Chapter 8

INTRODUCTION TO SUPERVISED METHODS

Oded Maimon
Department of Industrial Engineering
Tel-Aviv University
maimon@eng.tau.ac.il

Lior Rokach
Department of Industrial Engineering
Tel-Aviv University
liorr@eng.tau.ac.il

Abstract This chapter summarizes the fundamental aspects of supervised methods. The chapter provides an overview of concepts from various interrelated fields used in subsequent chapters. It presents basic definitions and arguments from the supervised machine learning literature and considers various issues, such as performance evaluation techniques and challenges for data mining tasks.

Keywords: Attribute, Classifier, Inducer, Regression, Training Set, Supervised Methods, Instance Space, Sampling, Generalization Error

1. Introduction

Supervised methods are methods that attempt to discover the relationship between input attributes (sometimes called independent variables) and a target attribute (sometimes referred to as a dependent variable). The relationship discovered is represented in a structure referred to as a *model*. Usually models describe and explain phenomena, which are hidden in the dataset and can be used for predicting the value of the target attribute knowing the values of the input attributes. The supervised methods can be implemented in a variety of domains such as marketing, finance and manufacturing.

It is useful to distinguish between two main supervised models: *classification models* (*classifiers)* and *Regression Models*. Regression models map the input space into a real-value domain. For instance, a regressor can predict the demand for a certain product given its characteristics. On the other hand, classifiers map the input space into pre-defined classes. For instance, classifiers can be used to classify mortgage consumers as good (fully payback the mortgage on time) and bad (delayed payback). There are many alternatives for representing classifiers, for example, support vector machines, decision trees, probabilistic summaries, algebraic function, etc.

Along with regression and probability estimation, classification is one of the most studied models, possibly one with the greatest practical relevance. The potential benefits of progress in classification are immense since the technique has great impact on other areas, both within Data Mining and in its applications.

2. Training Set

In a typical supervised learning scenario, a training set is given and the goal is to form a description that can be used to predict previously unseen examples.

The training set can be described in a variety of languages. Most frequently, it is described as a *bag instance* of a certain *bag schema*. A *bag instance* is a collection of tuples (also known as records, rows or instances) that may contain duplicates. Each tuple is described by a vector of attribute values. The bag schema provides the description of the attributes and their domains. A bag schema is denoted as $B(A \cup y)$. Where A denotes the set of input attributes containing n attributes: $A = \{a_1, \ldots, a_i, \ldots, a_n\}$ and y represents the class variable or the target attribute.

Attributes (sometimes called field, variable or feature) are typically one of two types: nominal (values are members of an unordered set), or numeric (values are real numbers). When the attribute a_i is nominal, it is useful to denote by $dom(a_i) = \{v_{i,1}, v_{i,2}, \ldots, v_{i,|dom(a_i)|}\}$ its domain values, where $|dom(a_i)|$ stands for its finite cardinality. In a similar way, $dom(y) = \{c_1, \ldots, c_{|dom(y)|}\}$ represents the domain of the target attribute. Numeric attributes have infinite cardinalities.

The instance space (the set of all possible examples) is defined as a Cartesian product of all the input attributes domains: $X = dom(a_1) \times dom(a_2) \times \ldots \times dom(a_n)$. The universal instance space (or the *labeled instance space)* U is defined as a Cartesian product of all input attribute domains and the target attribute domain, i.e.: $U = X \times dom(y)$.

The training set is a bag instance consisting of a set of m tuples. Formally the training set is denoted as $S(B) = (\langle x_1, y_1 \rangle, \ldots, \langle x_m, y_m \rangle)$ where $x_q \in X$ and $y_q \in dom(y)$.

It is usually assumed that the training set tuples are generated randomly and independently according to some fixed and unknown joint probability distribution D over U. Note that this is a generalization of the deterministic case when a supervisor classifies a tuple using a function $y = f(x)$.

We use the common notation of bag algebra to present projection (π) and selection (σ) of tuples (Grumbach and Milo, 1996).

3. Definition of the Classification Problem

Originally the machine learning community introduced the problem of *concept learning*. Concepts are mental categories for objects, events, or ideas that have a common set of features. According to Mitchell (1997): "each concept can be viewed as describing some subset of objects or events defined over a larger set" (e.g., the subset of a vehicle that constitues trucks). To learn a concept is to infer its general definition from a set of examples. This definition may be either explicitly formulated or left implicit, but either way it assigns each possible example to the concept or not. Thus, a concept can be regarded as a function from the Instance space to the Boolean set, namely: $c : X \rightarrow \{-1, 1\}$. Alternatively, one can refer a concept c as a subset of X, namely: $\{x \in X : c(x) = 1\}$. A *concept class* C is a set of concepts.

Other communities, such as the KDD community prefer to deal with a straightforward extension of *concept learning*, known as The *classification problem* or *multi-class classification problem*. In this case, we search for a function that maps the set of all possible examples into a pre-defined set of class labels which are not limited to the Boolean set. Most frequently the goal of the classifiers inducers is formally defined as follows.

DEFINITION 8.1 *Given a training set S with input attributes set $A = \{a_1, a_2, \ldots, a_n\}$ and a nominal target attribute y from an unknown fixed distribution D over the labeled instance space, the goal is to induce an optimal classifier with minimum generalization error.*

The generalization error is defined as the misclassification rate over the distribution D. In case of the nominal attributes it can be expressed as:

$$\varepsilon(I(S), D) = \sum_{\langle x,y \rangle \in U} D(x, y) \cdot L(y, I(S)(x))$$

where $L(y, I(S)(x)$ is the zero-one loss function defined as:

$$L(y, I(S)(x)) = \begin{cases} 0 & if\, y = I(S)(x) \\ 1 & if\, y \neq I(S)(x) \end{cases} \tag{8.1}$$

In case of numeric attributes the sum operator is replaced with the integration operator.

4. Induction Algorithms

An *induction algorithm*, or more concisely an *inducer* (also known as learner), is an entity that obtains a training set and forms a model that generalizes the relationship between the input attributes and the target attribute. For example, an inducer may take as an input, specific training tuples with the corresponding class label, and produce a *classifier*.

The notation I represents an inducer and $I(S)$ represents a model which was induced by performing I on a training set S. Using $I(S)$ it is possible to predict the target value of a tuple x_q. This prediction is denoted as $I(S)(x_q)$.

Given the long history and recent growth of the field, it is not surprising that several mature approaches to induction are now available to the practitioner.

Classifiers may be represented differently from one inducer to another. For example, C4.5 (Quinlan, 1993) represents a model as a decision tree while Naïve Bayes (Duda and Hart, 1973) represents a model in the form of probabilistic summaries. Furthermore, inducers can be deterministic (as in the case of C4.5) or stochastic (as in the case of back propagation)

The classifier generated by the inducer can be used to classify an unseen tuple either by explicitly assigning it to a certain class (crisp classifier) or by providing a vector of probabilities representing the conditional probability of the given instance to belong to each class (probabilistic classifier). Inducers that can construct probabilistic classifiers are known as probabilistic inducers. In this case it is possible to estimate the conditional probability $\hat{P}_{I(S)}(y = c_j \,|a_i = x_{q,i} \;; i = 1, \ldots, n)$ of an observation x_q. Note the addition of the "hat" - ˆ - to the conditional probability estimation is to distinguish it from the actual conditional probability.

The following chapters review some of the major approaches to concept learning.

5. Performance Evaluation

Evaluating the performance of an inducer is a fundamental aspect of machine learning. As stated above, an inducer receives a training set as input and constructs a classification model that can classify an unseen instance . Both the classifier and the inducer can be evaluated using an evaluation criteria. The evaluation is important for understanding the quality of the model (or inducer); for refining parameters in the KDD iterative process; and for selecting the most acceptable model (or inducer) from a given set of models (or inducers).

There are several criteria for evaluating models and inducers. Naturally, classification models with high accuracy are considered better. However, there are other criteria that can be important as well, such as the computational complexity or the comprehensibility of the generated classifier.

5.1 Generalization Error

Let $I(S)$ represent a classifier generated by an inducer I on S. Recall that the generalization error of $I(S)$ is its probability to misclassify an instance selected according to the distribution D of the instance labeled space. The *classification accuracy* of a classifier is one minus the generalization error. The *training error* is defined as the percentage of examples in the training set correctly classified by the classifier, formally:

$$\hat{\varepsilon}(I(S),S) = \sum_{\langle x,y\rangle \in S} L(y, I(S)(x)) \tag{8.2}$$

where $L(y, I(S)(x))$ is the zero-one loss function defined in Equation 61.2.

Although generalization error is a natural criterion, its actual value is known only in rare cases (mainly synthetic cases). The reason for that is that the distribution D of the instance labeled space is not known.

One can take the training error as an estimation of the generalization error. However, using the training error as-is will typically provide an optimistically biased estimate, especially if the learning algorithm *over-fits* the training data.

There are two main approaches for estimating the generalization error: theoretical and empirical.

5.2 Theoretical Estimation of Generalization Error

A low training error does not guarantee low generalization error. There is often a trade-off between the training error and the confidence assigned to the training error as a predictor for the generalization error, measured by the difference between the generalization and training errors. The capacity of the inducer is a determining factor for this confidence in the training error. Indefinitely speaking, the capacity of an inducer indicates the variety of classifiers it can induce. The notion of VC-Dimension presented below can be used as a measure of the inducers capacity.

Inducers with a large capacity, e.g. a large number of free parameters, relative to the size of the training set are likely to obtain a low training error, but might just be memorizing or over-fitting the patterns and hence exhibit a poor generalization ability. In this regime, the low error is likely to be a poor predictor for the higher generalization error. In the opposite regime, when the capacity is too small for the given number of examples, inducers may under-fit the data, and exhibit both poor training and generalization error. For inducers with an insufficient number of free parameters, the training error may be poor, but it is a good predictor for the generalization error. In between these capacity extremes, there is an optimal capacity for which the best generalization error is obtained, given the character and amount of the available training data.

In "Mathematics of Generalization" Wolpert (1995) discuss four theoretical frameworks for estimating the generalization error, namely: PAC, VC and Bayesian, and statistical physics. All these frameworks combine the training error (which can be easily calculated) with some penalty function expressing the capacity of the inducers.

5.2.1 VC-Framework. Of all the major theoretical approaches to learning from examples, the Vapnik–Chervonenkis theory (Vapnik, 1995) is the most comprehensive, applicable to regression, as well as classification tasks. It provides general, necessary and sufficient conditions for the consistency of the induction procedure in terms of bounds on certain measures. Here we refer to the classical notion of consistency in statistics: both the training error and the generalization error of the induced classifier must converge to the same minimal error value as the training set size tends to infinity. Vapnik's theory also defines a capacity measure of an inducer, the VC-dimension, which is widely used.

VC-theory describes a worst-case scenario: the estimates of the difference between the training and generalization errors are bounds valid for any induction algorithm and probability distribution in the labeled space. The bounds are expressed in terms of the size of the training set and the VC-dimension of the inducer.

THEOREM 8.2 *The bound on the generalization error of hypothesis space H with finite VC-Dimension d is given by:*

$$|\varepsilon(h,D) - \hat{\varepsilon}(h,S)| \leq \sqrt{\frac{d \cdot (\ln \frac{2m}{d} + 1) - \ln \frac{\delta}{4}}{m}} \quad \begin{array}{l} \forall h \in H \\ \forall \delta > 0 \end{array} \tag{8.3}$$

with probability of $1-\delta$ where $\hat{\varepsilon}(h,S)$ represents the training error of classifier h measured on training set S of cardinality m and $\varepsilon(h,D)$ represents the generalization error of the classifier h over the distribution D.

The VC-dimension is a property of a set of all classifiers, denoted by H, that have been examined by the inducer. For the sake of simplicity, we consider classifiers that correspond to the two-class pattern recognition case. In this case, the VC-dimension is defined as the maximum number of data points that can be shattered by the set of admissible classifiers. By definition, a set S of m points is shattered by H if and only if for every dichotomy of S there is some classifier in H that is consistent with this dichotomy. In other words, the set S is shattered by H if there are classifiers that split the points into two classes in all of the 2^m possible ways. Note that, if the VC-dimension of H is d, then there exists at least one set of d points that can be shattered by H. In general, however, it will not be true that every set of d points can be shattered by H.

A sufficient condition for consistency of an induction procedure is that the VC-dimension of the inducer is finite. The VC-dimension of a linear classifier is simply the dimension n of the input space, or the number of free parameters of the classifier. The VC-dimension of a general classifier may however be quite different from the number of free parameters and in many cases it might be very difficult to compute it accurately. In this case it is useful to calculate a lower and upper bound for the VC-Dimension. Schmitt (2002) have presented these VC bounds for neural networks.

5.2.2 PAC-Framework. The Probably Approximately Correct (PAC) learning model was introduced by Valiant (1984). This framework can be used to characterize the concept class "that can be reliably learned from a reasonable number of randomly drawn training examples and a reasonable amount of computation" (Mitchell, 1997). We use the following formal definition of PAC-learnable adapted from (Mitchell, 1997):

DEFINITION 8.3 *Let C be a concept class defined over the input instance space X with n attributes. Let I be an inducer that considers hypothesis space H. C is said to be PAC-learnable by I using H if for all $c \in C$, distributions D over X, ε such that $0 < \varepsilon < 1/2$ and δ such that $0 < \delta < 1/2$, learner I with a probability of at least $(1 - \delta)$ will output a hypothesis $h \in H$ such that $\varepsilon(h, D) \leq \varepsilon$, in time that is polynomial in $1/\varepsilon$, $1/\delta$, n, and $size(c)$, where $size(c)$ represents the encoding length of c in C, assuming some representation for C.*

The PAC learning model provides a general bound on the number of training examples sufficient for any consistent learner I examining a finite hypothesis space H with probability at least (1 - δ) to output a hypothesis $h \in H$ within error ε of the target concept $c \in C \subseteq H$. More specifically, the size of the training set should be: $m \geq \frac{1}{\varepsilon}(\ln(1/\delta) + \ln|H|)$

5.3 Empirical Estimation of Generalization Error

Another approach for estimating the generalization error is to split the available examples into two groups: training and test sets. First, the training set is used by the inducer to construct a suitable classifier and then we measure the misclassification rate of this classifier on the test set. This test set error usually provides a better estimation of the generalization error than the training error. The reason for this is that the training error usually under-estimates the generalization error (due to the overfitting phenomena).

When data is limited, it is common practice to *re-sample* the data, that is, partition the data into training and test sets in different ways. An inducer is trained and tested for each partition and the accuracies averaged. By doing

this, a more reliable estimate of the true generalization error of the inducer is provided.

Random sub-sampling and *n-fold cross-validation* are two common methods of re-sampling. In random subsampling, the data is randomly partitioned into disjoint training and test sets several times. Errors obtained from each partition are averaged. In n-fold cross-validation, the data is randomly split into n mutually exclusive subsets of approximately equal size. An inducer is trained and tested several times. Each time it is tested on one of the k folds and trained using the remaining $n-1$ folds.

The cross-validation estimate of the generalization error is the overall number of misclassifications, divided by the number of examples in the data. The random sub-sampling method has the advantage that it can be repeated an indefinite number of times. However, it has the disadvantage that the test sets are not independently drawn with respect to the underlying distribution of examples. Because of this, using a t-test for paired differences with random subsampling can lead to an increased chance of Type I error that is, identifying a significant difference when one does not actually exist. Using a t-test on the generalization error produced on each fold has a lower chance of Type I error but may not give a stable estimate of the generalization error. It is common practice to repeat n fold cross-validation n times in order to provide a stable estimate. However, this, of course, renders the test sets non-independent and increases the chance of Type I error. Unfortunately, there is no satisfactory solution to this problem. Alternative tests suggested by Dietterich (1998) have a low chance of Type I error but a high chance of Type II error - that is, failing to identify a significant difference when one does actually exist.

Stratification is a process often applied during random sub-sampling and n-fold cross-validation. Stratification ensures that the class distribution from the whole dataset is preserved in the training and test sets. Stratification has been shown to help reduce the variance of the estimated error especially for datasets with many classes. Stratified random subsampling with a paired t-test is used herein to evaluate accuracy.

5.4 Computational Complexity

Another useful criterion for comparing inducers and classifiers is their computational complexities. Strictly speaking, computational complexity is the amount of CPU consumed by each inducer. It is convenient to differentiate between three metrics of computational complexity:

- Computational complexity for generating a new classifier: This is the most important metric, especially when there is a need to scale the Data Mining algorithm to massive data sets. Because most of the algorithms have computational complexity, which is worse than linear in the num-

bers of tuples, mining massive data sets might be "prohibitively expensive".

- Computational complexity for updating a classifier: Given new data, what is the computational complexity required for updating the current classifier such that the new classifier reflects the new data?

- Computational Complexity for classifying a new instance: Generally this type is neglected because it is relatively small. However, in certain methods (like k-nearest neighborhood) or in certain real time applications (like anti-missiles applications), this type can be critical.

5.5 Comprehensibility

Comprehensibility criterion (also known as interpretability) refers to how well humans grasp the classifier induced. While the generalization error measures how the classifier fits the data, comprehensibility measures the "mental fit" of that classifier.

Many techniques, like neural networks or support vectors machines, are designed solely to achieve accuracy. However, as their classifiers are represented using large assemblages of real valued parameters, they are also difficult to understand and are referred to as black-box models.

It is often important for the researcher to be able to inspect an induced classifier. For domains such as medical diagnosis, the users must understand how the system makes its decisions in order to be confident of the outcome. Data mining can also play an important role in the process of scientific discovery. A system may discover salient features in the input data whose importance was not previously recognized. If the representations formed by the inducer are comprehensible, then these discoveries can be made accessible to human review (Hunter and Klein, 1993).

Comprehensibility can vary between different classifiers created by the same inducer. For instance, in the case of decision trees, the size (number of nodes) of the induced trees is also important. Smaller trees are preferred because they are easier to interpret. However, this is only a rule of thumb. In some pathologic cases, a large and unbalanced tree can still be easily interpreted (Buja and Lee, 2001).

As the reader can see, the accuracy and complexity factors can be quantitatively estimated, while comprehensibility is more subjective.

Another distinction is that the complexity and comprehensibility depend mainly on the induction method and much less on the specific domain considered. On the other hand, the dependence of error metrics on a specific domain cannot be neglected.

6. Scalability to Large Datasets

Obviously induction is one of the central problems in many disciplines such as machine learning, pattern recognition, and statistics. However the feature that distinguishes Data Mining from traditional methods is its scalability to very large sets of varied types of input data. The notion, "scalability" usually refers to datasets that fulfill at least one of the following properties: high number of records or high dimensionality.

"Classical" induction algorithms have been applied with practical success in many relatively simple and small-scale problems. However, trying to discover knowledge in real life and large databases, introduces time and memory problems.

As large databases have become the norm in many fields (including astronomy, molecular biology, finance, marketing, health care, and many others), the use of Data Mining to discover patterns in them has become a potentially very productive enterprise. Many companies are staking a large part of their future on these "Data Mining" applications, and looking to the research community for solutions to the fundamental problems they encounter.

While a very large amount of available data used to be the dream of any data analyst, nowadays the synonym for "very large" has become "terabyte", a hardly imaginable volume of information. Information-intensive organizations (like telecom companies and banks) are supposed to accumulate several terabytes of raw data every one to two years.

However, the availability of an electronic data repository (in its enhanced form known as a "data warehouse") has created a number of previously unknown problems, which, if ignored, may turn the task of efficient Data Mining into mission impossible. Managing and analyzing huge data warehouses requires special and very expensive hardware and software, which often causes a company to exploit only a small part of the stored data.

According to Fayyad *et al*. (1996) the explicit challenges for the data mining research community are to develop methods that facilitate the use of Data Mining algorithms for real-world databases. One of the characteristics of a real world databases is high volume data.

Huge databases pose several challenges:

- Computing complexity. Since most induction algorithms have a computational complexity that is greater than linear in the number of attributes or tuples, the execution time needed to process such databases might become an important issue.

- Poor classification accuracy due to difficulties in finding the correct classifier. Large databases increase the size of the search space, and the

chance that the inducer will select an overfitted classifier that generally invalid.

- Storage problems: In most machine learning algorithms, the entire training set should be read from the secondary storage (such as magnetic storage) into the computer's primary storage (main memory) before the induction process begins. This causes problems since the main memory's capability is much smaller than the capability of magnetic disks.

The difficulties in implementing classification algorithms as is, on high volume databases, derives from the increase in the number of records/instances in the database and of attributes/features in each instance (high dimensionality). Approaches for dealing with a high number of records include:

- Sampling methods - statisticians are selecting records from a population by different sampling techniques.
- Aggregation - reduces the number of records either by treating a group of records as one, or by ignoring subsets of "unimportant" records.
- Massively parallel processing - exploiting parallel technology - to simultaneously solve various aspects of the problem.
- Efficient storage methods that enable the algorithm to handle many records. For instance, Shafer *et al*. (1996) presented the SPRINT which constructs an attribute list data structure.
- Reducing the algorithm's search space - For instance the PUBLIC algorithm (Rastogi and Shim, 2000) integrates the growing and pruning of decision trees by using MDL cost in order to reduce the computational complexity.

7. The "Curse of Dimensionality"

High dimensionality of the input (that is, the number of attributes) increases the size of the search space in an exponential manner, and thus increases the chance that the inducer will find spurious classifiers that are generally invalid. It is well-known that the required number of labeled samples for supervised classification increases as a function of dimensionality (Jimenez and Landgrebe, 1998). Fukunaga (1990) showed that the required number of training samples is linearly related to the dimensionality for a linear classifier and to the square of the dimensionality for a quadratic classifier. In terms of non-parametric classifiers like decision trees, the situation is even more severe. It has been estimated that as the number of dimensions increases, the sample size needs to increase exponentially in order to have an effective estimate of multivariate densities (Hwang *et al*., 1994).

This phenomenon is usually called the "curse of dimensionality". Bellman (1961) was the first to coin this term, while working on complicated signal processing. Techniques, like decision trees inducers, that are efficient in low dimensions, fail to provide meaningful results when the number of dimensions increases beyond a "modest" size. Furthermore, smaller classifiers, involving fewer features (probably less than 10), are much more understandable by humans. Smaller classifiers are also more appropriate for user-driven Data Mining techniques such as visualization.

Most of the methods for dealing with high dimensionality focus on feature selection techniques, i.e. selecting a single subset of features upon which the inducer (induction algorithm) will run, while ignoring the rest. The selection of the subset can be done manually by using prior knowledge to identify irrelevant variables or by using proper algorithms.

In the last decade, feature selection has enjoyed increased interest by many researchers. Consequently many feature selection algorithms have been proposed, some of which have reported a remarkable improvement in accuracy. Please refer to Chapter 5 in this volume for further reading.

Despite its popularity, the usage of feature selection methodologies for overcoming the obstacles of high dimensionality has several drawbacks:

- The assumption that a large set of input features can be reduced to a small subset of relevant features is not always true. In some cases the target feature is actually affected by most of the input features, and removing features will cause a significant loss of important information.
- The outcome (i.e. the subset) of many algorithms for feature selection (for example almost any of the algorithms that are based upon the wrapper methodology) is strongly dependent on the training set size. That is, if the training set is small, then the size of the reduced subset will be also small. Consequently, relevant features might be lost. Accordingly, the induced classifiers might achieve lower accuracy compared to classifiers that have access to all relevant features.
- In some cases, even after eliminating a set of irrelevant features, the researcher is left with relatively large numbers of relevant features.
- The backward elimination strategy, used by some methods, is extremely inefficient for working with large-scale databases, where the number of original features is more than 100.

A number of linear dimension reducers have been developed over the years. The linear methods of dimensionality reduction include projection pursuit (Friedman and Tukey, 1973), factor analysis (Kim and Mueller, 1978), and principal components analysis (Dunteman, 1989). These methods are not

aimed directly at eliminating irrelevant and redundant features, but are rather concerned with transforming the observed variables into a small number of "projections" or "dimensions". The underlying assumptions are that the variables are numeric and the dimensions can be expressed as linear combinations of the observed variables (and vice versa). Each discovered dimension is assumed to represent an unobserved factor and thus to provide a new way of understanding the data (similar to the curve equation in the regression models).

The linear dimension reducers have been enhanced by constructive induction systems that use a set of existing features and a set of pre-defined constructive operators to derive new features (Pfahringer, 1994; Ragavan and Rendell, 1993). These methods are effective for high dimensionality applications only if the original domain size of the input feature can be in fact decreased dramatically.

One way to deal with the above-mentioned disadvantages is to use a very large training set (which should increase in an exponential manner as the number of input features increases). However, the researcher rarely enjoys this privilege, and even if it does happen, the researcher will probably encounter the aforementioned difficulties derived from a high number of instances.

Practically most of the training sets are still considered "small" not due to their absolute size but rather due to the fact that they contain too few instances given the nature of the investigated problem, namely the instance space size, the space distribution and the intrinsic noise.

8. Classification Problem Extensions

In this section we survey a few extensions to the classical classification problem.

In classic supervised learning problems, classes are mutually exclusive by definition. In "multiple label" classification problems each training instance is given a set of candidate class labels but only one of the candidate labels is the correct one (Jin and Ghahramani, 2002). The reader should not be confused with multi-class classification problems which usually refer to simply having more than two possible disjoint classes for the classier to learn.

In practice, many real problems are formalized as a "Multiple Labels" problem. For example, this occurs when there is a disagreement regarding the label of a certain training instance. Another typical example of "multiple labels" occurs when there is a hierarchical structure over the class labels and some of the training instances are given the labels of the superclasses instead of the labels of the subclasses. For instance a certain training instance representing a course can be labeled as "engineering", while this class consists of more specific classes such as "electrical engineering", "industrial engineering", etc.

A closely-related problem is the "multi-label" classification problem. In this case, the classes are not mutually exclusive. One instance is actually associated with many labels, and all labels are correct. Such problems exist, for example, in text classifications. Texts may simultaneously belong to more than one genre (Schapire and Singer, 2000). In bioinformatics, genes may have multiple functions, yielding multiple labels (Clare and King, 2001). Boutella *et al*. (2004) presented a framework to handle multi-label classification problems. They present approaches for training and testing in this scenario and introduce new metrics for evaluating the results.

The difference between "multi-label" and "multiple Label" should be clarified. In "multi-label" each training instance can have multiple class labels, and all the assigned class labels are actually correct labels while in "Multiple Labels" problem only one of the assigned multiple labels is the target label.

Another closely-related problem is the fuzzy classification problem (Janikow, 1998), in which class boundaries are not clearly defined. Instead, each instance has a ceratin membership function for each class which represents the degree to which the instance belongs to this class.

Another related problem is "preference learning" (Furnkranz, 1997). The training set consists of a collection of training instances which are associated with a set of pairwise preferences between labels, expressing that one label is preferred over another. The goal of "preference learning" is to predict a ranking, of all possible labels for a new training example. Cohen *et al*. (1999) have investigated a more narrow version of the problem, the learning of one single preference function. The "constraint classification" problem (Har-Peled *et al*., 2002) is a superset of the "preference learning" and "multi-label classification", in which each example is labeled according to some partial order.

In "multiple-instance" problems (Dietterich *et al*., 1997), the instances are organized into bags of several instances, and a class label is tagged for every bag of instances. In the "multiple-instance" problem, at least one of the instances within each bag corresponds to the label of the bag and all other instances within the bag are just noises. Note that in "multiple-instance" problem the ambiguity comes from the instances within the bag.

References

Boutella R. M., Luob J., Shena X., Browna C. M., Learning multi-label scene classification, Pattern Recognition, 37(9), pp. 1757-1771, 2004.

Buja, A. and Lee, Y.S., Data Mining criteria for tree based regression and classification, Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining, (pp 27-36), San Diego, USA, 2001.

Clare, A., King R.D., Knowledge Discovery in Multi-label Phenotype Data, Lecture Notes in Computer Science, Vol. 2168, Springer, Berlin, 2001.

Cohen, W. W., Schapire R.E., and Singer Y., Learning to order things. Journal of Artificial Intelligence Research, 10:243270, 1999.

Dietterich, T. G., Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation, 10(7): 1895-1924, 1998.

Dietterich, T. G., Lathrop, R. H. , and Perez, T. L., Solving the multiple-instance problem with axis-parallel rectangles, Artificial Intelligence, 89(1-2), pp. 31-71, 1997.

Duda, R., and Hart, P., Pattern Classification and Scene Analysis, New-York, Wiley, 1973.

Dunteman, G.H., Principal Components Analysis, Sage Publications, 1989.

Fayyad, U., Piatesky-Shapiro, G. & Smyth P., From Data Mining to Knowledge Discovery: An Overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds), Advances in Knowledge Discovery and Data Mining, pp 1-30, AAAI/MIT Press, 1996.

Friedman, J.H. & Tukey, J.W., A Projection Pursuit Algorithm for Exploratory Data Analysis, IEEE Transactions on Computers, 23: 9, 881-889, 1973.

Fukunaga, K., Introduction to Statistical Pattern Recognition. San Diego, CA: Academic, 1990.

Fürnkranz J. and Hüllermeier J., Pairwise preference learning and ranking. In Proc. ECML03, pages 145156, Cavtat, Croatia, 2003.

Grumbach S., Milo T., Towards Tractable Algebras for Bags. Journal of Computer and System Sciences 52(3): 570-588, 1996.

Har-Peled S., Roth D., and Zimak D., Constraint classification: A new approach to multiclass classification. In Proc. ALT02, pages 365379, Lubeck, Germany, 2002, Springer.

Hunter L., Klein T. E., Finding Relevant Biomolecular Features. ISMB 1993, pp. 190-197, 1993.

Hwang J., Lay S., and Lippman A., Nonparametric multivariate density estimation: A comparative study, IEEE Transaction on Signal Processing, 42(10): 2795-2810, 1994.

Janikow, C.Z., Fuzzy Decision Trees: Issues and Methods, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 28, Issue 1, pp. 1-14. 1998.

Jimenez, L. O., & Landgrebe D. A., Supervised Classification in High- Dimensional Space: Geometrical, Statistical, and Asymptotical Properties of Multivariate Data. IEEE Transaction on Systems Man, and Cybernetics - Part C: Applications and Reviews, 28:39-54, 1998.

Jin, R. , & Ghahramani Z., Learning with Multiple Labels, The Sixteenth Annual Conference on Neural Information Processing Systems (NIPS 2002) Vancouver, Canada, pp. 897-904, December 9-14, 2002.

Kim J.O. & Mueller C.W., Factor Analysis: Statistical Methods and Practical Issues. Sage Publications, 1978.

Mitchell, T., Machine Learning, McGraw-Hill, 1997.

Pfahringer, B., Controlling constructive induction in CiPF, In Bergadano, F. and De Raedt, L. (Eds.), Proceedings of the seventh European Conference on Machine Learning, pp. 242-256, Springer-Verlag, 1994.

Quinlan, J. R., C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos, 1993.

Ragavan, H. and Rendell, L., Look ahead feature construction for learning hard concepts. In Proceedings of the Tenth International Machine Learning Conference: pp. 252-259, Morgan Kaufman, 1993.

Rastogi, R., and Shim, K., PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning,Data Mining and Knowledge Discovery, 4(4):315-344, 2000.

Schapire R., Singer Y., Boostexter: a boosting-based system for text categorization, Machine Learning 39 (2/3):135168, 2000.

Schmitt , M., On the complexity of computing and learning with multiplicative neural networks, Neural Computation 14: 2, 241-301, 2002.

Shafer, J. C., Agrawal, R. and Mehta, M. , SPRINT: A Scalable Parallel Classifier for Data Mining, Proc. 22nd Int. Conf. Very Large Databases, T. M. Vijayaraman and Alejandro P. Buchmann and C. Mohan and Nandlal L. Sarda (eds), 544-555, Morgan Kaufmann, 1996.

Valiant, L. G. (1984). A theory of the learnable. Communications of the ACM 1984, pp. 1134-1142.

Vapnik, V.N., The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.

Wolpert, D. H., The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. In D. H. Wolpert, editor, The Mathematics of Generalization, The SFI Studies in the Sciences of Complexity, pages 117-214. AddisonWesley, 1995.