Pattern Recognition and Classification

Chapter · March 2015

DOI: 10.1002/9781118445112.stat06503.pub2

2 authors:

Ludmila Kuncheva
Bangor University
202 PUBLICATIONS **17,538** CITATIONS
SEE PROFILE

Chris Whitaker
Whitaker Research Ltd.
126 PUBLICATIONS **7,473** CITATIONS
SEE PROFILE

Pattern recognition and classification

Ludmila I. Kuncheva[†] *and Christopher J. Whitaker* [‡]

[†]*Bangor University, UK*, [‡]*Whitaker Research Ltd., UK* [1]

Keywords: *classification, discriminant analysis, cluster analysis, feature selection, classifier ensembles*

Abstract

Abstract: Pattern recognition is about assigning objects (also called observations, instances or examples) to classes. The objects are described by features and represented as points in the feature space. A classifier is an algorithm that assigns a class label to any given point in the feature space. Pattern recognition comprises supervised learning (predefined class labels) and unsupervised learning (unknown class labels). Supervised learning includes choosing a classifier model, training and testing the classifier and selecting relevant features. Classifier ensembles combine the outputs of a set of classifiers for improved accuracy. Unsupervised learning is usually approached by cluster analysis.

Introduction

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 their iris pattern, detecting fraudulent credit card transactions.

In some problems, the class labels are not defined in advance (*unsupervised learning).* 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. Cluster analysis procedures can be roughly grouped into *hierarchical* (*see* **single linkage**) and *iterative optimization methods* (*see* **K-Means Analysis**).

When the class labels are available (*supervised learning)* the task is to build a classifier that will predict the labels. Supervised pattern recognition can be roughly categorized as statistical, syntactic and structural. This article details further statistical pattern recognition [1, 4, 7, 14].

Statistical pattern recognition

Each object (test sample, person, transaction) is described by a set of d features (variables or measurements) and can be thought of as a point in some d-dimensional feature space.

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

A fundamental principle underlying much of statistical pattern recognition is **Bayes decision theory** [4, 14]. The c classes are treated as random entities that occur with prior probabilities $P(\omega_i)$, $i = 1$,

[1]This article was originally published online in 2005 in Encyclopedia of Statistics in Behavioral Science, © John Wiley & Sons, Ltd and republished in Wiley StatsRef: Statistics Reference Online, 2014.

..., c. The posterior probability for the event that the observed data point $\mathbf{x}$ comes from class ω_i is calculated using the Bayes rule

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

where $p(\mathbf{x}\,|\omega_i)$ is the class-conditional probability density function (pdf) of $\mathbf{x}$, given class ω_i. 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. Different misclassification costs are handled through a *loss matrix* $\Lambda = [\lambda_{ij}]$, where λ_{ij} is a measure of the loss incurred if we assign class ω_i when the true class is ω_j (*See* **Loss function**). The *minimum risk classifier* assigns $\mathbf{x}$ to the class with the minimum expected risk

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

Posterior probabilities or expected risk may not be easily available. In the *canonical model* of a classifier [4], a separate function, called *discriminant function*, is evaluated for each class. The discriminant functions can be interpreted as a set of c degrees of support, one for each class. We label $\mathbf{x}$ in the class with the largest support.

Classifiers

The prior probabilities and the class-conditional *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 the training data. Typically, the parametric group of classifier models assume multivariate normal distributions as $p(\mathbf{x}|\omega_\mathbf{i})$. This group includes the linear and the quadratic discriminant classifiers named after the type of boundary they build in the d dimensional space. The Naïve Bayes classifier assumes that the features are independent, and the pdf is a product of d one-dimensional pdfs.

A representative of the nonparametric group is the **k-nearest neighbor classifier** (k-nn) which assigns $\mathbf{x}$ to the class most represented among its closest k neighbors. It is derived from equation (1) by local approximation of the class-conditional pdfs.

Instead of trying to estimate the *pdfs*, some classifier models directly look for the best discrimination boundary between the classes, for example, **support vector machines** (SVM), **neural network** classifiers (NN) and **decision tree** classifiers (DT) [4]. DTs and NNs are important for classifier ensembles (see later) because of their instability property. The classification regions of the DT classifier are created using a single feature at each decision node. This leads to a mosaic of hyperboxes in the feature space. The regions may change significantly with small changes of the data. NNs' classification regions vary not only with small changes in the data but also depending on the initialization of the training algorithm.

Figure 1 (a) shows a two-dimensional data set with two classes. The dots represent the observed data points, and members of the classes are denoted by their color. The Bayes-optimal classification regions for this set are plotted in Figure 1 (b). The dark-shaded region is where

$$P(\text{class light}|\mathbf{x}) > P(\text{class dark}|\mathbf{x}).$$

Labelling the points in this region as belonging to class 'light' guarantees the minimum overall error rate.

Figure 2 depicts the classification regions obtained through training 7 classifier models.

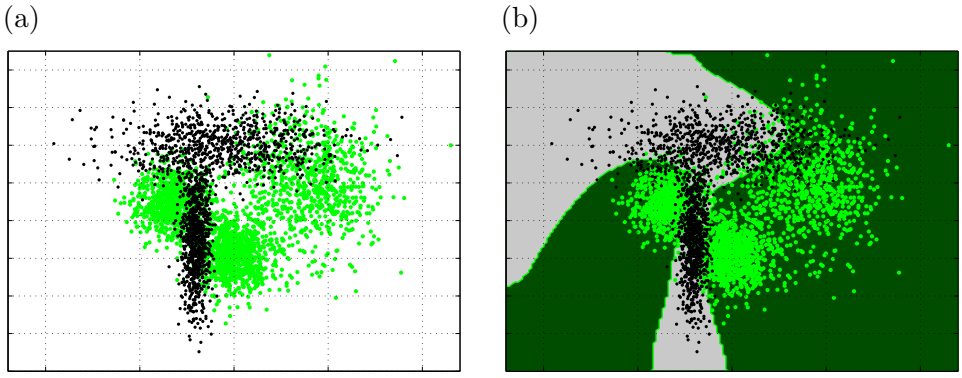

Figure 1: The two-class data set and the Bayes-optimal classification regions.

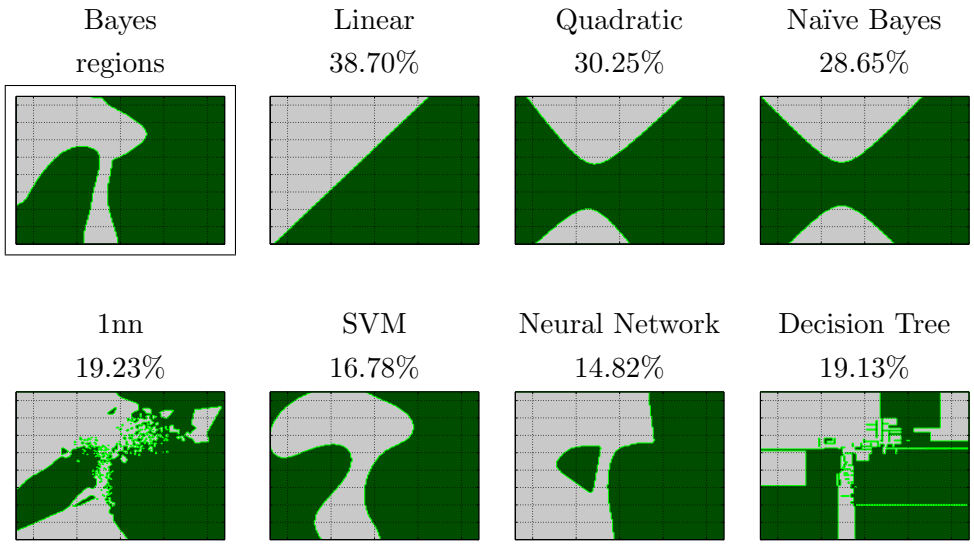

Figure 2: Classification regions and the testing error of 7 individual classifiers for the problem in Figure 1.

It can be seen that the linear discriminant classifier over-simplifies the task while 1nn over-complicates it. The best among the individual classifiers for this problem was the neural network classifier (NN).

Training and Testing Protocols

There is no classifier model that is best for all problems. Different classifiers need to be tried on the data in order asses their accuracy. The training depends on the classifier model [9]. The nearest neighbor classifier (1-nn) does not require any training; we can classify a new data point right away by finding its nearest neighbor in Z. The neural network classifier, on the other hand, is largely dependent upon the initialization and tuning of its **backpropagation training algorithm**. When trained, some classifiers can provide 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 validate the logic of the decision making, the ultimate judge of a classifier's performance is the classification error on unseen data drawn from the distribution of the problem. Estimating the **misclassification rate** is done through the training and testing protocol. Part of the data set 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 $K-1$ parts 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 folds.

Dimensionality reduction

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 models and the corresponding parameter estimates are imperfect. The task of *feature selection* is to identify a subset of the original feature set to improve the performance of the chosen classifier model. *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**). Feature extraction may also aid the classifier's performance but is often used for visualizing the data in two or three dimensions.

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 high-dimensional problems, checking all possible subsets of features is 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 d features and remove the single feature which gives the best remaining subset of $d-1$ features. Next we remove the feature that results in the best remaining subset of $d-2$ 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. In this case, the features are allowed to drop out from and re-enter the set of candidate features throughout the search. Among many others, *genetic algorithms* (GAs) have been applied for feature selection with various degrees of success. The feature set is represented as a d-dimensional binary vector where 0 indicates that the feature is absent and 1 indicates that the feature is present. Considering such a vector as a 'chromosome', GAs mimic natural evolution to evolve a population of chromosomes. The best individual is returned as the selected feature subset.

An alternative to filter and wrapper approaches is the *embedded* approach. The feature subset is selected as a part of the training of the classifier. A notable example of this group is the **Recursive Feature Elimination** (RFE) method where the feature selector is embedded in the SVM classifier [5]. RFE is suitable for very large data sets such as **functional Magnetic Resonance Image** (fMRI) data. RFE is originally designed for two-class problem, as is the SVM classifier. An SVM classifier with linear kernel is trained, resulting in a linear discriminant function between the classes. The K features corresponding to the coefficients with the smallest absolute values are discarded. Another SVM classifier with linear kernel is trained with the remaining features. The K least important features are discarded, and the process is continued until a desired cardinality of the feature set is achieved. K is a parameter of the algorithm.

Classifier Ensembles

Instead of using a single classifier, we may combine the outputs of several classifiers (base classifiers) in an attempt to reach a more accurate or reliable solution [10, 12, 16]. Ensembles of classifiers work if the individual ensemble members are both accurate and diverse. Diversity in this context means that the misclassification errors that the classifiers make, tend to be on different objects.

Base classifiers

Decision trees and neural networks have been picked as base classifiers because of their accuracy and *instability*. The instability causes the classifier to change significantly when trained on only slightly different training data. This change contributes towards the ensemble diversity. The linear discriminant classifier or k-nn are stable, and therefore are not good candidates for base classifiers in ensembles.

Combination of the classifier outputs

The most widely used method for combining the classifier outputs is *majority vote*. The class label given to object $\mathbf{x}$ is the one most represented among the classifier outputs for $\mathbf{x}$. In the *weighted majority vote*, the strength of a classifier's vote is determined by the overall accuracy of that classifier.

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 proposed in the literature that involve various degrees of training.

Ensemble methods

A list of popular ensemble methods and their most important diversifying heuristic is shown in Table 1.

Table 1: Popular ensemble methods

Ensemble method	Description
Bagging	Independent bootstrap samples of the data
Random Forest	Random tree as the base classifier & Bagging
AdaBoost	Dependent bootstrap samples of the data
Random Subspace	Independent sub-samples of the features
Rotation Forest	Bootstrap samples of the data and PCA rotation

Bagging has become a classic in pattern recognition and machine learning [2]. It takes L random bootstrap samples from 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 Forest** is a variant of bagging [3]. Its typical implementation uses a random tree as the base classifier with the bagging approach, thereby improving the ensembles diversity. Random trees differ from standard classification and regression trees by a minor training detail, which makes them particularly suitable for ensembles. During training of a node in the random tree, only a random subset of features are checked instead of all features. A feature subset is sampled afresh for each node. Random forests are therefore useful for high-dimensional problems.

Instead of drawing random bootstrap samples, **AdaBoost** designs the ensemble members one at a time, based on the performance of the previous member [13]. A set of weights is maintained across

the data points in 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.

Random subspace ensembles are suitable for high-dimensional problems and work well even with stable base classifiers such as k-nn [8]. Finally, the **Rotation Forest** ensemble benefits from partial rotation of the feature space [11]. In doing so, the class boundaries of the decision tree classifier, which are always parallel to the coordinate axes, are also rotated. Hence the classification regions can be approximated more accurately, and with fewer ensemble members.

Figure 3 gives the classification regions obtained from three ensemble methods: Bagging, AdaBoost and Rotation Forest. (Random Forest and Random Subspace are not useful in two dimensions.) Generally, ensemble methods can be expected to work better than individual classifiers. This example reinforces this message, with the exception of SVM classifier, which rates before AdaBoost.

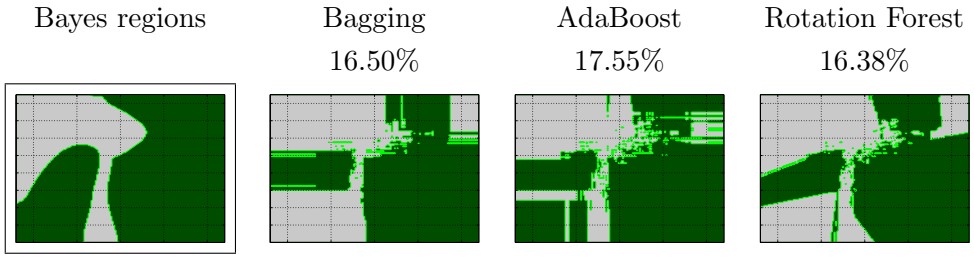

Figure 3: Classification regions and the testing error of 3 ensemble classifiers for the problem in Figure 1.

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**, network security, document classification, financial forecasting, organization and retrieval of multimedia databases, microarray data analysis (*see* **Microarrays**), fMRI data analysis and many more [9].

Software

WEKA[2] [6, 15] is a collection of machine learning algorithms for data mining tasks. It contains a wide collection of individual classifier and classifier ensemble methods (called meta-classifiers). It is open source software issued under the GNU General Public License.

PRTools[3] is a MATLAB toolbox for pattern recognition developed by the Pattern Recognition Research Group of the TU Delft, The Netherlands, led by Professor R.P.W. (Bob) Duin. An industry oriented spin-off toolbox, called 'perClass'[4] was designed later. The recent editions of MATLAB Statistics toolbox (since 2013b) include a suite of classifiers and classifier ensemble methods.

[2] `http://www.cs.waikato.ac.nz/ml/weka/`
[3] `http://prtools.org/`
[4] `http://perclass.com/index.php/html/`

Related Articles

Multivariate classification rules: calibration and discrimination, **Classification - Further Developments**, **Hierarchical Cluster Analysis**, **Fuzzy Cluster Analysis**, **Cluster Analysis**, **Graph-Theoretic**

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, NY, 2006.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[3] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, NY, second edition, 2001.

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

[8] T. K. Ho. The random space method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

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

[10] L. I. Kuncheva. *Combining Pattern Classifiers. Methods and Algorithms*. Wiley, 2nd edition, 2014.

[11] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, Oct 2006.

[12] L. Rokach. *Pattern Classification Using Ensemble Methods*. World Sceintific, 2010.

[13] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2012.

[14] Andrew R. Webb and Keith D. Copsey. *Statistical Pattern Recognition, 3rd Edition*. Wiley, 3rd edition, 2011.

[15] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[16] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithm*. CRC Press – Business & Economics, 2012.