Classification

Ian H. Witten
Department of Computer Science, University of Waikato, Hamilton, New Zealand

Synonyms

Classification learning; Supervised learning; Learning with a teacher, Concept learning; Statistical decision techniques

Definition

In *Classification learning*, an algorithm is presented with a set of classified examples or "instances" from which it is expected to infer a way of classifying unseen instances into one of several "classes". Instances have a set of features or "attributes" whose values define that particular instance. Numeric prediction, or "regression," is a variant of classification learning in which the class attribute is numeric rather than categorical. Classification learning is sometimes called *supervised* because the method operates under supervision by being provided with the actual outcome for each of the training instances. This contrasts with Data clustering (see entry Data Clustering), where the classes are not given, and with Association learning (see entry Association Learning), which seeks any association – not just one that predicts the class.

Historical Background

Classification learning grew out of two strands of work that began in the 1950s and were actively pursued throughout the 1960s: statistical decision techniques and the Perceptron model of neural networks. In 1955 statisticians Bush and Mosteller published a seminal book *Stochastic Models for Learning* which modeled in mathematical terms the psychologist B. F. Skinner's experimental analyses of animal behavior using reinforcement learning [2]. The "perceptron" was a one-level linear classification scheme developed by Rosenblatt around 1957 and published in his book *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms* [10]. In a response published in 1969, Minsky and Papert argued that perceptrons were simplistic in terms of their representational capability and had been greatly over-hyped as potentially universal learning machines [6]. This scathing response by widely-respected artificial intelligence pioneers dampened research in neural nets and machine learning in general. Meanwhile, in 1957 others were investigating the application of Bayesian decision schemes to pattern recognition; the general conclusion was that full Bayesian models were prohibitively expensive. In 1960 Maron investigated in the context of information retrieval what has since become known as the "naïve Bayes" approach, which assumes independence between attributes notwithstanding overwhelming evidence to the contrary [5]. Other early machine learning work was buried in cybernetics, the study of feedback and derived concepts such as communication and control in living and artificial organisms. Throughout the 1960s classification learning applied to pattern recognition was the central thread of the embryo field of machine learning, as underlined by the subtitle of Nilsson's 1965 landmark book *Learning Machines – Foundations of Trainable Pattern-Classifying Systems* [7].

Symbolic learning techniques began to recover from the doldrums in the late 1970s, with influential and almost simultaneous publications by Breiman et al. on *classification and regression trees* (the CART system) [1] and Quinlan on *decision tree induction* (the ID3 and later C4.5 systems) [8,9]. Whereas Breiman was a statistician, Quinlan was an experimental computer scientist who first used decision trees not to generalize but to condense large collections of chess end-games. Their work proceeded independently, and the similarities remained unnoticed until years later. CART (by default) produces multivariate trees whose tests can involve more than one attribute: these are more accurate and smaller than the univariate trees produced by Quinlan's systems, but take longer to generate.

The first workshop devoted to machine learning was held in 1980 at Carnegie-Mellon University; further workshops followed in 1983 and 1985. These invitation-only events became an open conference in 1988. Meanwhile the journal *Machine Learning* was established in 1986. By the 1990s the subject had become the poster child of artificial intelligence – a successful, burgeoning, practical technology that eschewed the classical topics of general knowledge representation, logical deduction, theorem proving, search techniques, computational linguistics, expert systems and philosophical foundations that still characterize the field today. Classification learning, which forms the core of machine learning, outgrew its behaviorist and neurological roots and moved into the practical realm of database systems.

Early work focused on the *process* of learning – learning curves, the possibility of sustained learning, and the like – rather than the *results* of learning. However, with the new emphasis on applications, objective techniques of empirical testing began to supplant the scenario-based style of evaluation that characterized the early days. A major breakthrough came during the 1980s when researchers finally realized that evaluating a learning system on its training data gave misleading results, and instead put the subject on a secure statistical footing.

Scientific Fundamentals

One of the most instructive lessons learned since the renaissance of classification in the 1980s is that simple schemes often work very well. Today, practitioners strongly recommend the adoption of a "simplicity-first" methodology when analyzing practical datasets. There are many different kinds of simple structure that datasets can exhibit. One dataset might have a single attribute that does all the work, the others being irrelevant or redundant. Alternatively, the attributes might contribute independently and equally to the final outcome. Underlying a third dataset might be a simple contingent structure involving just a few attributes. In a fourth, a few independent rules may govern the assignment of instances to classes. In a fifth, classifications appropriate to particular regions of instance space might depend on the distance between the instances themselves. A sixth might exhibit dependence among numeric attributes, determined by a sum of attribute values with appropriately chosen weights. This sum might represent the final output for numeric prediction, or be compared to a fixed threshold in a binary decision setting. Each of these examples leads to a different style of method suited to discovering that kind of structure.

Rules Based on a Single Attribute

Even when instances have several attributes, the classification decision may rest on the value of just one of them. Such a structure constitutes a set of rules that all test the same attribute (or, equivalently, a one-level decision tree). It can be found by evaluating the success, in terms of the total number of errors on the training data, of testing each attribute in turn, predicting the most prevalent class for each value of that attribute. If an attribute has many possible values – and particularly if it has numeric values – this may "overfit" the training data by generating a rule that has almost as many branches as there are instances. Minor modifications to the scheme overcome this problem.

A startling discovery published in 1993 was that "very simple classification rules perform well on most commonly used datasets" [3]. In an empirical investigation of the accuracy of rules that classify instances on the basis of a single attribute, on most standard datasets the resulting rule was found to be as accurate as the structures induced by the majority of machine learning systems – which are far more complicated. The moral? – always compare new methods with simple baseline schemes.

Statistical Modeling (see entry Bayesian Classification)

Another simple technique is to use all attributes and allow them to make contributions to the decision that are *equally important* and *independent* of one another, given the class. Although grossly unrealistic – what makes real-life datasets interesting is that the attributes are certainly not equally important or independent – it leads to a statistically-based scheme that works surprisingly well in practice. Employed in information retrieval as early as 1960 [5], the idea was rediscovered, dubbed "naïve Bayes," and introduced into machine learning 30 years later [4]. Despite the disparaging moniker it works well on many actual datasets. Over-reliance on the independence of attributes can be countered by applying attribute selection techniques.

Divide and Conquer Technique (see entry Decision Tree Classification)

The process of constructing a decision tree can be expressed recursively. First, select an attribute to use

at the root, and make a branch for each possible value. This splits the instance set into subsets, one for every value of the attribute. Now repeat the process recursively for each branch, using only those instances that actually reach the branch. If all instances at a node have the same classification, stop developing that part of the tree. This method of "top-down induction of decision trees" was explored and popularized by Quinlan [8,9]. The nub of the problem is to select an appropriate attribute at each stage. Of many heuristics that have been investigated, the dominant one is to measure the expected amount of information gained by knowing that attribute's actual value. Having generated the tree, it is selectively pruned back from the leaves to avoid over-fitting. A series of improvements include ways of dealing with numeric attributes, missing values, and noisy data; and generating rules from trees.

Covering Algorithms (see entry Rule-Based Classification)

Classification rules can be produced by taking each class in turn and seeking a rule that covers all its instances, at the same time excluding instances not in the class. This bottom-up approach is called *covering* because at each stage a rule is identified that "covers" some of the instances. Although trees can always be converted into an equivalent rule set, and vice versa, the perspicuity of the representation often differs. Rules can be symmetric whereas trees must select one attribute to split on first, which can produce trees that are much larger than an equivalent set of rules. In the multiclass case a decision tree split takes account of all classes and maximizes the information gained, whereas many rule generation methods concentrate on one class at a time, disregarding what happens to the others.

Instance-Based Learning (see entry Nearest Neighbor Classification)

Another approach is to store training instances verbatim and, given an unknown test instance, use a distance function to determine the closest training instance and predict its class for the test instance. Suitable distance functions are the Euclidean or Manhattan (city-block) metric; attributes should be normalized to lie between 0 and 1 to compensate for scaling effects. For nominal attributes that assume symbolic rather than numeric values, the distance between two values is 1 if they are not the same and 0 otherwise. In the *k*-nearest neighbor strategy, some fixed number of nearest neighbors – say five – are located and used together to determine the class of the test instance by majority vote. Another way of proofing the database against noise is to selectively and judiciously choose the exemplars that are added. Nearest-neighbor classification was notoriously slow until advanced data structures like *k*D-trees were applied in the early 1990s.

Linear Models (see entry Linear Regression)

When the outcome and all attributes are numeric, linear regression can be used. This expresses the class as a linear combination of the attributes, with weights that are calculated from the training data. Linear regression has been popular in statistical applications for decades. If the data exhibits a nonlinear dependency, the best-fitting straight line will be found, where "best" is interpreted in the least-mean-squared-difference sense. Although this line may fit poorly, linear models can serve as building blocks for more complex learning schemes.

Linear Classification (see entry Neural Networks, Support Vector Machine)

The idea of linear classification is to find a hyperplane in instance space that separates two classes. (In the multiclass case, a binary decision can be learned for each pair of classes.) If the linear sum exceeds zero the first class is predicted; otherwise the second is predicted. If the data is linearly separable – that is, it can be separated perfectly using a hyperplane – the perceptron learning rule espoused by Rosenblatt is guaranteed to find a separating hyperplane [10]. This rule adjusts the weight vector whenever the prediction for a particular instance is erroneous: if the first class is predicted the instance (expressed as a vector) is added to the weight vector (making it more likely that the result will be positive next time around); otherwise the instance is subtracted.

There have been many powerful extensions of this basic idea. Support vector machines use linear decisions to implement nonlinear class boundaries by transforming the input using a nonlinear mapping. Multilayer perceptrons connect many linear models in a hierarchical arrangement that can represent nonlinear decision boundaries, and use a technique called "back-propagation" to distribute the effect of errors through this hierarchy during training.

Missing Values

Most datasets encountered in practice contain missing values. Sometimes different kinds are distinguished (e.g., unknown vs. unrecorded vs. irrelevant values). They may occur for a variety of reasons. There may be some significance in the fact that a certain instance has an attribute value missing – perhaps a decision was taken not to perform some test – and that might convey information about the instance other than the mere absence of the value. If this is the case, *not tested* should be recorded as another possible value for this attribute. Only someone familiar with the data can make an informed judgment as to whether a particular value being missing has some significance or should simply be coded as an ordinary missing value. For example, researchers analyzing medical databases have noticed that cases may, in some circumstances, be diagnosable strictly from the tests that a doctor decides to make, regardless of the outcome of the tests. Then a record of which values are "missing" is all that is needed for a complete diagnosis – the actual measurements can be ignored entirely!

Meta-Learning

Decisions can often be improved by combining the output of several different models. Over the past decade or so the techniques of bagging (see entry Bagging), boosting (see entry Boosting), and *stacking* have been developed that learn an ensemble of models and deploy them together. Their performance is often astonishingly good. Researchers have struggled to understand why, and during that struggle new methods have emerged that are sometimes even better. For example, whereas human committees rarely benefit from noisy distractions, shaking up bagging by adding random variants of classifiers can improve performance. Boosting – perhaps the most powerful of the three methods – is related to the established statistical technique of additive models, and this realization has led to improved procedures.

Combined models share the disadvantage of being rather hard to analyze: they can comprise dozens or even hundreds of individual learners and it is not easy to understand in intuitive terms what factors are contributing to the improved decisions. In the last few years methods have been developed that combine the performance benefits of committees with comprehensible models. Some produce standard decision tree models; others introduce new variants of trees that provide optional paths.

Evaluation

For classification problems, performance is naturally measured in terms of the *error rate*. The classifier predicts the class of each test instance: if it is correct, that is counted as a success; if not, it is an error. The error rate is the proportion of errors made over a whole set of instances, and reflects the overall performance of the classifier. Performance on the training set is definitely *not* a good indicator of expected performance on an independent test set. A classifier is *overfitted* to a dataset if its structure reflects that particular set to an excessive degree. For example, the classifier might be generated by rote learning without any generalization whatsoever. An overfitted classifier usually exhibits performance on the training set which is excellent but far from representative of performance on other datasets from the same source.

In practice, one must predict performance bounds based on experiments with whatever data is available. Labeled data is required for both training and testing, and is often hard to obtain. A single data set can be partitioned for training and testing in various different ways. In a popular statistical technique called *cross-validation* the experimenter first decides on a fixed number of "folds," or partitions of the data – say three. The data is split into three approximately equal portions, and each in turn is used for testing while the remainder serves for training. The procedure is repeated three times so that in the end every instance has been used exactly once for testing. This is called *threefold cross-validation*. "Stratification" is the idea of ensuring that all classes are represented in all folds in approximately the right proportions. *Stratified tenfold cross-validation* has become a common standard for estimating the error rate of a classification learning scheme. Alternatives include *leave-one-out* cross-validation, which is effectively n-fold cross-validation where n is the size of the data set; and the *bootstrap*, which takes a carefully-judged number of random samples from the data with replacement and uses these for training, combining the error rate on the training data (an optimistic estimate) with that on the test data (a pessimistic estimate, since the classifier has only been trained on a subset of the full data) to get an overall estimate.

Key Applications

Classification learning is one of the flagship triumphs of research in artificial intelligence. It has been used for problems that range from selecting promising embryos

to implant in a human womb during in vitro fertilization to the selection of which cows in a herd to sell off to an abattoir. Fielded applications are legion. They include decisions involving judgment, such as whether a credit company should make a loan to a particular person; screening images, such as the detection of oil slicks from satellite images; load forecasting, such as combining historical load information with current weather conditions and other events to predict hourly demand for electricity; diagnosis, such as fault finding and preventative maintenance of electromechanical devices; marketing and sales, such as detecting customers who are likely to switch to a competitor.

URL to Code

The Weka machine learning workbench is a popular tool for experimental investigation and comparison of classification learning techniques, as well as other machine learning methods. It is described in [11] and available for download from http://www.cs.waikato.ac.nz/ml/weka.

Cross-references

- ► Association Rule Mining
- ► Bagging
- ► Bayesian Classification
- ► Boosting
- ► Bootstrap
- ► Classification by Association Rule Analysis
- ► Clustering Overview and Applications
- ► Cross Validation
- ► Data Mining
- ► Decision Tree Classification
- ► Fuzzy Set Approach
- ► Genetic Algorithm
- ► Linear Regression
- ► Log-Linear Regression
- ► Nearest Neighbor Classification
- ► Neural Networks
- ► ROC
- ► Rough Set Approach
- ► Rule-Based classification
- ► Support Vector Machine

Recommended Reading

1. Breiman L., Friedman J.H., Olshen R.A., and Stone C.J. Classification and Regression Trees. Wadsworth, Pacific Grove, CA, 1984.
2. Bush R.R. and Mosteller F. Stochastic Models for Learning. Wiley, New York, 1955.
3. Holte R.C. "Very simple classification rules perform well on most commonly used datasets." Mach. Learn., 11:63–91, 1993.
4. Kononenko I. "ID3, sequential Bayes, naïve Bayes and Bayesian neural networks." In Proceedings of the Fourth European Working Session on Learning. Montpellier, France, 1989, pp. 91–98.
5. Maron M.E. and Kuhns J.L. "On relevance, probabilistic indexing and information retrieval." J. ACM, 7(3):216–244, 1960.
6. Minsky M.L. and Papert S. Perceptrons. Cambridge, MIT Press, 1969.
7. Nilsson N.J. Learning Machines. McGraw-Hill, New York, 1965.
8. Quinlan J.R. "Induction of decision trees." Mach. Learn., 1(1):81–106, 1986.
9. Quinlan J.R. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, CA, 1993.
10. Rosenblatt F. Principles of Neurodynamics. Spartan, Washington, DC, 1961.
11. Witten I.H. and Frank E. Data Mining: Practical Machine Learning Tools and Techniques (2nd edn.). Morgan Kaufmann, San Francisco, CA, 2003.