Pattern Recognition

LUDMILA I. KUNCHEVA AND CHRISTOPHER J. WHITAKER

Volume 3, pp. 1532–1535

in

Encyclopedia of Statistics in Behavioral Science

ISBN-13: 978-0-470-86080-9
ISBN-10: 0-470-86080-4

Editors

Brian S. Everitt & David C. Howell

Pattern Recognition

Pattern recognition deals with classification problems that we would like to delegate to a machine, for example, scanning for abnormalities in smear test samples, identifying a person by voice and a face image for security purposes, detecting fraudulent credit card transactions, and so on. Each object (test sample, person, transaction) is described by a set of p features and can be thought of as a point in some p-dimensional feature space.

A classifier is a formula, algorithm or technique that outputs a class label for any collection of values of the p features submitted to its input. For designing a classifier, also called **discriminant analysis**, we use a *labeled data set*, Z, of n objects, where each object is described by its feature values and true class label.

The fundamental idea used in statistical pattern recognition is Bayes decision theory [2]. The c classes are treated as random entities that occur with prior probabilities $P(\omega_i), i = 1, \ldots, c$. The posterior probability of being in class ω_i for an observed data point $\mathbf{x}$ is calculated using Bayes rule

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j)}, \qquad (1)$$

where $p(\mathbf{x}|\omega_i)$ is the class-conditional probability density function (pdf) of $\mathbf{x}$, given class ω_i (*see* **Bayesian Statistics**; **Bayesian Belief Networks**). According to the Bayes rule, the class with the largest posterior probability is selected as the label of $\mathbf{x}$. Ties are broken randomly. The Bayes rule guarantees the minimum misclassification rate. Sometimes the misclassifications cost differently for different classes. Then we can use a *loss matrix* $\Lambda = [\lambda_{ij}]$, where λ_{ij} is a measure of the loss incurred if we assign class label ω_i when the true label is ω_j. The *minimum risk classifier* assigns $\mathbf{x}$ to the class with the minimum expected risk

$$R_{\mathbf{x}}(\omega_i) = \sum_{j=1}^{c} \lambda_{ij} P(\omega_j|\mathbf{x}). \qquad (2)$$

In general, the classifier output can be interpreted as a set of c degrees of support, one for each class (discriminant scores obtained through discriminant functions). We label $\mathbf{x}$ in the class with the largest support.

In practice, the prior probabilities and the class-conditional pdfs are not known. The pdfs can be estimated from the data using either a parametric or nonparametric approach. Parametric classifiers assume the form of the probability distributions and then estimate the parameters from $\mathbf{Z}$. The linear and quadratic discriminant classifiers, which assume multivariate normal distributions as $p(\mathbf{x}|\omega_i)$, are commonly used (*see* **Discriminant Analysis**). Nonparametric classifier models include the *k-nearest neighbor classifier* (k-nn) and *kernel classifiers* (e.g., Parzen, support vector machines (SVM)). The k-nn classifier assigns $\mathbf{x}$ to the class most represented among the closest k neighbors of $\mathbf{x}$.

Instead of trying to estimate the pdfs and applying Bayes' rule, some classifier models directly look for the best discrimination boundary between the classes, for example, **classification and regression trees**, and **neural networks**.

Figure 1 shows a two-dimensional data set with two banana-shaped classes. The dots represent the observed data points, and members of the classes are denoted by their color. Four classifiers are built, each one splitting the feature space into two classification regions. The boundary between the two regions is denoted by the white line. The linear discriminant classifier results in a linear boundary, while the quadratic discriminant classifier results in a quadratic boundary. While both these models are simple, for this difficult data set, neither is adequate. Here, the greater flexibility afforded by the 1-nn and neural network models result in more accurate but complicated boundaries. Also, the class regions found by 1-nn and the neural network may not be connected sets of points, which is not possible for the linear and quadratic discriminant classifiers.

Classifier Training and Testing

In most real life problems we do not have a ready made classification algorithm. Take for example, a classifier that recognizes expressions of various feelings from face images. We can only provide a rough guidance in a linguistic format and pick out features that we believe are relevant for the task. The classifier has to be trained by using a set of labeled examples. The training depends on the classifier model. The nearest neighbor classifier (1-nn) does not require any training; we can classify a new data point

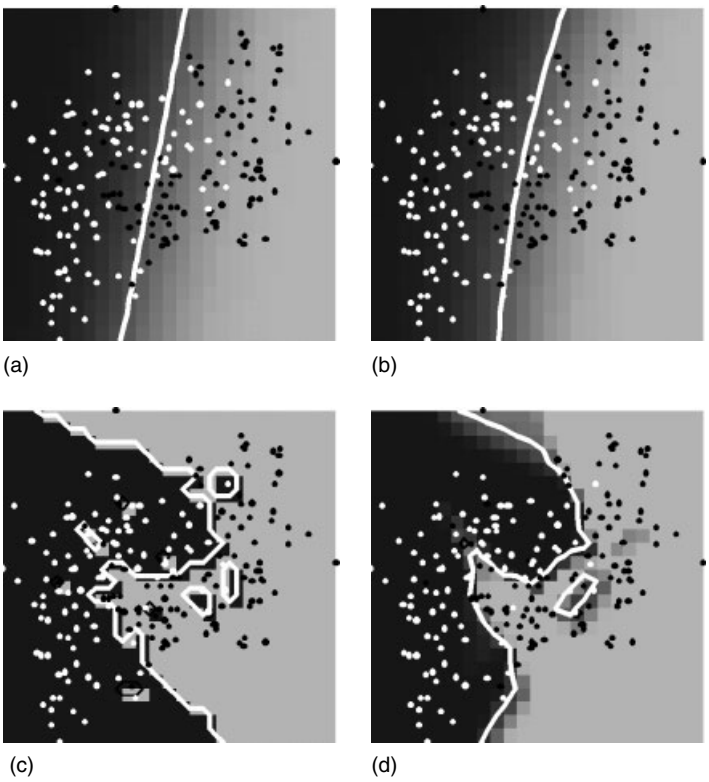

Figure 1 Classification regions for two banana-shaped classes found by four classifiers (a) Linear, (b) Quadratic, (c) 1-nn, (d) Neural Network

right away by finding its nearest neighbor in **Z**. On the other hand, the lack of a suitable training algorithm for neural networks was the cause for their dormant period between 1940s and 1980s (the error back-propagation algorithm revitalized their development). When trained, some classifiers can provide us with an interpretable decision strategy (e.g., tree models and k-nn) whereas other classifiers behave as black boxes (e.g., neural networks). Even when we can verify the logic of the decision making, the ultimate judge of the classifier performance is the classification error. Estimating the **misclassification rate** of our classifier is done through the training protocol. Part of the data set, **Z**, is used for training and the remaining part is left for testing. The most popular training/testing protocol is **cross-validation**. **Z** is divided into K approximately equal parts, one is left for testing, and the remaining $K-1$ are pooled as the training set. This process is repeated K times (K-fold cross-validation) leaving aside a different part each time. The error of the classifier is the averaged testing error across the K testing parts.

Variable Selection

Not all features are important for the classification task. Classifiers may perform better with fewer features. This is a paradox from an information-theoretic point of view. Its explanation lies in the fact that the classifiers that we use and the parameter estimates that we calculate are imperfect; therefore, some of the supposed information is actually noise to our model. *Feature selection* reduces the original feature set to a subset without adversely affecting the classification performance. *Feature extraction*, on the other hand, is a dimensionality reduction approach whereby all initial features are used and a small amount of new features are calculated from them (e.g., **principal component analysis**, **projection pursuit**, **multidimensional scaling**).

There are two major questions in feature selection: what criterion should we use to evaluate the subsets? and how do we search among all possible subset-candidates? Since the final goal is to have an accurate classifier, the most natural choice of a criterion is the minimum error of the classifier built on the subset-candidate. Methods using a direct estimate of the error are called *wrapper methods*. Even with modern computational technology, training a classifier and estimating its error for each examined subset of features might be prohibitive. An alternative class of feature selection methods where the criterion is indirectly related to the error are called *filter methods*. Here the criterion used is a measure of discrimination between the classes, for example, the **Mahalanobis distance** between the class centroids.

For large p, checking all possible subsets is often not feasible. There are various search algorithms, the simplest of which are the *sequential forward selection* (SFS) and the *sequential backward selection* (SBS). In SFS, we start with the single best feature (according to the chosen criterion) and add one feature at a time. The second feature to enter the selected subset will be the feature that makes the best pair with the feature already selected. The third feature is chosen so that it makes the best triple containing the already selected two features, and so on. In SBS, we start with the whole set of features and remove the single feature which gives the best remaining subset of $p - 1$ features. Next we remove the feature that results in the best remaining subset of $p - 2$ features, and so on. SFS and SBS, albeit simple, have been found to be surprisingly robust and accurate. A modification of these is the *floating search* feature selection algorithm, which leads to better results at the expense of an expanded search space. Feature selection is an art rather than science as it relies on heuristics, intuition, and domain knowledge. Among many others, *genetic algorithms* have been applied for feature selection with various degree of success.

Cluster Analysis

In some problems, the class labels are not defined in advance. Then, the problem is to find a class structure in the data set, if there is any. The number of clusters is usually not specified in advance, which makes the problem even more difficult. If we guess wrongly, we may impose a structure onto a data set that does not have one or may fail to discover an existing structure. Cluster analysis procedures can be roughly grouped into *hierarchical* and *iterative optimization methods* (*see* **Hierarchical Clustering**; ***k*-means Analysis**).

Classifier Ensembles

Instead of using a single classifier, we may combine the outputs of several classifiers in an attempt to reach a more accurate or reliable solution. At this stage, there are a large number of methods, directions, and paradigms in designing classifier ensembles but there is no agreed taxonomy for this relatively young area of research.

Fusion and Selection. In classifier fusion, we assume that all classifiers are 'experts' across the whole feature space, and therefore their votes are equally important for any $\mathbf{x}$. In classifier selection, first 'an oracle' or meta-classifier decides whose *region of competence* $\mathbf{x}$ is in, and then the class label of the nominated classifier is taken as the ensemble decision.

Decision Optimization and Coverage Optimization. Decision optimization refers to ensemble construction methods that are primarily concerned with the combination rule assuming that the classifiers in the ensembles are given. Coverage optimization looks at building the individual classifiers assuming a fixed classification rule.

Five simple combination rules (combiners) are illustrated below. Suppose that there are 3 classes and 7 classifiers whose outputs for a particular $\mathbf{x}$ (discriminant scores) are organized in a 7 by 3 matrix (called a *decision profile* for $\mathbf{x}$) as given in Table 1. The overall support for each class is calculated by applying a simple operation to the discriminant scores for that class only. The majority vote operates on the label outputs of the seven classifiers.

Each possible class label occurs as the final ensemble label in the five shaded cells. This shows the flexibility that we have in choosing the combination rule for the particular problem.

We can consider the classifier outputs as new features, disregard their context as discriminant scores, and use these features to build a classifier. We can thus build hierarchies of classifiers (*stacked generalization*). There are many combination methods

Table 1 An ensemble of seven classifiers, $D_1, \ldots, D_7$: the decision profile for $\mathbf{x}$ and five combination rules

	Support for ω_1	Support for ω_2	Support for ω_3	Label output
D_1	0.24	0.44	0.56	ω_3
D_2	0.17	0.13	0.59	ω_3
D_3	0.22	0.32	0.86	ω_3
D_4	0.17	0.40	0.49	ω_3
D_5	0.27	0.77	0.45	ω_2
D_6	0.51	0.90	0.06	ω_2
D_7	0.29	0.46	0.03	ω_2
Minimum	**0.17**	0.13	0.03	ω_1
Maximum	0.51	**0.90**	0.86	ω_2
Average	0.27	**0.49**	0.43	ω_2
Product	0.0001	**0.0023**	0.0001	ω_2
Majority	–	–	–	ω_3

proposed in the literature that involve various degrees of training.

Within the coverage optimization group are **bagging**, **random forests** and **boosting**. Bagging takes L random samples (with replacement) from $\mathbf{Z}$ and builds one classifier on each sample. The ensemble decision is made by the majority vote. The success of bagging has been explained by its ability to reduce the variance of the classification error of a single classifier model. Random forests are defined as a variant of bagging such that the production of the individual classifiers depends on a random parameter and independent sampling. A popular version of random forests is an ensemble where each classifier is built upon a random subsets of features, sampled with replacement from the initial feature set.

A boosting algorithm named *AdaBoost* has been found to be even more successful than bagging. Instead of drawing random bootstrap samples, AdaBoost designs the ensemble members one at a time, based on the performance of the previous member. A set of weights is maintained across the data points in $\mathbf{Z}$. In the resampling version of AdaBoost, the weights are taken as probabilities. Each training sample is drawn using the weights. A classifier is built on the sampled training set, and the weights are modified according to its performance. Points that have been correctly recognized get smaller weights and points that have been misclassified get larger weights. Thus difficult to classify objects will have more chance to be picked in the subsequent training sets. The procedure stops at a predefined number of classifiers (e.g., 50). The votes are combined by a weighted majority vote, where the classifier weight depends on its error rate. AdaBoost has been proved to reduce the training error to zero. In addition, when the training error does reach zero, which for most classifier models means that they have been overtrained and the testing error might be arbitrarily high, AdaBoost 'miraculously' keeps reducing the testing error further. This phenomenon has been explained by the *margin theory* but with no claim about a global convergence of the algorithm. AdaBoost has been found to be more sensitive to noise in the data than bagging. Nevertheless, AdaBoost has been declared by Leo Breiman to be the 'most accurate available off-the-shelf classifier' [1].

Pattern recognition is related to artificial intelligence and machine learning. There is renewed interest in this topic as it underpins applications in modern domains such as **data mining**, document classification, financial forecasting, organization and retrieval of multimedia databases, microarray data analysis (*see* **Microarrays**), and many more [3].

References

[1] Breiman, L. (1998). Arcing classifiers, *The Annals of Statistics* **26**(3), 801–849.

[2] Duda, R.O., Hart, P.E. & Stork, D.G. (2001). *Pattern Classification*, 2nd Edition, John Wiley & Sons, New York.

[3] Jain, A.K., Duin, R.P.W. & Mao, I. (2000). Statistical pattern recognition: a review, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1), 4–37.

LUDMILA I. KUNCHEVA AND CHRISTOPHER J. WHITAKER