We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

4

Methods for Pattern Classification

Yizhang Guan
South China University of Technology
China

1. Introduction

Pattern classification is to classify some object into one of the given categories called classes. For a specific pattern classification problem, a classifier is computer software. It is developed so that objects (x) are classified correctly with reasonably good accuracy. Through training using input-output pairs, classifiers acquire decision functions that classify an input datum into one of the given classes (ω_i). In pattern recognition applications we rarely if ever have the prior probability $P(\omega_i)$ or the class-conditional density $p(x|\omega_i)$. of complete knowledge about the probabilistic structure of the problem. In a typical case we merely have some vague, general knowledge about the situation, together with a number of design samples or training data—particular representatives of the patterns we want to training classify. The problem, then, is to find some way to use this information to design or data train the classifier.

The organization of this chapter is to address those cases where a great deal of information about the models is known and to move toward problems where the form of the distributions are unknown and even the category membership of training patterns is unknown. We begin in Bayes decision theory(Sec.2) by considering the ideal case in which the probability structure underlying the categories is known perfectly. In Sec.3(Maximum Likelihood) we address the case when the full probability structure underlying the categories is not known, but the general forms of their distributions are the models. Thus the uncertainty about a probability distribution is represented by the values of some unknown parameters, and we seek to determine these parameters to attain the best categorization. In Sec.4(Nonparametric techniques)we move yet further from the Bayesian ideal,and assume that we have no prior parameterized knowledge about the underlying probability structure;in essence our classification will be based on information provided by training samples alone. Classic techniques such as the nearest-neighbor algorithm and potential functions play an important role here. We then in Sec.5(Support Vector Machine) Next, in Sec.6(Nonlinear Discriminants and Neural Networks)we see how some of the ideas from such linear discriminants can be extended to a class of very powerful algorithms such as backpropagation and others for multilayer neural networks; these neural techniques have a range of useful properties that have made them a mainstay in contemporary pattern recognition research. In Sec.7(Stochastic Methods)we discuss simulated annealing by the Boltzmann learning algorithm and other stochastic methods. We explore the behaviour of such algorithms with regard to the matter of local minima that can plague other neural methods. Sec.8(Unsupervised Learning and

Clustering),by addressing the case when input training patterns are not labelled, and that our recognizer must determine the cluster structure.

2. Bayesian Decision Theory

Suppose that we know both the prior probabilities $P(\omega_j)$ and the conditional densities $p(x \mid \omega_j)$. Suppose further that we measure the features of a sample and discover that its value is x. How does this measurement influence our attitude concerning the true state of nature—that is, the category of the input? We note first that the(joint) probability density of finding a pattern that is in category ω_j and has feature value x can be written in two ways: $P(\omega_j, x) = P(\omega_j \mid x)p(x) = p(x \mid \omega_j)P(\omega_j)$. Rearranging these leads us to the answer to our question, which is called Bayes formula:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)} \tag{1}$$

where in this case of c categories

$$p(x) = \sum_{j=1}^{c} p(x \mid \omega_j)P(\omega_j) \tag{2}$$

2.1 Two-Category Classification

If we have an observation x for which $P(\omega_1 \mid x)$ is greater than $P(\omega_2 \mid x)$, we would naturally be inclined to decide that the true state of nature is ω_1. Similarly, if $P(\omega_2 \mid x)$ is greater than $P(\omega_1 \mid x)$, we would be inclined to choose ω_2. Thus we have justified the following Bayes decision rule for minimizing the probability of error:

$$Decide \;\; \omega_1 \;\; if \;\; P(\omega_1 \mid x) > P(\omega_2 \mid x), \;\; otherwise \;\; decide \;\; \omega_2 \tag{3}$$

In Eq. (1), $p(x)$ is a scale factor and unimportant for our problem. By using Eq.(1), we can instead express the rule in terms of the conditional and prior probabilities. And we notice $P(\omega_1 \mid x) + P(\omega_2 \mid x)) = 1$. By eliminating this scale factor, we obtain the following completely equivalent decision rule:

$$Decide \;\; \omega_1 \;\; if \;\; p(x \mid \omega_1)P(\omega_1) > p(x \mid \omega_2)P(\omega_2), \;\; otherwise \;\; decide \;\; \omega_2 \tag{4}$$

While the two-category case is just a special instance of the multi-category case, it has traditionally received separate treatment.Indeed,a classifier that places a pattern in one of only two categories has a special name—a dichotomizer. Instead of using two dichotomizer discriminant functions g_1 and g_2 and assigning x toω1 if $g_1 > g_2$, it is more common to define a single discriminant function

$$g(x) = g_1(x) - g_2(x) \tag{5}$$

and to use the following decision rule:

$$Decide\ \ \omega_1\ \ if\ \ g(x) > 0,\ \ otherwise\ \ decide\ \ \omega_2$$

Thus, a dichotomizer can be viewed as a machine that computes a single discriminant function $g(x)$, and classifies x according to the algebraic sign of the result. Of the various forms in which the minimum-error-rate discriminant function can be written, the following two(derived from Eqs.(1)&(5)are particularly convenient:

$$g(x) = P(\omega_1 \mid x) - P(\omega_2 \mid x) \tag{6}$$

$$g(x) = \ln \frac{p(x \mid \omega_1)}{p(x \mid \omega_2)} + \ln \frac{p(\omega_1)}{p(\omega_2)} \tag{7}$$

2.2 Multi-Category Classification

Let $\omega_1, \cdots, \omega_c$ be the finite set of c states of nature. Let the feature vector x be a d-component vector-valued random variable, and let $P(x \mid \omega_j)$ be the state- conditional probability density function for x —the probability density function for x conditioned on ω_j being the true state of nature. As before, $P(\omega_j)$ describes the prior probability that nature is in state ω_j. Then the posterior probability $P(\omega_j \mid x)$ can be computed from $p(x \mid \omega_j)$ by Bayes formula:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)} \tag{8}$$

where the evidence is now

$$p(x) = \sum_{j=1}^{c} p(x \mid \omega_j) P(\omega_j) \tag{9}$$

A Bayes classifier is easily and naturally represented in this way. For the minimum-error-rate case, we can simplify things further by taking gi(x)=P(ωi | x),so that the maximum discriminant function corresponds to the maximum posterior probability.

Clearly, the choice of discriminant functions is not unique. We can always multiply all the discriminant functions by the same positive constant or shift them by the same additive constant without influencing the decision. More generally, if we replace every $g\ (x)$ by $f(g\ (x))$, where $f(\cdot)$ is a monotonically increasing function, the resulting classification is unchanged. This observation can lead to significant analytical and computational simplifications. In particular, for minimum-error-rate classification, any of the following choices gives identical classification results, but some can be much simpler to understand or to compute than others:

$$g\ (x) = P(\omega_i \mid x) = \frac{p(x \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(x \mid \omega_j)P(\omega_j)} \tag{10}$$

$$g\ (x) = P(\omega_i \mid x) = p(x \mid \omega_i)P(\omega_i) \tag{11}$$

$$g\ (x) = P(\omega_i \mid x) = \ln p(x \mid \omega_i) + \ln P(\omega_i) \tag{12}$$

where ln denotes natural logarithm.

3. Maximum-likelihood Method

It is important to distinguish between supervised learning and unsupervised learning. In both cases, samples x are assumed to be obtained by selecting a state of nature ω_i with probability $P(\omega_i)$,and then independently selecting x according to the probability law $p(x \mid \omega_i)$. The distinction is that with supervised learning we know the state of nature(class label)for each sample, whereas with unsupervised learning we do not. As one would expect, the problem of unsupervised learning is the more difficult one. In this section we shall consider only the supervised case, deferring consideration of unsupervised learning to Section 8.

The problem of parameter estimation is a classical one in statistics, and it can be approached in several ways. We shall consider two common and reasonable procedures, maximum likelihood estimation and Bayesian estimation. Although the results obtained with these two procedures are frequently nearly identical, the approaches are conceptually quite different. Maximum likelihood and several other methods view the parameters as quantities whose values are fixed but unknown. The best estimate of their value is defined to be the one that maximizes the probability of obtaining the samples actually observed. In contrast, Bayesian methods view the parameters as random variables having some known a priori distribution. Observation of the samples converts this to a posterior density, thereby revising our opinion about the true values of the parameters. In the Bayesian case, we shall see that a typical effect of observing additional samples is to sharpen the a posteriori density function, causing it to peak near the true values of the parameters. This phenomenon is known as Bayesian learning. In either case, we use the posterior densities for our classification rule, as we have seen before.

3.1 Maximum Likelihood

Maximum likelihood estimation methods have a number of attractive attributes. First, they nearly always have good convergence properties as the number of train- ing samples increases. Further, maximum likelihood estimation often can be simpler than alternate methods, such as Bayesian techniques or other methods presented in subsequent section.

Suppose that we separate a collection of samples according to class, so that we have c sets, $D_1,\cdots,D_c$, with the samples in D_i having been drawn independently according to the probability law $p(x \mid \omega_i)$. We say such samples are i.i.d.—independent identically distributed random variables. We assume that $p(x \mid \omega_i)$ has a known parametric form, and is therefore determined uniquely by the value of a parameter vector θ_i . For example, we

might have $p(x|\omega_i) \sim N(\mu_i, \Sigma_i)$, where θ_i consists of the components of μ_i and Σ_i. To show the dependence of $p(x|\omega_i)$ on θ_i explicitly, we write $p(x|\omega_i)$ as $p(x|\omega_i, \theta_i)$. Our problem is to use the information provided by the training samples to obtain good estimates for the unknown parameter vectors $\theta_1, \cdots, \theta_c$ associated with each category.

To simplify treatment of this problem, we shall assume that samples in D_i give no information about θ_i if $i = j$ —that is, we shall assume that the parameters for the different classes are functionally independent. This permits us to work with each class separately, and to simplify our notation by deleting indications of class distinctions. With this assumption we thus have c separate problems of the following form: Use a set D of training samples drawn independently from the probability density p(x|θ)to estimate the unknown parameter vector θ.

Suppose that D contains n samples, $x_1, \cdots, x_n$. Then, since the samples were drawn independently, we have

$$p(D|\theta) = \prod_{k=1}^{n} p(x_k|\theta) \tag{13}$$

Viewed as a function of θ, $p(D|\theta)$ is called the likelihood of θ with respect to the set of samples. The maximum likelihood estimate of θ is, by definition, the value θ that maximizes $p(D|\theta)$. Intuitively, this estimate corresponds to the value of θ that in some sense best agrees with or supports the actually observed training samples.

For analytical purposes, it is usually easier to work with the logarithm of the likelihood than with the likelihood itself. Since the logarithm is monotonically increasing, the θ that maximizes the log-likelihood also maximizes the likelihood. If $p(D|\theta)$ is a well behaved, differentiable function of θ, θ can be found by the standard methods of differential calculus. If the number of parameters to be set is p, then we let θ denote the p-component vector $\theta = (\theta_1, \cdots, \theta_p)^T$, and ∇ be the gradient operator

$$\nabla_\theta = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_p} \end{bmatrix} \tag{14}$$

We define $L(\theta)$ as the log-likelihood function?

$$L(\theta) = \ln p(D|\theta) \tag{15}$$

We can then write our solution formally as the argument θ that maximizes the log-likelihood, i.e.,

$$\hat{\theta} = \arg\max_{\theta} L(\theta) \tag{16}$$

where the dependence on the data set D is implicit. Thus we have from Eq.(13)

$$L(\theta) = \sum_{k=1}^{n} \ln p(x_k \mid \theta) \tag{17}$$

and

$$\nabla_\theta L = \sum_{k=1}^{n} \nabla_\theta \ln p(x_k \mid \theta) \tag{18}$$

Thus, a set of necessary conditions for the maximum likelihood estimate for θ can be obtained from the set of p equations

$$\nabla_\theta L = 0 \tag{19}$$

A solution θ to Eq.(19) could represent a true global maximum, a local maximum or minimum, or(rarely)an inflection point of $L(\theta)$. One must be careful, too, to check if the extremum occurs at a boundary of the parameter space, which might not be apparent from the solution to Eq.(19). If all solutions are found, we are guaranteed that one represents the true maximum, though we might have to check each solution individually(or calculate second derivatives)to identify which is the global optimum. Of course, we must bear in mind that θ is an estimate; it is only in the limit of an infinitely large number of training points that we can expect that our estimate will equal to the true value of the generating function.

3.2 Bayesian estimation

We now consider the Bayesian estimation or Bayesian learning approach to pattern classification problems. Although the answers we get by this method will generally be nearly identical to those obtained by maximum likelihood, there is a conceptual difference: whereas in maximum likelihood methods we view the true parameter vector we seek, θ, to be fixed, in Bayesian learning we consider θ to be a random variable, and training data allows us to convert a distribution on this variable into a posterior probability density.

The computation of the posterior probabilities $P(\omega_i \mid x)$ lies at the heart of Bayesian classification. Bayes formula allows us to compute these probabilities from the prior probabilities $P(\omega_i)$ and the class-conditional densities $p(x \mid \omega_i)$, but how can we proceed when these quantities are unknown? The general answer to this question is that the best we can do is to compute $P(\omega_i \mid x)$ using all of the information at our disposal. Part of this information might be prior knowledge, such as knowledge of the functional forms for unknown densities and ranges for the values of unknown parameters. Part of this information might reside in a set of training samples. If we again let D denote the set of samples, then we can emphasize the role of the samples by saying that our goal is to compute the posterior probabilities $P(\omega_i \mid x, D)$. From these probabilities we can obtain the Bayes classifier.

Given the sample D, Bayes formula then becomes

$$P(\omega_i \mid x, D) = \frac{p(x \mid \omega_i, D)P(\omega_i \mid D)}{\sum_{j=1}^{c} p(x \mid \omega_j, D)P(\omega_j \mid D)} \tag{20}$$

As this equation suggests, we can use the information provided by the training samples to help determine both the class-conditional densities and the a priori probabilities.
Although we could maintain this generality, we shall henceforth assume that the true values of the a priori probabilities are known or obtainable from a trivial calculation; thus we substitute $P(\omega_i) = P(\omega_i \mid D)$. Furthermore, since we are treating the supervised case, we can separate the training samples by class into c subsets $D_1, \cdots, D_c$ with the samples in D_i belonging to ω_i. As we mentioned when addressing maximum likelihood methods, in most cases of interest(and in all of the cases we shall consider), the samples in D_i have no influence on $p(x \mid \omega_j, D)$ if $i = j$. This has two simplifying consequences. First, it allows us to work with each class separately, using only the samples in D_i to determine $p(x \mid \omega_i, D)$. Used in conjunction with our assumption that the prior probabilities are known, this allows us to write Eq. 23 as

$$P(\omega_i \mid x, D) = \frac{p(x \mid \omega_i, D_j)P(\omega_i)}{\sum_{j=1}^{c} p(x \mid \omega_j, D_j)P(\omega_j)} \tag{21}$$

Second, because each class can be treated independently, we can dispense with needless class distinctions and simplify our notation. In essence, we have c separate problems of the following form: use a set D of samples drawn independently according to the fixed but unknown probability distribution p(x)to determine $p(x \mid D)$ .This is the central problem of Bayesian learning.

4. Nonparametric Techniques

We treat supervised learning under the assumption that the forms of the underlying density functions are known in the last section. But in most pattern recognition applications, the common parametric forms rarely fit the densities. In this section we shall examine nonparametric procedures that can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known.
There are several types of nonparametric methods of interest in pattern recognition. One is to estimate the density functions $p(x \mid \omega_j)$ from sample. And it can be substituted for the true densities. Another is to estimate a posteriori probabilities $P(\omega_j \mid x)$ directly. such as the nearest-neighbor rule Finally, there are nonparametric procedures for transforming the feature space in the hope that it may be possible to employ parametric methods in the transformed space.
The following obvious estimate for $p(x)$:

$$p(x) = \frac{k_n / n}{V_n} \tag{22}$$

4.1 Parzen Windows

Assume that the region R^n is a d -dimensional hypercube. hn is the length of an edge of that hypercube, then its volume is given by

$$V_n = h_n^d \tag{23}$$

Define the window function as

$$\varphi(u) = \begin{cases} 1 & |u_j| \le 1/2 \\ 0 & otherwise \end{cases} \qquad j = 1, \cdots, d \tag{24}$$

The number of samples in this hypercube is given by

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{x - x_i}{h_n} \right) \tag{25}$$

Substitute this into Eq (22). we obtain the estimate

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi\left(\frac{x - x_i}{h_n} \right) \tag{26}$$

Eq.(26) expresses the estimate for $p(x)$ as an average of functions of x and the samples xi. In essence, the window function is being used for interpolation—each sample contributing to the estimate in accordance with its distance from x.

It is natural to ask that the estimate $p_n(x)$ be a legitimate density function. We can require that

$$\varphi(x) \ge 0, \tag{27}$$

and

$$\int \varphi(u) du = 1 \tag{28}$$

Maintain the relation $V_n = h_n^d$, then the $p_n(x)$ also satisfies these conditions.

Define $\delta_n(x)$ by

$$\delta_n(x) = \frac{1}{V_n} \varphi\left(\frac{x}{h_n} \right) \tag{29}$$

Then $p_n(x)$ can be written as the average

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\delta_n(x - x_i) \tag{30}$$

Since $V_n = h_n^d$, h_n clearly affects both the amplitude and the width of $\delta_n(x)$. If h_n is very large, the amplitude of $\delta_n(x)$ is small, and x must be far from xi before $\delta_n(x - x_i)$ changes much from $\delta_n(0)$ .In this case, pn(x) is the superposition of n broad, slowly changing functions and is a very smooth "out-of-focus" estimate of $p(x)$. On the other hand, if h_n is very small, the peak value of $\delta_n(x - x_i)$ is large and occurs near x=xi. In this case $p(x)$ is the superposition of n sharp pulses centered at the samples—an erratic, "noisy" estimate. For any value of h_n , the distribution is normalized, that is

$$\int \delta_n(x - x_i)dx = \int \frac{1}{V_n}\varphi\left(\frac{x - x_i}{h_n}\right)dx = \int \varphi(u)du = 1 \tag{31}$$

Let V_n slowly approach zero as n increases and $p_n(x)$ converges to the unknown density p(x). $p_n(x)$ has some mean $\overline{p}_n(x)$ and variance $\sigma_n^2(x)$. $p_n(x)$ converges to p(x) if

$$\lim_{n\to\infty}\overline{p}_n(x) = p(x) \tag{32}$$

and

$$\lim_{n\to\infty}\sigma_n^2(x) = 0 \tag{33}$$

To prove convergence we must place conditions on the unknown density $p(x)$, on the window function $\varphi(u)$, and on the window width h_n . In general, continuity of $p(\cdot)$ at x is required, and the conditions imposed by Eqs.(27)&(28) are customarily invoked. With care, it can be shown that the following additional conditions assure convergence:

$$\sup_{\mu}\varphi(u) < \infty \tag{34}$$

$$\lim_{\|\mu\|\to\infty}\varphi(u)\prod_{i=1}^{d}\mu_i = 0 \tag{35}$$

$$\lim_{n\to\infty}V_n = 0 \tag{36}$$

$$\lim_{n\to\infty}nV_n = \infty \tag{37}$$

Equations (34)&(35) keep $\varphi(\cdot)$ well behaved, and are satisfied by most density functions that one might think of using for window functions. Equations (36)&(37) state that the volume V_n must approach zero, but at a rate slower than $1/n$. We shall now see why these are the basic conditions for convergence.

4.2 k_n –Nearest-Neighbor Estimation

A potential remedy for the problem of the unknown "best" window function is to let the cell volume be a function of the training data, rather than some arbitrary function of the overall number of samples. For example, to estimate p(x)from n training samples or prototypes we can center a cell about x and let it grow until I captures k_n samples, where k_n is some specified function of n. These samples are the k_n nearest-neighbors of x. It the density is high near x, the cell will be relatively small, which leads to good resolution. If the density is low, it is true that the cell will grow large, but it will stop soon after it enters regions of higher density. In either case, if we take

$$p(x) = \frac{k_n / n}{V_n} \tag{38}$$

we want k_n to go to infinity as n goes to infinity, since this assures us that k_n / n will be a good estimate of the probability that a point will fall in the cell of volume V_n. However, we also want k_n to grow sufficiently slowly that the size of the cell needed to capture k_n training samples will shrink to zero. Thus, it is clear from Eq.(38) that the ratio k_n / n must go to zero. Although we shall not supply a proof, it can be shown that the conditions $\lim_{n\to\infty} k_n = \infty$ and $\lim_{n\to\infty} k_n / n = 0$ are necessary and sufficient for $p_n(x)$ to converge to p(x)in probability at all points where $p(x)$ is continuous. If we take $k_n = \sqrt{n}$ and assume that $p_n(x)$ is a reasonably good approximation to $p(x)$ we then see from Eq.(38) that $V_n = 1/(\sqrt{n}\, p(x))$. Thus, V_n again has the form $V_1 / \sqrt{n}$, but the initial volume V_1 is determined by the nature of the data rather than by some arbitrary choice on our part. Note that there are nearly always discontinuities in the slopes of these estimates, and these lie away from the prototypes themselves.

5. Support Vector Machine

In a support vector machine, the direct decision function that maximizes the generalization ability is determined for a two-class problem. Assuming that the training data of different classes do not overlap, the decision function is determined so that the distance from the training data is maximized. We call this the optimal decision function. Because it is difficult to determine a nonlinear decision function, the original input space is mapped into a high-dimensional space called feature space. And in the feature space, the optimal decision function, namely, the optimal hyper-plane is determined.

Support vector machines outperform conventional classifiers, especially when the number of training data is small and the number of input variables is large. This is because the conventional classifiers do not have the mechanism to maximize the margins of class boundaries. Therefore, if we introduce some mechanism to maximize margins, the generalization ability is improved.

If the decision function is linear, namely, $g_i(x)$ is given by

$$g_i(x) = w^T x + b \tag{39}$$

where w is an m-dimensional vector and b is a bias term, and if one class is on the positive side of the hyper-plane, i.e., $g_i(x) > 0$, and the other class is on the negative side, the given problem is said to be linearly separable.

5.1 Indirect Decision Functions

For an $n(>2)$ -class problem, suppose we have indirect decision functions $g_i(x)$ for classes i. To avoid unclassifiable regions, we classify x into class j given by

$$j = \arg\max_i g_i(x) \tag{40}$$

where arg returns the subscript with the maximum value of $g_i(x)$ .If more than one decision function take the same maximum value for x, namely, x is on the class boundary, it is not classifiable.

In the following we discuss several methods to obtain the direct decision functions for multi-class problems.

The first approach is to determine the decision functions by the one-against-all formulation. We determine the i th decision function $g_i(x)$ $i = 1, \cdots, n$, so that when x belongs to class i,

$$g_i(x) > 0 \tag{41}$$

and when x belongs to one of the remaining classes,

$$g_i(x) < 0 \tag{42}$$

When x is given, we classify x into class i if $g_i(x) > 0$ and $g_j(x) < 0$ $j \neq i, j = 1, \cdots, n$. But by these decision functions, unclassifiable regions exist when more than one decision function are positive or no decision functions are positive.

The second approach is based on a decision tree. It is considered to be a variant of one-against-all formulation. We determine the i th decision function $g_i(x)$ $i = 1, \cdots, n$, so that when x belongs to class i,

$$g_i(x) > 0 \tag{43}$$

and when x belongs to one of the classes $i+1, \cdots, n$,

$$g_i(x) < 0 \tag{44}$$

In classifying x, starting from $g_1(x)$, we find the first positive $g_i(x)$ and classify x into class i. If there is no such i among $g_i(x)$ ($i = 1, \cdots, n$), we classify x into class n.

The decision functions change if we determine decision functions in descending order or in an arbitrary order of class labels. Therefore, in this architecture, we need to determine the decision functions so that classification performance in the upper level of the tree is more accurate than in the lower one. Otherwise, the classification performance may not be good.

Pair-wise Formulation

The third approach is to determine the decision functions by pair-wise formulation. For classes i and j we determine the decision function $g_{ij}(x)$ ($i \neq j,,i,j=1,\cdots,n$, so that

$$g_{ij}(x) > 0 \tag{45}$$

when x belongs to class i and

$$g_{ij}(x) > 0 \tag{46}$$

when x belongs to class j .

In this formulation, $g_{ij}(x) = -g_{ji}(x)$, and we need to determine n(n?1)/2 decision functions. Classification is done by voting, namely, we calculate $g_i(x)$

$$g_i(x) = \sum_{j \neq i, j=1}^{n} sign(g_{ij}(x)) \tag{47}$$

where

$$sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{48}$$

and we classify x into the class with the maximum $g_i(x)$. By this formulation also, unclassifiable regions exist if $g_i(x)$ take the maximum value for more than one class. These can be resolved by decision-tree formulation or by introducing membership functions.

The fourth approach is to use error-correcting codes for encoding outputs. One-against-all formulation is a special case of error-correcting code with no error-correcting capability, and so is pair-wise formulation, as if"don't"care bits are introduced.

The fifth approach is to determine decision functions at all once. Namely, we determine the decision functions $g_i(x)$ by

$$g_i(x) > g_j(x) \text{ for } j = i, j = 1, \cdots, n \tag{49}$$

In this formulation we need to determine n decision functions at all once. This results in solving a problem with a larger number of variables than the previous methods. Unlike one-against-all and pair-wise formulations, there is no unclassifiable region.

Determination of decision functions using input-output pairs is called training. In training a multilayer neural network for a two-class problem, we can determine a direct decision function if we set one output neuron instead of two. But because for an n-class problem we set n output neurons with the i th neuron corresponding to the class i decision function, the obtained functions are indirect. Similarly, decision functions for fuzzy classifiers are indirect because membership functions are defined for each class. Conventional training methods

determine the indirect decision functions so that each training input is correctly classified into the class designated by the associated training output. Assuming that the circles and rectangles are training data for Classes 1 and 2, respectively, even if the decision function $g_2(x)$ moves to the right as shown in the dotted curve, the training data are still correctly classified. Thus there are infinite possibilities of the positions of the decision functions that correctly classify the training data. Although the generalization ability is directly affected by the positions, conventional training methods do not consider this.

5.2 Linear Learning Machines

In training a classifier, usually we try to maximize classification performance for the training data. But if the classifier is too fit for the training data, the classification ability for unknown data, i.e., the generalization ability is degraded. This phenomenon is called over-fitting. Namely, there is a trade-of between the generalization ability and fitting to the training data. For a two-class problem, a support vector machine is trained so that the direct decision function maximizes the generalization ability. Namely, the m -dimensional input space x is mapped into the l -dimensional($l \geq m$)feature space z . Then in z , the quadratic programming problem is solved to separate two classes by the optimal separating hyper-plane. One of the main ideas is, like support vector machines, to add a regularization term, which controls the generalization ability, to the objective function.

Let M m -dimensional training inputs x_i ($i = 1, \cdots, M$) belong to Class 1 or 2 and the associated labels be $y_i = 1$ for Class 1 and -1 for Class 2. If these data are linearly separable, we can determine the decision function:

$$D(x) = w^T x_i + b \tag{50}$$

where w is an m-dimensional vector, b is a bias term, and for $i = 1, \cdots, M$

$$w^T x_i + b \begin{cases} > 0 & for \quad y_i = 1 \\ < 0 & for \quad y_i = -1 \end{cases} \tag{51}$$

Because the training data are linearly separable, no training data satisfy $w^T x_i + b = 0$. Thus, to control separability, instead of(51),we consider the following inequalities:

$$w^T x_i + b \begin{cases} \geq 1 & for \quad y_i = 1 \\ \leq -1 & for \quad y_i = -1 \end{cases} \tag{52}$$

Equation(2.3)is equivalent to

$$y_i (w^T x_i + b) \geq 1 \quad \text{for} \quad i = 1, \cdots, M \tag{53}$$

The hyper-plane

$$D(x) = w^T x_i + b = c \quad \text{for} \quad -1 < c < 1 \tag{54}$$

forms a separating hyper-plane that separates x_i ($i = 1, \cdots, M$). When $c = 0$, the separating hyper-plane is in the middle of the two hyper-planes with $c = 1$ and > 1. The distance between the separating hyper-plane and the training datum nearest to the hyper-plane is called the margin. Assuming that the hyper-planes $D(x) = 1$ and > 1 include at least one training datum, the hyper-plane $D(x) = 0$ has the maximum margin for $-1 < c < 1$. The region $\{x \mid -1 \le D(x) \le 1\}$ is the generalization region for the decision function.

Now consider determining the optimal separating hyper-plane. The Euclidean distance from a training datum x to the separating hyper-plane is given by $|D(x)|/w$. Because the vector w is orthogonal to the separating hyper-plane, the line that goes through x and that is orthogonal to the hyper-plane is given by $aw/\|w\| + x$, where $|a|$ is the Euclidean distance from x to the hyper-plane. It crosses the hyper-plane at the point where

$$D(aw/\|w\| + x) = 0 \tag{55}$$

is satisfied. Solving(2.6)for a, we obtain $a = -D(x)/w$.

Then all the training data must satisfy

$$\frac{y_k D(x_k)}{\|w\|} \ge \delta \quad \text{for} \quad k = 1, \cdots, M \tag{56}$$

Where δ is the margin.

Now if (w, b) is a solution, (aw, ab) is also a solution, where a is a scalar. Thus we impose the following constraint:

$$\delta \cdot w = 1 \tag{57}$$

From (56) and (57),to find the optimal separating hyper-plane, we need to find w with the minimum Euclidean norm that satisfies(52). Therefore, the optimal separating hyper-plane can be obtained by minimizing

$$Q(w) = \frac{1}{2}\|w\|^2 \tag{58}$$

with respect to w and b subject to the constraints:

$$y_i(w^T x_i + b) \ge 1 \quad \text{for} \quad i = 1, \cdots, M \tag{59}$$

Here, the square of the Euclidean norm w in (58) is to make the optimization problem quadratic programming. The assumption of linear separability means that there exist w and b that satisfy (59).We call the solutions that satisfy (2.10) feasible solutions. Because the optimization problem has the quadratic objective function with the inequality constraints, even if the solutions are non-unique, the value of the objective function is unique(see Section 2.6.4).Thus non-uniqueness is not a problem for support vector machines. This is one of the

advantages of support vector machines over neural networks, which have numerous local minima.

5.3 SVM and Change-point Theory

To detect the change-points in signal data is an important practical problem. The classical method to solve this problem is using the statistical algorithms which are based on Bayesian theory. The efficiency of these methods always depends on the character of the given data. In this paper, we introduce support vector machine method to detect the abrupt change on signal data. The experience shows that the idea is effective, and it does not limit to the character of the distribution.

Consider time series $x_1^{(1)}, x_2^{(1)}, \cdots, x_m^{(1)}$ and time series $x_{m+1}^{(2)}, x_{m+2}^{(2)}, \cdots, x_n^{(2)}$, and assume some character has been change during $x_m^{(1)}$ to $x_{m+1}^{(2)}$, that means the first m points submit to the distribution (1), and the following data occur with the other distribution. The time series is recombination like this:

$$\begin{aligned} y_1^{(k)} &= (x_1^{(1)}, x_2^{(1)}, \cdots, x_k^{(1)}), \\ y_2^{(k)} &= (x_2^{(1)}, x_3^{(1)}, \cdots, x_{k+1}^{(1)}), \\ &\vdots \end{aligned} \tag{60}$$

where k is greatly less than m and $n-m$.

We assume that the change-point is at m, and the changing information has been distributed to several distinct and continuous y_i's. the vectors contain only the point on state (1) and state (2) are classifiable since the different distributions. Of course, it seems arbitrary here. We must do a great deal of experience to support this point of view.

In this paper, we illustrate our method to detect change-point with support vector machine method firstly in next section. The result of experience shows that the method is effective. And in the third section, we discuss on the detail of method. After that, the last section is our conclusion. .

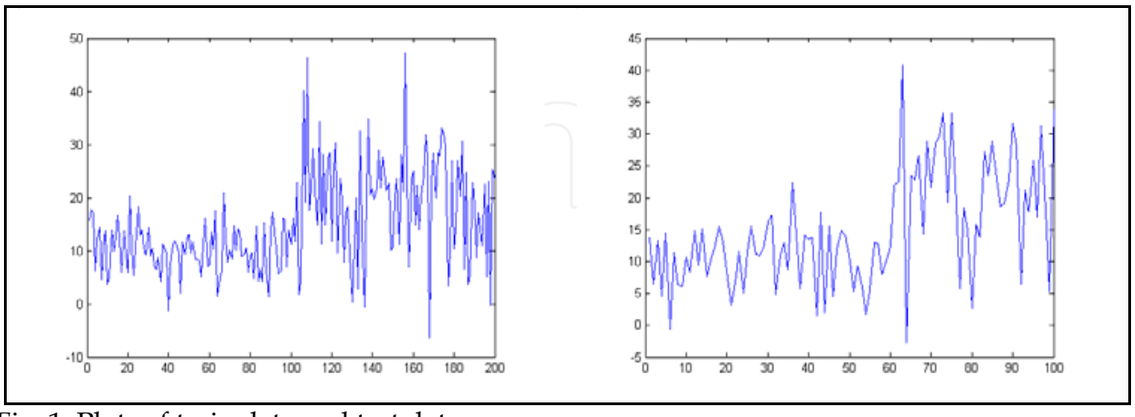


Fig. 1. Plots of train data and test data

The method is still effective for classifying the samples from the distributions with distinct variance. Let us consider a new simulation data set. The randomizer produces 100 samples

that submit to $N(\mu_1, \sigma_1^2)$ and the others that submit to $N(\mu_2, \sigma_2^2)$. Let $\mu_1 = 10$, $\mu_2 = 20$, $\sigma_1^2 = 5$, $\sigma_2^2 = 10$. The randomizer produce 100 samples under two group of parameter respectively. These samples are taken as training data set. Repeat the steps to produce more 60 and 40 samples under the different group of parameter. These samples act as test data set. The values of samples are described by figure below.

The result of output file is list in the table below. The terms of columns are: (1) Dimension; (2) Accuracy on test set; (3) support vectors; (4) Count of incorrect samples; and (5) Mean Squared Error.

k	Accuracy	SV	incorrect	MSE
2	88.78%	100	11	0.890499
3	90.63%	85	9	1.687711
4	94.68%	74	5	1.437032
5	96.74%	68	3	1.260453
6	97.78%	62	2	1.098423
7	97.73%	57	2	0.859724
8	100.00%	55	0	0.814199
9	100.00%	52	0	0.727360
10	100.00%	53	0	0.754256

Table 1. Result of experience

We are interested in considering the location of the incorrect samples. Figure 4 tell us the information.

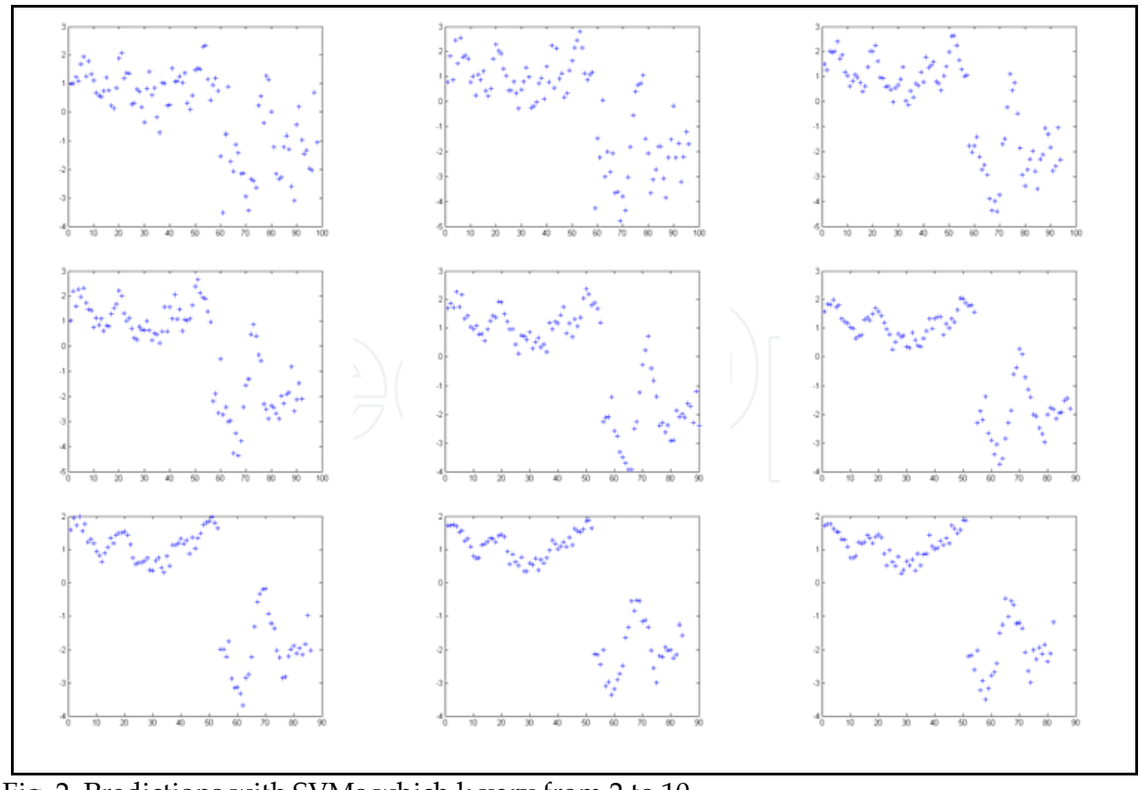


Fig. 2. Predictions with SVMs which k vary from 2 to 10

To detect the change-points in signal processing is an important practical problem. The classical method to solve this problem is using the statistical algorithms which are based on Bayesian theory. The efficiency of these methods always depends on the character of the given data. In this paper, we introduce support vector machine method to detect the abrupt change on signal data. A change-point detecting problem is transformed to a classification problem. The experience shows that the idea is effective,

6. Neural Networks

For classification, we will have c output units, one for each of the categories, and the signal from each output unit is the discriminant function $g_k(x)$ as:

$$g_k(x) = z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right) \tag{61}$$

This, then, is the class of functions that can be implemented by a three-layer neural network. An even broader generalization would allow transfer functions at the output layer to differ from those in the hidden layer, or indeed even different functions at each individual unit. Kolmogorov proved that any continuous function $g(x)$ defined on the unit hypercube I^n (I=[0,1]and $n \geq 2$)can be represented in the form

$$g(x) = \sum_{j=1}^{2n+1} E_j\left(\sum_{i=1}^{d} \psi_{ij}(x_i)\right) \tag{62}$$

or properly chosen functions E_j and $\psi_{ij}(x_i)$.

We consider the training error on a pattern to be the sum over output units of the training squared difference between the desired output t_k (given by a teacher)and the actual error output z_k , much as we had in the LMS algorithm for two-layer nets:

$$J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}(t - z)^2 \tag{63}$$

where t and z are the target and the network output vectors of length c ; w represents all the weights in the network.

The back propagation learning rule is based on gradient descent. The weights are initialized with random values, and are changed in a direction that will reduce the error:

$$\Delta w = -\eta \frac{\partial J}{\partial w} \tag{64}$$

or in component form

$$\Delta w_{\min} = -\eta \frac{\partial J}{\partial w_{\min}} \tag{65}$$

where η is the learning rate, and merely indicates the relative size of the change in weights. This iterative algorithm requires taking a weight vector at iteration m and updating it as:

$$w(m+1) = w(m) + \Lambda w(m) \tag{66}$$

where m indexes the particular pattern presentation

7. Stochastic Search

Search methods based on evolution—genetic algorithms and genetic programming —perform highly parallel stochastic searches in a space set by the designer. The fundamental representation used in genetic algorithms is a string of bits, or chromosome; the representation in genetic programming is a snippet of computer code. Variation is introduced by means of crossover, mutation and insertion. As with all classification methods, the better the features, the better the solution. There are many heuristics that can be employed and parameters that must be set. As the cost of computation contiues to decline, computationally intensive methods, such as Boltzmann networks and evolutionary methods, should become increasingly popular.

7.1 Simulated annealing

In physics, the method for allowing a system such as many magnets or atoms in an alloy to find a low-energy configuration is based on annealing. Annealing proceeds by gradually lowering the temperature of the system—ultimately toward zero and thus no randomness—so as to allow the system to relax into a low-energy configuration. Such annealing is effective because even at moderately high temperatures, the system slightly favors regions in the configuration space that are overall lower in energy, and hence are more likely to contain the global minimum. As the temperature is lowered, the system has increased probability of finding the optimum configuration.

This method is successful in a wide range of energy functions Fortunately, the problems in learning we shall consider rarely involve such pathological functions.

The statistical properties of large number of interacting physical components at a temperature T, such as molecules in a gas or magnetic atoms in a solid, have been thoroughly analyzed. A key result, which relies on a few very natural assumptions, is the following: the probability the system is in a(discrete)configuration indexed by γ having energy $E\gamma$ is given by

$$P(\gamma) = \frac{e^{-E_\gamma / T}}{Z(T)} \tag{67}$$

where Z is a normalization constant. The numerator is the Boltzmann factor and the denominator the partition function, the sum over all possible configurations

$$Z(T) = \sum_{\gamma'} e^{-E_{\gamma'} / T} \tag{68}$$

which guarantees Eq. 2 represents a true probability. The number of configurations is very high, 2^N, and in physical systems Z can be calculated only in simple cases. Fortunately, we need not calculate the partition function, as we shall see.

7.2 Genetic Algorithms

In basic genetic algorithms, the fundamental representation of each classifier is a binary string, called a chromosome. The mapping from the chromosome to the features chromosome and other aspects of the classifier depends upon the problem domain, and the designer has great latitude in specifying this mapping. In pattern classification, the score is usually chosen to be some monotonic function of the accuracy on a data set, possibly with penalty term to avoid overfitting. We use a desired fitness, θ, as the stopping criterion. Before we discuss these points in more depth, we first consider more specifically the structure of the basic genetic algorithm, and then turn to the key notion of genetic operators, used in the algorithm.

There are three primary genetic operators that govern reproduction: Crossover, Mutation and Selection.: Crossover involves the mixing—"mating"—of two chromosomes. A mating split point is chosen randomly along the length of either chromosome. The first part of chromosome A is spliced to the last part of chromosome B, and vice versa, thereby yielding two new chromosomes. Each bit in a single chromosome is given a small chance, Pmut, of being changed from a 1 to a 0 or vice versa. Other genetic operators may be employed, for instance inversion—where the chromosome is reversed front to back. This operator is used only rarely since inverting a chromosome with a high score nearly always leads to one with very low score. Below we shall briefly consider another operator, insertions.

The process of selection specifies which chromosomes from one generation will be sources for chromosomes in the next generation. Up to here, we have assumed that the chromosomes would be ranked and selected in order of decreasing fitness until the next generation is complete. This has the benefit of generally pushing the population toward higher and higher scores. Nevertheless, the average improvement from one generation to the next depends upon the variance in the scores at a given generation, and because this standard fitness-based selection need not give high variance, other selection methods may prove superior.

The principle alternative selection scheme is fitness-proportional selection, or fitness proportional reproduction, in which the probability that each chromosome is selecte is proportional to its fitness. While high-fitness chromosomes are preferentially selected, occasionally low-fitness chromosomes are selected, and this may preserve diversity and increase variance of the population.

A minor modification of this method is to make the probability of selection proportional to some monotonically increasing function of the fitness. If the function instead has a positive second derivative, the probability that high-fitness chromosomes is enhanced. One version of this heuristic is inspired by the Boltzmann factor of Eq.2; the probability that chromosome i with fitness fi will be selected is

$$P(i) = \frac{e^{f_i / T}}{E\left(e^{f_i / T}\right)} \tag{69}$$

where the expectation is over the current generation and T is a control parameter loosely referred to as a temperature. Early in the evolution the temperature is set high, giving all chromosomes roughly equal probability of being selected. Late in the evolution the temperature is set lower so as to find the chromosomes in the region of the optimal classifier. We can express such search by analogy to biology: early in the search the population remains diverse and explores the fitness landscape in search of promising areas; later the population exploits the specific fitness opportunities in a small region of the space of possible classifiers.

When a pattern recognition problem involves a model that is discrete or of such high complexity that analytic or gradient descent methods are unlikely to work, we may employ stochastic techniques—ones that at some level rely on randomness to find model parameters.Simulated annealing,based on physical annealing of metals, consists in randomly perturbing the system,and gradually decreasing the randomness to a low final level,in order to find an optimal solution.Boltzmann learning trains the weights in a network so that the probability of a desired final output is increased. Such learning is based on gradient descent in the Kullback-Liebler divergence between two distributions of visible states at the output units:one distribution describes these units when clamped at the known category information, and the other when they are free to assume values based on the activations throughout the network. Some graphical models, such as hidden Markov models and Bayes belief networks, have counterparts in structured Boltzmann networks, and this leads to new applications of Boltzmann learning.

8. Unsupervised Learning and Clustering

Until now we have assumed that the training samples used to design a classifier were labelled by their category membership. Procedures that use labelled samples are said to be supervised. Now we shall investigate a number of unsupervised procedures, which use unlabeled samples.

Let us reconsider our original problem of learning something of use from a set of unlabeled samples. Viewed geometrically, these samples may form clouds of points in a d-dimensional space. Suppose that we knew that these points came from a single normal distribution. Then the most we could learn form the data would be contained in the sufficient statistics—the sample mean and the sample covariance matrix. In essence, these statistics constitute a compact description of the data. The sample mean locates the centre of gravity of the cloud; it can be thought of as the single point m that best represents all of the data in the sense of minimizing the sum of squared distances from m to the samples. The sample covariance matrix describes the amount the data scatters along various directions. If the data points are actually normally distributed, then the cloud has a simple hyperellipsoidal shape, and the sample mean tends to fall in the region where the samples are most densely concentrated.

If we assume that the samples come from a mixture of c normal distributions, we can approximate a greater variety of situations. In essence, this corresponds to assuming that the samples fall in hyperellipsoidally shaped clouds of various sizes and orientations. If the number of component densities is sufficiently high, we can approximate virtually any density function as a mixture model in this way, and use the parameters of the mixture to describe the data. Alas, we have seen that the problem of estimating the parameters of a

mixture density is not trivial. Furthermore, in situations where we have relatively little prior knowledge about the nature of the data, the assumption of particular parametric forms may lead to poor or meaningless results. Instead of finding structure in the data, we would be imposing structure on it.

One alternative is to use one of the nonparametric methods to estimate the unknown mixture density. If accurate, the resulting estimate is certainly a complete description of what we can learn from the data. Regions of high local density, which might correspond to significant subclasses in the population, can be found from the peaks or modes of the estimated density.

If the goal is to find subclasses,a more direct alternative is to use a clustering procedure. Roughly speaking, clustering procedures yield a data description in terms clustering of clusters or groups of data points that possess strong internal similarities. Formalprocedure clustering procedures use a criterion function, such as the sum of the squared distances from the cluster centres, and seek the grouping that extremizes the criterion function. Because even this can lead to unmanageable computational problems, other procedures have been proposed that are intuitively appealing but that lead to solutions having few if any established properties. Their use is usually justified on the ground that they are easy to apply and often yield interesting results that may guide the application of more rigorous procedures.

8.1 Similarity Measures

The most obvious measure of the similarity(or dissimilarity)between two samples is the distance between them. One way to begin a clustering investigation is to define a suitable distance function and compute the matrix of distances between all pairs of samples. If distance is a good measure of dissimilarity, then one would expect the distance between samples in the same cluster to be significantly less than the distance between samples in different clusters.

Suppose for the moment that we say that two samples belong to the same cluster if the Euclidean distance between them is less than some threshold distance d_0 . It is immediately obvious that the choice of d0 is very important. If d_0 is very large, all of the samples will be assigned to one cluster. If d_0 is very small, each sample will form an isolated, singleton cluster. To obtain "natural" clusters, d_0 will have to be greater than the typical within-cluster distances and less than typical between-cluster distances.

Less obvious perhaps is the fact that the results of clustering depend on the choice of Euclidean distance as a measure of dissimilarity. That particular choice is generally justified if the feature space is isotropic and the data is spread roughly evenly a long all directions. Clusters defined by Euclidean distance will be invariant to translations or rotations in feature space—rigid-body motions of the data points. However, they will not be invariant to linear transformations in general, or to other transformations that distort the distance relationships. Thus, a simple scaling of the coordinate axes can result in a different grouping of the data into clusters. Of course, this is of no concern for problems in which arbitrary rescaling is an unnatural or meaningless transformation. However, if clusters are to mean anything, they should be invariant to transformations natural to the problem.

One way to achieve invariance is to normalize the data prior to clustering. For example, to obtain invariance to displacement and scale changes, one might translate and scale the axes

so that all of the features have zero mean and unit variance— standardize the data. To obtain invariance to rotation, one might rotate the axes so that they coincide with the eigenvectors of the sample covariance matrix. This trans- formation to principal components can be preceded and/or followed by normalization for scale.

However, we should not conclude that this kind of normalization is necessarily desirable. Consider, for example, the matter of translating and whitening—scaling the axes so that each feature has zero mean and unit variance. The rationale usually given for this normalization is that it prevents certain features from dominating distance calculations merely because they have large numerical values, much as we saw in networks trained with backpropagation. Subtracting the mean and dividing by the standard deviation is an appropriate normalization if this spread of values is due to normal random variation;however,it can be quite inappropriate if the spread is due to the presence of subclasses.

Instead of scaling axes, we can change the metric in interesting ways. For instance, one broad class of distance metrics is of the form

$$d(x,x') = \left(\sum_{k=1}^{d} | x_k - x_k' |^q \right)^{1/q} \tag{70}$$

where $q \geq 1$ is a selectable parameter—the general Minkowski metric we considered in. Setting q =2 gives the familiar Euclidean metric while setting q =1 the Manhattan or city block metric—the sum of the absolute distances along each of the d coordinate axes. Note that only q =2 is invariant to an arbitrary rotation or translation in feature space. Another alternative is to use some kind of metric based on the data itself, such as the Mahalanobis distance.

More generally, one can abandon the use of distance altogether and introduce a nonmetric similarity function $s(x,x')$ to compare two vectors x and x . Convention-similarity ally, this is a symmetric functions whose value is large when x and x are somehowfunction "similar." For example, when the angle between two vectors is a meaningful measure of their similarity, then the normalized inner product

$$s(x,x') = \frac{x^t x'}{\| x \| \cdot \| x' \|} \tag{71}$$

may be an appropriate similarity function. This measure, which is the cosine of the angle between x and x ,is invariant to rotation and dilation, though it is not invariant to translation and general linear transformations.

8.2 Criterion Functions

8.2.1 The Sum-of-Squared-Error Criterion

The simplest and most widely used criterion function for clustering is the sum-of- squared-error criterion. Let ni be the number of samples in D_i and let mi be the mean of those samples,

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x \tag{72}$$

Then the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{c} \sum_{x \in D_i} \| x - m_i \|^2 \tag{73}$$

8.2.2 Related Minimum Variance Criteria

By some simple algebraic manipulation we can eliminate the mean vectors from the expression for J_e and obtain the equivalent expression

$$J_e = \frac{1}{2} \sum_{i=1}^{c} n_i \vec{s}_i \tag{74}$$

where

$$\vec{s}_i = \frac{1}{n_i^2} \sum_{x \in D_i} \sum_{x' \in D_i} \| x - x' \|^2 \tag{75}$$

Equation 51 leads us to interprets $\vec{s}_i$ as the average squared distance between points in the i-th cluster, and emphasizes the fact that the sum-of-squared-error criterion uses Euclidean distance as the measure of similarity. It also suggests an obvious way of obtaining other criterion functions. For example, one can replaces $\vec{s}_i$ by the average, the median, or perhaps the maximum distance between points in a cluster. More generally, one can introduce an appropriate similarity function s(x,x)and replaces $\vec{s}_i$ by functions such as

$$\vec{s}_i = \frac{1}{n_i^2} \sum_{x \in D_i} s(x, x') \tag{76}$$

or

$$\vec{s}_i = \min_{x, x' \in D_i} s(x, x') \tag{77}$$

8.3 Hierarchical Clustering

The most natural representation of hierarchical clustering is a corresponding tree, called a dendrogram, which shows how the samples are grouped. If it is possible to measure the similarity between clusters, then the dendrogram is usually drawn to scale to show the similarity between the clusters that are grouped. We shall see shortly how such similarity values can be obtained, but first note that the similarity values can be used to help determine whether groupings are natural or forced. If the similarity values for the levels are roughly evenly distributed throughout the range of possible values, then there is no principled argument that any particular number of clusters is better or "more natural" than another. Conversely, suppose that there is a unusually large gap between the similarity

values for the levels corresponding to c =3 and to c =4 clusters. In such a case, one can argue that c =3 is the most natural number of clusters.

8.3.1 The Nearest-Neighbor Algorithm

When $d_{\max}$ is used to measure the distance between clusters the algorithm is sometimes called the nearest-neighbor cluster algorithm, or minimum algorithm Moreover, if it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the single-linkage algorithm. Suppose that we think of the data points as being nodes of a graph, with edges forming a path between the nodes in the same subset D_i. When dmin is used to measure the distance between subsets, the nearest neighbor nodes determine the nearest subsets. The merging of D_i and D_j corresponds to adding an edge between the nearest pair of nodes in D_i and D_j. Since edges linking clusters always go between distinct clusters, the resulting graph never has any closed loops or circuits;in the terminology of graph theory, this procedure generates a tree. If it is allowed to continue until all of the subsets are linked, the result is a spanning tree—a tree with a path from any node to any other node. Moreover,it can be shown that the sum of the edge lengths of the resulting tree will not exceed the sum of the edge lengths for any other spanning tree for that set of samples. Thus, with the use of $d_{\min}$ as the distance measure, the agglomerative clustering procedure becomes an algorithm for generating a minimal spanning tree.

8.3.2 The Farthest-Neighbor Algorithm

When dmax(Eq.75)is used to measure the distance between subsets, the algorithm is sometimes called the farthest-neighbor clustering algorithm, or maximum algorithm. If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the complete-linkage algorithm. The farthest-neighbor algorithm discourages the growth of elongated clusters. Application of the procedure can be thought of as producing a graph in which edges connect all of the nodes in a cluster. In the terminology of graph theory, every cluster constitutes a complete subgraph. The distance between two clusters is determined by the most distant nodes in the two clusters. When the nearest clusters are merged, the graph is changed by adding edges between every pair of nodes in the two clusters.

Unsupervised learning and clustering seek to extract information from unlabeled samples. If the underlying distribution comes from a mixture of component densities described by a set of unknown parameters θ, then θ can be estimated by Bayesian or maximum- likelihood methods. A more general approach is to define some measure of similarity between two clusters, as well as a global criterion such as a sum-squared- error or trace of a scatter matrix. Since there are only occasionally analytic methods for computing the clustering which optimizes the criterion, a number of greedy(locally step-wise optimal)iterative algorithms can be used, such as k-means and fuzzy k-means clustering.

If we seek to reveal structure in the data at many levels—i.e., clusters with sub-clusters and sub-subcluster—then hierarchical methods are needed. Agglomerative or bottom-up methods start with each sample as a singleton cluster and iteratively merge clusters that are "most similar" according to some chosen similarity or distance measure.Conversely, divisive or top-down methods start with a single cluster representing the full data set and

iteratively splitting into smaller clusters, each time seeking the subclusters that are most dissimilar. The resulting hierarchical structure
is revealed in a dendrogram. A large disparity in the similarity measure for successive cluster levels in a dendrogram usually indicates the "natural" number of clusters. Alternatively, the problem of cluster validity—knowing the proper number of clusters —can also be addressed by hypothesis testing. In that case the null hypothesis is that there are some number c of clusters; we then determine if the reduction of the cluster criterion due to an additional cluster is statistically significant.
Competitive learning is an on-line neural network clustering algorithm in which the cluster center most similar to an input pattern is modified to become more like that pattern. In order to guarantee that learning stops for an arbitrary data set, the learning rate must decay. Competitive learning can be modified to allow for the creation of new cluster centers, if no center is sufficiently similar to a particular input pattern, as in leader-follower clustering and Adaptive Resonance. While these methods have many advantages, such as computational ease and tracking gradual variations in the data, they rarely optimize an easily specified global criterion such as sum-of-squared error.
Component analysis seeks to find directions or axes in feature space that provide an improved, lower-dimensional representation for the full data space. In(linear) principal component analysis, such directions are merely the largest eigenvectors of the covariance matrix of the full data; this optimizes a sum-squared-error criterion. Nonlinear principal components, for instance as learned in an internal layer an auto- encoder neural network, yields curved surfaces embedded in the full d-dimensional feature space, onto which an arbitrary pattern x is projected. The goal in independent component analysis—which uses gradient descent in an entropy criterion—is to determine the directions in feature space that are statistically most independent. Such directions may reveal the true sources(assumed independent)and can be used for segmentation and blind source separation.
Two general methods for dimensionality reduction is self-organizing feature maps and multidimensional scaling. Self-organizaing feature maps can be highly nonlinear, and represents points close in the source space by points close in the lower-dimensional target space. In preserving neighborhoods in this way, such maps also called "topologically correct." The source and target spaces can be of very general shapes, and the mapping will depend upon the the distribution of samples within the source space. Multidimensional scaling similarly learns a nonlinear mapping that, too, seeks to preserve neighborhoods, and is often used for data visualization. Because the basic method requires all the inter-point distances for minimizing a global criterion function, its space complexity limits the usefulness of multidimensional scaling to problems of moderate size.

9. Conclusion

One approach to this problem is to use the samples to estimate the unknown probabilities and probability densities, and to use the resulting estimates as if they were the true values. In typical supervised pattern classification problems, the estimation of the prior probabilities presents no serious difficulties. However, estimation of the class-conditional densities is quite another matter. The number of available samples always seems too small, and serious problems arise when the dimensionality of the feature vector x is large. If we know the number of parameters in advance and our general knowledge about the problem permits us

to parameterize the conditional densities, then the severity of these problems can be reduced significantly. Suppose, for example, that we can reasonably assume that $p(x|\omega_i)$ is a normal density with mean μ_i and covariance matrix Σ_i, although we do not know the exact values of these quantities. This knowledge simplifies the problem from one of estimating an unknown function $p(x|\omega_i)$ to one of estimating the parameters μ_i and Σ_i.

In general there are two approaches to develop classifiers: a parametric approach and a nonparametric approach. in a parametric approach, a priori knowledge of data distributions is assumed, otherwise, a nonparametric approach will be employed. Neural networks, fuzzy systems, and support vector machines are typical nonparametric classifiers.

10. References

Abe, S. (2005). *Support vector machines for pattern classification,* Springer, 1852339292, USA

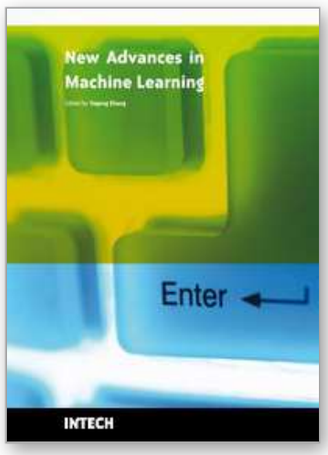
Richard, D.; Peter, H. & David, S. (1999). *Pattern Classification,* Wiley, John & Sons, Incorporated, 0471056693, USA

Simon S. Haykin (2008). *Neural Networks and Learning Machines,* Prentice Hall, 0131471392

Thorsten Joachims(2002), Learning to Classify Text Using Support Vector Machines. Dissertation, Kluwer,.

Vapnik, V. (1998). Statistical learning theory. Chichester, UK: Wiley

Yizhang, G.; Zhifeng, H. (2007). Using SVMs Method to Detect Abrupt Change, Proceedings of 2007 International Conference on Machine Learning and Cybernetics, pp. 3298-3301, 1-4244-0972-1, Aug. 2007, Hong Kong



New Advances in Machine Learning

Edited by Yagang Zhang

ISBN 978-953-307-034-6
Hard cover, 366 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The purpose of this book is to provide an up-to-date and systematical introduction to the principles and algorithms of machine learning. The definition of learning is broad enough to include most tasks that we commonly call “learning” tasks, as we use the word in daily life. It is also broad enough to encompass computers that improve from experience in quite straightforward ways. The book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides a good introduction to many approaches of machine learning, and it is also the source of useful bibliographical information.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen