs as trees. He was able to find analogues to the genetic operators within the standard set of tree operators. For example, swapping sub-trees is equivalent to the corresponding process of genetic crossover, where sub-strings of a genetic code are transplanted into an individual of the next generation. Fitness is measured by scoring the output from the functions of the lisp code. Similar analogues between the tree structured lisp representation and the representation of grammars as trees, made the application of genetic programming techniques possible for grammar induction.

In the case of Grammar Induction, the transplantation of sub-trees corresponds to the swapping of production rules that enable the parsing of phrases from some language. The fitness operator for the grammar is based upon some measure of how well it performed in parsing some group of sentences from the target language. In a tree representation of a grammar, a terminal symbol of a production rule corresponds to a leaf node of the tree. Its parent nodes corresponds to a non-terminal symbol (e.g. a noun phrase or a verb phrase) in the rule set. Ultimately, the root node might correspond to a sentence non-terminal.

12.3.3 Grammatical inference by greedy algorithms

Like all greedy algorithms, greedy grammar inference algorithms make, in iterative manner, decisions that seem to be the best at that stage. These made decisions deal usually with things like the making of a new or the removing of the existing rules, the choosing of the applied rule or the merging of some existing rules. Because there are several ways to define 'the stage' and 'the best', there are also several greedy grammar inference algorithms.

These context-free grammar generating algorithms make the decision after every read symbol:

- Lempel-Ziv-Welch algorithm creates a context-free grammar in a deterministic way such that it is necessary to store only the start rule of the generated grammar.

- Sequitur and its modifications.

These context-free grammar generating algorithms first read the whole given symbol-sequence and then start to make decisions:

- Byte pair encoding and its optimizations.

12.3.4 Distributional Learning

A more recent approach is based on Distributional Learning. Algorithms using these approaches have been applied to learning context-free grammars and mildly context-sensitive languages and have been proven to be correct and efficient for large subclasses of these grammars.[3]

12.3.5 Learning of Pattern languages

Angluin defines a **pattern** to be a string of constant symbols from Σ and **variable symbols** from a disjoint set. The language of such a pattern is the set of all its nonempty ground instances i.e. all strings resulting from consistent replacement of its variable symbols by nonempty strings of constant symbols.[note 1] A pattern is called **descriptive** for a finite input set of strings if its language is minimal (with respect to set inclusion) among all pattern languages subsuming the input set.

Angluin gives a polynomial algorithm to compute, for a given input string set, all descriptive patterns in one variable x.[note 2] To this end, she builds an automaton representing all possibly relevant patterns; using sophisticated arguments about word lengths, which rely on x being the only variable, the state count can be drastically reduced.[4]

Erlebach et al. give a more efficient version of Angluin's pattern learning algorithm, as well as a parallelized version.[5]

Arimura et al. show that a language class obtained from limited unions of patterns can be learned in polynomial time.[6]

12.3.6 Pattern theory

Pattern theory, formulated by Ulf Grenander,[7] is a mathematical formalism to describe knowledge of the world as patterns. It differs from other approaches to artificial intelligence in that it does not begin by prescribing algorithms and machinery to recognize and classify patterns; rather, it prescribes a vocabulary to articulate and recast the pattern concepts in precise language.

In addition to the new algebraic vocabulary, its statistical approach was novel in its aim to:

- Identify the hidden variables of a data set using real world data rather than artificial stimuli, which was commonplace at the time.

- Formulate prior distributions for hidden variables and models for the observed variables that form the vertices of a Gibbs-like graph.

- Study the randomness and variability of these graphs.

- Create the basic classes of stochastic models applied by listing the deformations of the patterns.
- Synthesize (sample) from the models, not just analyze signals with it.

Broad in its mathematical coverage, Pattern Theory spans algebra and statistics, as well as local topological and global entropic properties.

12.4 Applications

The principle of grammar induction has been applied to other aspects of natural language processing, and have been applied (among many other problems) to morpheme analysis, and even place name derivations. Grammar induction has also been used for lossless data compression and statistical inference via MML and MDL principles.

12.5 See also

- Artificial grammar learning
- Syntactic pattern recognition
- Inductive inference
- Straight-line grammar
- Kolmogorov complexity
- Automatic distillation of structure
- Inductive programming

12.6 Notes

[1] The language of a pattern with at least two occurrences of the same variable is not regular due to the pumping lemma.

[2] x may occur several times, but no other variable y may occur

12.7 References

[1] de la Higuera, Colin (2010). *Grammatical Inference: Learning Automata and Grammars*. Cambridge: Cambridge University Press.

[2] D'Ulizia, A., Ferri, F., Grifoni, P. (2011) "A Survey of Grammatical Inference Methods for Natural Language Learning", Artificial Intelligence Review, Vol. 36, No. 1, pp. 1-27.

[3] Clark and Eyraud (2007) Journal of Machine Learning Research, Ryo Yoshinaka (2011) Theoretical Computer Science

[4] Dana Angluin (1980). "Finding Patterns Common to a Set of Strings" (PDF). *Journal of Computer and System Sciences* **21**: 46–62. doi:10.1016/0022-0000(80)90041-0.

[5] T. Erlebach, P. Rossmanith, H. Stadtherr, A. Steger, T. Zeugmann (1997). "Learning One-Variable Pattern Languages Very Efficiently on Average, in Parallel, and by Asking Queries". In M. Li and A. Maruoka. *Proc. 8th International Workshop on Algorithmic Learning Theory — ALT'97*. LNAI **1316**. Springer. pp. 260–276.

[6] Hiroki Arimura, Takeshi Shinohara, Setsuko Otsuki (1994). "Finding Minimal Generalizations for Unions of Pattern Languages and Its Application to Inductive Inference from Positive Data". *Proc. STACS 11*. LNCS **775**. Springer. pp. 649–660.

[7] Grenander, Ulf, and Michael I. Miller. Pattern theory: from representation to inference. Vol. 1. Oxford: Oxford university press, 2007.

- Duda, Richard O.; Hart, Peter E.; Stork, David G. (2001), *Pattern Classification* (2 ed.), New York: John Wiley & Sons
- Fu, King Sun (1982), *Syntactic Pattern Recognition and Applications*, Englewood Cliffs, NJ: Prentice-Hall
- Fu, King Sun (1977), *Syntactic Pattern Recognition, Applications*, Berlin: Springer-Verlag
- Horning, James Jay (1969), *A Study of Grammatical Inference* (Ph.D. Thesis ed.), Stanford: Stanford University Computer Science Department
- Gold, E. Mark (1967), *Language Identification in the Limit* (PDF) **10**, Information and Control, pp. 447–474

, see also the corresponding Wikipedia article

12.8 Text and image sources, contributors, and licenses

12.8.1 Text

jeremy, A m sheldon, AntiSpamBot, LeighvsOptimvsMaximvs, Ramkumar.krishnan, Shoessss, Josephjthomas, Parikshit Basrur, Doug4, Cometstyles, DH85868993, DorganBot, Bonadea, WinterSpw, Mark.hornick, Andy Marchbanks, Yecril, BernardZ, RJASE1, Idioma-bot, RonFredericks, Jeff G., Jimmaths, DataExp, Philip Trueman, Adamminstead, TXiKiBoT, Deleet, Udufruduhu, Deanabb, Valerie928, TyrantX, OlavN, Arpabr, Vlad.gerchikov, Don4of4, Raymondwinn, Mannafredo, 1yesfan, Bearian, Jkosik1, Wykypydya, Billinghurst, Atannir, Hadleywickham, Hherbert, Falcon8765, Sebastjanmm, Pjoef, Mattelsen, AlleborgoBot, Burkeangirl, NHRHS2010, Rknasc, Pdfpdf, Equilibrioception, Calliopejen1, VerySmartNiceGuy, Euryalus, Dawn Bard, Estard, Srp33, Jerryobject, Kexpert, Mark Klamberg, Curuxz, Flyer22, Eikoku, JCLately, Powtroll, Jpcedenog, Strife911, Pyromaniaman, Oxymoron83, Gpswiki, Dodabe~enwiki, Gargvikram07, Mátyás, Fratrep, Chrisguyot, Odo Benus, Stfg, StaticGull, Sanya r, DixonD, Kjtobo, Melcombe, 48states, LaUs3r, Pinkadelica, Ypouliot, Denisarona, Sbacle, Kotsiantis, Loren.wilton, Sfan00 IMG, Nezza 4 eva, ClueBot, The Thing That Should Not Be, EoGuy, Supertouch, Kkarimi, Blanchardb, Edayapattiarun, Lbertolotti, Shaw76, Verticalsearch, Sebleouf, Hanifbbz, Abrech, Sterdeus, DrCroco, Nano5656, Aseld, Amossin, Dekisugi, SchreiberBike, DyingIce, Atallcostsky, 9Nak, Dank, Versus22, Katanada, Qwfp, DumZiBoT, Sunsetsky, XLinkBot, Articdawg, Cgfjpfg, Ecmalthouse, Little Mountain 5, WikHead, SilvonenBot, Badgernet, Foxyliah, Freestyle-69, Texterp, Addbot, DOI bot, Mabdul, Landon1980, Mhahsler, AndrewHZ, Elsendero, Matt90855, Jpoelma13, Cis411, Drkknightbatman, MrOllie, Download, RTG, M.r santosh kumar., Glane23, Delaszk, Chzz, Swift-Epic (Refectory), AtheWeatherman, Fauxstar, Jesuja, Luckas-bot, Yobot, Adelpine, Bunnyhop11, Ptbotgourou, Cflm001, Hulek, Alusayman, Ryanscraper, Carleas, Nallimbot, SOMart, Tiffany9027, AnomieBOT, Rjanag, Jim1138, JackieBot, Fahadsadah, OptimisticCynic, Dudukeda, Materialscientist, Citation bot, Schul253, Cureden, Capricorn42, Gtfjbl, Lark137, Liwaste, The Evil IP address, Tomwsulcer, BluePlateSpecial, Dr Oldekop, Rosannel, Rugaaad, RibotBOT, Charvest, Tareq300, Cmccormick8, Smallman12q, Andrzejrauch, Davgrig04, Stekre, Whizzdumb, Thehelpfulbot, Kyleamiller, OlafvanD, FrescoBot, Mark Renier, Ph92, W Nowicki, X7q, Colewaldron, Er.piyushkp, HamburgerRadio, Atlantia, Webzie, Citation bot 1, Killian441, Manufan 11, Rustyspatula, Pinethicket, Guerrerocarlos, Toohuman1, BRUTE, Elsevierеditormath, Stpasha, MastiBot, SpaceFlight89, Jackverr, UngerJ, Juliustch, Priyank782, TobeBot, Pamparam, Btcoal, Kmettler, Jonkerz, GregKaye, Glenn Maddox, Jayrde, Angelorf, Reaper Eternal, Chenzheruc, Pmauer, DARTH SIDIOUS 2, Mean as custard, RjwilmsiBot, Mike78465, D vandyke67, Ripchip Bot, Slon02, Aaronzat, Helwr, Ericmortenson, EmausBot, Acather96, BillyPreset, Fly by Night, WirlWhind, GoingBatty, Emilescheepers444, Stheodor, Lawrykid, Uploadvirus, Wikipelli, Dcirovic, Joanlofe, Anir1uph, Chire, Cronk28, Zedutchgandalf, Vangelis12, T789, Rick jens, Donner60, Terryholmsby, MainFrame, Phoglenix, Raomohsinkhan, ClueBot NG, Mathstat, Aiwing, Nuwanmenuka, Statethatiamin, CherryX, Candace Gillhoolley, Robiminer, Leonardo61, Twillisjr, Widr, WikiMSL, Luke145, EvaJamax, Debuntu, Helpful Pixie Bot, AlbertoBetulla, HMSSolent, Ngorman, Inoshika, Data.mining, ErinRea, BG19bot, Wanming149, PhnomPencil, Lisasolomonsalford, Uksas, Naeemmalik036, Chafe66, Onewhohelps, Netra Nahar, Aranea Mortem, Jasonem, Flaticida, Funkykeith777, Moshiurbd, Nathanashleywild, Anilkumar 0587, Mpaye, Rabarbaro70, Thundertide, BattyBot, Aacruzr, Warrenxu, IjonTichyIjonTichy, Harsh 2580, Dexbot, Webclient101, Mogism, TwoTwoHello, Frosty, Bradhill14, 7376a73b3bf0a490fa04bea6b76f4a4b, L8fortee, Dougs campbell, Mark viking, Cmartines, Epicgenius, THill182, Delafé, Melonkelon, Herpderp1235689999, Revengetechy, Amykam32, The hello doctor, Mimarios1, Huang cynthia, DavidLeighEllis, Gnust, Rbrandon87, Astigitana, Alihaghi, Philip Habing, Wccsnow, Jianhui67, Tahmina.tithi, Yeda123, Skr15081997, Charlotth, Jfrench7, Zjl9191, Davidhart007, Routerdecomposer, Augt.pelle, Justincahoon, Gstoel, Wiki-jonne, MatthewP42, 115ash, LiberumConsilium, Ran0512, Daniel Bachar, Galaktikasoft, Prof PD Hoy, Gary2015 and Anonymous: 973

- **Statistical classification** *Source:* http://en.wikipedia.org/wiki/Statistical%20classification?oldid=630022839 *Contributors:* The Anome, Michael Hardy, GTBacchus, Hike395, Robbot, Benwing, Giftlite, Beland, Violetriga, Kierano, Jérôme, Anthony Appleyard, Denoir, Oleg Alexandrov, Bkkbrad, Qwertyus, Bgwhite, Roboto de Ajvol, YurikBot, Jrbouldin, Tiffanicita, Tobi Kellner, SmackBot, Object01, Mcld, Chris the speller, Nervexmachina, Can't sleep, clown will eat me, Memming, Cybercobra, Richard001, Bohunk, Beetstra, Hu12, Billgaitas@hotmail.com, Trauber, Juansempere, Thijs!bot, Prolog, Mack2, Peteymills, VoABot II, Robotman1974, Quocminh9, RJASE1, Jamelan, ThomHImself, Gdupont, Junling, Melcombe, WikiBotas, Agor153, Addbot, Giggly37, Fgnievinski, SpBot, Movado73, Yobot, Oleginger, AnomieBOT, Ashershow1, Verbum Veritas, FrescoBot, Gire 3pich2005, DrilBot, Classifier1234, Jonkerz, Fly by Night, Microfries, Chire, Sigma0 1, Rmashhadi, ClueBot NG, Girish280, MerlIwBot, Helpful Pixie Bot, Chyvve, Swsboarder366, Klilidiplomus, Ferrarisailor, Mark viking, Francisbach, Imphil, I Less than3 Maths, LdyBruin and Anonymous: 65

- **Cluster analysis** *Source:* http://en.wikipedia.org/wiki/Cluster%20analysis?oldid=662268192 *Contributors:* The Anome, Fnielsen, Nealmcb, Michael Hardy, Shyamal, Kku, Tomi, GTBacchus, Den fjättrade ankan~enwiki, Cherkash, BAxelrod, Hike395, Dbabbitt, Phil Boswell, Robbot, Gandalf61, Babbage, Aetheling, Giftlite, Lcgarcia, Cfp, BenFrantzDale, Soundray~enwiki, Ketil, Khalid hassani, Angelo.romano, Dfrankow, Gadfium, Pgan002, Gene s, EBB, Sam Hocevar, Pwaring, Jutta, Abdull, Bryan Barnard, Rich Farmbrough, Mathiasl26, NeuronExMachina, Yersinia~enwiki, Bender235, Alex Kosorukoff, Aaronbrick, John Vandenberg, Greenleaf~enwiki, Ahc, NickSchweitzer, 3mta3, Jonsafari, Jumbuck, Jérôme, Terrycojones, Denoir, Jnothman, Stefan.karpinski, Hazard, Oleg Alexandrov, Soultaco, Woohookitty, Linas, Uncle G, Borb, Ruud Koot, Tabletop, Male1979, Joerg Kurt Wegner, DESiegel, Ruziklan, Sideris, BD2412, Qwertyus, Rjwilmsi, Koavf, Salix alba, Michal.burda, Denis Diderot, Klonimus, FlaBot, Mathbot, BananaLanguage, Kcarnold, Payo, Jrtayloriv, Windharp, BMF81, Roboto de Ajvol, The Rambling Man, YurikBot, Wavelength, Argav, SpuriousQ, Pseudomonas, NawlinWiki, Gareth Jones, Bayle Shanks, TCrossland, JFD, Hirak 99, Zzuuzz, Rudrasharman, Zigzaglee, Closedmouth, Dontaskme, Kevin, Killerandy, Airconswitch, SmackBot, Drakyoko, Jtneill, Pkirlin, Object01, Mcld, Ohnoitsjamie, KaragouniS, Bryan Barnard1, MalafayaBot, Drewnoakes, Tenawy, DHN-bot~enwiki, Iwaterpolo, Zacronos, MatthewKarlsen, Krexer, Bohunk, MOO, Lambiam, Friend of facts, Benash, ThomasHofmann, Dfass, Beetstra, Ryulong, Nabeth, Hu12, Iridescent, Ralf Klinkenberg, Madla~enwiki, Alanbino, Origin415, Bairam, Ioannes Pragensis, Joaoluis, Megannnn, Nczempin, Harej bot, Slack---line, Playtime, Endpoint, Dgtized, Skittleys, DumbBOT, Talgalili, Thijs!bot, Barticus88, Vinoduec, Mailseth, Danhoppe, Phoolimin, Onasraou, Denaxas, AndreasWittenstein, Daytona2, MikeLynch, JAnDbot, Inverse.chi, .anacondabot, Magioladitis, Andrimirzal, Fallschirmjäger, JBIdF, David Eppstein, User A1, Eeera, Varun raptor, LedgendGamer, Jiuguang Wang, Sommersprosse, Koko90, Smite-Meister, McSly, Dvdpwiki, DavidCBryant, AStrathman, Camrn86, TXiKiBoT, Rnc000, Tamás Kádár, Mundhenk, Maxim, Winterschlaefer, Lamro, Wheatin, Arrenbas, Sesilbumfluff, Tomfy, Kerveros 99, Seemu, WRK, Drdan14, Harveydrone, Graham853, Wcdriscoll, Zwerglein~enwiki, Osian.h, FghIJklm, Melcombe, Kotsiantis, Freeman77, Victor Chmara, Kl4m, Mugvin, Manuel freire, Boing! said Zebedee, Tim32, PixelBot, Lartoven, Chaosdruid, Aprock, Practical321, Qwfp, FORTRANslinger, Sunsetsky, Ocean931, Phantom xxiii, XLinkBot, Pichpich, Gnowor, Sujaykoduri, WikHead, Addbot, Allenchue, DOI bot, Bruce rennes, Fgnievinski, Gangcai, MrOllie, FerrousTigrus, Delaszk, Tide rolls, Lightbot, PAvdK, Fjrohlf, Tobi, Luckas-bot, Yobot, Gulfera, Hungpuiki, AnomieBOT, Flamableconcrete, Materialscientist, Citation bot, Xqbot, Erud, Sylwia Ufnalska, Simeon87, Omnipaedista, Kamitsaha, Playthebass, FrescoBot, Sacomoto, D'ohBot, Dan Golding, JohnMeier, Slowmo0815, Atlantia, Citation bot 1, Boxplot, Edfox0714, MondalorBot, Lotje, E.V.Krishnamurthy, Capez1, Koozedine, Tbalius, RjwilmsiBot, Ripchip Bot, Jchemmanoor, GodfriedToussaint, Aaronzat, Helwr, EmausBot, John of Reading, Stheodor, Elixirrixile, BOUMEDJOUT, ZéroBot, Sgoder, Chire, Darthhappyface, Jucypsycho, RockMagnetist, Wakebrdkid, Fazlican, Anita5192, ClueBot NG, Marion.cuny, Ericfouh, Simeos, Poirel, Robiminer, Michael-stanton, Girish280, Helpful Pixie Bot, Novusuna, BG19bot, Cpkex0102, Wiki13, TimSwast, Cricetus, Douglas H Fisher, Mu.ting, ColanR, Cornelius3, Illia Connell, Compsim, Mogism, Frosty, Abewley, Mark viking, Metcalm, Ninjarua, Trouveur de faits, TCMemoire, Monkbot, Leegrc, Imsubhashjha, Екатерина Конь, Olosko, Angelababy00 and Anonymous: 325

- **Anomaly detection** *Source:* http://en.wikipedia.org/wiki/Anomaly%20detection?oldid=657858542 *Contributors:* Andreas Kaufmann, Vonkje, Wavelength, Gareth Jones, Henryyan, Elonka, Krexer, Kvng, Nick Number, Madmarigold, Mark.hornick, Clangin, Javhar, Persian oracle, KirbyMaster14, Melcombe, Qwfp, Dthomsen8, Addbot, Yobot, AnomieBOT, Mario777Zelda, Professor Augustus Barclay Yaffle, Lkarsten~enwiki, Chire, EvaJamax, Vrlab, BG19bot, QualitycontrolUS, Gforman44, Andrea.venturini65, Nikaleido, Dexbot, Bigdata turn, Stamptrader, Monkbot, Bippina, Hfanaee and Anonymous: 28
- **Association rule learning** *Source:* http://en.wikipedia.org/wiki/Association%20rule%20learning?oldid=661174139 *Contributors:* SimonP, Michael Hardy, Angela, Azazello, Witbrock, Dfrankow, Neilc, Raand, Urhixidur, Adambro, Stesmo, WilliamKF, Rjwilmsi, Pseudomonas, Grafen, Gareth Jones, Crasshopper, Chughgaurav~enwiki, NHSavage, SmackBot, Reedy, Amux, Chris the speller, Mitar, Lambiam, Dicklyon, Beefyt, CmdrObot, ShelfSkewed, Harrigan, UberScienceNerd, Qwertyplus, Jeffreydiehl, A3nm, David Eppstein, Jnnnnn, Samtheboy, Dvdpwiki, Cobi, Hamman Samuel, Themacolyte, TXiKiBoT, Coastside, Kotsiantis, Jlpinar83, Autofei, Niceguyedc, Auntof6, Xodarap00, Stephengmatthews, Alokito, Rahul234, Life of Riley, Sunsetsky, Addbot, MichaelMampaey, Mhahsler, Aelkris, MrOllie, Greg4cr, Favonian, Yobot, Wim Leers, KamikazeBot, AnomieBOT, Broncobus, Citation bot, LilHelpa, Andrewmc123, FrescoBot, Citation bot 1, RedBot, Geoffrey I Webb, Trappist the monk, Cincoutprabu, Ali hadian, RjwilmsiBot, Mango bush, 2aprilboy, Frostyandy2k, Jbr jbr, Donner60, Chiu.chienpei, ChuispastonBot, Phoglenix, Pokbot, Kounoupis, ClueBot NG, Helpful Pixie Bot, HMSSolent, BG19bot, Uksas, Himanshujain123, Jdubin, AnsafSalleb, Ftrxx, Rahulkj, TwoTwoHello, Behroozomidvar, Dataesp, Dexterous1802, Rmasba, Kr4gfo87, Dsousacosta, Denny73, Monkbot, 4costlygrace, D Bhalodia, Dr.shaheen.khan, Joselunaariza, Ramezanics, Gingerlime, SnazzyFiend, Dndm97 and Anonymous: 113
- **Reinforcement learning** *Source:* http://en.wikipedia.org/wiki/Reinforcement%20learning?oldid=655647708 *Contributors:* Wmorgan, Imran, Mrwojo, Michael Hardy, Togelius, DopefishJustin, Kku, Delirium, Hike395, Charles Matthews, Robbot, Altenmann, Giftlite, Dratman, Gene s, Juxi, Urhixidur, Bender235, Tobacman, Diego Moya, Nvrmnd, Oleg Alexandrov, Olethros, Qwertyus, Seliopou, Mathbot, Banazir, Kri, Chobot, Bgwhite, YurikBot, Wavelength, Masatran, Digfarenough, SmackBot, Fabrice.Rossi, Vermorel, Jcarroll, Chris the speller, Ash.dyer, DHN-bot~enwiki, Mitar, Beetstra, Flohack, Ceran, Janrpeters, XApple, ShelfSkewed, Perimosocordiae, Skittleys, Rev.bayes, Escarbot, Tremilux, Parunach, R'n'B, Wfu, Jiuguang Wang, Shyking, Kpmiyapuram, Qsung, Szepi~enwiki, Nedrutland, Mdchang, Sebastjanmm, MrinalKalakrishnan, Flyer22, Melcombe, Rinconsoleao, MBK004, XLinkBot, Addbot, DOI bot, MrOllie, Download, Mianarshad, Yobot, Maderlock, Citation bot, LilHelpa, DSisyphBot, J04n, Gosavia, FrescoBot, Fgpilot, Kartoun, Mr ashyash, D'ohBot, Citation bot 1, Albertzeyer, Wikinacious, Skyerise, Trappist the monk, Dpbert, Stuhlmueller, RjwilmsiBot, Claggierk, EmausBot, Macopema, Chire, Jcautilli, DrewNoakes, Correction45, Rlguy, ChuispastonBot, Mbdts, Dvir-ad, Albertttt, Uymj, Helpful Pixie Bot, BG19bot, Stephen Balaban, ChrisGualtieri, Rbabuska, Ra ules, Chrislgarry, Awliehr, Monkbot, SoloGen and Anonymous: 117
- **Structured prediction** *Source:* http://en.wikipedia.org/wiki/Structured%20prediction?oldid=643965303 *Contributors:* Edward, Kku, Nowozin, Qwertyus, Brendan642, Semifinalist, Geo g guy, Yobot, AnomieBOT, Venustas 12, Alfaisanomega, SwimmingFox, Weiping.thu, Papertoys, Mathewk1300 and Anonymous: 3
- **Feature learning** *Source:* http://en.wikipedia.org/wiki/Feature%20learning?oldid=661746836 *Contributors:* Phil Boswell, Tobias Bergemann, Qwertyus, Rjwilmsi, Mcld, Kotabatubara, Dsimic, Yobot, AnomieBOT, BG19bot, Mavroudisv, TonyWang0316, Ixjlyons and Anonymous: 7
- **Online machine learning** *Source:* http://en.wikipedia.org/wiki/Online%20machine%20learning?oldid=656630296 *Contributors:* Mrwojo, Pgan002, Leondz, Qwertyus, Gmelli, Kri, BrotherE, R'n'B, Funandtrvl, Carriearchdale, P.r.newman, Themfromspace, AnomieBOT, Mesterharm, Surv1v4l1st, Masterhot93, X7q, Larry.europe, Chire, Helpful Pixie Bot, Ledkas82, BattyBot, Peg49, Ss044 and Anonymous: 10
- **Semi-supervised learning** *Source:* http://en.wikipedia.org/wiki/Semi-supervised%20learning?oldid=649528667 *Contributors:* Edward, Delirium, Furrykef, Benwing, Rajah, Arthena, Facopad, Soultaco, Bkkbrad, Ruud Koot, Qwertyus, Gmelli, Chobot, DaveWF, Cedar101, Jcarroll, Drono, Phoxhat, Rahimiali, Bookuser, Lamro, Tbmurphy, Addbot, MrOllie, Luckas-bot, Yobot, Gelbukh, AnomieBOT, Xqbot, Omnipaedista, Romainbrasselet, D'ohBot, Wokonen, EmausBot, Grisendo, Stheodor, Rahulkmishra, Pintaio, Helpful Pixie Bot, BG19bot, CarrieVS, AK456, Techerin, M.shahriarinia, Rcpt2 and Anonymous: 28
- **Grammar induction** *Source:* http://en.wikipedia.org/wiki/Grammar%20induction?oldid=661963338 *Contributors:* Delirium, Aabs, Jim Horning, NTiOzymandias, MCiura, Marudubshinki, Rjwilmsi, Koavf, SmackBot, Took, Bluebot, Rizzardi, Antonielly, Dfass, Hukkinen, Gregbard, Wikid77, Bobblehead, Erxnmedia, Tremilux, Stassa, Mgalle, KoenDelaere, Aclark17, 1ForTheMoney, Bility, Hiihammuk, Josve05a, Chire, KLBot2, BG19bot, Jochen Burghardt, Superploro and Anonymous: 7

12.8.2 Images

- **File:Ambox_important.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/b4/Ambox_important.svg *License:* Public domain *Contributors:* Own work, based off of Image:Ambox scales.svg *Original artist:* Dsmurat (talk · contribs)
- **File:Animation2.gif** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/c0/Animation2.gif *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* MG (talk · contribs)
- **File:Cluster-2.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/c8/Cluster-2.svg *License:* Public domain *Contributors:*
- Cluster-2.gif *Original artist:* Cluster-2.gif: hellisp
- **File:Commons-logo.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* ? *Contributors:* ? *Original artist:* ?
- **File:Edit-clear.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/f/f2/Edit-clear.svg *License:* Public domain *Contributors:* The *Tango! Desktop Project*. *Original artist:*

  The people from the Tango! project. And according to the meta-data in the file, specifically: “Andreas Nilsson, and Jakub Steiner (although minimally).”
- **File:Example_of_unlabeled_data_in_semisupervised_learning.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/d0/Example_of_unlabeled_data_in_semisupervised_learning.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Techerin
- **File:Fisher_iris_versicolor_sepalwidth.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/4/40/Fisher_iris_versicolor_sepalwidth.svg *License:* CC BY-SA 3.0 *Contributors:* en:Image:Fisher iris versicolor sepalwidth.png *Original artist:* en:User:Qwfp (original); Pbroks13 (talk) (redraw)

12.8.3 Content license