Chapter 19

Pattern recognition

19.1 Highlights

Recognizing objects in images as members of a certain class is – as many other aspects of image processing and analysis – a truly interdisciplinary problem. On one hand, it requires solid knowledge concerning the application area. The reason is that images are very high-dimensional objects (an $N \times M$ image can be considered a point in an $N \cdot M$-dimensional feature space) and do not lend themselves to direct classification, without any preprocessing or feature extraction reducing noise and dimensionality.

On the other hand, predicting class membership from measured features is a problem that has surfaced in many different disciplines, ranging from speech recognition (which word was uttered?) to industrial quality control (part intact or defect?).

This chapter presents algorithms which "learn" automatically how to predict the true class membership from previously extracted features. This process is known as pattern recognition, discriminant analysis, classification or supervised learning.

Another family of techniques which seek to detect inherent structure in a data set with no class labels goes by the name of unsupervised learning or cluster analysis (or, unfortunately, also by the name of classification). These methods, though sometimes subsumed under the title of pattern recognition, are not treated here.

19.2 Task

In this chapter, we assume that the goal is to predict as well as possible the origin, or the type, or the class membership of an object in an image.

Previous chapters have dealt with the extraction of appropriate features describing, for instance, shape, geometrical or chromatic properties of an object. We further assume that the user has access to a historical database or *training set* of objects and their features, along with their true origin, or type, or class membership.

The following sections discuss a quality criterion for classification, and how a model is validated. In section 19.4, we introduce a very stiff classifier (19.4.1) and a generalization thereof (19.4.2), as well as an extremely flexible classifier (19.4.3), and we discuss their respective merits. Section 19.4.4 gives guidance on the informed choice of features or classification methods.

19.3 Concepts

Objects can be characterized in many ways; in particular, their features can be measured on the nominal scale, the ordinal scale or the interval scale [1]. For simplicity, we will assume that a total of P features have been measured on the continuous scale. Assuming that our training set contains a total of N objects, we can interpret these as N points in a P-dimensional feature space. If P is two or three, it is easy to visualize the data in a scatter plot. The class membership could then be illustrated using, for instance, different colors; in a quality control problem, one might choose to color those parts that have passed a test green and the others red.

We now seek to parametrize an algorithm such that it takes the measured features of an object as input and gives an estimate or prediction of the true class membership as output. Such an algorithm is a *classifier*. The more modest term "estimate" already indicates that this may not always be possible without error. The frequency of false predictions depends on the degree of overlap of the two classes in feature space: if the two classes from the training set are clearly separated, and if the examples in the training set are representative for the process under investigation, a prediction with no or little error can be expected.

If, on the other hand, the two classes overlap severely, and if the algorithm does not have access to information other than the objects' position in feature space, it cannot be expected to make accurate predictions.

In a region of feature space in which two or more classes overlap, the algorithm's estimate should depend on the relative seriousness of the consequences of a wrong prediction. For argument's sake, assume you need to set up an automatic quality control for a small but vital part; assume, in addition, that the costs incurred by delivering a faulty part are very high. In this case, the algorithm should take a "conservative" stance: if there is only

a trace of doubt, that is, if an object lies in a part of feature space in which the faulty class has some density (i.e., the training set contained a few faulty objects lying in that region of feature space), it is safer to assume the part is faulty and either discard it or subject it to closer scrutiny.

	pos. prediction	neg. prediction
truth: pos.	true positive	false negative
truth: neg.	false positive	true negative

Table 19.1: Loss matrix: diagonal elements are typically set to zero, off-diagonal elements to positive constants which can differ. An optimal classifier minimizes the expected loss.

The relative costs incurred by the decision process can be coded in a *loss matrix*. Typically, correct predictions (true positives, true negatives) incur zero loss, whereas false predictions (false positives, false negatives) are undesirable; as was illustrated above, the seriousness of wrong decisions can be unequal, and the off-diagonal elements in the loss matrix can be chosen to reflect this imbalance.

We stated above that the aim of classification was to "predict as well as possible" the class of an object. We can now sharpen this concept, by choosing to minimize the *expected loss* of a classifier. The idea is to choose the parameters in an algorithm such that, when an infinite number of training samples are drawn randomly and classified and the loss of each single prediction is looked up in the loss matrix and added up, the sum of all these losses becomes minimal.

The resultant classifier will be the *best possible* for this choice of problem and loss matrix. It is also called the Bayes classifier. Details can be found e.g. in [2]. In the following and for simplicity, we will assume that false positive and false negative predictions are equally bad, and that correct predictions incur no loss; in short, we choose the loss matrix to be of the form $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. We state here without proof that for this specific choice of loss function, the best possible or Bayes classifier is the one which predicts, at each point in feature space, that class which has the highest density at that point [2]. Unfortunately, this statements shifts our problem without solving it; for we do not know the true class densities. We will, in the procedures section, introduce two methods: linear discriminant analysis, discussed in section 19.4.1, seeks to estimate the densities and relative frequencies of the classes; whereas k-nearest neighbors, discussed in 19.4.3, aims to directly predict the locally dominant class.

19.3.1 Model optimization and validation

The solution of a classification problem requires a number of choices. The first of these is the proper choice of features, and this is of paramount importance! For what information is lost in the feature extraction can never be compensated for later on, no matter how sophisticated the machinery. We stated above that classification results are best if two classes have little or no overlap in feature space. Accordingly, it makes sense to include (combinations of) features that separate the classes as well as possible. Note that this separation need not be linear, all that is required is that a scatter plot would reveal, say, the "green" and the "red" areas to be as distinct as possible. "Then why", may you ask, "not include all possible features, just to be on the safe side?" The reason is that the effective number of parameters that need be determined in an algorithm grows with the dimensionality of the feature space. When the training set is small, there may not be enough data to reliably determine all these extra parameters. As a consequence, it is best to choose a small subset of all conceivable features, that subset which allows for the best discrimination. If subsets with only two members are sought, it is possible to produce scatter plots of the training set in this representation. While this may become tedious if the set of candidate features is large (for P candidate features, you would need to consider $P(P-1)/2$ scatter plots), it becomes outright impossible if the subset sought should comprise more than three features: straightforward visualization in a scatter plot is no longer feasible. In this case, an automatic selection is desirable, which will be discussed below.

Another choice required is that of the capacity, complexity, or flexibility of the classifier. Some classifiers allow for an explicit tuning of their complexity, e.g. neural nets with their variable number of units in the hidden layer. Finally, there is the choice between different classifiers; modern software packages for the statistical analysis of data (such as [3]) offer a wide range of classification methods and it is, unfortunately, impossible to ascertain the general superiority of a specific method.

In summary, there is a number of choices which it would be desirable to automate and render more objective. If no additional constraints such as easy interpretability of the algorithm or meagre computational resources exist, one possibility is to take the expected loss (see above) as only criterion.

Now, in practice, we do not have an infinite training set. One way forward is to split the available training set as indicated in figure 19.1. The performance on the validation set serves as guide in the parametrization of the algorithm; whereas the classification performance on the test set gives an estimate for the method's performance on unknown data.

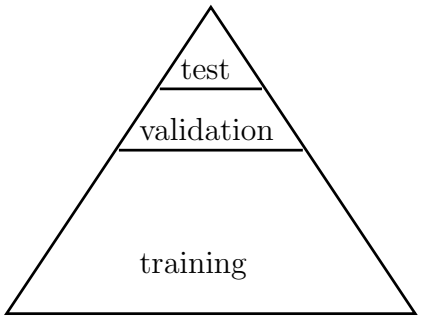


Figure 19.1: If the database is sufficiently large, it can be split into training, validation and test sets. The latter should be used only once, for a final evaluation before the algorithm is deployed.

While honest, this method makes poor use of the data; frequently, experiments are so expensive or tedious that the historical dataset is small, and it would be wasteful not to use all of the data to improve the method. On the other hand, it is a very bad idea to use all of the data for training with no external check; the 1-nearest neighbor method discussed below will make zero errors no matter how large or complex the training set, i.e., it is a perfect fit to the data. However, due to overfitting, generalization performance on unseen data can be poor. Generally speaking, the performance on the training set is better than the true performance, because the method is optimized to do as well as possible on the training set. The difference between the empirical error on the training set and the true error can be called optimism, and it grows with the flexibility of a classifier.

Besides certain analytical schemes (such as Akaike's Information Criterion, AIC [2]) which seek to avoid overfitting, there are a number of *resampling techniques*, such as *cross-validation* or bootstrapping [4]. In these schemes, the data is repeatedly partitioned into training and test sets, see section 19.4.4.

19.4 Procedures

19.4.1 Linear discriminant analysis (LDA)

In this approach, each class is modeled using a multivariate normal (i.e., P-dimensional Gaussian) distribution. Formally, each class k is assumed to

have density p at position x in feature space:

$$p(x|k) = \frac{1}{((2\pi)^P|\Sigma_k|)^{\frac{1}{2}}} \times \exp\left\{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right\}$$

In this expression, Σ is the covariance matrix describing the shape of a normal distribution, and μ is its center of mass. The diagonal elements of the covariance matrix (see section **??**) are the variances. These indicate the spread of the distribution along the various coordinate axes. The off-diagonal elements indicate the amount of correlation: a spherically symmetric distribution has zero covariances; whereas one which is extended and "points" in a direction other than one of the coordinate axes is characterized by positive or negative correlations, see figure 19.2.

We stated that the best possible classifier is the one predicting the locally most dominant class. Assuming that we seek to solve a two-class problem ($k \in \{1,2\}$), the discrimination surface separating these two classes must then be the set of all points at which the classes have the same probability. In other words, the discrimination surface is the solution x of

$$p(1|x) = p(2|x) \tag{19.1}$$

$$\frac{p(x|1)\pi(1)}{p(x)} = \frac{p(x|2)\pi(2)}{p(x)} \tag{19.2}$$

where the transition from the first equation to the second is by Bayes theorem, $p(k|x)$ is the posterior density of class k at location x, $p(x|k)$ is the density of class k at x and $\pi(k)$ is the prior probability, or relative frequency of occurrence of class k, with $\sum_k \pi(k) = 1$.

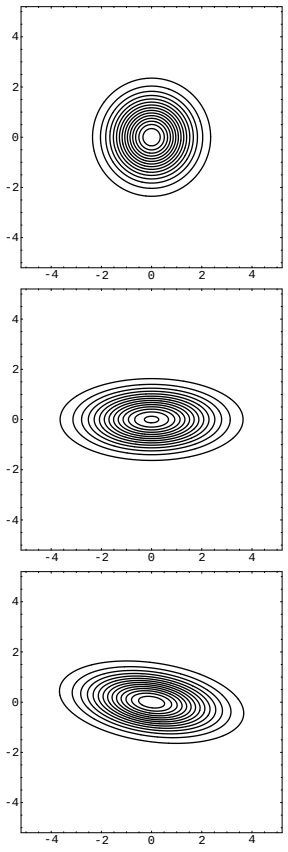
If the two classes are described by the same covariance matrix $\Sigma_1 = \Sigma_2$, then the quadratic terms in x cancel and what remains is a (hyper-) plane, that is, a straight line in two dimensions, a plane in three dimensions, etc. This property has inspired the name of *linear* discriminant analysis.

The estimation of the paramaters involved is simple: the vectors μ_k are given by the arithmetic mean or center of mass of all the samples from class k

$$\mu_k = \frac{1}{N_k}\sum_{i=1}^{N_k} x_i^k \tag{19.3}$$

where x_i^k is the ith sample out of a total of N_k examples of class k in the training set. The relative frequencies are simply given by $\pi_k = N_k/(N_1+N_2)$. The pooled covariance matrix is computed by

$$\Sigma = \frac{1}{N_1+N_2}\left(\sum_{i=1}^{N_1}\left(x_i^1-\mu_1\right)\left(x_i^1-\mu_1\right)^T + \sum_{i=1}^{N_2}\left(x_i^2-\mu_2\right)\left(x_i^2-\mu_2\right)^T\right) \tag{19.4}$$



$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

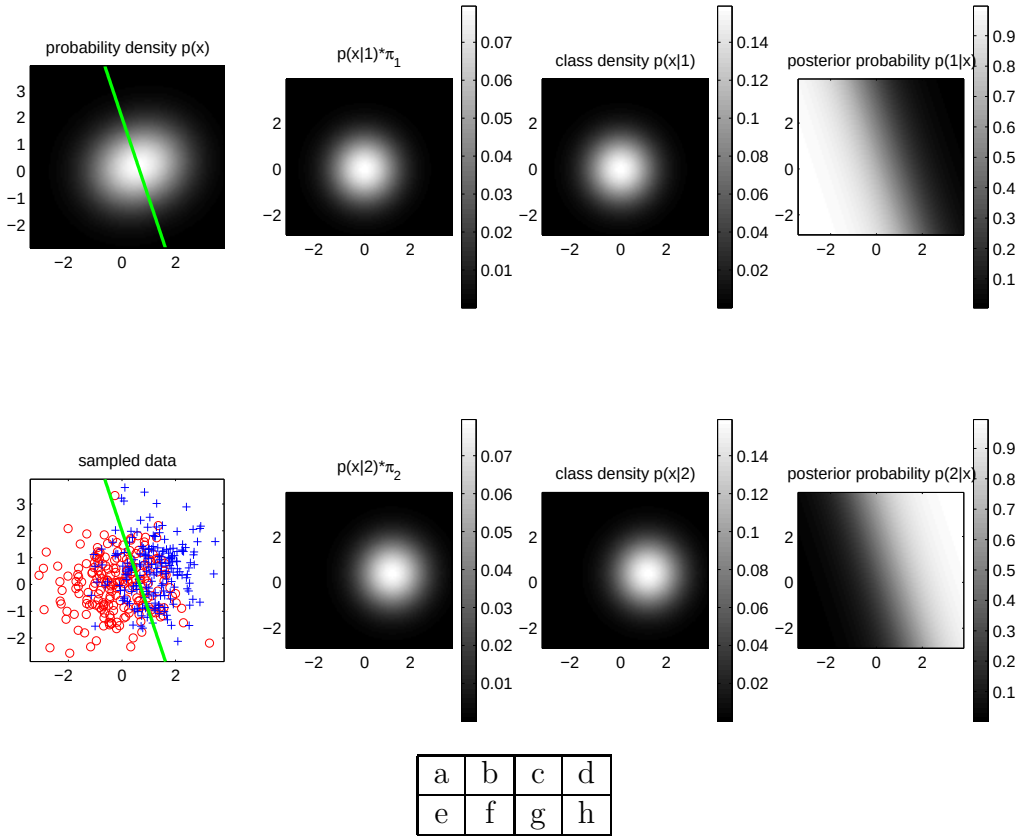
$$\begin{bmatrix} 2.5 & -0.3 \\ -0.3 & 0.5 \end{bmatrix}$$

Figure 19.2: Contour plots of the densities of two-dimensional normal distributions along with their covariance matrices. The first density is spherical or isotropic, the second has a diagonal covariance matrix, and the third shows some anti-correlation: a positive value of the first component tends to go together with a negative value of the second component and vice versa.

where the superscripts on x again indicate class membership.

We now illustrate linear discriminant analysis on two training sets: figure 19.3 shows what kind of data LDA is suitable for (in fact, the data has in this case been sampled from the model), while figure 19.4 illustrates a case for which LDA is not appropriate.

Starting with figure 19.3, panels c) and g) give two isotropic class densities; these are identical except for their location. Panels b) and f) show these densities multiplied by their prior probabilities which were chosen as $\pi(1) = \pi(2) = 0.5$. Panel a) illustrates the total density $p(x) = p(x|1)\pi(1) + p(x|2)\pi(2)$. Panels d) and h) give the posterior probabilities of the classes at each point in this two-dimensional feature space; they can be computed from



a	b	c	d
e	f	g	h

Figure 19.3: Linear discriminant analysis as a generative model: data has been sampled from the model. See text for a detailed description of the panels.

the information in the other panels by application of Bayes' theorem, equations 19.1, 19.2. The set of all points at which the two posterior probabilities are equally large is indicated by the straight line in panels a) and e). This is the linear discriminant function. Since it was computed from the exact densities, it is in this case the best possible or Bayes classifier. Finally, panel e) shows a set of points which have been sampled from this model. Note that there is significant overlap of the two classes and thus a significant number of prediction errors even for the best possible classifier. On an intuitive level, the discriminant function seems appropriate for the data.

Figure 19.4 shows a training set in panel e), along with the linear discriminant model obtained by estimating parameters according to equations 19.3, 19.4. In this example, the assumptions inherent in linear discriminant analysis do not match the data: one of the classes is much more compact,

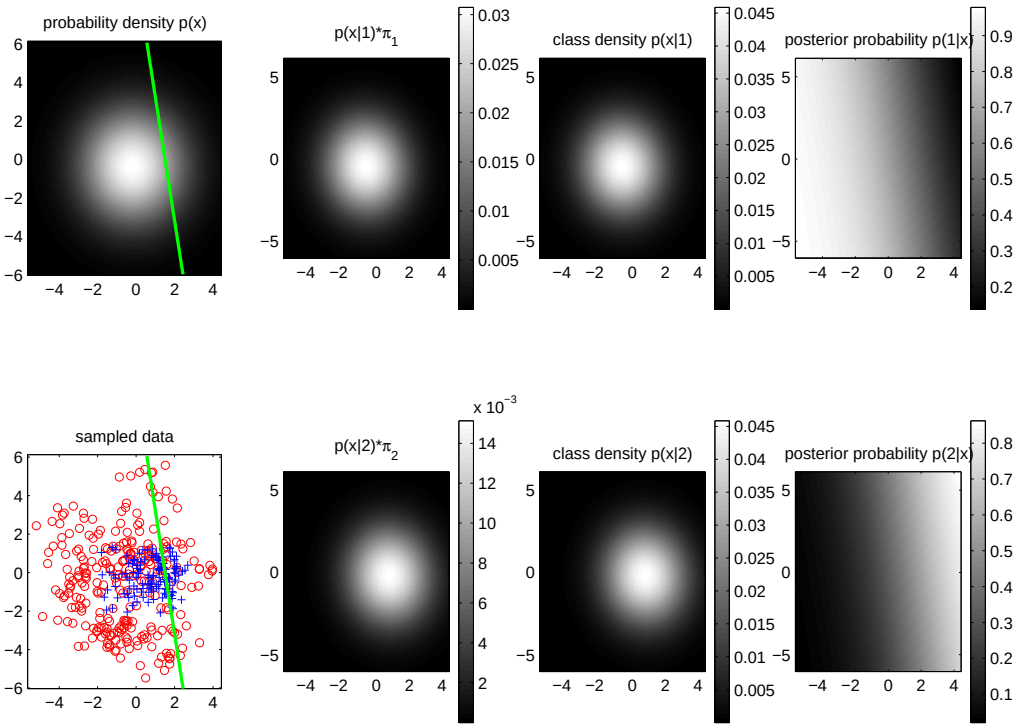


Figure 19.4: Panel e) shows a given training set, and the LDA classifier fit to it. Other panels show the estimated class densities, total density, and posterior densities, cf. description of figure 19.3 in text. Clearly, the linear classifier does not do the data justice in this example.

such that one covariance matrix Σ cannot adequately describe both classes. The discriminant function obtained is non-sensical. Apparently, we need to relax some of our assumptions in order to give the model more flexibility and allow for nonlinear classifiers.

19.4.2 Quadratic discriminant analysis (QDA)

As in linear discriminant analysis, each class is assumed to follow a multivariate normal distribution. The important difference is that the classes are now allowed to have different covariance matrices, and these are estimated separately for each class. As a consequence, the term quadratic in x does not cancel anymore in equation 19.2 and the resultant discriminant functions are quadrics, i.e. parabolas, hyperbolas, etc. Figure 19.5 shows an example.

Since QDA offers added flexibility and contains LDA as a special case, why not use it always? We had asked a similar question in section 19.3.1, and

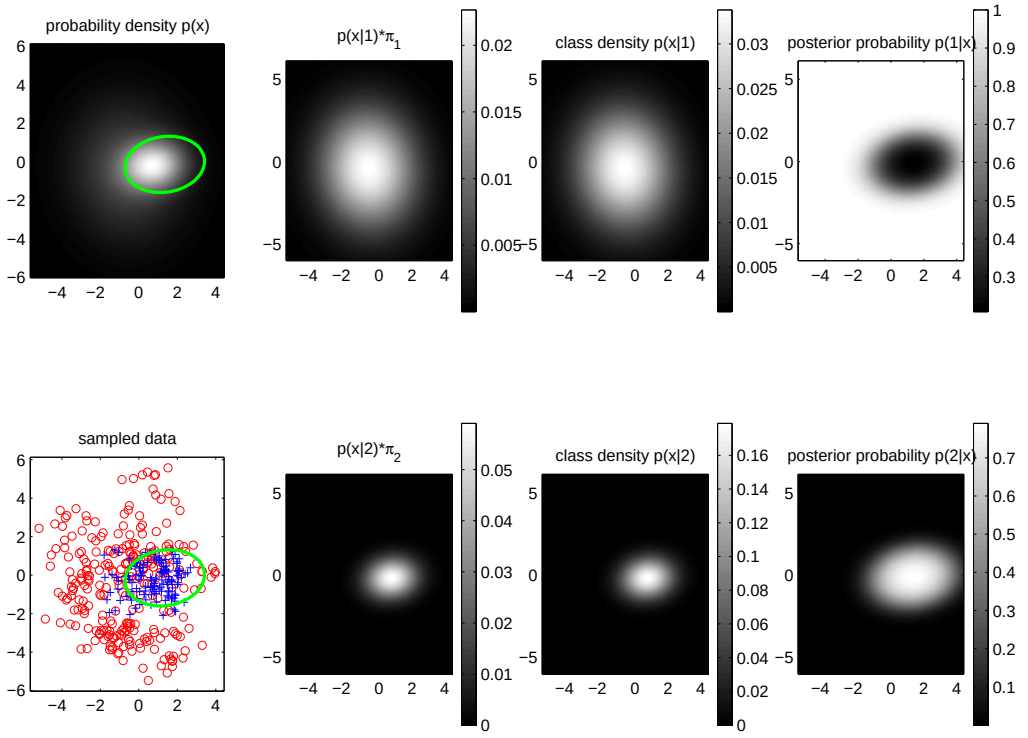


Figure 19.5: Panel e) shows the same training set as in figure 19.4, and the QDA classifier fit to it. Other panels show the estimated class densities, total density, and posterior densities. This non-linear classifier offers a much improved fit to the data, and promises superior classification performance.

our answer here is similar: QDA requires determination of more parameters, and this endeavor becomes difficult if the training set is small compared to the dimensionality P of the feature space. Which of these two should then be used? As always, cross-validation (see section 19.4.4) offers a convenient answer. Let us add that it is possible to "compromise" [4] between LDA and QDA by using a covariance matrix of the form $\Sigma_k^{\mathrm{mod}}(\alpha) = \alpha\Sigma_k + (1 - \alpha)\Sigma, \quad 0 \leq alpha \leq 1$ where Σ_k are estimates of the class-covariance matrices and Σ is an estimate of the pooled covariance matrix, cf. equation 19.4.

If even linear discriminant analysis has too many parameters compared to the size of the training set, further simplifications are possible: all off-diagonal elements of the covariance matrix may be set to zero, or the entire covariance matrix may be biased towards sphericity using $\Sigma^{\mathrm{mod}}(\beta) = \beta\Sigma + \sigma^2(1 - \beta)I, \quad 0 \leq \beta \leq 1$ with I the unit matrix and σ^2 the variance of the data [4]. Again, an optimal value of β can be found using cross-validation.

This modification of the estimated covariance may be necessary in the case of very high-dimensional data. For the sake of argument, assume that you compute a large number P of features which are thought to be relevant. N examples in feature space can only span an $(N-1)$-dimensional subspace. If $N-1 < P$, then the covariance matrix becomes singular and a stabilization is required.

19.4.3 k-nearest neighbors (k-NN)

The methods discussed above make strong assumptions concerning the distribution of the data, and yield succinctly formulated models. k-nearest neighbor methods lie at the other extreme in that they make very little assumptions, but are difficult to formulate in a concise manner: these memory-based methods "grow" with the size of the training set.

The basic idea is simple: for each new sample which should be classified, find that example from the training set which lies closest in feature space. Give the label of that closest sample as prediction.

An obvious generalization is to search not only for the closest sample, but for the closest k[1] , k uneven, samples from the training set. A "democratic vote" then reveals the locally dominant class which is given as prediction. Taking this vote with a larger number of neighbors gives the result with greater certainty (the change in the classifier when going from one training set to another will be smaller), but it also involves averaging over a larger region, making the discrimination surface smoother, and maybe overly so. A suitable value of k can again be found by cross-validation, see section 19.4.4.

It may appear as if the method had only one parameter, k. This is far from the truth: the effective number of parameters is much larger, up to N/k [4]. One justification of this view is to consider 1-NN: the training set partitions the feature space into influence regions of each sample (these are the Voronoi regions). Each sample transmits its class membership to that region, thus each individual region has one parameter.

Since Euclidean distance (or its square, which is computationally cheaper) is used for identifying the k nearest neighbors, proper scaling of the feature axes is vital: if the units on one axis are converted, say, from meters to millimeters, the classification results may change drastically. This is in contrast to LDA which can correct affine distortions of the entire training set, such as scaling, "by itself" through the estimation of the covariance structure.

An advantage of k-NN is its flexibility and ease of use, disadvantages are

[1] In previous sections, k was a class index; now it has become an integer denoting the number of nearest neighbors that are used for prediction.

the computational burden in routine use if the available training set is large, and the difficult interpretation of this "model".

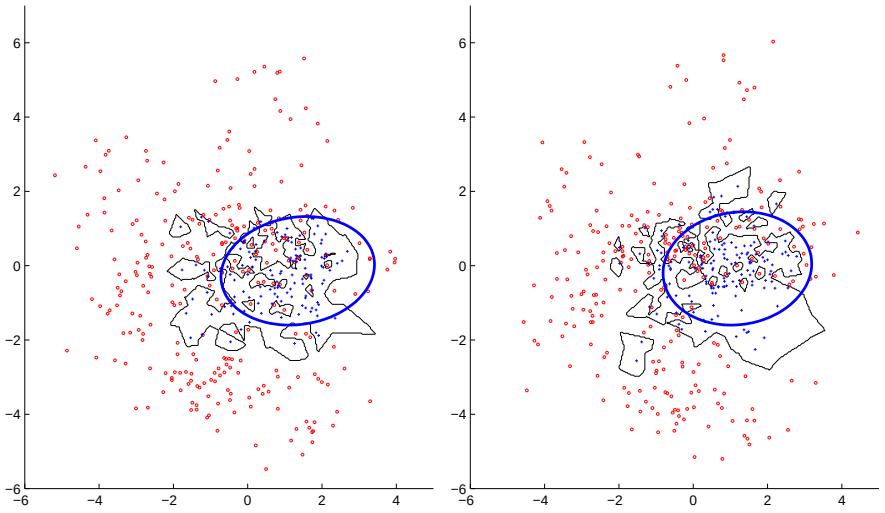


Figure 19.6: The left panel shows the same training set as in figures 19.4, 19.5. The right panel shows another training set which was sampled from the same underlying distribution. The ellipses are from a quadratic discriminant analysis on these data sets, the contours come from a 1-NN analysis. The two ellipses are quite similar, whereas the 1-NN classifier varies quite a lot from one training set to the next. This is what statisticians call the *variance*, and the severity of this undesirable feature generally increases with the flexibility of a classifer. In turn, overly stiff classifiers as illustrated in figure 19.4 cannot always do the data justice, leading to large *bias* [4].

Overall, the classification schemes presented here, though simple, are among the most performant [5]. If results prove unsatisfactory, advanced methods such as support vector machines [6, 7] may be applied, but generally speaking, no quantum leap in performance should be expected. In many practical situations, improper selection, evaluation or transformation (taking the logarithm, the inverse, etc.) of features is the weakest link in the chain and should be amended first.

19.4.4 Cross-validation

Cross-validation can be used to estimate the predictive power of different methods when there is not enough data to split it in the way indicated in figure 19.1. In this context, LDA, QDA, 5-NN and 7-NN would all qualify

as "different methods", and so would LDA based on two different sets of features (one of which may be a subset of the other). The complete available database is split randomly[2] into K fractions of equal size. A specific method is then parametrized using all but one of these fractions as the training set. The classifier is then used to predict the labels of the samples in the fraction that had been left out, and the predicted labels are compared to the true ones. The error rate is stored. This procedure of taking one subset out while training on all others is repeated for each of the K fractions, and the mean error rate is computed. The method with the smallest mean error rate can be expected to yield the greatest prediction performance.

There is a trade-off concerning the size of the fractions: on one hand, the smallest possible amount of data should be left out in order not to "waste" it; on the other hand, leaving out a small number of samples corresponds to a small perturbation of the complete training set only. This in turn means that results may differ if a totally different training set from the same process is used. A good compromise seems to be to choose K of the order $5, \ldots, 10$. Another frequent choice is $K = N$, the number of samples, which is also called "leave-one-out-cross-validation".

There are many variations on this theme, summarized under the term *resampling methods*. One enjoying much popularity recently is the *bootstrap* [4] in which the fractions are chosen by sampling from the complete training set with replacement.

In summary, cross-validation has a number of beneficial properties which were not discussed here, and is easy to implement and use. While often computationally prohibitive in the past, it is a method that merits its popularity nowadays.

19.5 Advanced Reference Material

There are a number of very accessible textbooks today, two of which we would like to single out. [8] is a bit more verbose and possibly more recommendable for absolute beginners; it also features a very well-structured introduction to clustering methods. [4] puts more emphasis on recent developments, though its derivations are sometimes not very detailed. [2] is a fine introduction to the statistical foundations of pattern recognition, but requires more background knowledge than the other two.

Throughout this chapter, reference has been made to these three textbooks where appropriate, rather than to the original articles which are mostly more difficult.

[2] It is important to really choose samples randomly to avoid biasing by temporal trends.

Bibliography

[1] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Monographs on Statistics and Applied Probability. Chapman & Hall, 1995.

[2] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge Univ. Press, 1997.

[3] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *J. Comput. Graph. Stat.*, 5(3):299–314, 1996. http://www.r-project.org/.

[4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, 2001.

[5] S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 2002.

[6] B. Schölkopf and A. J. Smola. *Learning with kernels*. MIT Press, Boston, 2002.

[7] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.

[8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2000.