Predictive Data Mining: A Survey of Regression Methods

Sotiris Kotsiantis
University of Patras, Greece & University of Peloponnese, Greece

Panayotis Pintelas
University of Patras, Greece & University of Peloponnese, Greece

INTRODUCTION

Data mining is the extraction of implicit, previously unknown, and potentially useful information from data. The idea is to build computer programs that sift through databases automatically, seeking regularities or patterns. Strong patterns, if found, will likely generalize to make accurate predictions on future data. Machine learning (ML) provides the technical basis of data mining. It is used to extract information from the raw data in databases—information that is expressed in a comprehensible form and can be used for a variety of purposes.

Every instance in any data set used by ML algorithms is represented using the same set of features. The features may be continuous, categorical, or binary. If instances are given with known labels (the corresponding correct outputs), then the learning is called supervised in contrast to unsupervised learning, where instances are unlabeled (Kotsiantis & Pintelas, 2004). This work is concerned with regression problems in which the output of instances admits real values instead of discrete values in classification problems.

BACKGROUND

A brief review of what ML includes can be found in Dutton and Conroy (1996). A historical survey of logic and instance-based learning is also presented in De Mantaras and Armengol (1998). The first step of predictive data mining is collecting the data set. If a requisite expert is available, then he or she can suggest which fields (attributes, features) are the most informative. If not, then the simplest method is that of "brute force," which means measuring everything available in the hope that the right (informative, relevant) features can be isolated. However, a data set collected by the brute-force method is not directly suitable for induction. It contains, in most cases, noise and missing feature values, and therefore requires significant preprocessing (Zhang, Zhang, & Yang, 2002). Hodge and Austin (2004) have recently introduced a survey of contemporary techniques for outlier (noise) detection. Depending on the circumstances, researchers have a number of methods to choose from to handle missing data (Batista & Monard, 2003). Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible (Yu & Liu, 2004). The fact that many features depend on one another often unduly influences the accuracy of supervised ML models. This problem can be addressed by constructing new features from the basic feature set (Markovitch & Rosenstein, 2002).

The problem of regression consists of obtaining a functional model that relates the value of a target continuous-variable y with the values of variables x_1, x_2...x_n (the predictors). This model is obtained using samples of the unknown regression function. These samples describe different mappings between the predictor and the target variables. The traditional approach for prediction of a continuous target is the classical linear least-squares regression (Fox, 1997). The model constructed for regression in this traditional approach is a linear equation. By estimating the parameters of this equation with a computationally simple process on the training set, a model is created. However, the linearity assumption between input features and predicted value introduces a large bias error for most domains. That is why most studies are directed to nonlinear and nonparametric techniques for the regression problem.

Murthy (1998) provided an overview of work in decision trees and a sample of their usefulness to newcomers as well as practitioners in the field of data mining. Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Regression trees are the counterpart of decision trees for regression tasks. A regression tree, or any learned hypothesis h, is said to overfit training data if another hypothesis h´ exists that has a larger error than h when tested on the training data, but a smaller error than h when tested on the entire data set. There are two common approaches that regression-tree induction algorithms can use to avoid overfitting training data: (a) Stop the training algorithm before it reaches a point at which it perfectly fits the training data, and (b) prune the induced regression tree.

Most algorithms use a pruning method. Model trees generalize the concepts of regression trees, which have constant values at their leaves (Torgo, 2000). Thus, they are analogous to piecewise linear functions (and hence nonlinear functions). The major advantage of model trees over regression trees is that model trees are much smaller than regression trees, the decision strength is clear, and regression functions do not normally involve many variables. The most well-known model-tree inducer is the M5´ (Wang & Witten, 1997). Model trees can tackle tasks with very high dimensionality—up to hundreds of attributes; however, computational requirements grow rapidly with dimensionality.

Regression trees can be translated into a set of rules by creating a separate rule for each path from the root to a leaf in the tree (Torgo, 1995). The M5´ rules algorithm produces propositional regression rules using routines for generating a decision list from M5´ model trees (Witten & Frank, 2005). The algorithm is able to deal with both continuous and nominal variables, and obtains a piecewise linear model of the data. However, rules can also be directly induced from training data using a variety of rule-based algorithms. Furnkranz (1999) provided an excellent overview of existing work in rule-based methods. Regression rules represent each result by a disjunctive normal form (DNF). A *k*-DNF expression is of the form $(X_1 \wedge X_2 \wedge \ldots \wedge X_n) \vee (X_{n+1} \wedge X_{n+2} \wedge \ldots X_{2n}) \vee \ldots \vee (X_{(k-1)n+1} \wedge X_{(k-1)n+2} \wedge \ldots \wedge X_{kn})$, where k is the number of disjunctions, n is the number of conjunctions in each disjunction, and X_n is defined over the alphabet $X_1, X_2 \ldots X_j \cup \sim X_1, \sim X_2 \ldots \sim X_j$. The goal is to construct the smallest rule set that is consistent with the training data. A large number of learned rules is usually a sign that the learning algorithm is attempting to remember the training set instead of discovering the assumptions that govern it. The difference between heuristics for rule learning and heuristics for regression trees is that the latter evaluate the average quality of a number of disjointed sets (one for each value of the feature that is tested), while rule learners only evaluate the quality of the set of instances that is covered by the candidate rule (Torgo, 1995).

Artificial neural networks (ANNs) are another method of inductive learning based on computational models of biological neurons and networks of neurons as found in the central nervous system of humans (Witten & Frank, 2005). A multilayer neural network consists of a large number of units (neurons) joined together in a pattern of connections. Units in a net are usually segregated into three classes: input units, which receive information to be processed; output units, where the results of the processing are found; and units in between known as hidden units. Feed-forward ANNs allow signals to travel one way only, from input to output. First, the network is trained on a set of paired data to determine input-output mapping. The weights of the connections between neurons are then fixed and the network is used to predict the value of a new set of data. Generally, properly determining the size of the hidden layer is a problem. An excellent argument regarding this topic can be found in Camargo and Yoneyama (2001). Kon and Plaskota (2000) also studied the minimum amount of neurons and the number of instances necessary to program a given task into feed-forward neural networks. ANN depends upon three fundamental aspects: input and activation functions of the unit, network architecture, and the weight of each input connection. Given that the first two aspects are fixed, the behavior of the ANN is defined by the current values of the weights. The weights of the net to be trained are initially set to random values, and then instances of the training set are repeatedly exposed to the net. The values for the input of an instance are placed on the input units and the output of the net is compared with the desired output for this instance. Then, all the weights in the net are adjusted slightly in the direction that would bring the output values of the net closer to the values for the desired output. There are several algorithms with which a network can be trained (Neocleous & Schizas, 2002). However, the most well-known and widely used learning algorithm to estimate the values of the weights is the back-propagation (BP) algorithm.

Instance-based learning algorithms are lazy learning algorithms as they delay the induction or generalization process until the regression process is performed. *k*-nearest neighbor (k-NN) is based on the principle that the instances within a data set will generally exist in close proximity with other instances that have similar properties (Aha, 1997). The k-NN algorithm first finds the closest instances to the query point in the instance space according to a distance measure, and then outputs the average of the target values of those instances as the prediction for that query instance. Many different metrics for the relative distance between instances have been presented (Witten & Frank, 2005). For more accurate results, several algorithms use weighting schemes that alter the distance measurements and voting influence of each instance. A survey of weighting schemes is given by Wettschereck, Aha, and Mohri (1997). As the prediction of the target value of a query instance requires one to measure its distance to all training instances, which might be a very huge set, the prediction in k-NN is very costly.

An excellent survey of support vector machines (SVMs) can be found in (Burges, 1998). The sequential minimal optimization (SMO) algorithm has been shown to be an effective method for training support vector machines on classification tasks defined on sparse data sets (Platt, 1999). SMO differs from most SVM algorithms in that it does not require a quadratic programming solver. In Shevade, Keerthi, Bhattacharyya, and Murthy (2000) and Flake and Lawrence (2002), SMO is generalized so that it can handle regression problems. This implementation globally replaces all missing values and transforms nominal attributes into binary ones.