Article type: Overview

Pattern Recognition CSV0050

Lasse Holmström

Department of Mathematical Sciences, University of Oulu

Petri Koistinen

Department of Mathematics and Statistics, University of Helsinki

Keywords

Decision theory, feature extraction, nearest neighbor methods, neural networks, support vector machines

Abstract

We give an overview of pattern recognition, concentrating on the problem of pattern classification. Several popular discrimination methods are reviewed using decision theory as a unifying framework.

Pattern recognition is an engineering discipline, where the goal is to build systems which are able to classify real-world objects of interest into one of a number classes on the basis of measurements. The objects or patterns can be, e.g., printed or handwritten characters, biological cells or acoustic or electronic signals. The measurements can be, e.g., signal waveforms, images or image sequences. Table 1 lists typical application areas. To achieve the goal, one

Table 1 Typical application areas for pattern recognition.

Automated visual inspection of products in manufacturing
Automatic speech recognition
Classification of text into categories (e.g., spam vs. non-spam)
Character recognition (for printed or for handwritten text)
Computer-aided diagnosis using a variety of medical data
Classification of ground cover types in remote sensing
Face and gesture recognition from images or image sequences

may apply techniques from signal and image processing, computer science, neural computation and statistics.

Often the raw measurements one can make are so many in number, that inferring their statistical properties is hopeless. Therefore they are usually first transformed into a vector whose components are called features. This transformation is called feature selection or feature extraction and it is very much application dependent. While some generic methods are available, one should study carefully the application-specific literature in order to identify promising features.

When each pattern is represented by a numerical feature vector, then one speaks of statistical pattern recognition. There are approaches grouped under the term syntactic or structural pattern recognition, where the patterns are represented using more complicated structures such as strings in a formal grammar. The rest of this overview discusses statistical pattern recognition. If we have available a set of patterns whose classes and feature vectors are already known, then one speaks of supervised pattern recognition or discriminant analysis. However, sometimes the classes have not yet been defined, and one attempts to find classes of objects with similar properties. Then one has a problem in unsupervised pattern recognition or clustering. This article concentrates on supervised pattern recognition.

Reading material

Pattern recognition has a long history. It had its beginnings in the statistical literature of the 1930s. The advent of computers in 1950s and 1960s brought a demand for practical applications, and the field developed significantly. The 1970s brought great developments in the probabilistic theory of pattern recognition. One significant boost for the field was the sudden growth of neural network research in the late 1980s and 1990s. New developments take currently place, e.g., in the machine learning community. Also emerging applications drive the development of the field.

For further study of pattern recognition, we recommend one of the many texts currently available, such as [13, 9, 27, 24]. Older but still useful texts include [28, 7, 25, 22, 11]. On a more advanced level, McLachlan [20] gives a scholarly treatment of discriminant analysis, Ripley [21] presents an advanced synthesis of statistical pattern recognition and neural networks, while the monograph [8] can be consulted for rigorous probabilistic results on the generalization ability of a range of classification rules. Several texts on multivariate statistical analysis such as [19, 18, 2, 16] and texts and monographs on machine learning methods such as [12, 14, 3] are relevant. Handbooks are also available, including [17, 29, 4]. A variety of scientific journals and conferences contain both theoretical and application oriented articles on pattern recognition.

Classifiers based on decision theory

In the pattern classification problem, an object is to be classified as belonging to one of c mutually exclusive classes (or categories), labeled $1, \ldots, c$. We denote the true class of the object by J. The classification is to be made on the basis of features X_i measured from the object. Together they form the feature vector $\mathbf{X} = [X_1, \ldots, X_d]^T$. The feature vector exhibits random variation, partly due to the different properties of the different classes and partly due to variation within each class. We regard the class J and the feature vector $\mathbf{X}$ to be random quantities, which have a joint distribution. Whereas the class J is a discrete random variable, the feature vector $\mathbf{X}$ can have either a continuous or a discrete distribution, or some of its components may have discrete and the others have a continuous distribution.

Many authors avoid the use of a separate random variable to denote the class of the object, and instead denote the j'th class, e.g., by ω_j and use a phrase like "$\mathbf{X}$ belongs

to class ω_j" or even the corresponding notation $\mathbf{X} \in \omega_j$ to indicate that the class of the object with feature vector $\mathbf{X}$ is j. Especially the notation $\mathbf{X} \in \omega_j$ can be confusing for the newcomer, who has not yet internalized the central idea that the feature vector of the object does not determine its class uniquely. It is therefore better to use a separate random variable (such as J) to denote the class of the object.

The pattern classification (or discrimination) task is to design a classifier, which tries to guess the class J of the object based on the value of the feature vector $\mathbf{X}$. This guess is calculated by a classifier g (also called a decision or discrimination rule), which is simply a function defined on $\mathbb{R}^d$ such that $g(\mathbf{x})$ is the classifier's guess of J, when the feature vector $\mathbf{X}$ has the value $\mathbf{x}$. We then say that $\mathbf{X}$ is classified, allocated or assigned to class $g(\mathbf{X})$. The classifier errs, when $g(\mathbf{X}) \neq J$. An alternative viewpoint is that the classifier $g(\mathbf{X})$ determines to which of the decision regions $A_j = \{\mathbf{x} \in \mathbb{R}^d : g(\mathbf{x}) = j\}$ the feature vector $\mathbf{X}$ belongs. Their boundaries are called decision boundaries.

Usually, the classifier g returns one of the valid class labels $1, \ldots, c$. In some applications, however, the classifier is also allowed to reject the feature vector, which is known as the reject option. Rejected feature vectors are set aside, e.g., to be classified by a human expert. Rejection can be represented by a separate label, such as $g(\mathbf{x}) = 0$.

If the joint distribution of $\mathbf{X}$ and J is known, then decision theory allows us to find classifiers which are optimal according to various criteria. These theoretically optimal classifiers can be kept as guides, when some aspects of the joint distribution must be estimated. According to the multiplication rule of probability theory, the joint density of two random quantities can be factorized as the marginal density of one times the conditional density of the other. Therefore the joint density $f(\mathbf{x}, j)$ of $\mathbf{X}$ and J can be factorized as

$$f(\mathbf{x}, j) = P_j \, f_j(\mathbf{x}) = f_{\mathbf{X}}(\mathbf{x}) \, P(j|\mathbf{x}). \tag{1}$$

In the first factorization, $P_j = P(J = j)$ is the marginal probability of class j, and $f_j(\mathbf{x})$ is the class-conditional density (probability density function or probability mass function) of $\mathbf{X}$ given that $J = j$. In the second factorization, $f_{\mathbf{X}}(\mathbf{x})$ is the marginal density of $\mathbf{X}$, and $P(j|\mathbf{x})$ is the conditional probability of $J = j$ given that $\mathbf{X} = \mathbf{x}$. P_j is called the prior probability and $P(j|\mathbf{x})$ the posterior probability of class j. Both the prior and the posterior probabilities of the classes sum to one. From the factorizations (1), we obtain the posterior probabilities as

$$P(j|\mathbf{x}) = \frac{P_j f_j(\mathbf{x})}{f_{\mathbf{X}}(\mathbf{x})}, \quad \text{where} \quad f_{\mathbf{X}}(\mathbf{x}) = \sum_{j=1}^{c} P_j \, f_j(\mathbf{x}). \tag{2}$$

See Fig. 1 for an illustration of these concepts.

We next derive the form of the classifier, which has the least possible risk. Let $\lambda(j_0, j)$ be the loss (or cost), when $J = j_0$ and $g(\mathbf{X}) = j$ for $j_0 = 1, \ldots, c$ and for $j = 1, \ldots, c$. We may allow the reject option by defining $\lambda(j_0, j)$ also for $j = 0$. Then the expected loss or risk for classifier g is given by

$$R(g) = E[\lambda(J, g(\mathbf{X}))]. \tag{3}$$

Usually, one would penalize only for misclassifications, which corresponds to choosing $\lambda(j, j) = 0$ for all j. If we further select $\lambda(j_0, j)$ equal to one when $j_0 \neq j$, then $R(g)$ is simply the error probability of g. The situation where certain kinds of misclassifications are more serious than others can be modeled by using unequal losses.

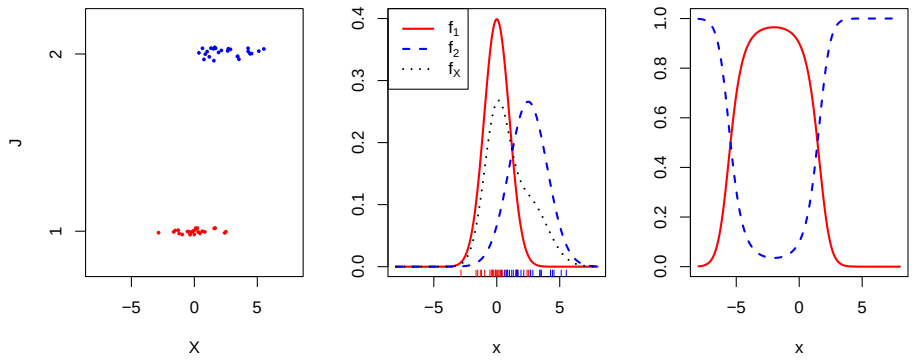

Figure 1: The joint distribution of class J and feature vector X, when there are two classes, and the feature vector dimension is one. The first panel shows a jittered scatter plot of (X, J), the middle panel shows the class conditional densities f_1 and f_2 as well as the marginal density f_X, while the last panel shows the two posterior probabilities. The prior probabilities are $P_1 = 0.6$ and $P_2 = 1 - P_1$.

By writing the expectation (3) as an iterated expectation, we obtain

$$R(g) = E[E[\lambda(J, g(\mathbf{X})) \mid \mathbf{X}]] = \int_{\mathbb{R}^d} \left[\sum_{j_0=1}^{c} \lambda(j_0, g(\mathbf{x}))\, P(j_0|\mathbf{x}) \right] f_{\mathbf{X}}(\mathbf{x})\, \mathrm{d}\mathbf{x}.$$

(This formula is valid when $\mathbf{X}$ is continuously distributed, but in the general case, the result is a Lebesgue integral with respect to the marginal distribution of $\mathbf{X}$.) Since $f_{\mathbf{X}}(\mathbf{x}) \geq 0$, it is clear that the optimal classifier g^* is obtained by minimizing at each $\mathbf{x}$ the conditional risk

$$E[\lambda(J, g(\mathbf{X})) \mid \mathbf{X} = \mathbf{x}] = \sum_{j_0=1}^{c} \lambda(j_0, g(\mathbf{x}))\, P(j_0|\mathbf{x}).$$

Such is always the case in Bayesian decision theory. Hence the optimal classifier is

$$g^*(\mathbf{x}) = \arg\min_j \sum_{j_0=1}^{c} \lambda(j_0, j)\, P(j_0|\mathbf{x}), \tag{4}$$

where $\arg\min_j$ selects that value of the argument j, which minimizes the expression following it, when j ranges over the possible values: over $1, \ldots, c$ when rejection is not allowed, and over $0, 1, \ldots, c$ otherwise. If there are ties, then the minimizing argument can be selected arbitrarily among the minimizers. (We also use the $\arg\max$ operator, which is defined similarly.) The resulting classifier g^* is the Bayes classifier for minimum risk.

If we penalize one unit for each misclassification, and do not allow rejections, then the loss $\lambda(j_0, j)$ is zero when $j_0 = j$ and one when $j_0 \neq j$, and here $j_0, j \in \{1, \ldots, c\}$. The corresponding risk is the classifier error probability, which is the most widely used criterion in practice. In this case

$$\sum_{j_0=1}^{c} \lambda(j_0, j)\, P(j_0|\mathbf{x}) = \sum_{j_0 \neq j} P(j_0|\mathbf{x}) = 1 - P(j|\mathbf{x}).$$

Here we used the fact that the posterior probabilities of the classes sum to one. Since minimizing $1 - P(j|\mathbf{x})$ is the same as maximizing $P(j|\mathbf{x})$, we see that the Bayes classifier for minimum error probability is given by

$$g^*(\mathbf{x}) = \arg\max_{j=1,\ldots,c} P(j|\mathbf{x}). \tag{5}$$

The resulting classifier is often called simply the Bayes classifier. It assigns $\mathbf{x}$ to that class, whose posterior probability is greatest. When the class-conditional distributions overlap, even the Bayes classifier has positive error probability. Using (2), and noticing that the positive factor $f_{\mathbf{X}}(\mathbf{x})$ is common to all the classes, we obtain the following alternative expression for the Bayes classifier,

$$g^*(\mathbf{x}) = \arg\max_{j=1,\ldots,c} P_j f_j(\mathbf{x}). \tag{6}$$

Besides the risk, other kinds of criteria are important, such as the Neyman-Pearson criterion, which is available in a two-class problem, see e.g. [27]. There the classification error of one class is minimized given a fixed error probability for the other class.

Designing classifiers on the basis of training data

Although the formulas for the optimal classifiers depend on the joint distribution of the pair $(\mathbf{X}, J)$, one does not know it in practice, but only has available a sequence of pairs $\mathcal{D} = ((\mathbf{X}_1, J_1), \ldots, (\mathbf{X}_n, J_n))$ of feature vectors and classes of objects. One typically assumes that the training data is obtained either by mixture sampling or by separate sampling. In mixture sampling the training data are assumed to be an i.i.d. sample from the joint distribution of $(\mathbf{X}, J)$. In separate sampling the feature vectors are assumed to be sampled independently and separately from each of the classes. Consequently, in separate sampling the sample sizes of the different classes do not necessarily reflect their prior probabilities. These training data (or design data) are then used for designing the classifier applying, e.g., some of the following ideas.

1. One first forms estimates $\hat{P}_j$ and $\hat{f}_j(\mathbf{x})$ for the prior probabilities and class-conditional densities, which are then plugged in the formula for the optimal classifier. E.g., instead of (6), one can use the plug-in classifier

   $$g(\mathbf{x}) = \arg\max_{j=1,\ldots,c} \hat{P}_j \hat{f}_j(\mathbf{x}). \tag{7}$$

   The prior probabilities P_j are usually estimated by the relative frequencies of the classes in some large data set (which does not need to be the training set). The class-conditional densities $f_j(\mathbf{x})$ can be estimated using various parametric or non-parametric approaches.

2. The optimal classifiers depend on the joint distribution only through the posterior probabilities $P(j|\mathbf{x})$. Therefore one viable approach is to form directly estimates $\hat{P}(j|\mathbf{x})$ of the posterior probabilities and plug them in the formulas for optimal classifiers. E.g., instead of (5), one can use

   $$g(\mathbf{x}) = \arg\max_{j=1,\ldots,c} \hat{P}(j|\mathbf{x}). \tag{8}$$

The posterior probabilities can be estimated using various parametric or non-parametric regression approaches.

3. While the two previous approaches use estimates as if they were equal to the unknown population quantities, in the Bayesian approach to statistics one usually averages over such uncertainty. A natural idea is to base the classification decision on the predictive distribution of the class label (i.e., its conditional distribution) conditional on the training data $\mathcal{D}$ and the observed feature vector $\mathbf{x}$. The predictive distributions of the classes are then used in place of the theoretical class posterior probabilities. In some cases, the predictive distributions can be calculated in closed form, but otherwise they must be approximated, e.g., by Markov chain Monte Carlo methods.

4. Another approach is to estimate the decision regions directly, e.g., by making assumptions about the forms of the decision boundaries.

Normal-based classifiers

A classical approach is to assume that the class-conditional distributions belong to a family of distributions described by a finite set of parameters. Once these parameters have been estimated, one can use the plug-in rule (7). The most popular choice is to model the distribution of $\mathbf{X}$ conditional on $J = j$ as a multinormal distribution $N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, where $\boldsymbol{\mu}_j$ is the mean vector and $\boldsymbol{\Sigma}_j$ the covariance matrix of class j. The resulting quadratic and linear classifiers are among the most popular classifiers in applications, and are discussed in all pattern recognition texts.

Letting $f_j(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ denote the density of $N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, one sees immediately that the logarithm of $P_j\, f_j(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ is a quadratic function of $\mathbf{x}$. The parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ can be estimated by the sample mean $\hat{\boldsymbol{\mu}}_j$ and sample covariance matrix $\hat{\boldsymbol{\Sigma}}_j$ of those training feature vectors originating from class j, but other estimates of $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ can be used as well. Once the parameter estimates have been plugged in, the classifier (7) is equivalent to selecting the maximal one out of c quadratic functions of $\mathbf{x}$, which are called discriminant functions. This procedure is often called Quadratic Discriminant Analysis (QDA), and the classifier may also be called the Gaussian classifier or the (normal-based) quadratic classifier.

Instead of the previous heteroscedastic model, where each class has its own covariance matrix, one can also use a homoscedastic model, where the covariance matrix is the same in all the classes, i.e., $\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$. In this case the quadratic term is common to all the classes and can be canceled from the discriminant functions. The common covariance matrix can be estimated by pooling the within-class covariance matrix estimates needed for the quadratic classifier. The resulting classifier selects the maximizing argument among c first degree polynomials. In a pioneering work published in 1936, R. A. Fisher proposed a method, which is equivalent with this in the two-class case. This approach can be called Gaussian Linear Discriminant Analysis (LDA), or the (normal-based) linear classifier.

Often the sample sizes from each of the classes are so small compared to the feature vector dimension that the covariance estimates in the heteroscedastic model are highly variable. One option is then to use the homoscedastic model, even when there are no

grounds for believing that the covariance matrices are equal. Another option is to use Regularized Discriminant Analysis (RDA) [10], which regularizes the covariance estimates of QDA by shrinking them, firstly, toward the pooled covariance estimate and, secondly, toward a multiple of the identity matrix. The method uses two regularizing parameters, which can be selected by cross-validation.

As discussed by Ripley [21, Sec. 2.4], several predictive classifiers can be expressed in closed form within the normal model. See [23] for more recent work.

Non-parametric approaches

The normal-based classifier assumes that the class-conditional densities are Gaussian. In many situations non-parametric estimates of the class-conditional densities may lead to a better plug-in classifier (7). The most popular non-parametric density estimator is the kernel method. Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be a random sample from a d-variate distribution with a density f. The kernel estimator of f is

$$\hat{f}(\mathbf{x}; h) = \frac{1}{n} \sum_{i=1}^{n} K_h(\mathbf{x} - \mathbf{X}_i), \tag{9}$$

where the kernel K satisfies $\int_{\mathbb{R}^d} K = 1$, $h > 0$ is the smoothing parameter, and $K_h(\mathbf{x}) = h^{-d} K(h^{-1}\mathbf{x})$ is the scaled kernel. The performance of the estimator does not depend much on the choice of K and one often uses the standard multinormal kernel. What matters is a proper value for the smoothing parameter h. A small value of h produces a spiky estimate with high variance whereas a large value of h results in a smooth estimate with a large bias. Using training data from class j, the class-conditional density f_j can be estimated by a kernel estimator $\hat{f}_j(\mathbf{x}; h_j)$ and then used in the classifier (7). This is often called kernel discriminant analysis (KDA). The smoothing parameters h_j are selected to minimize the classification error probability using, e.g., cross-validation on the training set. While accurate estimates $\hat{f}_j(\mathbf{x}; h_j)$ of the class-conditional densities f_j of course will lead to a good classifier, what really matters is that the classifier properly models the decision boundaries. For this reason even biased kernel estimates may produce a classifier with a low classification error.

Sometimes the kernel estimate of a density improves if the shapes of the kernel and the density are chosen to be similar. The shape of the kernel can be adjusted by generalizing (9) to $\hat{f}(\mathbf{x}; \mathbf{H}) = \frac{1}{n} \sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{x} - \mathbf{X}_i)$, where $\mathbf{H}$ is a symmetric positive definitive scaling matrix and $K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}^{-1/2}| K(\mathbf{H}^{-1/2}\mathbf{x})$. A special case is a diagonal matrix $\mathbf{H} = \mathrm{diag}(h_1, \ldots, h_d)$ which allows different levels of smoothing for different variables of the feature vector.

Another popular non-parametric approach to classification has been the k-nearest neighbor method (k-NN). It can be interpreted as an instance of the plug-in rule (7) as follows. Let $1 \leq k \leq n$ and $\mathbf{x} \in \mathbb{R}^d$. Consider the training data fixed and permute the k training vectors $\mathbf{X}_i$ nearest to $\mathbf{x}$ in an order of increasing distance from $\mathbf{x}$,

$$\|\mathbf{x} - \mathbf{X}_{i_1}\| \leq \|\mathbf{x} - \mathbf{X}_{i_2}\| \leq \cdots \leq \|\mathbf{x} - \mathbf{X}_{i_k}\|. \tag{10}$$

Let $\delta_k = \|\mathbf{x} - \mathbf{X}_{i_k}\|$ be the distance from $\mathbf{x}$ to its kth nearest neighbor and denote by $B = B(\mathbf{x}, \delta_k)$ the ball with radius δ_k centered at $\mathbf{x}$. Let m_j be the number of training

vectors from class j in this ball. If the the total number of class j training vectors is n_j and the class-conditional density f_j does not change much in the neighborhood B, then

$$\frac{m_j}{n_j} \approx P\left(\mathbf{X} \in B \mid J = j\right) = \int_B f_j(\mathbf{y})\,\mathrm{d}\mathbf{y} \approx f_j(\mathbf{x}) \cdot \mathrm{Vol}(B), \tag{11}$$

where $\mathrm{Vol}(B) = c_d \delta_k^d$ and the constant c_d depends on the dimension d of the feature space. Solving for $f_j(\mathbf{x})$ in (11) one then obtains a natural density estimate

$$\hat{f}_j(\mathbf{x}; k) = \frac{m_j/n_j}{c_d \delta_k^d}. \tag{12}$$

Here k plays the role of a smoothing parameter similar to h in kernel estimation: a small k produces a highly variable density function estimate whereas a large k results in a smooth, possibly biased estimate. If the training set is obtained using mixture sampling, a convenient estimate of the class prior probability is $\hat{P}_j = n_j/n$ and the rule (7) then takes the form

$$g(\mathbf{x}) = \underset{j=1,\ldots,c}{\arg\max}\, \frac{n_j}{n} \cdot \frac{m_j/n_j}{c_d \delta_k^d} = \underset{j=1,\ldots,c}{\arg\max}\, m_j,$$

so that $\mathbf{x}$ is assigned to the class with the most training vectors among its k nearest neighbors. Ties need to be resolved separately, and one may for example choose that class whose farthest training vector from $\mathbf{x}$ is closest to it among the tied classes. The justification of the k-nearest neighbor classifier through density estimation is most plausible when $k \gg 1$ and n is large with $k \ll n$. Then even a small neighborhood of $\mathbf{x}$ can contain many training vectors from all classes and the approximations in (11) are credible. In practice, however, very small values of k are often used, one of the most popular non-parametric discrimination methods being in fact the 1-nearest neighbor classifier. Figure 2 shows decision regions obtained with QDA and the 1-nearest neighbor classifier.

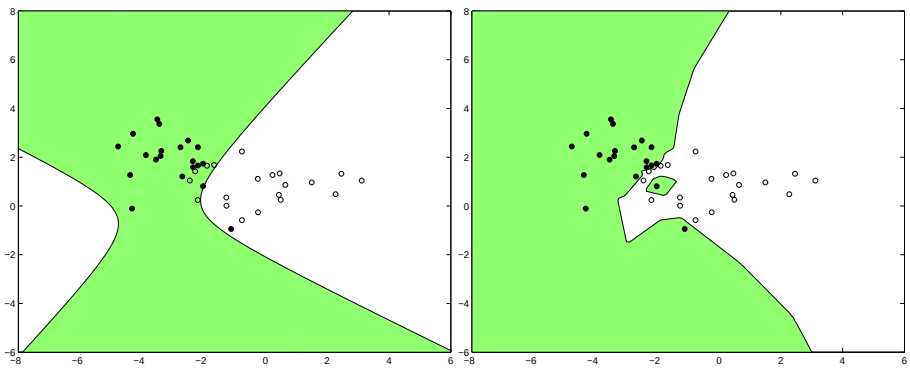

Figure 2: Decision regions obtained using QDA (left) and the 1-nearest neighbor classifier (right) for the same training data.

For large n finding the nearest neighbors can incur a heavy computational overhead and various training set preprocessing schemes have been proposed to alleviate this problem. Computational efficiency can also be improved using so-called editing techniques

that aim to select representative subsets of training vectors for each class and construct a classifier based on the training set thus reduced in size. For further reading on kernel discriminant analysis and nearest neighbor methods, see for example [20, 8, 27].

Logistic discrimination

A popular statistical approach to classification in a two-class problem is logistic regression. There one models the log-odds, in favor of class one, using a linear function of $\mathbf{x}$,

$$\log \frac{P(1|\mathbf{x})}{P(2|\mathbf{x})} = \alpha + \boldsymbol{\beta}^T \mathbf{x}, \tag{13}$$

where $P(2|\mathbf{x}) = 1 - P(1|\mathbf{x})$. The assumption (13) is equivalent with

$$P(1|\mathbf{x}) = \frac{\exp(\alpha + \boldsymbol{\beta}^T \mathbf{x})}{1 + \exp(\alpha + \boldsymbol{\beta}^T \mathbf{x})}, \qquad P(2|\mathbf{x}) = \frac{1}{1 + \exp(\alpha + \boldsymbol{\beta}^T \mathbf{x})}. \tag{14}$$

The marginal distribution of $\mathbf{X}$ is here typically left unmodelled by conditioning on the feature vectors in the training data.

Conditionally on $\mathbf{X}_i$, the class J_i has the value 1 with probability $P(1|\mathbf{X}_i)$ and the value 2 with probability $P(2|\mathbf{X}_i)$, and so the conditional likelihood for $(\alpha, \boldsymbol{\beta})$ is

$$\prod_{i=1}^{n} \left[P(1|\mathbf{X}_i)\right]^{1(J_i=1)} \left[P(2|\mathbf{X}_i)\right]^{1(J_i=2)},$$

where the two posterior probabilities are as in (14), and the indicators of the two classes $1(J_i = 1)$ and $1(J_i = 2)$ select the first term when $J_i = 1$ and the second term when $J_i = 2$. This is suitable both for separate and for mixture sampling. The parameter values maximizing the conditional likelihood can be calculated with widely available software for fitting generalized linear models (GLMs), and then one can use (8), which is equivalent with classifying to class one whenever $\hat{P}(1|\mathbf{x}) > 0.5$.

The multi-class analog of the logistic regression model is the multinomial (or multiple) logistic model, where the posterior probabilities are modeled as

$$\log \frac{P(j|\mathbf{x})}{P(c|\mathbf{x})} = r_j(\mathbf{x}), \qquad j = 1, \ldots, c-1, \tag{15}$$

where $r_j(\mathbf{x}) = \alpha_j + \boldsymbol{\beta}_j^T \mathbf{x}$ are $c - 1$ linear functions. Equivalently,

$$P(j|\mathbf{x}) = \frac{\exp(r_j(\mathbf{x}))}{\sum_{k=1}^{c} \exp(r_k(\mathbf{x}))}, \qquad j = 1, \ldots, c \tag{16}$$

where we have made use of the convention that $r_c(\mathbf{x}) \equiv 1$. The conditional likelihood is now multinomial,

$$\prod_{i=1}^{n} \prod_{j=1}^{c} \left[P(j|\mathbf{X}_i)\right]^{1(J_i=j)},$$

and also this model can be fitted using software for GLMs. Instead of the linear form, one can also choose the functions r_j to be nonlinear in their parameters, e.g., they could be neural networks of the feedforward type, such as multilayer perceptrons or radial basis function networks. However, then the model falls outside the class of GLMs.

Multilayer perceptrons

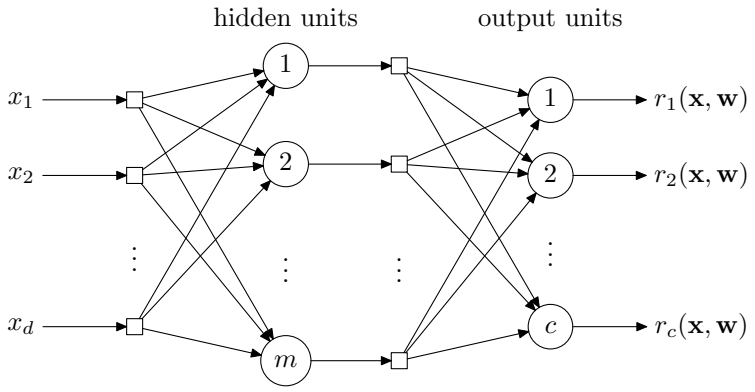

Figure 3: A multilayer perceptron with one layer of hidden units.

Fig. 3 shows the structure of a multilayer perceptron (MLP) with d inputs, one layer of m hidden units and c output units. Each unit calculates a linear combination of its outputs, optionally applies a nonlinear function, and passes the result as its output, which becomes either an input to other units or the output of the network. If there are no nonlinearities in the output layer, then the MLP evaluates the function $\mathbf{r}(\mathbf{x}, \mathbf{w})$, where

$$r_j(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{m} w_{ji}^{(2)} \, \varphi \left(\sum_{\ell=1}^{d} w_{i\ell}^{(1)} x_\ell + w_{i0}^{(1)} \right) + w_{j0}^{(2)}, \qquad j = 1, \ldots, c. \tag{17}$$

Here the weight vector $\mathbf{w}$ contains all the weights (and biases) of the MLP: $w_{i\ell}^{(1)}$ is the weight from the ℓth input to the ith hidden unit and $w_{i0}^{(1)}$ is its bias weight; $w_{ji}^{(2)}$ is the weight from the ith hidden unit to the jth output unit and $w_{j0}^{(2)}$ is its bias weight. The most popular choice for the nonlinearity φ is the logistic sigmoid $\varphi(z) = 1/(1 + \exp(-z))$. Thanks to their flexibility, MLPs have been very popular in all kinds of nonlinear regression and classification tasks.

As already mentioned, one approach to fitting a MLP is to use $c-1$ outputs (or even an overparametrized model with c outputs) together with the structure (16) (which is often called the softmax function) and the multinomial conditional likelihood. However, perhaps the most widely used fitting criterion in classification tasks is the squared error criterion

$$\sum_{i=1}^{n} \|\mathbf{t}_i - \mathbf{r}(\mathbf{X}_i, \mathbf{w})\|^2 = \min_{\mathbf{w}}!,$$

where the ith target $\mathbf{t}_i$ is the indicator vector of the class J_i, i.e., t_{ij} is one if $J_i = j$ and zero otherwise. Under mixture sampling, the outputs of the estimated network $r_j(\mathbf{x}, \widehat{\mathbf{w}})$ can be interpreted as estimates of the posterior probabilities of the classes, and then one can use the rule (8). One often adds a regularization term to the optimization criterion in order to make the estimates more stable. This can be, e.g., of the weight decay form,

$\lambda\|\mathbf{w}\|^2$, where $\lambda > 0$. The number of hidden units and the value of the regularization parameter λ can be chosen, e.g., with cross-validation.

All the formulations lead to a nonconvex optimization problem which has several local and global extrema. One can fit the MLP using either general purpose optimization routines or special optimization methods tailored for MLPs. The gradient (and higher derivatives) of the optimization criterion can be calculated using formulas, which are explained in the literature in connection with the backpropagation method.

See [15] for a view on the relationship between neural and statistical classifiers and, e.g., [21, 3] for more information on neural network models for classification.

Support vector machines

Support vector machines (SVMs) [5] are currently of great interest to theoretical and applied researchers and they have strong connections to computational learning theory. The basic idea is easiest to understand, when we have a linearly separable two-class problem. The resulting classifier is called the maximal margin classifier. The idea is to search the optimal separating hyperplane which has the maximal margin of separation between the training vectors from the two classes, so maximal margin classifiers estimate directly the decision boundary. Being a separating hyperplane means that the training vectors from the two classes lie on different sides of the hyperplane, and having maximal margin means that the distance from the hyperplane to the nearest training vector is maximal. The support vectors are those training vectors which lie nearest to the optimal hyperplane. This optimization problem can be formulated as a quadratic programming problem. In real applications, the training data is usually not linearly separable and then the maximal margin hyperplane does not exist. A solution is to seek the so-called soft-margin hyperplane instead. Also this leads to a quadratic program. Since the construction of SVM classifiers leads to standard convex optimization problems, there are no complications with local minima as there are with MLPs. These quadratic programs can be solved either by general purpose quadratic program solvers or by techniques developed specially for SVMs.

Suppose we transform the original feature vectors into some high-dimensional or even infinite-dimensional (Hilbert) space using a nonlinear mapping φ before constructing the maximal margin of the soft-margin hyperplane. Using a dual formulation of the original quadratic program, one obtains another quadratic program, which depends on the training vectors only through their inner products. In the transformed space the inner products can be represented using a kernel function

$$\langle \varphi(\mathbf{X}_i), \varphi(\mathbf{X}_j)) \rangle = K(\mathbf{X}_i, \mathbf{X}_j).$$

Thus the inner products needed for the construction of the SVM classifier can be can be calculated in the original feature space. Also the resulting classifier can be implemented using the kernel and then working out what the non-linear mapping φ is becomes unnecessary. This is the so-called kernel trick. By Mercer's theorem, the kernel has to be a non-negative definite function. In applications, one may start by choosing a convenient form for the kernel among the wide selection of valid kernels available in the literature.

SVM classifiers can be used also in a multiclass problem using one of several approaches. For more information on SVMs, see e.g. [6, 16, 24, 14] and the web page [1].

Other methods

A number of methods construct the decision regions using recursive partitioning of the feature space. A well-known approach in this vein are the tree-structured classifiers such as the classification and regression tree method that includes the original CART and its variants. An important property of this approach is that, where most classifiers operate as "black boxes", tree classifiers are also able to provide a potentially useful explanation for the assignment of a pattern to a particular class in the form of threshold values on its features X_i. Another, graphical model based approach are the Bayesian belief networks.

To improve discrimination performance, combination of several classifiers into a single "committee" classifier has also received much attention in recent years. This approach includes for example bagging and boosting which, by perturbing the training set, generate a collection of classifiers that are combined into a single combined classifier.

For more information on these and other techniques not covered in this article, see e.g. [16, 26, 21].

Feature extraction and selection

Often the measured pattern vectors are too high dimensional for useful estimation of a classifier. The raw patterns are therefore first transformed to lower dimensional feature vectors in a way that hopefully preserves the salient class information of the original measurements. However, such a transformation can never improve the theoretically optimal classification result. Indeed, let $\varphi : \mathbb{R}^D \to \mathbb{R}^d$ be the transformation of the raw pattern vector $\mathbf{Y}$ into the feature vector $\mathbf{X}$ (here usually $d \ll D$). Then any classifier g in $\mathbb{R}^d$ induces a classifier $g \circ \varphi$ in the original pattern space and if the lowest (Bayes) error for $\mathbf{Y}$ is e^*, then we have $P(g(\mathbf{X}) \neq J) = P((g \circ \varphi)(\mathbf{Y}) \neq J) \geq e^*$. Still, in practice, dimension reduction makes classifier estimation easier which often more than compensates for the possible loss of classification information in the transformation. Also, if one wishes to use a particular classifier type, a clever feature transformation can sometimes be used to improve classifier performance. A prime example of this idea is the support vector machine that combines a non-linear mapping with a simple linear classifier.

Ideally, the classifier and the feature transformation should designed in a joint process but often in practice these two design steps are separated. While application-specific information is usually needed for best results, there are also some generic methods for transforming the raw patterns into feature vectors. These include feature selection, feature extraction typically based on a linear mapping, as well as general purpose dimension reduction techniques.

In feature selection one tries to choose from the raw pattern $\mathbf{Y} = [Y_1, \ldots, Y_D]^T$ variables Y_i that are most useful in discrimination. Thus $\mathbf{X} = \varphi(\mathbf{Y}) = [Y_{i_1}, \ldots, Y_{i_d}]^T$, where $i_1 < \cdots < i_d$. Estimated classification error or some other measure of class separation can be used to rank the performance of different subsets of indices. Because of combinatorial complexity, exhaustive search through all possible subsets is rarely feasible and one resorts to various suboptimal, incremental schemes that add or delete one feature at a time.

A typical linear feature extraction method uses a transformation of the form $\varphi(\mathbf{Y}) = [\mathbf{a}_1^T\mathbf{Y}, \ldots, \mathbf{a}_d^T\mathbf{Y}]^T$, where $\mathbf{a}_1, \ldots, \mathbf{a}_d$ are orthonormal vectors chosen to optimize some measure of within-class spread and between-class separation. The $\mathbf{a}_i$s can be selected for example from among the eigenvectors of the sample within-class covariance matrix $\mathbf{S}_W$ to maximize the ratio $\mathbf{a}_i^T\mathbf{S}_B\mathbf{a}_i/\lambda_i$, where $\mathbf{S}_B$ is the sample between-class covariance matrix and λ_i is the eigenvalue corresponding to $\mathbf{a}_i$. Thus,

$$\mathbf{S}_W = \sum_{j=1}^{c} \frac{n_j}{n}\hat{\mathbf{\Sigma}}_j, \qquad \mathbf{S}_B = \sum_{j=1}^{c} \frac{n_j}{n}(\hat{\boldsymbol{\mu}}_j - \hat{\boldsymbol{\mu}})(\hat{\boldsymbol{\mu}}_j - \hat{\boldsymbol{\mu}})^T,$$

where n_j is the number of class j training vectors and $\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\mu}}$ and $\hat{\mathbf{\Sigma}}_j$ are sample versions of the class j mean, the overall mean and the class j sample covariance matrix, respectively, all computed from the original raw training data.

The most widely used dimension reduction technique is principal components analysis. The features are of the form $X_i = \mathbf{u}_i^T\mathbf{Y}$, where $\mathbf{u}_i$ is the ith eigenvector of the covariance matrix of $\mathbf{Y}$ which needs to be estimated from the training data. The X_is are uncorrelated and when the eigenvectors are indexed in an order of decreasing eigenvalues we have that $\mathrm{Var}(X_1) \geq \cdots \geq \mathrm{Var}(X_D)$. By defining a feature vector as $\mathbf{X} = [X_1, \ldots, X_d]^T$, where $d \ll D$, dimension reduction is achieved while at the same time most of the variability in the original pattern vector $\mathbf{Y}$ can be captured. Of course, this approach does not take into account the class labels of the patterns in any way and indeed there are versions of principal component analysis which incorporate this information, too. Another general purpose dimension reduction and feature extraction technique that has found some use in pattern recognition is metric multidimensional scaling.

Good accounts of various feature selection and extraction methods can be found, e.g., in [7] and [27].

Classifier assessment

Once the classifier has been implemented, one should assess whether it meets the design criteria such as being sufficiently quick to compute and having adequate error rate. A naive way of estimating the error rate (assuming mixture sampling) is to calculate the relative frequency of errors in the design sample. This is called the resubstitution estimator or the apparent error rate. It is optimistically biased, and the bias can be severe for complex classifiers. Instead, the recommended approach is to split the data into two separate sets, the training set and the test set. The classifier is estimated using data in the training set, and its performance is assessed on the independent test set. This is called the holdout estimate. Several classifiers require the choice of tuning parameters or model or architecture or kernel selection. This is typically based on cross-validation. In order to keep strict separation between the design and the test set, the cross-validation then needs to be done using the training set only.

Conclusion

Pattern recognition has a long history, starting from the investigations of statisticians in the 1930s and continuing with the groundwork of 1950s and 1960s.

In more recent times, the field has received new impetus especially from the neural networks and machine learning communities.

However exciting the new classification methods may seem, one should keep in mind that the greatest progress usually comes from a careful formulation of the real-world problem and from the invention of good features, not so much from new classification algorithms. This is especially relevant in new application domains.

References

[1] Kernel-machines.org. http://www.kernel-machines.org/.

[2] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons, third edition, 2003.

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

[4] C. H. Chen and P. S. P. Wang, editors. *Handbook of Pattern Recognition and Computer Vision*. World Scientific, 3rd edition, 2005.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, pages 273–297, 1995.

[6] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[7] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.

[8] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2nd edition, 2000.

[10] J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[11] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990. Original ed. published 1972.

[12] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[13] D. J. Hand. *Construction and Assessment of Classification Rules*. Wiley, 1997.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2nd edition, 2001.

[15] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja. Neural and statistical classifiers—taxonomy and two case studies. *IEEE Transactions on Neural Networks*, 8(1):5–17, 1997. doi:10.1109/72.554187.

[16] A. J. Izenman. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer, 2008.

[17] P. R. Krishnaiah and L. N. Kanal, editors. *Handbook of Statistics 2: Classification, Pattern Recognition and Reduction of Dimensionality*. North-Holland, 1982.

[18] W. J. Krzanowski and F. H. C. Marriott. *Multivariate Analysis: Part 2, Classification, Covariance Structures and Repeated Measurements*. Kendall's Library of Statistics, 2. Edward Arnold, 1995.

[19] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[20] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.

[21] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[22] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. Wiley, 1992.

[23] S. Srivastava, M. R. Gupta, and B. A. Frigyik. Bayesian quadratic discriminant analysis. *Journal of Machine Learning Research*, 8:1277–1305, 2007.

[24] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 4th edition, 2008.

[25] C. W. Therrien. *Decision, Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*. Wiley, 1989.

[26] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, 4th edition, 2002.

[27] A. Webb. *Statistical Pattern Recognition*. Wiley, 2nd edition, 2002.

[28] T. Y. Young and T. W. Calvert. *Classification, Estimation and Pattern Recognition*. American Elsevier Publishing Co., Inc., New York, 1974.

[29] T. Y. Young and K.-S. Fu, editors. *Handbook of Pattern Recognition and Image Processing*. Academic Press, 1986.

Cross-References

Discriminant analysis, Classification, Bayesian classification