Chapter 8
Supervised Classification Techniques

8.1 Introduction

Supervised classification is the technique most often used for the quantitative analysis of remote sensing image data. At its core is the concept of segmenting the spectral domain into regions that can be associated with the ground cover classes of interest to a particular application. In practice those regions may sometimes overlap. A variety of algorithms is available for the task, and it is the purpose of this chapter to cover those most commonly encountered. Essentially, the different methods vary in the way they identify and describe the regions in spectral space. Some seek a simple geometric segmentation while others adopt statistical models with which to associate spectral measurements and the classes of interest. Some can handle user-defined classes that overlap each other spatially and are referred to as *soft classification* methods; others generate firm boundaries between classes and are called *hard classification* methods, in the sense of establishing boundaries rather than having anything to do with difficulty in their use. Often the data from a *set* of sensors is available to help in the analysis task. Classification methods suited to multi-sensor or multi-source analysis are the subject of Chap. 12.

The techniques we are going to develop in this chapter come from a field that has had many names over the years, often changing as the techniques themselves develop. Most generically it is probably called pattern recognition or pattern classification but, as the field has evolved, the names

learning machines
pattern recognition
classification, and
machine learning

have been used. Learning machine theory commenced in the late 1950s in an endeavour to understand brain functioning and to endow machines with a degree of decision making intelligence[1]; so the principle of what we are going to develop here is far from new, although some of the procedures are.

8.2 The Essential Steps in Supervised Classification

Recall from Fig. 3.3 that supervised classification is essentially a mapping from the measurement space of the sensor to a field of labels that represent the ground cover types of interest to the user. It depends on having enough pixels available, whose class labels are known, with which to train the classifier. In this context "training" refers to the estimation of the parameters that the classifier needs in order to be able to recognise and label unseen pixels. The labels represent the classes on the map that the user requires. The map is called the *thematic map*, meaning a map of themes.

An important concept must be emphasised here, which is often overlooked in practice and about which we will have much to say in Chap. 11. That concerns whether the classes of interest to the user occupy unique regions in spectral space, and whether there is a one-to-one mapping between the measurement vectors and class labels. A simple illustration is shown in Fig. 8.1, which is an infrared versus visible red spectral space of a geographical region used for growing crops. The data is shown to be in three moderately distinct clusters, corresponding to rich, fully growing crops (on loam), a loam soil and a sandy soil. One user might be interested in a map of crops and soil—just two classes, but the soil class has two sub-classes—loam and sand. Another user might be interested in which parts of the region are used to support cropping, and those that are not. The former will also have two sub-classes—fully growing crops and loam—but they are different from those of the first user. We call the classes into which the data naturally groups the *spectral classes* (in this simple example the crops, loam and sand) whereas those that match user requirements are called *information classes.*[2]

This simple example has been used to introduce the distinction. In practice the difference may be more subtle. Spectral classes are groupings of pixel vectors in spectral space that are well matched to the particular classifier algorithm being used. Sometimes it will be beneficial to segment even simple classes like vegetation into sub-groups of data and let the classifier work on the sub-groups. After classification is complete the analyst then maps the sub-groups to the information classes of interest. For most of this chapter we will not pursue any further the distinction between the two class types. Generally, we will assume the information and spectral classes are the same.

[1] N.J. Nilsson, *Learning Machines*, McGraw-Hill, N.Y., 1965.

[2] The important distinction between information and spectral classes was first made in P.H. Swain and S.M. Davis, eds., *Remote Sensing: the Quantitative Approach*, McGraw-Hill, N.Y., 1978.

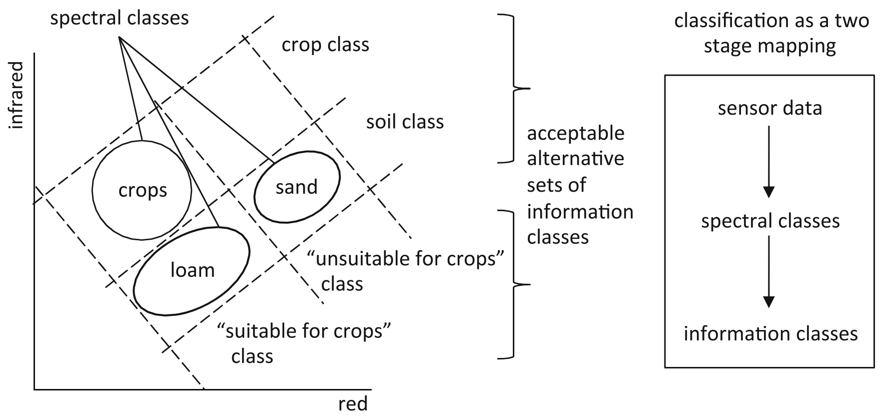

spectral classes
infrared
red
crops
loam
sand
crop class
soil class
"unsuitable for crops" class
"suitable for crops" class
acceptable alternative sets of information classes
classification as a two stage mapping
sensor data
spectral classes
information classes

Fig. 8.1 Simple illustration of the difference between spectral and information classes: in this case the spectral classes have identifiable names; in practice they are more likely to be groupings of data that match the characteristics of the classifier to be used

Many different algorithms are available for supervised classification, ranging from those based on probability distributions to those in which the spectral measurement space is partitioned geometrically into class-specific regions. Irrespective of the method chosen, the essential practical steps in applying a classifier usually include:

1. Deciding on the set of ground cover type classes into which to segment the image.
2. Choosing known, representative pixels for each of the classes. Those pixels are said to form *training data*. Training sets for each class can be established using site visits, maps, air photographs or even photointerpretation of image products formed from the data. Sometimes the training pixels for a given class will lie in a common region enclosed by a border. That region is called a *training field*.
3. Using the training data to estimate the parameters of the particular classifier algorithm to be employed; the set of parameters is sometimes called the *signature* of that class.
4. Using the trained classifier to label every pixel in the image as belonging to one of the classes specified in step 1. Here the whole image is classified. Whereas in step 2 the user may need to know the labels of about 1% or so of the pixels, the remainder are now labelled by the classifier.
5. Producing thematic (class) maps and tables which summarise class memberships of all pixels in the image, from which the areas of the classes can be measured.
6. Assessing the accuracy of the final product using a labelled *testing data* set.

In practice it might be necessary to decide, on the basis of the results obtained at step 6, to refine the training process in order to improve classification accuracy. Sometimes it might even be desirable to classify just the training samples themselves to ensure that the parameters generated at step 3 are acceptable.

It is our objective now to consider the range of algorithms that could be used in steps 3 and 4. Recall that we are assuming that each information class consists of just one spectral class, unless specified otherwise, and we will use the names interchangeably. That allows the development of the basic algorithms to proceed without the added complexity of considering spectral classes, which are also called sub-classes. Sub-classes are treated in Sect. 8.4 and in Chaps. 9 and 11.

8.3 Maximum Likelihood Classification

Maximum likelihood classification is one of the most common supervised classification techniques used with remote sensing image data, and was the first rigorous algorithm to be employed widely. It is developed in the following in a statistically acceptable manner. A more general derivation is given in Appendix E, although the present approach is sufficient for most remote sensing purposes.

8.3.1 Bayes' Classification

Let the classes be represented by $\omega_i, i = 1 \ldots M$ where M is the number of classes. In determining the class or category to which a pixel with measurement vector $\mathbf{x}$ belongs, the conditional probabilities

$$p(\omega_i|\mathbf{x}), i = 1 \ldots M$$

play a central role. The vector $\mathbf{x}$ is a column vector of the brightness values for the pixel in each measurement band. It describes the pixel as a point in spectral space. The probability $p(\omega_i|\mathbf{x})$ tells us the likelihood that ω_i is the correct class for the pixel at position $\mathbf{x}$ in the spectral space. If we knew the complete set of $p(\omega_i|\mathbf{x})$ for the pixel, one for each class, then we could label the pixel—classify it—according to the *decision rule*

$$\mathbf{x} \in \omega_i \text{ if } p(\omega_i|\mathbf{x}) > p(\omega_j|\mathbf{x}) \text{ for all } j \neq i \tag{8.1}$$

This says that the pixel with measurement vector $\mathbf{x}$ is a member of class ω_i if $p(\omega_i|\mathbf{x})$ is the largest probability of the set. This intuitive statement is a special case of a more general rule in which the decisions can be biased according to different degrees of significance being attached to different incorrect classifications. That general approach is called Bayes' classification which is the subject of Appendix E.

The rule of (8.1) is sometimes called a *maximum posterior* or MAP *decision rule*, a name which will become clearer in the next section.

8.3.2 The Maximum Likelihood Decision Rule

Despite the simplicity of (8.1) the $p(\omega_i|\mathbf{x})$ are unknown. What we can find relatively easily, though, are the set of class conditional probabilities $p(\mathbf{x}|\omega_i)$, which describe the chances of finding a pixel at position $\mathbf{x}$ in spectral space from each of the classes ω_i. Those conditional probability functions are estimated from labelled training data for each class. Later we will adopt a specific form for the distribution function, but for the moment we will retain it in general form.

The desired $p(\omega_i|\mathbf{x})$ in (8.1) and the available $p(\mathbf{x}|\omega_i)$, estimated from training data, are related by Bayes' theorem[3]

$$p(\omega_i|\mathbf{x}) = p(\mathbf{x}|\omega_i)\,p(\omega_i)/p(\mathbf{x}) \tag{8.2}$$

in which $\mathrm{p}(\omega_i)$ is the probability that pixels from class ω_i appear anywhere in the image. If 15% of the pixels in the scene belong to class ω_i then we could use $p(\omega_i) = 0.15$. The $p(\omega_i)$ are referred to as *prior probabilities*—or sometimes just *priors*. If we knew the complete set then they are the probabilities with which we could guess the class label for a pixel before (prior to) doing any analysis. In contrast the $p(\omega_i|\mathbf{x})$ are the *posterior probabilities* since, in principle, they are the probabilities of the class labels for a pixel at position $\mathbf{x}$, *after* analysis. What we strive to do in statistical classification is to estimate, via (8.2), the set of class posterior probabilities for each pixel so that the pixels can be labelled according to the largest of the posteriors, as in (8.1).

The term $p(\mathbf{x})$ is the probability of finding a pixel with measurement vector $\mathbf{x}$ in the image, from any class. Although $p(\mathbf{x})$ is not important in the following development it should be noted that

$$p(\mathbf{x}) = \sum_{i=1}^{M} p(\mathbf{x}|\omega_i)p(\omega_i)$$

On substituting from (8.2) the decision rule of (8.1) reduces to

$$\mathbf{x} \in \omega_i \text{ if } p(\mathbf{x}|\omega_i)p(\omega_i) > p\big(\mathbf{x}|\omega_j\big)p(\omega_j) \text{ for all } j \neq i \tag{8.3}$$

in which $p(\mathbf{x})$ has been removed as a common factor, since it is not class dependent. The decision rule of (8.3) is more acceptable than that of (8.1) since the $p(\mathbf{x}|\omega_i)$ are known from training data, and it is conceivable that the priors $p(\omega_i)$ are also known or can be estimated.

It is mathematically convenient now to define the *discriminant function*

$$g_i(\mathbf{x}) = \ln\{p(\mathbf{x}|\omega_i)p(\omega_i)\} = \ln p(\mathbf{x}|\omega_i) + \ln p(\omega_i) \tag{8.4}$$

Because the natural logarithm is a monotonic function we can substitute (8.4) directly into (8.3) to give as the decision rule:

[3] J.E. Freund, *Mathematical Statistics,* 5th *ed*., Prentice Hall, N.J., 1992.

$$\mathbf{x} \in \omega_i \text{ if } g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ for all } j \neq i \tag{8.5}$$

8.3.3 Multivariate Normal Class Models

To develop the maximum likelihood classifier further we now choose a particular probability model for the class conditional density function $p(\mathbf{x}|\omega_i)$. The most common choice is to assume $p(\mathbf{x}|\omega_i)$ is a multivariate normal distribution, also called a Gaussian distribution. This tacitly assumes that the classes of pixels of interest in spectral space are normally distributed. That is not necessarily a demonstrable property of natural spectral or information classes, but the Gaussian is a simple distribution to handle mathematically and its multivariate properties are well known. For an N dimensional space the specific form of the multivariate Gaussian distribution function is[4]

$$p(\mathbf{x}|\omega_i) = (2\pi)^{-N/2}|\mathbf{C}_i|^{-1/2}\exp\left\{-\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)\right\} \tag{8.6}$$

where $\mathbf{m}_i$ and $\mathbf{C}_i$ are the mean vector and covariance matrix of the data in class ω_i. We will sometimes write the normal distribution in the shorthand form $\mathcal{N}(\mathbf{x}|\mathbf{m}_i, \mathbf{C}_i)$.

Substituting (8.6) into (8.4) gives the discriminant function

$$g_i(\mathbf{x}) = -\tfrac{1}{2}N\ln 2\pi - \tfrac{1}{2}\ln|\mathbf{C}_i| - \tfrac{1}{2}(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i) + \ln p(\omega_i)$$

Since the first term is not class dependent is doesn't aid discrimination and can be removed, leaving as the discriminant function

$$g_i(\mathbf{x}) = \ln p(\omega_i) - \tfrac{1}{2}\ln|\mathbf{C}_i| - \tfrac{1}{2}(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i) \tag{8.7}$$

If the analyst has no useful information about the values of the prior probabilities they are assumed all to be equal. The first term in (8.7) is then ignored, allowing the ½ to be removed as well, leaving

$$g_i(\mathbf{x}) = -\ln|\mathbf{C}_i| - (\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i) \tag{8.8}$$

which is the discriminant function for the *Gaussian maximum likelihood classifier,* so-called because it essentially determines class membership of a pixel based on the highest of the class conditional probabilities, or likelihoods. Its implementation requires use of either (8.7) or (8.8) in (8.5). There is an important practical consideration concerning whether all the classes have been properly represented in the training data available. Because probability distributions like the Gaussian exist over the full domain of their argument (in this case the spectral measurements) use of (8.5) will yield a class label even in the remote tails of the distribution functions. In Sect. 8.3.5 the use of thresholds allows the user to avoid such inappropriate labelling and thereby to identify potentially missing classes.

[4] See Appendix D.

8.3.4 Decision Surfaces

The probability distributions in the previous section have been used to discriminate among the candidate class labels for a pixel. We can also use that material to see how the spectral space is segmented into regions corresponding to the classes. That requires finding the shapes of the separating boundaries in vector space. As will become evident, an understanding of the boundary shapes will allow us to assess the relative classification abilities, or strengths, of different classifiers.

Spectral classes are defined by those regions in spectral space where their discriminant functions are the largest. Those regions are separated by surfaces where the discriminant functions for adjoining spectral classes are equal. The *i*th and *j*th spectral classes are separated by the surface

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0$$

which is the actual equation of the surface. If all such surfaces were known then the class membership of a pixel can be made on the basis of its position in spectral space relative to the set of surfaces.

The construction $(\mathbf{x} - \mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x} - \mathbf{m}_i)$ in (8.7) and (8.8) is a quadratic function of $\mathbf{x}$. Consequently, the decision surfaces generated by Gaussian maximum likelihood classification are quadratic and thus take the form of parabolas, ellipses and circles in two dimensions, and hyperparaboloids, hyperellipsoids and hyperspheres in higher dimensions.

8.3.5 Thresholds

It is implicit in the rule of (8.1) and (8.5) that pixels at every point in spectral space will be classified into one of the available classes ω_i, irrespective of how small the actual probabilities of class membership are. This is illustrated for one dimensional data in Fig. 8.2a. Poor classification can result as indicated. Such situations can arise if spectral classes have been overlooked or, if knowing other classes existed, enough training data was not available to estimate the parameters of their distributions with any degree of accuracy (see Sect. 8.3.6 following).

In situations such as those it is sensible to apply thresholds to the decision process in the manner depicted in Fig. 8.2b. Pixels which have probabilities for all classes below the threshold are not classified. In practice, thresholds are applied to the discriminant functions and not the probability distributions, since the latter are never actually computed. With the incorporation of a threshold the decision rule of (8.5) becomes

$$\mathbf{x} \in \omega_i \quad \text{if} \quad g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ for all } j \neq i \tag{8.9a}$$

$$\text{and } g_i(\mathbf{x}) > T_i \tag{8.9b}$$

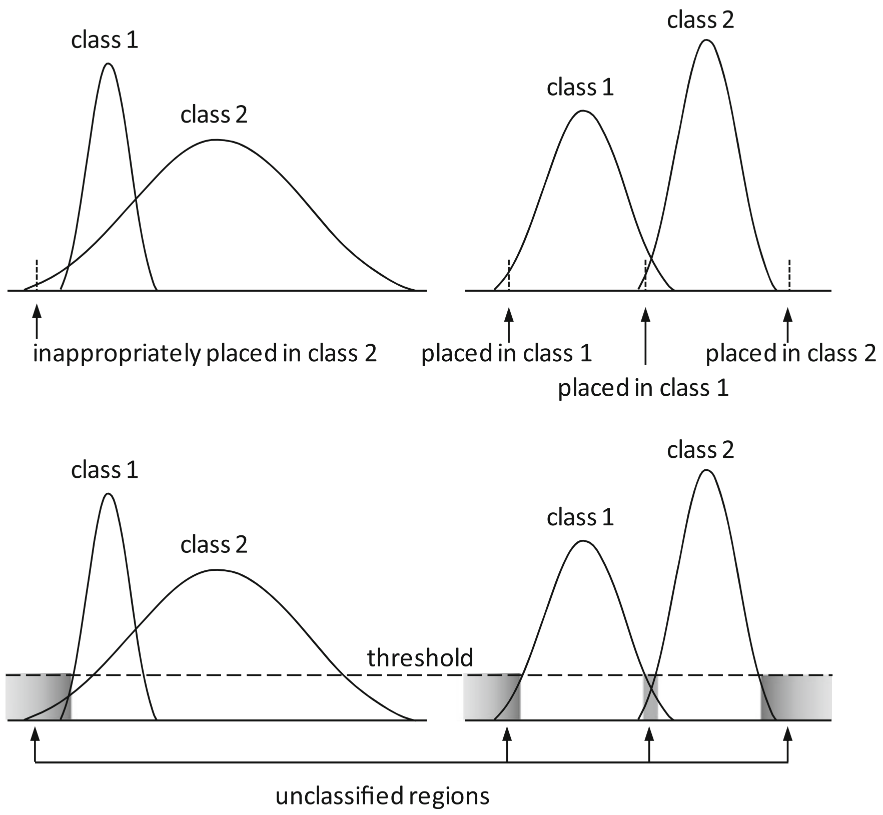


Fig. 8.2 Illustration of the use of thresholds to avoid questionable classification decisions

where T_i is the threshold to be used on class ω_i.

We now need to understand how a reasonable value for T_i can be chosen. Substituting (8.7) into (8.9b) gives

$$\ln p(\omega_i) - \tfrac{1}{2}\ln|\mathbf{C}_i| - \tfrac{1}{2}(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i) > T_i$$

or

$$(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i) < -2T_i + 2\ln p(\omega_i) - \ln|\mathbf{C}_i| \tag{8.10}$$

If $\mathbf{x}$ is normally distributed, the expression on the left hand side of (8.10) has a χ^2 distribution with N degrees of freedom,[5] where N is the number of bands in the data. We can, therefore, use the properties of the χ^2 distribution to determine that value of $(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)$ below which a desired percentage of pixels will

[5] See Swain and Davis, *loc. cit.*

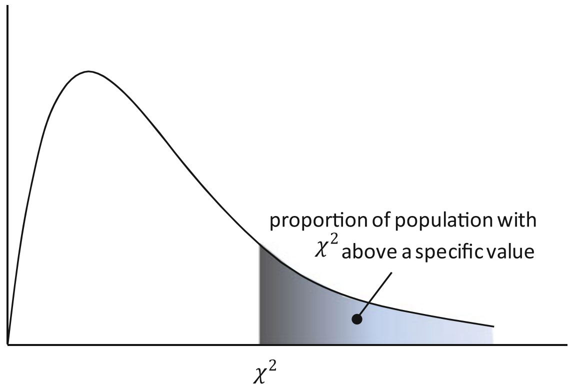


Fig. 8.3 Use of the χ^2 distribution for obtaining classifier thresholds

exist. Larger values of $(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)$ correspond to pixels lying further out in the tails of the normal distribution. That is shown in Fig. 8.3.

As an example of how this is used consider the need to choose a threshold such that 95% of all pixels in a class will be classified, or such that the 5% least likely pixels for the class will be rejected. From the χ^2 distribution we find that 95% of all pixels have χ^2 values below 9.488. Thus, from (8.10)

$$T_i = -4.744 + \ln p(\omega_{\mathrm{i}}) - {}^1\!/\!_2 \ln|\mathbf{C}_i|$$

which can be calculated from a knowledge of the prior probability and covariance matrix for the *i*th spectral class.

8.3.6 Number of Training Pixels Required

Enough training pixels for each spectral class must be available to allow reasonable estimates to be obtained of the elements of the class conditional mean vector and covariance matrix. For an N dimensional spectral space the mean vector has N elements. The covariance matrix is symmetric of size $N \times N$; it has ${}^1\!/\!_2 N(N+1)$ distinct elements that need to be estimated from the training data. To avoid the matrix being singular, at least $N(N+1)$ independent samples are needed. Fortunately, each N dimensional pixel vector contains N samples, one in each waveband; thus the minimum number of independent training *pixels* required is $(N+1)$. Because of the difficulty in assuring independence of the pixels, usually many more than this minimum number are selected. A practical minimum of $10N$ training pixels per spectral class is recommended, with as many as $100N$ per class if possible.[6] If the covariance matrix is well estimated then the mean vector will be also, because of its many fewer elements.

[6] See Swain and Davis, *loc. cit.*, although some authors regard this as a conservatively high number of samples.

For data with low dimensionality, say up to 10 bands, 10*N* to 100*N* can usually be achieved, but for higher order data sets, such as those generated by hyperspectral sensors, finding enough training pixels per class is often not practical, making reliable training of the traditional maximum likelihood classifier difficult. In such cases, dimensionality reduction procedures can be used, including methods for feature selection. They are covered in Chap. 10.

Another approach that can be adopted when not enough training samples are available is to use a classification algorithm with fewer parameters, several of which are treated in later sections. It is the need to estimate all the elements of the covariance matrix that causes problems for the maximum likelihood algorithm. The minimum distance classifier, covered in Sect. 8.5, depends only on knowing the mean vector for each training class—consequently there are only *N* unknowns to estimate.

8.3.7 The Hughes Phenomenon and the Curse of Dimensionality

Although recognised as a potential problem since the earliest application of computer-based interpretation to remotely sensed imagery, the Hughes phenomenon had not been a significant consideration until the availability of hyperspectral data. It concerns the number of training samples required per class to train a supervised classifier reliably and is thus related to the material of the previous section. Although that was focussed on Gaussian maximum likelihood classification, in principle any classifier that requires parameters to be estimated is subject to the problem.[7] It manifests itself in the following manner.

Clearly, if too few samples are available good estimates of the class parameters cannot be found; the classifier will not be properly trained and will not predict well on data from the same classes that it has not already seen. The extent to which a classifier predicts on previously unseen data, referred to as *testing data*, is called *generalisation*. Increasing the set of random training samples would be expected to improve parameter estimates, which indeed is generally the case, and thus lead to better generalisation.

Suppose now we have enough samples for a given number of dimensions to provide reliable training and good generalisation. Instead of changing the number of training samples, imagine now we can add more features, or dimensions, to the problem. On the face of it that suggests that the results should be improved because more features should bring more information into the training phase. However, if we don't increase the number of training samples commensurately we may not be able to get good estimates of the larger set of parameters created by the

[7] See M. Pal and G.F. Foody, Feature selection for classification of hyperspectral data for SVM, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 5, May 2010, pp. 2297–2307 for a demonstration of the Hughes phenomenon with the support vector machine of Sect. 8.14.

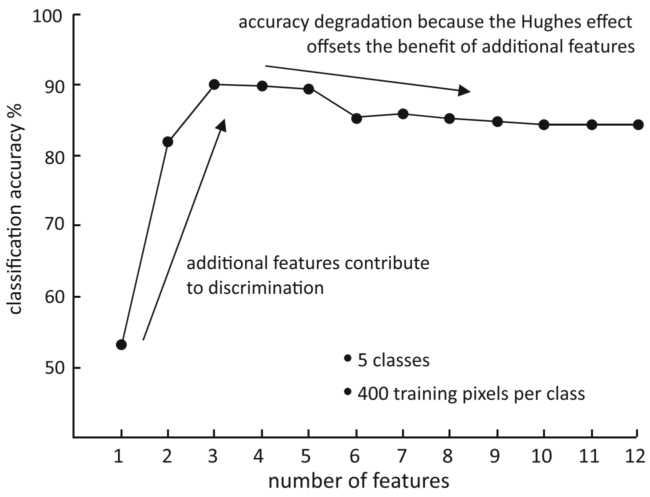


Fig. 8.4 Actual experimental classification results showing the Hughes phenomenon—the loss of classification accuracy as the dimensionality of the spectral domain increases

increased dimensionality, and both training and generalisation will suffer. That is the essence of the Hughes phenomenon.

The effect of poorly estimated parameters resulting from too few training samples with increased dimensionality was noticed very early in the history of remote sensing image classification. Figure 8.4 shows the results of a five class agricultural classification using 12 channel aircraft scanner data.[8] Classification accuracy is plotted as a function of the number of features (channels) used. As noted, when more features are added there is an initial improvement in classification accuracy until such time as the estimated maximum likelihood statistics apparently become less reliable.

As another illustration consider the determination of a reliable linear separating surface; here we approach the problem by increasing the number of training samples for a given dimensionality, but the principle is the same. Figure 8.5 shows three different training sets of data for the same two dimensional data set. The first has only one training pixel per class, and thus the same number of training pixels as dimensions. As seen, while a separating surface can be found it may not be accurate. The classifier performs at the 100% level on the training data but is very poor on testing data.

Having two training pixels per class, as in the case of the middle diagram, provides a better estimate of the separating surface, but it is not until we have many pixels per class compared to the number of channels, as in the right hand figure, that we obtain good estimates of the parameters of the supervised classifier, so that generalisation is good.

[8] Based on results presented in K.S. Fu, D.A. Landgrebe and T.L. Phillips, Information processing of remotely sensed agricultural data, *Proc. IEEE*, vol. 57, no. 4, April 1969, pp. 639–653.

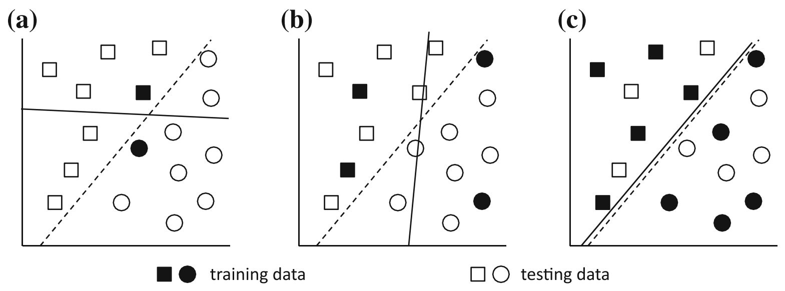


Fig. 8.5 Estimating a linear separating surface using an increasing number of training samples, showing poor generalisation if too few per number of features are used; the *dashed curve* represents the optimal surface

The Hughes phenomenon was first examined in 1968[9] and has become widely recognised as a major consideration whenever parameters have to be estimated in a high dimensional space. In the field of pattern recognition it is often referred to as the curse of dimensionality.[10]

8.3.8 An Example

As a simple example of the maximum likelihood approach, the 256×276 pixel segment of Landsat Multispectral Scanner image shown in Fig. 8.6 is to be classified. Four broad ground cover types are evident: water, fire burn, vegetation and urban. Assume we want to produce a thematic map of those four cover types in order to enable the distribution of the fire burn to be evaluated.

The first step is to choose training data. For such a broad classification, suitable sets of training pixels for each of the four classes are easily identified visually in the image data. The locations of four training fields for this purpose are seen in solid colours on the image. Sometimes, to obtain a good estimate of class statistics it may be necessary to choose several training fields for each cover type, located in different regions of the image, but that is not necessary in this case.

The signatures for each of the four classes, obtained from the training fields, are given in Table 8.1. The mean vectors can be seen to agree generally with the known spectral reflectance characteristics of the cover types. Also, the class variances, given by the diagonal elements in the covariance matrices, are small for water as might be expected but on the large side for the developed/urban class, indicative of its heterogeneous nature.

[9] G. F Hughes, On the mean accuracy of statistical pattern recognizers, *IEEE Transactions on Information Theory*, vol. IT-14, no.1, 1968, pp. 55–63.

[10] C.M. Bishop, *Pattern Recognition and Machine Learning,* Springer Science + Business Media, LLC, N.Y., 2006.

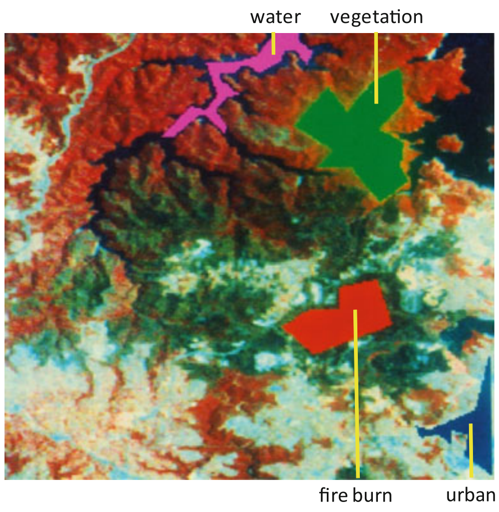


Fig. 8.6 Image segment to be classified, consisting of a mixture of natural vegetation, waterways, urban regions, and vegetation damaged by fire. Four training areas are identified in solid colour. They are water (*violet*), vegetation (*green*), fire burn (*red*) and urban (*dark blue in the bottom right hand corner*). Those pixels were used to generate the signatures in Table 8.1

Table 8.1 Class signatures generated from the training areas in Fig. 8.6

Class	Mean vector	Covariance matrix			
Water	44.27	14.36	9.55	4.49	1.19
	28.82	9.55	10.51	3.71	1.11
	22.77	4.49	3.71	6.95	4.05
	13.89	1.19	1.11	4.05	7.65
Fire burn	42.85	9.38	10.51	12.30	11.00
	35.02	10.51	20.29	22.10	20.62
	35.96	12.30	22.10	32.68	27.78
	29.04	11.00	20.62	27.78	30.23
Vegetation	40.46	5.56	3.91	2.04	1.43
	30.92	3.91	7.46	1.96	0.56
	57.50	2.04	1.96	19.75	19.71
	57.68	1.43	0.56	19.71	29.27
Urban	63.14	43.58	46.42	7.99	−14.86
	60.44	46.42	60.57	17.38	−9.09
	81.84	7.99	17.38	67.41	67.57
	72.25	−14.86	−9.09	67.57	94.27

When used in a maximum likelihood algorithm to classify the four bands of the image in Fig. 8.6, the signatures generate the thematic map of Fig. 8.7. The four classes, by area, are given in Table 8.2. Note that there are no unclassified pixels, since a threshold was not used in the labelling process. The area estimates are obtained by multiplying the number of pixels per class by the effective area of a pixel. In the case of the Landsat multispectral scanner the pixel size is 0.4424 hectare.

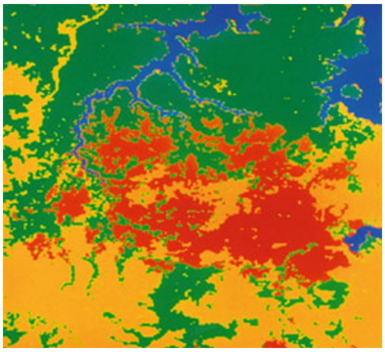


Fig. 8.7 Thematic map produced using the maximum likelihood classifier; *blue* represents water, *red* is fire damaged vegetation, *green* is natural vegetation and *yellow* is urban development

Table 8.2 Tabular summary of the thematic map of Fig. 8.7

Class	Number of pixels	Area (ha)
Water	4,830	2,137
Fire burn	14,182	6,274
Vegetation	28,853	12,765
Urban	22,791	10,083

8.4 Gaussian Mixture Models

In order that the maximum likelihood classifier algorithm work effectively it is important that the classes be as close to Gaussian as possible. In Sect. 8.2 that was stated in terms of knowing the number of spectral classes, sometimes also referred to as modes or sub-classes, of which each information class is composed. In Chap. 11 we will employ the unsupervised technique of clustering, to be developed in the next chapter, to help us do that. Here we present a different approach, based on the assumption that the image data is composed of a set of Gaussian distributions, some subsets of which may constitute information classes. This material is included at this point because it fits logically into the flow of the chapter, but it is a little more complex and could be passed over on a first reading.[11]

We commence by assuming that the pixels in a given image belong to a probability distribution that is a linear mixture of K Gaussian distributions of the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}|\mathbf{m}_k, \mathbf{C}_k) \tag{8.11}$$

The α_k are a set of mixture proportions, which are also considered to be the prior probabilities of each of the components that form the mixture. They satisfy

$$0 \le \alpha_k \le 1 \tag{8.12a}$$

[11] A very good treatment of Gaussian mixture models will be found in C.M. Bishop, *loc. cit.*

and

$$\sum_{k=1}^{K} \alpha_k = 1 \tag{8.12b}$$

Equation (8.11) is the probability of finding a pixel at position $\mathbf{x}$ in spectral space from the mixture. It is the mixture equivalent of the probability $p(\mathbf{x}|\omega_i)$ of finding a pixel at position $\mathbf{x}$ from the single class ω_i. Another way of writing the expression for the class conditional probability for a single distribution is $p(\mathbf{x}|\mathbf{m}_i, \mathbf{C}_i)$, in which the class has been represented by its parameters—the mean vector and covariance matrix—rather than by class label itself. Similarly, $p(\mathbf{x})$ in the mixture equation above could be shown as conditional on the parameters of the mixture explicitly, viz.

$$p(\mathbf{x}|\alpha_\mathbf{K}, \mathbf{m}_\mathbf{K}, \mathbf{C}_\mathbf{K}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}|\mathbf{m}_k, \mathbf{C}_k) \tag{8.13}$$

in which $\mathbf{m}_\mathbf{K}$ is the set of all K mean vectors, $\mathbf{C}_\mathbf{K}$ is the set of all K covariance matrices and $\boldsymbol{\alpha}_\mathbf{K}$ is the set of mixture parameters.

What we would like to do now is find the parameters in (8.13) that give the best match of the model to the data available. In what follows we assume we know, or have estimated, the number of components K; what we then have to find are the sets of mean vectors, covariance matrices and mixture proportions.

Suppose we have available a data set of J pixels $\mathbf{X} = \{\mathbf{x}_1 \ldots \mathbf{x}_J\}$ assumed to be independent samples from the mixture. Their joint likelihood can be expressed

$$p(\mathbf{X}|\boldsymbol{\alpha}_\mathbf{K}, \mathbf{m}_\mathbf{K}, \mathbf{C}_\mathbf{K}) = \prod_{j=1}^{J} p\left(\mathbf{x}_j|\boldsymbol{\alpha}_\mathbf{K}, \mathbf{m}_\mathbf{K}, \mathbf{C}_\mathbf{K}\right)$$

the logarithm[12] of which is

$$\ln p(\mathbf{X}|\boldsymbol{\alpha}_\mathbf{K}, \mathbf{m}_\mathbf{K}, \mathbf{C}_\mathbf{K}) = \sum_{j=1}^{J} \ln p\left(\mathbf{x}_j|\boldsymbol{\alpha}_\mathbf{K}, \mathbf{m}_\mathbf{K}, \mathbf{C}_\mathbf{K}\right) \tag{8.14}$$

We now want to find the parameters that will maximise the log likelihood that $\mathbf{X}$ comes from the mixture. In other words we wish to maximise (8.14) with respect to the members of the sets $\mathbf{m}_\mathbf{K} = \{\mathbf{m}_k, k = 1 \ldots K\}$, $\mathbf{C}_\mathbf{K} = \{\mathbf{C}_k, k = 1 \ldots K\}$, $\boldsymbol{\alpha}_\mathbf{K} = \{\alpha_k, k = 1 \ldots K\}$. Substituting (8.13) in (8.14) we have

$$\ln p(\mathbf{X}|\boldsymbol{\alpha}_\mathbf{K}, \mathbf{m}_\mathbf{K}, \mathbf{C}_\mathbf{K}) = \sum_{j=1}^{J} \ln \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k) \tag{8.15}$$

[12] As in Sect. 8.3, using the logarithmic expression simplifies the analysis to follow.

Differentiating this with respect to the mean vector of, say, the kth component, and equating the result to zero gives

$$\sum_{j=1}^{J}\frac{\partial}{\partial\mathbf{m}_k}\ln\sum_{k=1}^{K}\alpha_k\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k,\mathbf{C}_k)=0$$

$$\sum_{j=1}^{J}\frac{1}{\sum_{k=1}^{K}\alpha_k\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k,\mathbf{C}_k)}\frac{\partial}{\partial\mathbf{m}_k}\sum_{k=1}^{K}\alpha_k\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k,\mathbf{C}_k)=0 \tag{8.16}$$

$$\sum_{j=1}^{J}\frac{\alpha_k\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k,\mathbf{C}_k)}{\sum_{k=1}^{K}\alpha_k\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k,\mathbf{C}_k)}\frac{\partial}{\partial\mathbf{m}_k}\left\{-1/2(\mathbf{x}_j-\mathbf{m}_k)^{\mathrm{T}}\mathbf{C}_k^{-1}\left(\mathbf{x}_j-\mathbf{m}_k\right)\right\}=0$$

Using the matrix property[13] $\frac{\partial}{\partial\mathbf{x}}\{\mathbf{x}^T\mathbf{A}\mathbf{x}\}=2\mathbf{A}\mathbf{x}$ this can be written

$$\sum_{j=1}^{J}\frac{\alpha_k\,p(\mathbf{x}_j|\mathbf{m}_k,\mathbf{C}_k)}{p\left(\mathbf{x}_j|\alpha_\mathbf{k},\mathbf{m}_\mathbf{K},\mathbf{C}_\mathbf{K}\right)}\mathbf{C}_k^{-1}\left(\mathbf{x}_j-\mathbf{m}_k\right)\equiv\sum_{j=1}^{J}\frac{\alpha_k\,p(\mathbf{x}_j|\mathbf{m}_k,\mathbf{C}_k)}{p\left(\mathbf{x}_j\right)}\mathbf{C}_k^{-1}\left(\mathbf{x}_j-\mathbf{m}_k\right)=0$$

If we regard α_k as the prior probability of the kth component then the fraction just inside the sum will be recognised, from Bayes' Theorem, as $p(\mathbf{m}_k,\mathbf{C}_k|\mathbf{x}_j)$, the posterior probability for that component given the $\mathbf{x}_j$th sample,[14] so that the last equation becomes

$$\sum_{j=1}^{J}p(\mathbf{m}_k,\mathbf{C}_k|\mathbf{x}_j)\mathbf{C}_k^{-1}\left(\mathbf{x}_j-\mathbf{m}_k\right)=0$$

The inverse covariance matrix cancels out; rearranging the remaining terms gives

$$\mathbf{m}_k=\frac{\sum_{j=1}^{J}p(\mathbf{m}_k,\mathbf{C}_k|\mathbf{x}_j)\mathbf{x}_j}{\sum_{j=1}^{J}p(\mathbf{m}_k,\mathbf{C}_k|\mathbf{x}_j)}$$

The posterior probabilities in the denominator, $p\left(\mathbf{m}_k,\mathbf{C}_k|\mathbf{x}_j\right)$, express the likelihood that the correct component is k for the sample $\mathbf{x}_j$. It would be high if the sample belongs to that component, and low otherwise. The sum over all samples of the posterior probability for component k in the denominator is a measure of the effective number of samples likely to belong to that component,[15] and we define it as N_k:

---

[13] See K.B. Petersen and M.S. Pedersen, *The Matrix Cookbook*, 14 Nov 2008, http://matrixcookbook.com

[14] In C.M. Bishop, *loc. cit.*, it is called the *responsibility*.

[15] See Bishop, *loc. cit.*

$$N_k = \sum_{j=1}^{J} p(\mathbf{m}_k, \mathbf{C}_k|\mathbf{x}_j) \tag{8.17}$$

Note that the numerator in the expression for $\mathbf{m}_k$ above is a sum over all the samples weighted by how likely it is that the kth component is the correct one in each case. The required values of the $\mathbf{m}_k$ that maximise the log likelihood in (8.15) can now be written

$$\mathbf{m}_k = \frac{1}{N_k}\sum_{j=1}^{J} p(\mathbf{m}_k, \mathbf{C}_k|\mathbf{x}_j)\mathbf{x}_j \tag{8.18}$$

To find the covariance matrices of the components that maximise (8.15) we proceed in a similar manner by taking the first derivative with respect to $\mathbf{C}_k$ and equating the result to zero. Proceeding as before we have, similar to (8.16),

$$\sum_{j=1}^{J} \frac{\alpha_k}{\sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k)} \frac{\partial}{\partial \mathbf{C}_k}\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k) = 0 \tag{8.19}$$

The derivative of the normal distribution with respect to its covariance matrix is a little tedious but relatively straightforward in view of two more useful matrix calculus properties[16]:

$$\frac{\partial}{\partial \mathbf{M}}|\mathbf{M}| = |\mathbf{M}|(\mathbf{M}^{-1})^{\mathrm{T}} \tag{8.20a}$$

$$\frac{\partial}{\partial \mathbf{M}}\mathbf{a}^{\mathrm{T}}\mathbf{M}^{-1}\mathbf{a} = -(\mathbf{M}^{-1})^{\mathrm{T}}\mathbf{a}\mathbf{a}^{\mathrm{T}}(\mathbf{M}^{-1})^{\mathrm{T}} \tag{8.20b}$$

Using these we find

$$\frac{\partial}{\partial \mathbf{C}_k}\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k) = \mathcal{N}\left(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k\right)\left\{(\mathbf{x}_j - \mathbf{m}_k)(\mathbf{x}_j - \mathbf{m}_k)^{\mathrm{T}}(\mathbf{C}_k^{-1})^{\mathrm{T}} - 1\right\}$$

When substituted in (8.19) this gives

$$\sum_{j=1}^{J} \frac{\alpha_k \mathcal{N}\left(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k\right)}{\sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k)}\left\{(\mathbf{x}_j - \mathbf{m}_k)(\mathbf{x}_j - \mathbf{m}_k)^{\mathrm{T}}(\mathbf{C}_k^{-1})^{\mathrm{T}} - 1\right\} = 0$$

Recognising as before that the fraction just inside the summation is the posterior probability, and multiplying throughout by $\mathbf{C}_k^{-1}$, and recalling that the covariance matrix is symmetric, we have

[16] See K.B. Petersen and M.S. Pedersen, *loc. cit.*

$$\sum_{j=1}^{J} p(\mathbf{m}_k, \mathbf{C}_k|\mathbf{x}_j)\left\{(\mathbf{x}_j - \mathbf{m}_k)(\mathbf{x}_j - \mathbf{m}_k)^{\mathrm{T}} - \mathbf{C}_{\mathrm{k}}\right\} = 0$$

so that, with (8.17), this gives

$$\mathbf{C}_k = \frac{1}{N_k}\sum_{j=1}^{J} p(\mathbf{m}_k, \mathbf{C}_k|\mathbf{x}_j)\left\{(\mathbf{x}_j - \mathbf{m}_k)(\mathbf{x}_j - \mathbf{m}_k)^{\mathrm{T}}\right\} \tag{8.21}$$

which is the expression for the covariance matrix of the *k*th component that will maximise (8.15). Note how similar this is in structure to the covariance matrix definition for a single Gaussian component in (6.3). In (8.21) the terms are summed with weights that are the likelihoods that the samples come from the *k*th component. If there were only one component then the weight would be unity, leading to (6.3).

The last task is to find the mixing parameter of the *k*th component that contributes to maximising the log likelihood of (8.15). The α_k are also constrained by (8.12b). Therefore we maximise the following expression with respect to α_k using Lagrange multipliers λ to constrain the maximisation[17]:

$$\sum_{j=1}^{J} \ln \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k) + \lambda\left\{\sum_{k=1}^{K}\alpha_k - 1\right\}$$

Putting the first derivative with respect to α_k to zero gives

$$\sum_{j=1}^{J} \frac{\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k)}{\sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k)} + \lambda = 0$$

which, using (8.13), is

$$\sum_{j=1}^{J} \frac{\mathcal{N}(\mathbf{x}_j|\mathbf{m}_k, \mathbf{C}_k)}{p\left(\mathbf{x}_j|\alpha_{\mathbf{k}}, \mathbf{m}_{\mathbf{K}}, \mathbf{C}_{\mathbf{K}}\right)} + \lambda = 0$$

and, using Bayes' Theorem

$$\sum_{j=1}^{J} \frac{p\left(\mathbf{m}_k, \mathbf{C}_k|\mathbf{x}_j\right)}{\alpha_k} + \lambda = 0$$

Multiplying throughout by α_k and using (8.17) this gives

$$N_k + \alpha_k\lambda = 0 \tag{8.22}$$

[17] For a good treatment of Lagrange multipliers see C.M. Bishop, *loc. cit.*, Appendix E

so that if we take the sum $$\sum_{k=1}^{K}\{N_k + \alpha_k\lambda\} = 0$$

we have, in view of (8.12b) $$\lambda = -N$$

Thus, from (8.22), $$\alpha_k = \frac{N_k}{N} \tag{8.23}$$

With (8.18), (8.21) and (8.23) we now have expressions for the parameters of the Gaussian mixture model that maximise the log likelihood function of (8.14). Unfortunately, though, they each depend on the posterior probabilities $p(\mathbf{m}_k, \mathbf{C}_k|\mathbf{x}_j)$ which themselves are dependent on the parameters. The results of (8.18), (8.21) and (8.23) do not therefore represent solutions to the Gaussian mixing problem. However, an iterative procedure is available that allows the parameters and the posteriors to be determined progressively. It is called *Expectation–Maximisation* (EM) and is outlined in the algorithm of Table 8.3. It consists of an initial guess for the parameters, following which values for the posteriors are computed—that is called the expectation step. Once the posteriors have been computed, new estimates for the parameters are computed from (8.18), (8.21) and (8.23). That is the maximisation step. The loop is repeated until the parameters don't change any further. It is also helpful at each iteration to estimate the log likelihood of (8.14) and use it to check convergence.

8.5 Minimum Distance Classification

8.5.1 The Case of Limited Training Data

The effectiveness of maximum likelihood classification depends on reasonably accurate estimation of the mean vector **m** and the covariance matrix **C** for each spectral class. This in turn depends on having a sufficient number of training pixels for each of those classes. In cases where that is not possible, inaccurate estimates of the elements of **C** result, leading to poor classification.

When the number of training samples per class is limited it may sometimes be more effective to resort to an algorithm that does not make use of covariance information but instead depends only on the mean positions of the spectral classes, noting that for a given number of samples they can be more accurately estimated than covariances. The minimum distance classifier, or more precisely, minimum distance to class means classifier, is such an approach. With this algorithm, training data is used only to determine class means; classification is then performed by placing a pixel in the class of the nearest mean.

Table 8.3 The Expectation–Maximisation Algorithm for Mixtures of Gaussians

Step 1: Initialisation	Choose initial values for $\alpha_k, \mathbf{m}_k, \mathbf{C}_k$. Call these $\alpha_k^{\text{old}}, \mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}}$
Step 2: Expectation	Evaluate the posterior probabilities using the current estimates for $\alpha_k, \mathbf{m}_k, \mathbf{C}_k$, according to: $p(\mathbf{m}_k, \mathbf{C}_k\|\mathbf{x}_j) = \dfrac{\alpha_k^{\text{old}} p(\mathbf{x}_j\|\mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}})}{p(\mathbf{x}_j\|\boldsymbol{\alpha}_\mathbf{K}^{\mathbf{old}}, \mathbf{m}_\mathbf{K}^{\mathbf{old}}, \mathbf{C}_\mathbf{K}^{\mathbf{old}})}$ i.e. $p(\mathbf{m}_k, \mathbf{C}_k\|\mathbf{x}_j) = \dfrac{\alpha_k^{\text{old}} p(\mathbf{x}_j\|\mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}})}{\sum_{k=1}^{K} \alpha_k^{\text{old}} \mathcal{N}(\mathbf{x}_j\|\mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}})}$
Step 3: Maximisation	Compute new values for $\alpha_k, \mathbf{m}_k, \mathbf{C}_k$ from (8.18), (8.21) and (8.23) in the following way: First compute $N_k = \sum_{j=1}^{J} p(\mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}}\|\mathbf{x}_j)$ then $\mathbf{m}_k^{\text{new}} = \dfrac{1}{N_k} \sum_{j=1}^{J} p(\mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}}\|\mathbf{x}_j)\mathbf{x}_j$ $\mathbf{C}_k^{\text{new}} = \dfrac{1}{N_k} \sum_{j=1}^{J} p(\mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}}\|\mathbf{x}_j)\{(\mathbf{x}_j - \mathbf{m}_k^{\text{new}})(\mathbf{x}_j - \mathbf{m}_k^{\text{new}})^{\text{T}}\}$ $\alpha_k^{\text{new}} = \dfrac{N_k}{N}$ Also, evaluate the log likelihood $\ln p(\mathbf{X}\|\boldsymbol{\alpha}_\mathbf{k}, \mathbf{m}_\mathbf{K}, \mathbf{C}_\mathbf{K}) = \sum_{j=1}^{J} \ln \sum_{k=1}^{K} \alpha_k^{\text{old}} \mathcal{N}(\mathbf{x}_j\|\mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{old}})$
Step 4: Evaluation	If $\mathbf{m}_k^{\text{new}} \approx \mathbf{m}_k^{\text{old}}, \mathbf{C}_k^{\text{new}} \approx \mathbf{C}_k^{\text{old}}, \alpha_k^{\text{new}} \approx \alpha_k^{\text{old}}$ then terminate the process. Otherwise put $\mathbf{m}_k^{\text{old}} = \mathbf{m}_k^{\text{new}}, \mathbf{C}_k^{\text{old}} = \mathbf{C}_k^{\text{new}}, \alpha_k^{\text{old}} = \alpha_k^{\text{new}}$ and return to Step 2. This check can also be carried out on the basis of little or no change in the log likelihood calculation

The minimum distance algorithm is also attractive because it is faster than maximum likelihood classification, as will be seen in Sect. 8.5.6. However, because it does not use covariance data it is not as flexible. In maximum likelihood classification each class is modelled by a multivariate normal class model that can account for spreads of data in particular spectral directions. Since covariance data is not used in the minimum distance technique class models are symmetric in the spectral domain. Elongated classes will, therefore, not be well modelled. Instead several spectral classes may need to be used with this algorithm in cases where one might be suitable for maximum likelihood classification.

8.5.2 The Discriminant Function

Suppose $\mathbf{m}_i, i = 1\ldots M$ are the means of the M classes determined from training data, and $\mathbf{x}$ is the position of the pixel in spectral space to be classified. Compute the set of squared Euclidean distances of the unknown pixel to each of the class means:

$$d(\mathbf{x},\mathbf{m}_i)^2 = (\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}(\mathbf{x}-\mathbf{m}_i) = (\mathbf{x}-\mathbf{m}_i)\cdot(\mathbf{x}-\mathbf{m}_i)\ i = 1\ldots M$$

Expanding the dot product form gives

$$d(\mathbf{x},\mathbf{m}_i)^2 = \mathbf{x}\cdot\mathbf{x} - 2\mathbf{m}_i\cdot\mathbf{x} + \mathbf{m}_i\cdot\mathbf{m}_i$$

Classification is performed using the decision rule

$$\mathbf{x}\in\omega_i \text{ if } d(\mathbf{x},\mathbf{m}_i)^2 < d(\mathbf{x},\mathbf{m}_j)^2 \text{ for all } j\neq i$$

Since $\mathbf{x}\cdot\mathbf{x}$ is common to all squared distance calculations it can be removed from both sides in the decision rule. The sign of the remainder can then be reversed so that the decision rule can be written in the same way as (8.5) to give

$$\mathbf{x}\in\omega_i \text{ if } g_{\mathrm{i}}(\mathbf{x}) > g_{\mathrm{j}}(\mathbf{x}) \text{ for all } j\neq i$$

in which

$$g_{\mathrm{i}}(\mathbf{x}) = 2\mathbf{m}_i\cdot\mathbf{x} - \mathbf{m}_i\cdot\mathbf{m}_i \tag{8.24}$$

Equation (8.24) is the discriminant function for the minimum distance classifier.[18]

8.5.3 Decision Surfaces for the Minimum Distance Classifier

The implicit surfaces in spectral space separating adjacent classes are defined by the respective discriminant functions being equal. The surface between the *i*th and *j*th spectral classes is given by

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0$$

which, when substituting from (8.24), gives

$$2\left(\mathbf{m}_i-\mathbf{m}_j\right)\cdot\mathbf{x} - \left(\mathbf{m}_i\cdot\mathbf{m}_i - \mathbf{m}_j\cdot\mathbf{m}_j\right) = 0$$

This defines a linear surface, called a *hyperplane* in more than three dimensions. The surfaces between each pair of classes define a set of first degree separating hyperplanes that partition the spectral space linearly. The quadratic decision

[18] It is possible to implement a minimum distance classifier using distance measures other than Euclidean: see A.G. Wacker and D.A. Landgrebe, Minimum distance classification in remote sensing, *First Canadian Symposium on Remote Sensing*, Ottawa, 1972.

surface generated by the maximum likelihood rule in Sect. 8.3.4 renders that algorithm potentially more powerful than the minimum distance rule if properly trained; the minimum distance classifier nevertheless is effective when the number of training samples is limited or if linear separation of the classes is suspected.

8.5.4 Thresholds

Thresholds can be applied to minimum distance classification by ensuring not only that a pixel is closest to a candidate class but also that it is within a prescribed distance of that class in spectral space. Such an assessment is often used with the minimum distance rule. The distance threshold is usually specified in terms of a number of standard deviations from the class mean.

8.5.5 Degeneration of Maximum Likelihood to Minimum Distance Classification

The major difference between the minimum distance and maximum likelihood classifiers lies in the use, by the latter, of the sample covariance information. Whereas the minimum distance classifier labels a pixel as belonging to a particular class on the basis only of its distance from the relevant mean in spectral space, irrespective of its direction from that mean, the maximum likelihood classifier modulates its decision with direction, based on the information contained in the covariance matrix. Furthermore, the entry $\frac{1}{2}\ln|\mathbf{C}_i|$ in its discriminant function shows explicitly that patterns have to be closer to some means than others to have equivalent likelihoods of class membership. As a result, superior performance is expected of the maximum likelihood classifier. The following situation however warrants consideration since there is then no advantage in maximum likelihood procedures. It could occur in practice when class covariance is dominated by systematic noise rather than by the natural spectral spreads of the individual spectral classes.

Consider the covariance matrices of all classes to be diagonal and equal, and the variances in each component to be identical, so that

$$\mathbf{C}_i = \sigma^2\mathbf{I} \text{ for all } i$$

in which $\mathbf{I}$ is the identity matrix. The discriminant function for the maximum likelihood classifier in (8.7) then becomes

$$g_i(\mathbf{x}) = -\frac{1}{2}\ln\sigma^{2N} - \frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}(\mathbf{x}-\mathbf{m}_i) + \ln p(\omega_i)$$

The first term on the right hand side is common to all discriminant functions and can be ignored. The second term can be expanded in dot product form in which the resulting $\mathbf{x}\cdot\mathbf{x}$ terms can also be ignored, leaving

$$g_i(\mathbf{x}) = \frac{1}{2\sigma^2}(2\mathbf{m}_i\cdot\mathbf{x} - \mathbf{m}_i\cdot\mathbf{m}_i) + \ln p(\omega_i)$$

If all the prior probabilities are assumed to be equal then the last term can be ignored, allowing $1/2\sigma^2$ to be removed as a common factor, leaving

$$g_i(\mathbf{x}) = 2\mathbf{m}_i\cdot\mathbf{x} - \mathbf{m}_i\cdot\mathbf{m}_i$$

which is the same as (8.24), the discriminant function for the minimum distance classifier. Therefore, minimum distance and maximum likelihood classification are the same for identical and symmetric spectral class distributions.

8.5.6 Classification Time Comparison of the Maximum Likelihood and Minimum Distance Rules

For the minimum distance classifier the discriminant function in (8.24) must be evaluated for each pixel and each class. In practice $2\mathbf{m}_i$ and $\mathbf{m}_i\cdot\mathbf{m}_i$ would be calculated beforehand, leaving the computation of N multiplications and N additions to check the potential membership of a pixel to one class, where N is the number of components in $\mathbf{x}$. By comparison, evaluation of the discriminant function for maximum likelihood classification in (8.7) requires $N^2 + N$ multiplications and $N^2 + 2N + 1$ additions to check a pixel against one class, given that $\ln p(\omega_i) - \frac{1}{2}\ln|\mathbf{C}_i|$ would have been calculated beforehand. Ignoring additions by comparison to multiplications, the maximum likelihood classifier takes $N + 1$ times as long as the minimum distance classifier to perform a classification. It is also significant to note that classification time, and thus cost, increases quadratically with the number of spectral components for the maximum likelihood classifier, but only linearly for minimum distance classification.

8.6 Parallelepiped Classification

The parallelepiped classifier is a very simple supervised classifier that is trained by finding the upper and lower brightness values in each spectral dimension. Often that is done by inspecting histograms of the individual spectral components in the available training data, as shown in Fig. 8.8. Together the upper and lower bounds in each dimension define a multidimensional box or *parallelepiped*; Fig. 8.9 shows a set of two dimensional parallelepipeds. Unknown pixels are labelled as coming from the class of the parallelepiped within which they lie.

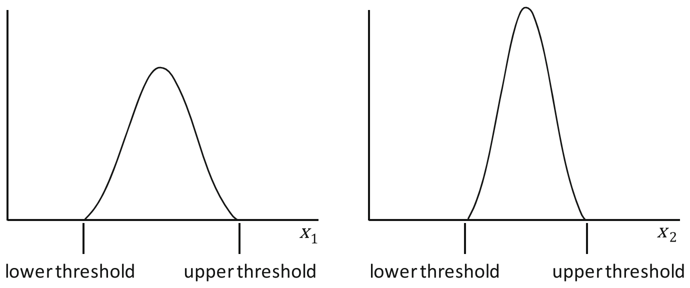


Fig. 8.8 Setting the parallelepiped boundaries by inspecting class histograms in each band

Fig. 8.9 A set of two dimensional parallelepipeds

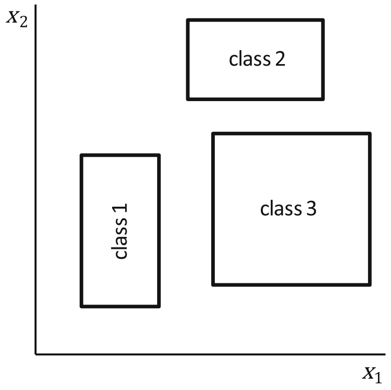
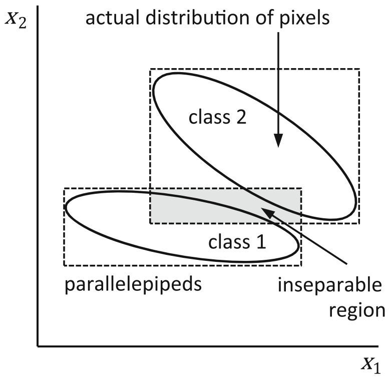


Fig. 8.10 Classification of correlated data showing regions of inseparability



While it is a very simple, and fast, classifier, it has several limitations. First, there can be considerable gaps between the parallelepipeds in spectral space; pixels in those regions cannot be classified. By contrast, the maximum likelihood and minimum distance rules will always label unknown pixels unless thresholds are applied. Secondly, for correlated data some parallelepipeds can overlap, as illustrated in Fig. 8.10, because their sides are always parallel to the spectral axes.

As a result, there are some parts of the spectral domain that can't be separated. Finally, as with the minimum distance classifier, there is no provision for prior probability of class membership with the parallelepiped rule.

8.7 Mahalanobis Classification

Consider the discriminant function for the maximum likelihood classifier for the special case of equal prior probabilities in (8.8). If the sign is reversed the function can be considered as a distance squared measure because the quadratic entry has those dimensions and the other term is a constant. Thus we can define

$$\mathrm{d}(\mathbf{x}, \mathbf{m}_i)^2 = \ln|\mathbf{C}_i| + (\mathbf{x} - \mathbf{m}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x} - \mathbf{m}_i) \tag{8.25}$$

and classify an unknown pixel on the basis of the smallest $\mathrm{d}(\mathbf{x}, \mathbf{m}_{\mathrm{i}})$, as for the Euclidean minimum distance classifier. Thus the maximum likelihood classifier (with equal priors) can be considered as a minimum distance algorithm but with a distance measure that is sensitive to direction in spectral space.

Assume now that all class covariances are the same and given by $\mathbf{C}_i = \mathbf{C}$ for all i. The $\ln|\mathbf{C}|$ term is now not discriminating and can be ignored. The squared distance measure then reduces to

$$\mathrm{d}(\mathbf{x}, \mathbf{m}_i)^2 = (\mathbf{x} - \mathbf{m}_i)^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m}_i) \tag{8.26}$$

A classifier using this simplified distance measure is called a *Mahalanobis classifier* and the distance measure shown squared in (8.26) is called the *Mahalanobis distance*. Under the additional simplification of $\mathbf{C} = \sigma^2\mathbf{I}$ the Mahalanobis classifier reduces to the minimum Euclidean distance classifier of Sect. 8.5.

The advantage of the Mahalanobis classifier over the maximum likelihood procedure is that it is faster and yet retains a degree of direction sensitivity via the covariance matrix $\mathbf{C}$, which could be a class average or a pooled covariance.

8.8 Non-parametric Classification

Classifiers such as the maximum likelihood and minimum distance rules are often called *parametric* because the class definitions and discriminant functions are defined in terms of sets of parameters, such as the mean vectors and covariance matrices for each class.

One of the valuable aspects of the parametric, statistical approach is that a set of relative likelihoods is produced. Even though, in the majority of cases, the maximum of the likelihoods is chosen to indicate the most probable label for a pixel, there exists nevertheless information in the remaining likelihoods that could be made use of in some circumstances, either to initiate processes such as relaxation labelling (Sect. 8.20.4) and Markov random fields (Sect. 8.20.5) or simply to provide the user with some feeling for the other likely classes. Those situations are not common however and, in most remote sensing applications, the maximum selection is made. That being so, the material in Sects. 8.3.4 and 8.5.3 shows that the decision process has a geometric counterpart in that a comparison of statistically derived discriminant functions leads equivalently to a decision rule that allows a pixel to be classified on the basis of its position in spectral space compared with the location of a decision surface.

There are also classifiers that, in principle, don't depend on parameter sets and are thus called *non-parametric*. Two simple non-parametric methods are given in Sects. 8.9 and 8.10, although the table look up approach of Sect. 8.9 is now rarely used.

Non-parametric methods based on finding geometric decision surfaces were popular in the very early days of pattern recognition[19] but fell away over the 1980s and 1990s because flexible training algorithms could not be found. They have, however, been revived in the past two decades with the popularity of the support vector classifier and the neural network, which we treat in Sects. 8.14 and 8.19 respectively, after we explore the fundamental problems with non-parametric approaches.

8.9 Table Look Up Classification

Since the set of discrete brightness values that can be taken by a pixel in each spectral band is limited by the radiometric resolution of the data, there is a finite, although often very large, number of distinct pixel vectors in any particular image. For a given class there may not be very many different pixel vectors if the radiometric resolution is not high, as was the case with the earliest remote sensing instruments. In such situations a viable classification scheme is to note the set of pixel vectors corresponding to a given class based on representative training data, and then use those vectors to classify the image by comparing unknown pixels with each pixel in the training data until a match is found. No arithmetic operations are required and, notwithstanding the number of comparisons that might be necessary to determine a match, it is a fast classifier. It is referred to as a *look up table* approach since the training pixel brightnesses are stored in tables that point to the corresponding classes.

An obvious drawback with this technique is that the chosen training data must contain an example of every possible pixel vector for each class. Should some be missed then the corresponding pixels in the image will be left unclassified. With modern data sets, in which there can be billions of individual data vectors, this approach is impractical.

[19] See N.J. Nilsson, *Learning Machines*, McGraw-Hill, N.Y., 1965.

8.10 *k*NN (Nearest Neighbour) Classification

A classifier that is particularly simple in concept, but can be time consuming to apply, is the k Nearest Neighbour classifier. It assumes that pixels close to each other in spectral space are likely to belong to the same class. In its simplest form an unknown pixel is labelled by examining the available training pixels in the spectral domain and choosing the class most represented among a pre-specified number of nearest neighbours. The comparison essentially requires the distances from the unknown pixel to all training pixels to be computed.

Suppose there are k_i neighbours labelled as class ω_i among the k nearest neighbours of a pixel vector $\mathbf{x}$, noting that $\sum\limits_{i=1}^{M} k_i = k$ where M is the total number of classes. In the basic kNN rule we define the discriminant function for the ith class as

$$g_i(\mathbf{x}) = k_i$$

and the decision rule as $\quad \mathbf{x} \in \omega_i \text{ if } g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ for all } j \neq i$

The basic rule does not take the distance from each neighbour to the current pixel vector into account, and may lead to tied results. An improvement is to distance-weight and normalise the discriminant function:

$$g_i(\mathbf{x}) = \frac{\sum_{j=1}^{k_i} 1/d(\mathbf{x}, \mathbf{x}_i^j)}{\sum_{i=1}^{M} \sum_{j=1}^{k_i} 1/d(\mathbf{x}, \mathbf{x}_i^j)} \tag{8.27}$$

in which $d(\mathbf{x}, \mathbf{x}_i^j)$ is the distance (usually Euclidean) from the unknown pixel vector $\mathbf{x}$ to its neighbour $\mathbf{x}_i^j$, the jth of the k_i pixels in class ω_i.

If the training data for each class is not in proportion to its respective population $p(\omega_i)$ in the image, a Bayesian version of the simple nearest neighbour discriminant function is

$$g_i(\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{\sum_{i=1}^{M} p(\mathbf{x}|\omega_i)p(\omega_i)} = \frac{k_i p(\omega_i)}{\sum_{i=1}^{M} k_i p(\omega_i)} \tag{8.28}$$

In the kNN algorithm as many spectral distances as there are training pixels must be evaluated for each unknown pixel to be labelled. That requires an impractically high computational load, particularly when the number of spectral bands and/or the number of training samples is large. The method is not well-suited therefore to hyperspectral datasets, although it is possible to improve the efficiency of the distance search process.[20]

[20] See B.V. Dasarathy, *Nearest neighbour (NN) norms*: *NN Pattern Classification Techniques*, IEEE Computer Society Press, Los Alamitos, California, 1991.

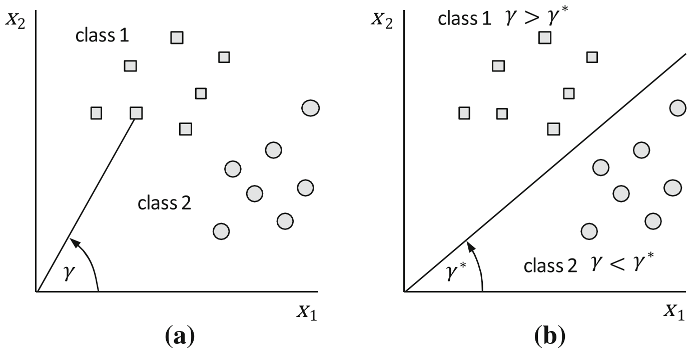


Fig. 8.11 **a** Representing pixel vectors by their angles; **b** segmenting the spectral space by angle

8.11 The Spectral Angle Mapper

A classifier sometimes used with data of high spectral dimensionality, such as that recorded by imaging spectrometers, is the spectral angle mapper[21] (SAM) which segments the spectral domain on the basis of the angles of vectors measured from the origin, as illustrated in Fig. 8.11 for two dimensions. Every pixel point in spectral space has both a magnitude and angular direction when expressed in polar, as against the more usual Cartesian, form. The decision boundary shown in Fig. 8.11b is based on the best angular separation between the training pixels in different classes, usually in an average sense. Only first order parameters are estimated, putting the SAM in the same class as the minimum distance classifier in terms of its suitability for high dimensional imagery.

8.12 Non-Parametric Classification from a Geometric Basis

8.12.1 The Concept of a Weight Vector

Consider the simple two class spectral space shown in Fig. 8.12, which has been constructed intentionally so that a simple straight line can be drawn between the pixels as shown. This straight line will be a multidimensional linear surface, or hyperplane in general, and will function as a decision surface for classification. In the two dimensions shown, the equation of the line can be expressed

$$w_1x_1 + w_2x_2 + w_3 = 0$$

[21] F.A. Kruse, A.B. Letkoff, J.W. Boardman, K.B. Heidebrecht, A.T. Shapiro, P.J. Barloon and A.F.H. Goetz, The spectral image processing system (SIPS)—interactive visualization and analysis of imaging spectrometer data, *Remote Sensing of Environment*, vol. 44, 1993, pp. 145–163.

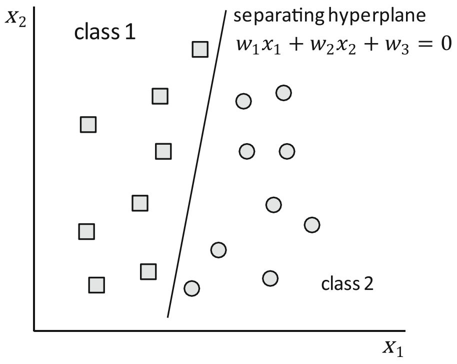


Fig. 8.12 Two dimensional spectral space, with two classes that can be separated by a simple linear surface

where the x_i are the brightness value coordinates in the spectral space and the w_i are a set of coefficients, usually called *weights*. There will be as many weights as the number of channels in the data, plus one. If the number of channels or bands is N, the equation of a general linear surface is

$$w_1x_1 + w_2x_2 + \ldots + w_Nx_N + w_{N+1} = 0$$

which can be written $$\mathbf{w}^T\mathbf{x} + w_{N+1} \equiv \mathbf{w} \cdot \mathbf{x} + w_{N+1} = 0 \qquad (8.29)$$

where **x** is the pixel measurement vector and **w** is called the *weight vector*. The transpose operation has the effect of turning the column vector into a row vector.

The position of the separating hyperplane would generally not be known initially and it would have to be found by training based on sets of reference pixels, just as the parameters of the maximum likelihood classifier are found by training. Note that there is not a unique solution; inspection of Fig. 8.12 suggests that any one of an infinite number of slightly different hyperplanes would be acceptable.

8.12.2 Testing Class Membership

The calculation in (8.29) will be zero only for values of **x** lying exactly on the hyperplane (the decision surface). If we substitute into that equation values of **x** corresponding to any of the pixel points shown in Fig. 8.12 the left hand side will be non-zero. Pixels in one class will generate a positive inequality, while pixels in the other class will generate a negative inequality. Once the decision surface, or more specifically the weights **w** that define its equation, has been found through training then a decision rule for labelling unknown pixels is

$$\begin{aligned}\mathbf{x} &\in \text{class 1 if } \mathbf{w}^T\mathbf{x} + w_{N+1} > 0\\ \mathbf{x} &\in \text{class 2 if } \mathbf{w}^T\mathbf{x} + w_{N+1} < 0\end{aligned} \qquad (8.30)$$

8.13 Training a Linear Classifier

The simple linear classifier in Fig. 8.12 can be trained in several ways. A particularly simple approach is to choose the hyperplane as the perpendicular bisector of the line between the mean vectors of the two classes. Effectively, that is the minimum distance to class means classifier of Sect. 8.5. Another is to guess an initial position for the separating hyperplane and then iterate it into position by reference, repetitively, to each of the training samples. Such a method has been known for almost 50 years.[22] There is however a more elegant training method that is the basis of the support vector classifier treated in the next section.

8.14 The Support Vector Machine: Linearly Separable Classes

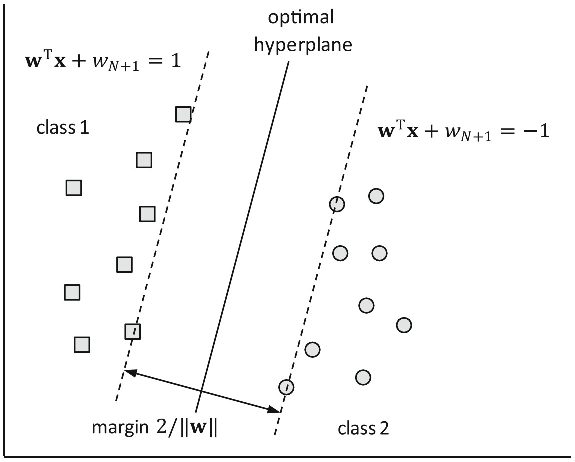
Inspection of Fig. 8.12 suggests that the only training patterns that need to be considered in finding a suitable hyperplane are those from each class nearest the hyperplane. Effectively, they are the patterns closest to the border between the classes. If a hyperplane can be found that satisfies those pixels then the pixels further from the border must, by definition, also be satisfied. Moreover, again by inspecting Fig 8.12, we can induce that the "best" hyperplane would be that which would be equidistant, on the average, between the bordering pixels for each of the two classes. This concept, along with the concentration just on the border pixels, forms the basis of the support vector machine, which was introduced into remote sensing in 1998.[23]

If we expand the region in the vicinity of the hyperplane in Fig. 8.12 we can see that the optimal position and orientation of the separating hyperplane is when there is a maximum separation between the patterns of the two classes in the manner illustrated in Fig. 8.13. We can draw two more hyperplanes, parallel to the separating hyperplane, that pass through the nearest training pixels from the classes, as indicated. We call them marginal hyperplanes. Their equations are shown on the figure, suitably scaled so that the right hand sides have a magnitude of unity. For pixels that lie beyond the marginal hyperplanes we have

$$\begin{aligned} &\text{for class 1 pixels } \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{N+1} \geq 1 \\ &\text{for class 2 pixels } \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{N+1} \leq -1 \end{aligned} \tag{8.31}$$

[22] See Nilsson, *ibid*.

[23] J.A. Gualtieri and R.F. Cromp, Support vector machines for hyperspectral remote sensing classification, *Proc. SPIE*, vol. 3584, 1998, pp. 221–232.



optimal hyperplane
$\mathbf{w}^T\mathbf{x} + w_{N+1} = 1$
class 1
$\mathbf{w}^T\mathbf{x} + w_{N+1} = -1$
margin $2/\|\mathbf{w}\|$
class 2

Fig. 8.13 An optimal separating hyperplane can be determined by finding the maximum separation between classes; two marginal hyperplanes can be constructed using the pixel vectors closest to the separating hyperplane

On the marginal hyperplanes we have

$$\text{for class 1 pixels}\ \mathbf{w}^T\mathbf{x} + w_{N+1} = 1$$
$$\text{for class 2 pixels}\ \mathbf{w}^T\mathbf{x} + w_{N+1} = -1$$

or

$$\text{for class 1 pixels}\ \mathbf{w}^T\mathbf{x} + w_{N+1} - 1 = 0$$
$$\text{for class 2 pixels}\ \mathbf{w}^T\mathbf{x} + w_{N+1} + 1 = 0$$

The perpendicular distances of these hyperplanes from the origin are, respectively, $|1 - w_{N+1}|/\|\mathbf{w}\|$ and $|-1 - w_{N+1}|/\|\mathbf{w}\|$ in which $\|\mathbf{w}\|$ is the Euclidean length, or norm, of the weight vector.[24] The separation of the hyperplanes is the difference in these perpendicular distances

$$\text{margin} = 2/\|\mathbf{w}\| \tag{8.32}$$

The best position for the separating hyperplane is that for which the *margin* of (8.32) is largest or, equivalently, when the weight vector norm $\|\mathbf{w}\|$ is smallest. This provides a goal for optimal training of the linear classifier. However, we must always ensure that every pixel vector stays on its correct side of the separating hyperplane, which gives us a set of constraints that must be observed when seeking the optimal hyperplane. We can capture those constraints mathematically in the following manner.

Describe the class label for the *i*th training pixel by the variable y_i, which takes the values of +1 for class 1 pixels and −1 for class 2 pixels. The two equations in (8.31) can then be written in single expression, valid for pixels from both classes:

[24] See Prob. 8.13.

for pixel $\mathbf{x}_i$ in its correct class $\quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}) \geq 1$

or $\quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}) - 1 \geq 0 \qquad (8.33)$

In seeking to minimise $\|\mathbf{w}\|$ we must observe the constraints of (8.33), one for each training pixel. Constrained minimisation can be handled by the process of Lagrange multipliers.[25] This entails setting up a function, called the Lagrangian $\mathcal{L}$, which consists of the property to be minimised but from which is subtracted a proportion of each constraint. For later convenience we will seek to minimise half the square of the vector norm, so that the Lagrangian has the form:

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left\{ y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}\right) - 1 \right\} \qquad (8.34)$$

The $\alpha_i \geq 0$ for all i, are called the Lagrange multipliers and are positive by definition. How are they treated during the minimisation process? Suppose we choose a training pixel and find it is on the wrong side of the separating hyperplane, thus violating (8.30) and (8.33). Given that α_i is positive that would cause $\mathcal{L}$ to increase. What we need to do is find values for $\mathbf{w}$ and w_{N+1} that minimise $\mathcal{L}$ while the α_i are trying to make it larger for incorrectly located training patterns. In other words we are shifting the hyperplane via its weights to minimise $\mathcal{L}$ in the face of the α_i trying to make it bigger for wrongly classified training data.

Consider, first, the values of $\mathbf{w}$ and w_{N+1} that minimise $\mathcal{L}$. That requires equating to zero the derivatives of $\mathcal{L}$ with respect to the weights. Thus, noting $\|\mathbf{w}\|^2 \equiv \mathbf{w}^{\mathrm{T}}\mathbf{w}$,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

which gives
$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \qquad (8.35)$$

Now
$$\frac{\partial \mathcal{L}}{\partial w_{N+1}} = -\sum_i \alpha_i y_i = 0 \qquad (8.36)$$

We can use (8.35) and (8.36) to simplify (8.34). First, using (8.35), we can write

$$\|\mathbf{w}\|^2 = \mathbf{w}^{\mathrm{T}}\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j^{\mathrm{T}} \sum_i \alpha_i y_i \mathbf{x}_i$$

[25] See C.M. Bishop, *loc. cit.*, Appendix E.

Substituting into (8.34) gives

$$\mathcal{L} = \frac{1}{2}\sum_j \alpha_j y_j \mathbf{x}_j^{\mathrm{T}} \sum_i \alpha_i y_i \mathbf{x}_i - \sum_i \alpha_i \{y_i \left(\sum_j \alpha_j y_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i + w_{N+1} \right) - 1\}$$

i.e. $$\mathcal{L} = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i - w_{N+1} \sum_i \alpha_i y_i + \sum_i \alpha_i$$

Using (8.36) this simplifies to

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i \tag{8.37}$$

We are now in the position to find the α_i. Remember they are trying to make the Lagrangian as large as possible, so we seek to maximise (8.37) with respect to the α_i. This is referred to as a *dual representation* in the field of optimisation. Note that there are constraints on this optimisation too. They are that

$$\alpha_i \geq 0 \tag{8.38a}$$

and, from (8.36) $$\sum_i \alpha_i y_i = 0 \tag{8.38b}$$

Equation (8.37), for any real problem, has to be solved numerically, following which the values of α_i have been found. The one remaining unknown is w_{N+1} which we will come to in a moment. First, however, there is another constraint on (8.37) which, together with (8.38a, b), give what are called the *Karush–Kuhn–Tucker* conditions.[26] This further condition is

$$\alpha_i \{ y_i (\mathbf{w}^{\mathrm{T}} \mathbf{x}_i + w_{N+1}) - 1 \} = 0 \tag{8.38c}$$

This is very interesting because it says that either $\alpha_i = 0$ or $y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}) = 1$. The latter expression is valid only for training vectors that lie on the marginal hyperplanes—what we call the *support vectors*. For any other training sample this condition is not valid so that (8.38c) can then only be satisfied if $\alpha_i = 0$. In other words it seems that a whole lot of the training pixels are irrelevant. In a sense that is true, but must be interpreted carefully. When maximising (8.37) we have no way of knowing beforehand which of the training samples will end up being support vectors because we don't yet know the value of $\mathbf{w}$. However once (8.37) has been optimised we know the optimal hyperplane and thus the support vectors. We are then in the position to discard all the other training data. When is that important? It is in the classification phase. That is done via the test of (8.30) or (8.33). To test the

[26] See Bishop, *loc. cit.*

class membership of a pixel at position $\mathbf{x}$ in multispectral space we evaluate the sign of $\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{N+1}$ with $\mathbf{w}$ given by (8.35) but computed using only the support vectors. Thus the test of class membership for pixel $\mathbf{x}$ is

$$\operatorname{sgn}\left\{\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{N+1}\right\} = \operatorname{sgn}\left\{\sum_{i\in\mathcal{S}} \alpha_i y_i \mathbf{x}_i^{\mathrm{T}}\mathbf{x} + w_{N+1}\right\} \tag{8.39}$$

where the symbol $\mathcal{S}$ in the sum refers only to the support vectors.

How do we now find the value of w_{N+1}? A simple approach is to choose two support vectors, one from each class; call these $\mathbf{x}(1)$ and $\mathbf{x}(-1)$ for which $y = 1$ and $y = -1$ respectively. From (8.33) we have for those vectors

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}(1) + w_{N+1} - 1 = 0$$

$$-\mathbf{w}^{\mathrm{T}}\mathbf{x}(-1) - w_{N+1} - 1 = 0$$

so that

$$w_{N+1} = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\{\mathbf{x}(1) + \mathbf{x}(-1)\} \tag{8.40}$$

We could alternately choose sets of $\mathbf{x}(1)$ and $\mathbf{x}(-1)$ and average the values of w_{N+1} so generated. Following Bishop,[27] this can be generalised by noting from the argument of (8.39) that

$$y_{\mathbf{x}}\left(\sum_{i\in\mathcal{S}} \alpha_i y_i \mathbf{x}_i^{\mathrm{T}}\mathbf{x} + w_{N+1}\right) = 1 \tag{8.41}$$

in which $y_{\mathbf{x}}$ is associated with the pixel $\mathbf{x}$. Multiplying throughout by $y_{\mathbf{x}}$, and noting $y_{\mathbf{x}}^2 = 1$, we have from (8.41)

$$w_{N+1} = \mathrm{y}_{\mathbf{x}} - \sum_{i\in\mathcal{S}} \alpha_i y_i \mathbf{x}_i^{\mathrm{T}}\mathbf{x}$$

This is for one support vector $\mathbf{x}$. We now average over all support vectors to give

$$w_{N+1} = \frac{1}{N_{\mathcal{S}}}\sum_{\mathbf{x}\in\mathcal{S}}\left\{\mathrm{y}_{\mathbf{x}} - \sum_{i\in\mathcal{S}} \alpha_i y_i \mathbf{x}_i^{\mathrm{T}}\mathbf{x}\right\} \tag{8.42}$$

in which $N_{\mathcal{S}}$ is the number of support vectors.

[27] *ibid.*

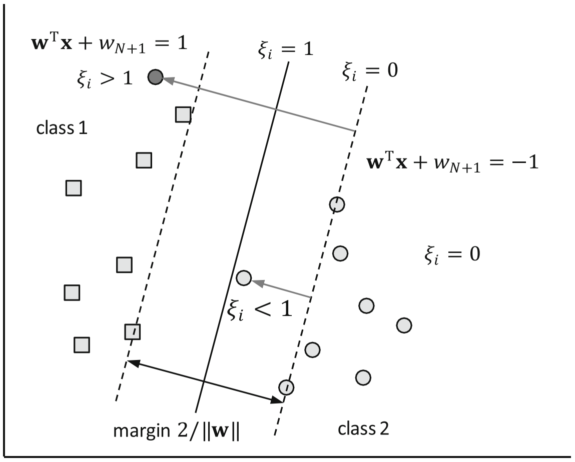

$\mathbf{w}^T\mathbf{x} + w_{N+1} = 1$
$\xi_i > 1$
$\xi_i = 1$
$\xi_i = 0$
class 1
$\mathbf{w}^T\mathbf{x} + w_{N+1} = -1$
$\xi_i = 0$
$\xi_i < 1$
margin $2/\|\mathbf{w}\|$
class 2

Fig. 8.14 Slack variables and overlapping classes

8.15 The Support Vector Machine: Overlapping Classes

It is unrealistic to expect that the pixel vectors from two ground cover classes will be completely separated, as implied in Fig. 8.13. Instead, there is likely to be class overlap more like the situation depicted in Fig. 8.2. Any classifier algorithm, to be effective, must be able to cope with such a situation by generating the best possible discrimination between the classes in the circumstances. As it has been developed in the previous section the support vector classifier will not find a solution for overlapping classes and requires modification. That is done by relaxing the requirement on finding a maximum margin solution by agreeing that such a goal is not possible for *all* training pixels and that we will have to accept that some will not be correctly separated during training. Such a situation is accommodated by introducing a degree of "slackness" in the training step. To develop this variation consider the set of training pixels in Fig. 8.14.

We introduce a set of positive "slack variables" ξ_i, one for each of the training patterns, which are used to modify the constraint of (8.33) such that it now becomes

$$(\mathbf{w}^T\mathbf{x}_i + w_{N+1})y_i \geq 1 - \xi_i \ \forall i \tag{8.43a}$$

The slack variables are defined such that:

$\xi_i = 0$	for training pixels that are on or on the correct side of the marginal hyperplane
$\xi_i = 1$	for a pixel on the separating hyperplane—the decision boundary—because $\mathbf{w}^T\mathbf{x}_i + w_{N+1} = 0$ and $\lvert y_i\rvert = 1$
$\xi_i > 1$	for pixels that are on the wrong side of the separating hyperplane since $\mathbf{w}^T\mathbf{x}_i + w_{N+1}$ has the opposite sign to y_i for misclassified pixels
$\xi_i = \lvert y_i - (\mathbf{w}^T\mathbf{x}_i + w_{N+1})\rvert$	for all other training pixels

When training the support vector machine with overlapping data we minimise the number of pixels in error while maximising the margin by minimising $\|\mathbf{w}\|$. A measure of the number of pixels in error is the sum of the slack variables over all the training pixels; they are all positive and the sum increases with misclassification error because the corresponding slack variables are greater than one. Minimising misclassification error and maximising the margin together can be achieved by seeking to minimise

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \tag{8.43b}$$

in which the positive weight C, called the *regularisation parameter*, adjusts the relative importance of the margin versus misclassification error.

As before the minimisation is subject to constraints. One is (8.43a); the other is that the slack variables are positive. We again accomplish the minimisation by introducing Lagrange multipliers. However, now there is a different multiplier associated with each constraint, so that the Lagrangian is

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\left\{y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}\right) - 1 + \xi_i\right\} - \sum_i \mu_i\xi_i \tag{8.44}$$

in which the α_i and the μ_i are the Lagrange multipliers. We now equate to zero the first derivatives with respect to the weight vector and the slack variables in an attempt to find the values that minimise the Lagrangian.

First
$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

which gives
$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{8.45}$$

while
$$\frac{\partial\mathcal{L}}{\partial w_{N+1}} = -\sum_i \alpha_i y_i = 0 \tag{8.46}$$

Now
$$\frac{\partial\mathcal{L}}{\partial\xi_i} = C - \alpha_i - \mu_i = 0 \tag{8.47}$$

Remarkably, (8.47) removes the slack variables when substituted into (8.44). Since (8.45) and (8.46) are the same as (8.35) and (8.36), (8.44) reduces to

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_j^{\mathrm{T}}\mathbf{x}_i \tag{8.48}$$

which is identical to the dual formulation of (8.37) which has to be maximised with respect to the Langrange multipliers α_i. However, the constraints on the α_i are now different. Since, by definition the Lagrange multipliers, both α_i and μ_i, are non-negative we have

$$0 \leq \alpha_i \leq C \tag{8.49a}$$

and, from (8.46)

$$\sum_i \alpha_i y_i = 0 \tag{8.49b}$$

Again, (8.48) needs to be solved numerically subject to the constraints of (8.49). Once the α_i are found (8.39) is used to label unknown pixels. As with the linearly separable case some of the α_i will be zero, in which case the corresponding training pixels do not feature in (8.39). The training pixels for which $\alpha_i \neq 0$ are again the support vectors.

There are more *Karush–Kuhn–Tucker* conditions in the case of slack variables as seen in Bishop[28] and Burges[29]; one is $\mu_i \xi_i = 0$. We can use this to generate a means for finding w_{N+1}. We know that $\alpha_i > 0$ for support vectors. If, in addition, we have some vectors for which $\alpha_i < C$ then (8.47) shows that μ_i must be non-zero. Since $\mu_i \xi_i = 0$ this requires $\xi_i = 0$ so that (8.42) can again be used to find w_{N+1} but with the sums over those support vectors for which also $\alpha_i < C$.

8.16 The Support Vector Machine: Nonlinearly Separable Data and Kernels

When we examine the central equations for support vector classification, viz. (8.37), (8.39), (8.42) and (8.48) we see that the pixel vectors enter only via a scalar (or dot) product of the form $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}$. As is often the case in thematic mapping, it is possible to transform the original pixel vectors to a new set of features before we apply the support vector approach, in an endeavour to improve separability. For example, we could use a function ϕ to generate the transformed feature vector $\phi(\mathbf{x})$, so that equations such as (8.39) become

$$\operatorname{sgn}\left\{\phi(\mathbf{w})^{\mathrm{T}}\phi(\mathbf{x}) + w_{N+1}\right\} = \operatorname{sgn}\left\{\sum_{i \in \mathcal{S}} \alpha_i y_i \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}) + w_{N+1}\right\} \tag{8.50}$$

We refer to the scalar product of the transformed features as a *kernel function*, written as

$$k(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}) \tag{8.51}$$

---

[28] *ibid.*

[29] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, vol. 2, 1998, pp. 121–166.

which has a scalar value. Since the pixel vectors enter the calculations only in this product form it is not necessary to know the actual transformation $\phi(\mathbf{x})$; all we have to do is specify a scalar kernel function $k(\mathbf{x}_i, \mathbf{x})$ of the two pixel vectors.

What functions are suitable as kernels? Effectively, any function that is expressible in the scalar (or dot) product form in (8.51) is suitable. Clearly, we could build up kernels by choosing the transformations first, but that defeats the purpose: the real benefit of the process known as *kernel substitution* or, sometimes the *kernel trick*, is that we don't need to know the transform but can just choose appropriate kernels. All we need do is satisfy ourselves that the kernel chosen is equivalent to a scalar product. The test comes in the form of satisfying the Mercer condition[30] which, for the kernels below, can always be assumed.

It is instructive to examine a classical example. Consider a kernel composed of the simple square of the scalar product:

$$k(\mathbf{x}, \mathbf{y}) = \left[\mathbf{x}^{\mathrm{T}}\mathbf{y}\right]^2$$

To avoid a complication with subscripts in the following the kernel has been expressed in terms of the variables $\mathbf{x}$ and $\mathbf{y}$. We now restrict attention to the case of two dimensional data, so that the column vectors can be written as $\mathbf{x} = [x_1, x_2]^{\mathrm{T}}$ and $\mathbf{y} = [y_1, y_2]^{\mathrm{T}}$. Expanding the kernel operation we have

$$k(\mathbf{x}, \mathbf{y}) = \left[\mathbf{x}^{\mathrm{T}}\mathbf{y}\right]^2 = [x_1 y_1 + x_2 y_2]^2$$

Expanding, we get $[x_1 y_1 + x_2 y_2]^2 = x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2$

which can be expressed

$$[x_1 y_1 + x_2 y_2]^2 = [x_1^2, \sqrt{2}x_1 x_2, x_2^2]\begin{bmatrix} y_1^2 \\ \sqrt{2}y_1 y_2 \\ y_2^2 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1 y_2 \\ y_2^2 \end{bmatrix}$$

This shows that the quadratic kernel $k(\mathbf{x}, \mathbf{y}) = [\mathbf{x}^{\mathrm{T}}\mathbf{y}]^2$ can be written in the scalar product form of (8.51) and is thus valid. The transformation is now seen explicitly to be

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{8.52}$$

which transforms the original two dimensional space into three dimensions defined by the squares and products of the original variables. Figure 8.15 shows how this transformation leads to linear separability of a two class data set that is not linearly separable in the original coordinates. In this example the original data lies either

[30] *ibid.*

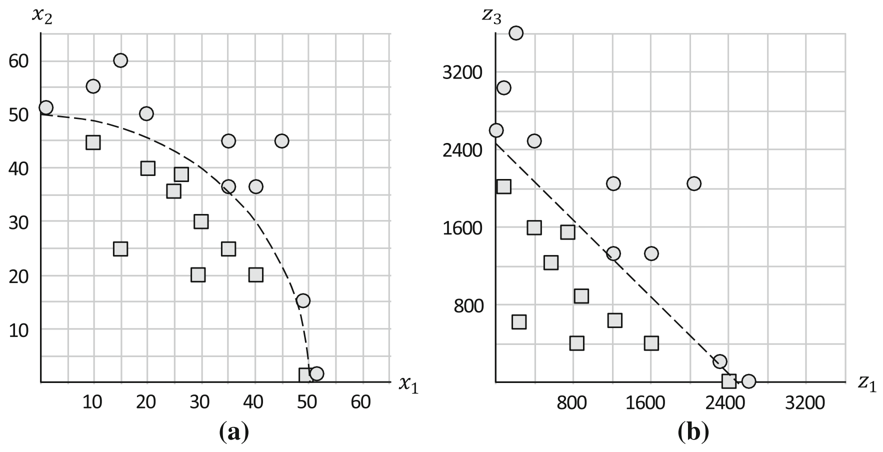


Fig. 8.15 Use of (8.52) to transform the linearly inseparable data set shown in **a** into the separable case in **b**; the z_2 dimension is out of the page, but doesn't contribute to separation

side of the quadrant of a circle defined by $x_1^2 + x_2^2 = 2500$, shown dotted in Fig. 8.15a. Following transformation the circle becomes the straight line $z_1 + z_3 = 2500$, which has a negative unity slope and intersects the axes at 2,500, as shown in Fig. 8.15b. In this simple example the third dimension is not required for separation and can be thought of as coming out of the page; Fig 8.15b is the two dimensional sub-space projection in the z_1, z_3 plane.

The simple quadratic kernel is of restricted value, but the above example serves to demonstrate the importance of using kernels as substitutions for the scalar product in the key equations. A more general polynomial kernel that satisfies the Mercer condition is of the form

$$k(\mathbf{x}_i, \mathbf{x}) = [\mathbf{x}_i^T \mathbf{x} + b]^m \tag{8.53}$$

with $b > 0$. Values for the parameters b and m have to be found to maximise classifier performance.

A popular kernel in remote sensing applications is the Gaussian radial basis function kernel, which has one parameter $\gamma > 0$ and is based on the distance between the two vector arguments:

$$k(\mathbf{x}_i, \mathbf{x}) = \exp\left\{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\right\} \tag{8.54}$$

The distance metric chosen is normally Euclidean, although others are also possible. It is interesting to note that the dimensionality of the transformed space with the radial basis function kernel is infinite, which explains its power and popularity. That can be seen by expanding

$$\|\mathbf{x} - \mathbf{x}_i\|^2 = (\mathbf{x} - \mathbf{x}_i)^T (\mathbf{x} - \mathbf{x}_i) = \mathbf{x}^T \mathbf{x} - 2\mathbf{x}^T \mathbf{x}_i + \mathbf{x}_i^T \mathbf{x}_i$$

so that

$$k(\mathbf{x}_i, \mathbf{x}) = \exp\{-\gamma\mathbf{x}^{\mathrm{T}}\mathbf{x}\}\exp\{2\gamma\mathbf{x}^{\mathrm{T}}\mathbf{x}_i\}\exp\{-\gamma\mathbf{x}_i^{\mathrm{T}}\mathbf{x}\}$$

Taking just the first exponential and replacing it by its power series we have

$$\exp\{-\gamma\mathbf{x}^{\mathrm{T}}\mathbf{x}\} = 1 - \gamma\mathbf{x}^{\mathrm{T}}\mathbf{x} + \frac{\gamma^2}{2}(\mathbf{x}^{\mathrm{T}}\mathbf{x})^2 - \frac{\gamma^3}{6}(\mathbf{x}^{\mathrm{T}}\mathbf{x})^3 + \ldots$$

Note the quadratic transformation of the scalar product in (8.52) led to one more dimension than the power. It is straightforward to demonstrate for two dimensional data that the cubic term leads to a four dimensional transformed space. Thus the first exponential, consisting of an infinite set of terms, leads to an infinite transformed space, as do the other exponentials in $k(\mathbf{x}_i, \mathbf{x})$.

A third option, that has an association with neural network classifiers treated later, is the sigmoidal kernel:

$$k(\mathbf{x}_i, \mathbf{x}) = \tanh(\kappa\mathbf{x}_i^{\mathrm{T}}\mathbf{x} + b) \tag{8.55}$$

which has two parameters to be determined. There is a set of rules that allow new kernels to be constructed from valid kernels, such as those above.[31]

When applying the support vector machine we have a number of parameters to find, along with the support vectors. The latter come out of the optimisation step but the parameters—the weight C in (8.43b) and γ for the radial basis function kernel, or m and b for the polynomial kernel—have to be estimated to give best classifier performance. Generally that is done through a search procedure that we come to in Sect. 11.6.

8.17 Multi-Category Classification with Binary Classifiers

Since the support vector classifier places unknown pixels into one of just two classes, a strategy is needed to allow its use in the multi-class situations encountered in remote sensing. Several approaches are possible. A simple method often used with binary classifiers in the past is to construct a decision tree, at each node of which one of M classes is separated from the remainder as depicted in Fig. 8.16a. The disadvantage with this method is that the training classes are unbalanced, especially in the early decision nodes, and it is not known optimally which classes should be separated first.

More recently, parallel networks have been used to perform multiclass decisions from binary classifiers, as illustrated in Fig. 8.16b. They have been used principally in one of two ways. First, each classifier can be trained to separate one class from the rest, in which case there are as many classifiers as there are classes, in this case M. Again, this approach suffers from unbalanced training sets. If the classifiers are

[31] See Bishop, *loc. cit.* p. 296.

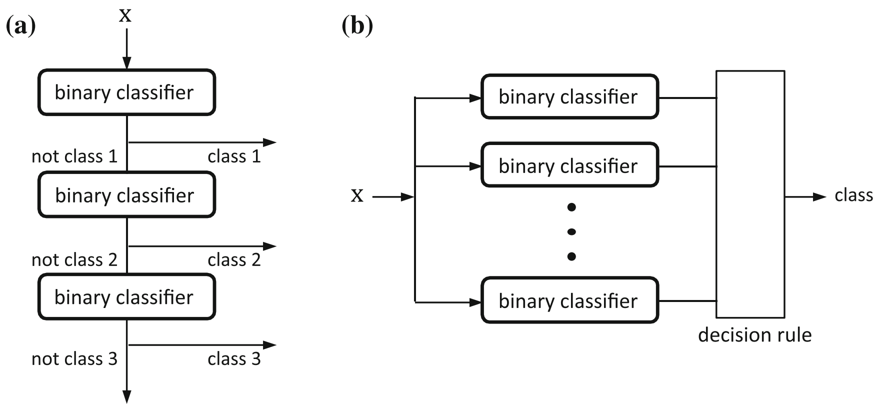


Fig. 8.16 **a** A simple binary decision tree, and **b** multiclass decisions using binary classifiers in parallel, followed by decision logic

support vector machines there is evidence to suggest that such imbalances can affect classifier performance.[32] Also, some form of logic needs to be applied in the decision rule to choose the most favoured class recommendation over the others; that could involve selecting the class which has the largest argument in (8.39).[33] Despite these considerations, the method is commonly adopted with support vector classifiers and is known as the *one-against-the-rest* or *one-against-all* (OAA) technique.

A better approach, again using Fig. 8.16b, is to train $M(M-1)/2$ separate binary classifiers, each of which is designed to separate a pair of the classes of interest. This number covers all possible class pairs. Once trained, an unknown pixel is subjected to each classifier and is placed in the class with the highest number of classification recommendations in favour. Of course, each individual classifier will be presented with pixels from classes for which it was not trained. However, by training for every possible class pair, choosing the most recommended class always works, in principle.[34] The method is called *one-against-one* (OAO). The disadvantage of this method is the very large number of binary classifiers needed. For example, if there were 10 classes, the OAO multiclass strategy requires 45 separate binary classifiers, whereas the OAA only needs 10. Offsetting that problem, however, is the fact that the class-wise binary decisions are generally easier than those in the one against the rest strategy, usually involving simpler SVMs with fewer support vectors for each separate classifier.[35]

[32] K. Song, Tackling Uncertainties and Errors in the Satellite Monitoring of Forest Cover Change, *Ph.D. Dissertation*, The University of Maryland, 2010.

[33] See F. Melgani and L. Bruzzone, Classification of hyperspectral remote sensing images with support vector machines, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 8, August 2004, pp. 1778–1790.

[34] See Prob. 8.7.

[35] See Melgani and Bruzzone, *loc. cit.*

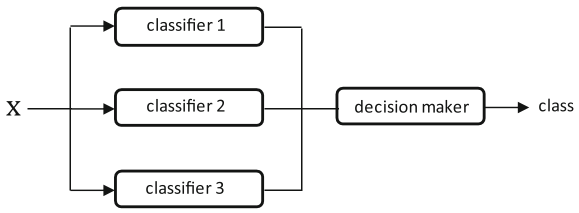


Fig. 8.17 A committee of three classifiers

8.18 Committees of Classifiers

Classically, a committee classifier consists of a number of algorithms that all operate on the same data set to produce individual, sometimes competing, recommendations about the class membership of a pixel, as shown in Fig. 8.17. Those recommendations are fed to a decision maker, or chairman, who resolves the final class label for the pixel.[36]

The decision maker resolves the conflicts by using one of several available logics. One is the majority vote, in which the chairman decides that the class most recommended by the committee members is the most appropriate one for the pixel. Another is veto logic, in which all classifiers have to agree about class membership before the decision maker will label the pixel. Yet another is seniority logic, in which the decision maker always consults one particular classifier first (the most "senior") to determine the label for a pixel. If that classifier is unable to recommend a class label, then the decision maker consults the next most senior member of the committee, and so on until the pixel is labelled. Seniority logic has been used as a means for creating piecewise linear decision surfaces.[37]

8.18.1 Bagging

Apart from the simple approach of committee labelling based on logical decisions as in the previous section, several other procedures using a number of similar classifiers have been devised in an endeavour to improve thematic mapping accuracy. One is called bagging which entails training a set of classifiers on randomly chosen subsets of data, and then combining the results.

The different data sets are chosen by the process called *bootstrapping*, in which the available K training pixels are used to generate L different data sets, each containing N training pixels, in the following manner. N pixels are chosen randomly from the K available, with replacement; in other words the same pixel could appear more than once among the N chosen. Each of the L data sets is used to train a classifier. The results of the individual classifiers are then combined by voting. The name **bagging** derives from **b**ootstrap **agg**regat**ing**.

[36] See N.J. Nilsson, *Learning Machines*, McGraw-Hill, N.Y., 1965.

[37] See T. Lee and J.A. Richards, A low cost classifier for multi-temporal applications, *Int. J. Remote Sensing*, vol. 6, 1985, pp. 1405–1417.

8.18.2 Boosting and AdaBoost

Adaptive **boost**ing, called **AdaBoost**, is a committee of binary classifiers in which the members are trained sequentially, in the following manner. The first classifier is trained. Training pixels that are found to be in error are then emphasised in the training set and the next classifier trained on that enhanced set. Training pixels unable to be separated correctly by the second classifier are given a further emphasis and the third classifier is trained, and so on. The final label allocated to a pixel is based on the outputs of all classifiers. Algorithmically, it proceeds in the following steps.

Suppose there are K training pixels; the correct label for the kth pixel is represented by the binary variable y_k which takes the values {+1,−1} according to which class the kth training pixel belongs. Define $t_k \in \{+1, -1\}$ as the actual class that the pixel is placed into by the trained classifier. For correctly classified pixels $t_k = y_k$ while for incorrectly classified pixels $t_k \neq y_k$.

1. Initialise a set of weights for each of the training pixels in the first classifier step according to[38]

$$w_k^1 = 1/K$$

2. For $l = 1\ldots L$, where L is the number of classifiers in the committee, carry out the following steps in sequence
   a. Train a classifier using the available weighted training data. Initially each training pixel in the set is weighted equally, as in 1. above.
   b. Set up an error measure after the lth classification step according to

$$e^l = \frac{\sum_k w_k^l I(t_k, y_k)}{\sum_k w_k^l}$$

in which $I(t_k, y_k) = 1 \; for \; t_k \neq y_k$
$\qquad = 0$ otherwise

   c. Form the weights for the $(l+1)$th classifier according to

either $$w_k^{l+1} = w_k^l \exp\{\alpha^l I(t_k, y_k)\} \tag{8.56a}$$

or $$w_k^{l+1} = w_k^l \exp\{-\alpha^l t_k y_k\} \tag{8.56b}$$

with $$\alpha^l = \ln\{(1 - e^l)/e^l\} \tag{8.56c}$$

38 All superscripts in this section are stage (iteration) indices and not powers.

Both (8.56a) and (8.56b) enhance the weights of the incorrectly classified pixels; (8.56a) leaves the weights of the correct pixels unchanged, while (8.56b) de-emphasises them.

d. Test the class membership of the kth pixel according to

$$T_L(\mathbf{x}_k) = \operatorname{sgn}\sum_l \alpha^l t_k^l(\mathbf{x}_k) \tag{8.56d}$$

This weights the class memberships recommended by the individual classifiers to come up with the final label for the pixel. Note that $T_L \in \{1, -1\}$ for this binary classifier.

Bishop[39] gives a simple example of AdaBoost in which the improvement of accuracy is seen as more classifiers are added. Accuracy may not always improve initially and many, sometimes 100s, of stages are needed to achieve good results.

8.19 Networks of Classifiers: The Neural Network

Decision trees, committee classifiers and processes such as boosting and bagging are examples of what more generally can be referred to as classifier networks, or sometimes layered classifiers.[40] Perhaps the most popular network classifier in the past two decades has been the artificial neural network (ANN), which is available in several configurations. Here we will develop the most common—the multilayer Perceptron. The basic Perceptron is a simple, binary linear classifier that, once trained, places patterns into one of the two available classes by checking on which side of the linear separating surface they lie. The surface is defined in (8.29) as $\mathbf{w}^T\mathbf{x} + w_{N+1} = 0$, so that the class test is given by (8.30):

$$\mathbf{x} \in \text{class 1 if } \mathbf{w}^T\mathbf{x} + w_{N+1} > 0$$
$$\mathbf{x} \in \text{class 2 if } \mathbf{w}^T\mathbf{x} + w_{N+1} < 0$$

Diagrammatically this can be depicted as shown in Fig. 8.18, in which the sign is checked via a thresholding operation. Together the weighting and thresholding operation are called a *threshold logic unit* (TLU), which is the essential building block of the Perceptron but which effectively limits its value to linearly separable, binary data unless layered procedures are used.[41]

[39] See C.M. Bishop, *loc. cit.*

[40] Nilsson, *loc. cit.*

[41] See Lee and Richards, *loc. cit.*, for one approach.

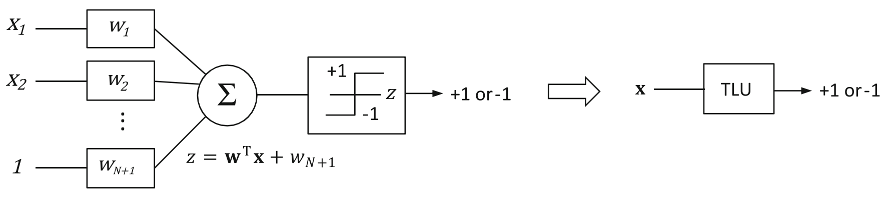


Fig. 8.18 The threshold logic unit (TLU)

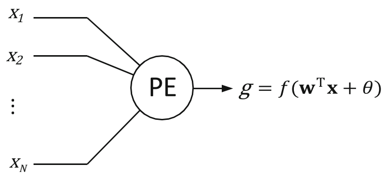


Fig. 8.19 Neural network processing element

8.19.1 The Processing Element

One of the breakthroughs that makes the ANN able to handle data that is not linearly separable is that the hard thresholding of the TLU is replaced by the softer thresholding operation shown in Fig. 8.19. The building block is then referred to as a *processing element* (PE) and is described by the operation

$$g = f(\mathbf{w}^T\mathbf{x} + \theta) \tag{8.57}$$

where $\mathbf{w}$ is the vector of weighting coefficients and $\mathbf{x}$ is the vector of inputs. θ is a threshold, sometimes set to zero, which takes the place of the weighting coefficient w_{N+1}. f is called the *activation function* which can take many forms, the most common of which is

$$g = f(z) = \frac{1}{1 + \exp(-z/b)} \tag{8.58}$$

where $z = \mathbf{w}^T\mathbf{x} + \theta$. It approaches 1 for z large and positive, and 0 for z large and negative, and is thus asymptotically thresholding. For a very small value of b it approaches a hard thresholding operation and the PE behaves like a TLU. Usually we choose $b = 1$. This is seen in Fig. 8.20, along with the behaviour of the activation function for several other values of b.

A typical neural network used in remote sensing applications will appear as shown in Fig. 8.21. It is a layered classifier composed of processing elements of the type in Fig. 8.19. It is conventionally drawn with an input layer of nodes, which has the function of distributing the inputs, possibly with scaling, to a set of processing elements that form the second layer. An output layer generates the class labels for the input provided. The central layer is referred to as a hidden layer.

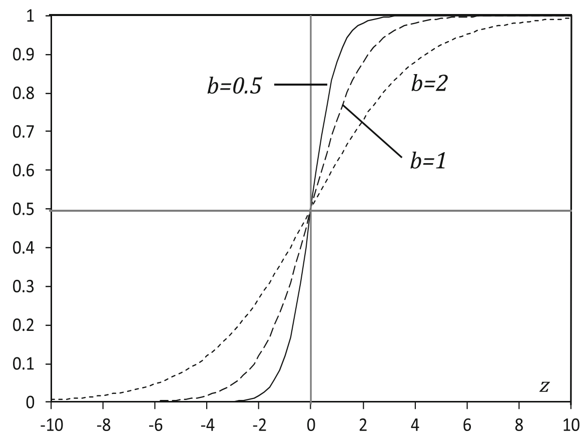


Fig. 8.20 Plots of the activation function of (8.58)

While we have shown only one here it is possible to use more than a single hidden layer. Even though one is usually sufficient in most remote sensing applications, choosing the number of processing elements to use in that layer is not simple and is often done by trial and error. We return to that in Sect. 8.19.3 below.

8.19.2 Training the Neural Network—Backpropagation

As with any supervised classifier the neural network must be trained before it can be used for thematic mapping. Available training data is used to determine the elements of the weight vector **w** and offset θ for each processing element in the network. The parameter b, which governs the slope of the activation function, as seen in Fig. 8.20, is generally pre-specified and does not need to be estimated from training data.

Part of the complexity in understanding the training process for a neural net is caused by the need to keep careful track of the parameters and variables over all layers and processing elements, how they vary with the presentation of training pixels and, as it turns out, with iteration count. This can be achieved with a detailed subscript convention, or by the use of a simpler generalised notation. We will adopt the latter approach, following essentially the development given by Pao.[42] The derivation will be focussed on a 3 layer neural net, since this architecture has been found sufficient for many applications. However the results generalise to more layers.

Figure 8.21 incorporates the nomenclature used. The three layers are lettered as i, j, k with k being the output. The set of weights linking layer i PEs with those in layer j are represented by w_{ji}, while those linking layers j and k are represented by w_{kj}. There will be a very large number of these weights, but in deriving the training

[42] Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading Mass., 1989.

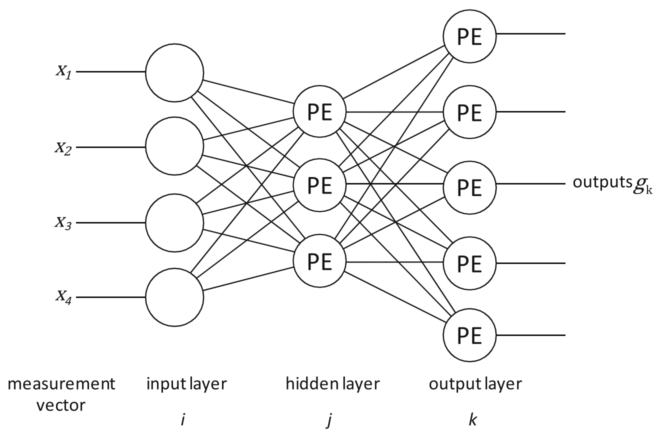


Fig. 8.21 A three layer multilayer Perceptron neural network and the nomenclature used in the derivation of the backpropagation training algorithm

algorithm it is not necessary to refer to them all individually. Similarly, the activation function arguments z_i and outputs g_i, can be used to represent all the arguments and outputs in the corresponding layer. For j and k layer PEs (8.58) is

$$g_j = f(z_j) \text{ with } z_j = \sum_j (w_{ji} g_i + \theta_j) \tag{8.59a}$$

$$g_k = f(z_k) \text{ with } z_k = \sum_k (w_{kj} g_j + \theta_k) \tag{8.59b}$$

The sums in (8.59) are shown with respect to the indices j and k. This should be read as meaning the sums are taken over all inputs of the layer j and layer k PEs respectively. Note also that the sums are expressed in terms of the outputs of the previous layer since those outputs form the inputs to the PEs in question.

An untrained or poorly trained network will give erroneous outputs. As a measure of how well a network is functioning during training we assess the outputs at the last layer (k). A suitable performance measure is the sum of the squared output error. The error made by the network when presented with *a single training pixel* is expressed

$$E = 1/2 \sum_k (t_k - g_k)^2 \tag{8.60}$$

where the t_k represent the desired or target outputs[43] and g_k represents the actual outputs from the output layer PEs in response to the training pixel. The factor of

[43] These will be specified in the labelling of the training data pixels. The actual value taken by t_k will depend on how the set of outputs is used to represent classes. Each individual output could be a specific class indicator, e.g. 1 for class 1 and 0 for class 2 as with the y_i in (8.33); alternatively some more complex coding of the outputs could be adopted. This is considered in Sect. 8.19.3.

1/2 is included for arithmetic convenience in the following. The sum is taken over all output layer PEs.

A logical training strategy is to adjust the weights in the processing elements until the error has been minimised, at which stage the actual outputs are as close as possible to the desired outputs.

A common approach for adjusting weights to minimise the value of a function of which they are arguments, is to modify their values proportional to the negative of the partial derivative of the function. This is called a gradient descent technique.[44] Thus for the weights linking the j and k layers let

$$w'_{kj} = w_{kj} + \Delta w_{kj}$$

with
$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}}$$

in which η is a positive constant that controls the degree of adjustment. This requires an expression for the partial derivative, which can be found using the chain rule

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial g_k}\frac{\partial g_k}{\partial z_k}\frac{\partial z_k}{\partial w_{kj}} \tag{8.61}$$

each term of which is now evaluated. From (8.58) and (8.59b) we see

$$\frac{\partial g_k}{\partial z_k} = \frac{1}{b}(1 - g_k)g_k \tag{8.62a}$$

and
$$\frac{\partial z_k}{\partial w_{kj}} = g_j \tag{8.62b}$$

From (8.60) we have
$$\frac{\partial E}{\partial g_k} = -(t_k - g_k) \tag{8.62c}$$

so that the correction to be applied to the weights is

$$\Delta w_{kj} = \eta(t_k - g_k)(1 - g_k)g_k g_j \tag{8.63}$$

in which we have chosen $b = 1$. For a given training pixel all the terms in (8.63) are known, so that a beneficial adjustment can be made to the weights that link the hidden layer to the output layer.

Now consider the weights that link layers i and j. The weight adjustments are determined from

[44] The conjugate gradient method can also be used: see J.A. Benediktsson, P.H. Swain and O.K. Esroy, Conjugate-gradient neural networks in classification of multisource and very high dimensional remote sensing data, *Int. J. Remote Sensing*, vol. 14, 1993, pp. 2883–2903.

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$$

In a manner similar to the above we can show, with $b = 1$, that

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial g_j} (1 - g_j) g_j g_i$$

Unlike the case with the output layer we cannot generate an expression for $\frac{\partial E}{\partial g_j}$ directly because it requires an expression for the output error in terms of hidden layer responses. Instead, we proceed by the following chain rule

$$\frac{\partial E}{\partial g_j} = \sum_k \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial g_j} = \sum_k \frac{\partial E}{\partial z_k} w_{kj}$$

The remaining partial derivative, from (8.62a) and (8.62c) with $b = 1$, is

$$\frac{\partial E}{\partial z_k} = -(t_k - g_k)(1 - g_k) g_k$$

so that

$$\Delta w_{ji} = \eta (1 - g_j) g_j g_i \sum_k (t_k - g_k)(1 - g_k) g_k w_{kj} \tag{8.64}$$

Having determined the w_{kj} via (8.63) it is now possible to find values for w_{ji} using (8.64) since all the other entries in (8.64) are known or can be determined readily.

We now define

$$\delta_k = (t_k - g_k)(1 - g_k) g_k \tag{8.65a}$$

and

$$\delta_j = (1 - g_j) g_j \sum_k \delta_k w_{kj} \tag{8.65b}$$

so that

$$\Delta w_{kj} = \eta \delta_k g_j \tag{8.66a}$$

and

$$\Delta w_{ji} = \eta \delta_j g_i \tag{8.66b}$$

The thresholds θ_j and θ_k in (8.59) are found in exactly the same manner as for the weights, using (8.66), but with the corresponding inputs set to unity.

Now that we have the mathematics in place it is possible to describe how training is carried out. The network is initialised with an arbitrary set of weights so that it can generate a nominal output. The training pixels are then presented one at a time to the network. For a given pixel the output of the network is computed using the network equations. Almost certainly the output will be incorrect to start with—i.e. g_k will not match the desired class t_k for the training pixel. Correction to the output PE weights, described in (8.66a), is then carried out, using the definition of δ_k in (8.65a). With these new values of δ_k, and thus w_{kj}, (8.65b) and (8.66b) can be applied to find the new weight values in the earlier layers. In this way the effect of the output being in error is propagated back through the network in order to correct the weights. The technique is thus referred to as *backpropagation*.

Pao[45] recommends that the weights not be corrected on each presentation of a single training pixel; instead the corrections for all pixels in the training set should be aggregated into a single adjustment. For p training patterns the bulk adjustments are[46]

$$\Delta w'_{kj} = \sum_p \Delta w_{kj} \quad \Delta w'_{ji} = \sum_p \Delta w_{ji}$$

After the weights have been adjusted the training pixels are presented to the network again and the outputs re-calculated to see if they correspond better to the desired classes. Usually they will still be in error and the process of weight adjustment is repeated. The process is iterated as many times as necessary in order that the network respond with the correct class for each of the training pixels, or until the number of errors in classifying the training pixels is reduced to an acceptable level.

8.19.3 Choosing the Network Parameters

When considering the application of the neural network to thematic mapping it is necessary to make several key decisions beforehand. First, the number of layers to use must be chosen. Generally, a three layer network is sufficient, with the purpose of the first layer being simply to distribute, or fan out, the components of the input pixel vector to each of the processing elements in the second layer. The first layer does no processing as such, apart perhaps from scaling the input data if required.

The next choice relates to the number of elements in each layer. The input layer will generally be given as many nodes as there are components (features) in the pixel vectors. The number to use in the output node will depend on how the outputs are used to represent the classes. The simplest method is to let each separate output signify a different class, in which case the number of output processing elements will be the same as the number of training classes. Alternatively, a single PE could be used to represent all classes, in which case a different value or level of the output variable will be attributed to a given class. A further possibility is to use the outputs as a binary code, so that two output PEs can represent four classes, three can represent eight classes and so on.

[45] Y.H. Pao, *loc. cit.*

[46] This is tantamount to deriving the algorithm with the error being calculated over all pixels p in the training set, viz $E = \sum_p E_p$, where E_p is the error for a single pixel in (8.60).

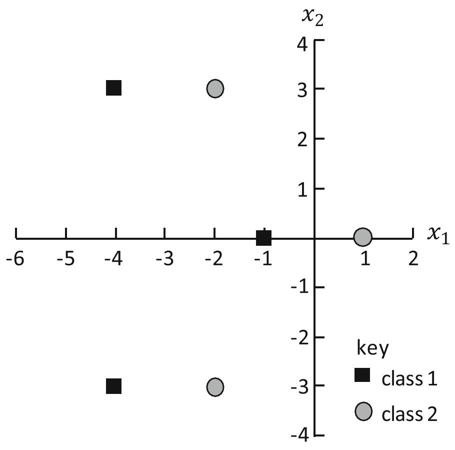


Fig. 8.22 Non-linearly separable two class data set

As a general guide, the number of PEs to choose for the hidden or processing layers should be the same as, or larger than, the number of nodes in the input layer.[47]

8.19.4 Example

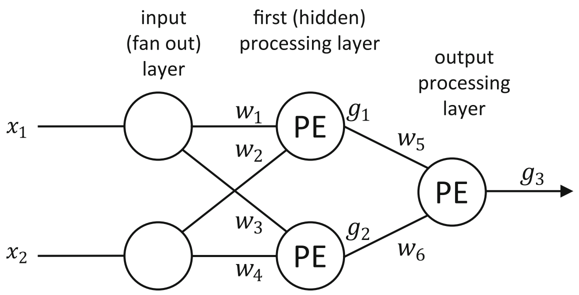
We now consider a simple example to see how a neural network is able to develop the solution to a classification problem. Figure 8.22 shows two classes of data, with three points in each, arranged so that they cannot be separated linearly. The network shown in Fig. 8.23 will be used to discriminate the data. The two PEs in the first processing layer are described by activation functions with no thresholds—i.e. $\theta = 0$ in (8.57), while the single output PE has a non-zero threshold in its activation function.

Table 8.4 shows the results of training the network with the backpropagation method of the previous section, along with the error measure of (8.60) at each step. As seen, the network approaches a solution quickly (approximately 50 iterations), but takes more iterations (approximately 250) to converge to a final result.

Having trained the network it is now possible to understand how it implements a solution to the nonlinear pattern recognition problem. The arguments of the activation functions of the PEs in the first processing layer each define a straight line (hyperplane in general) in the pattern space. Using the result at 250 iterations, these are:

$$2.901x_1 - 2.976x_2 = 0$$
$$2.902x_1 + 2.977x_2 = 0$$

47 R.P. Lippman, An introduction to computing with neural nets, *IEEE ASSP Magazine*, April 1987, pp. 4–22.



$$g_1 = f(z_1),\ z_1 = w_1x_1 + w_2x_2$$
$$g_2 = f(z_2),\ z_2 = w_3x_1 + w_4x_2$$
$$g_3 = f(z_3),\ z_3 = w_5g_1 + w_6g_2 + \theta$$

Fig. 8.23 Neural network with two processing layers, to be applied to the data of Fig. 8.22

which are shown plotted in Fig. 8.24. Either one of these lines goes some way towards separating the data but cannot accomplish the task fully. It is now important to consider how the output PE operates on the outputs of the first layer PEs to complete the discrimination of the two classes. For pattern points lying exactly on one of the above lines, the output of the respective PE will be 0.5, given that the activation function of (8.58) has been used. However, for patterns a little distance away from those lines the output of the first layer PEs will be close to 0 or 1 depending on which side of the hyperplane they lie. We can therefore regard the pattern space as being divided into two regions (0,1) by a particular hyperplane. Using these extreme values, Table 8.5 shows the possible responses of the output layer PE for patterns lying somewhere in the pattern space.

As seen, for this example the output PE functions in the nature of a logical OR operation; patterns that lie on the 1 side of either of the first processing layer PE hyperplanes are labelled as belonging to one class, while those that lie on the 0 side of both hyperplanes will be labelled as belonging to the other class. Therefore, patterns which lie in the shaded region shown in Fig. 8.24 will generate a 0 at the output of the network and will be labelled as belonging to class 1, while patterns in the unshaded region will generate a 1 response and thus will be labelled as belonging to class 2.

Although this exercise is based on just two classes, similar functionality of the PEs in a more complex network can, in principle, be identified. The input PEs will always set up hyperplane divisions of the data and the later PEs will operate on those results to generate a solution to a non-linearly separable problem.

An alternative way of considering how the network determines a solution is to regard the first processing layer PEs as transforming the data in such a way that later PEs (in this example only one) can apply linear discrimination. Figure 8.25 shows the outputs of the first layer PEs when fed with the training data of Fig. 8.22. After transformation the data is seen to be linearly separable. The hyperplane shown in the figure is that generated by the argument of the activation function of the output layer PE.

Table 8.4 Training the network of Fig. 8.23; iteration 0 shows an arbitrary set of initial weights

Iteration	w_1	w_2	w_3	w_4	w_5	w_6	θ	Error
0	0.050	0.100	0.300	0.150	1.000	0.500	−0.500	0.461
1	0.375	0.051	0.418	0.121	0.951	0.520	−0.621	0.424
2	0.450	0.038	0.455	0.118	1.053	0.625	−0.518	0.408
3	0.528	0.025	0.504	0.113	1.119	0.690	−0.522	0.410
4	0.575	0.016	0.541	0.113	1.182	0.752	−0.528	0.395
5	0.606	0.007	0.570	0.117	1.240	0.909	−0.541	0.391
10	0.642	−0.072	0.641	0.196	1.464	1.034	−0.632	0.378
20	0.940	−0.811	0.950	0.882	1.841	1.500	−0.965	0.279
30	1.603	−1.572	1.571	1.576	2.413	2.235	−1.339	0.135
50	2.224	−2.215	2.213	2.216	3.302	3.259	−1.771	0.040
100	2.670	−2.676	2.670	2.677	4.198	4.192	−2.192	0.010
150	2.810	−2.834	2.810	2.835	4.529	4.527	−2.352	0.007
200	2.872	−2.919	2.872	2.920	4.693	4.692	−2.438	0.006
250	2.901	−2.976	2.902	2.977	4.785	4.784	−2.493	0.005

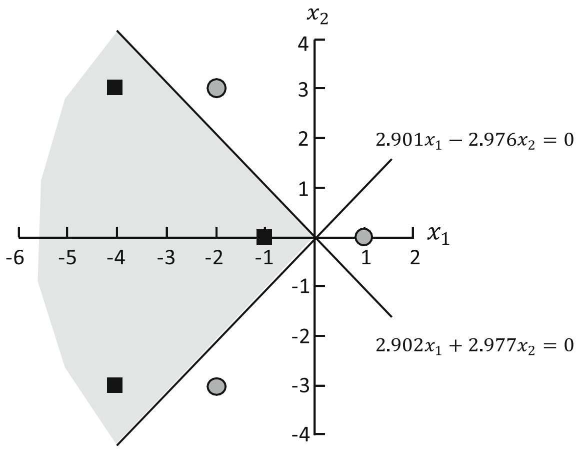


Fig. 8.24 Neural network solution for the data of Fig. 8.22

To illustrate how the network of Fig. 8.23 functions on unseen data Table 8.6 shows its response to the testing patterns indicated in Fig. 8.26. For this simple example all patterns are correctly classified.

8.20 Context Classification

8.20.1 The Concept of Spatial Context

The classifiers treated above are often categorised as point, or pixel-specific, in that they label a pixel on the basis of its spectral properties alone. No account is taken of how their neighbours might be labelled. In any real image adjacent pixels are related

Table 8.5 Response of the output layer PE

g_1	g_2	g_3
0	0	$0.076 \approx 0$
0	1	$0.908 \approx 1$
1	0	$0.908 \approx 1$
1	1	$0.999 \approx 1$

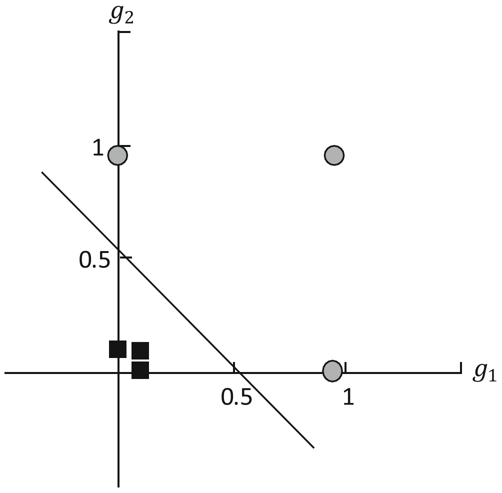


Fig. 8.25 Illustration of how the first processing layer PEs transform the input data into a linearly separable set, which is then separated by the output layer hyperplane

or correlated because imaging sensors acquire significant portions of energy from adjacent pixels[48] and because ground cover types generally occur over a region that is large compared with the size of a pixel. In an agricultural area, for example, if a particular pixel represents wheat it is highly likely that its neighbouring pixels will also be wheat. Knowledge of neighbourhood relationships is a rich source of information that is not exploited in simple, traditional classifiers. In this section we consider the importance of spatial relationships—called spatial context—and see the benefit of taking context into account when making classification decisions. Not only is the inclusion of context important because it exploits spatial properties, but sensitivity to the correct context for a pixel can improve a thematic map by removing individual pixel labelling errors that might result from noisy data, or from unusual classifier performance (see Problem 8.6).

Classification methods that take into account the labelling of neighbours when seeking to determine the most appropriate class for a pixel are said to be context sensitive and are called context classifiers. They attempt to develop a thematic map that is consistent both spectrally and spatially.

The degree to which adjacent pixels are strongly correlated will depend on the spatial resolution of the sensor and the scale of natural and cultural features on the

[48] This is known as the point spread function effect.

Table 8.6 Performance of the network of Fig. 8.23 on the test data of Fig. 8.26

Pattern	x_1	x_2	z_1	g_1	z_2	g_2	z_3	g_3	Class
a	−3.0	2.8	−17.036	0.000	−0.370	0.408	−0.539	0.368	1
b	−3.0	2.0	−14.655	0.000	−2.752	0.056	−2.206	0.099	1
c	−2.0	−1.0	−2.826	0.056	−8.781	0.000	−2.224	0.098	1
d	−2.0	−1.94	−0.029	0.493	−11.579	0.000	−0.135	0.466	1
e	−1.0	1.1	−6.175	0.002	0.373	0.592	0.350	0.587	2
f	1.0	2.0	−3.051	0.045	8.856	1.000	2.506	0.925	2
g	−1.0	−2.0	3.051	0.955	−8.856	0.000	2.077	0.889	2
h	2.0	−2.0	11.754	1.000	−0.150	0.463	4.505	0.989	2

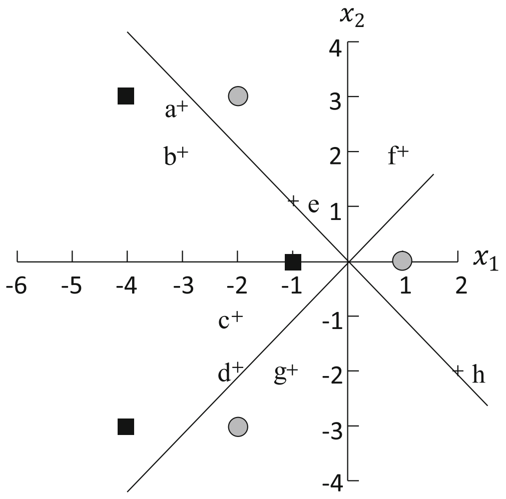


Fig. 8.26 Location of the test data, indicated by the lettered crosses

earth's surface. Adjacent pixels over an agricultural region will be strongly correlated, whereas for the same sensor, adjacent pixels over a busier, urban region would not show strong correlation. Likewise, for a given area, neighbouring Landsat MSS pixels, being larger, may not demonstrate as much correlation as adjacent SPOT HRV pixels. In general terms, context classification techniques usually warrant consideration when processing higher resolution imagery.

There are several approaches to context classification, the most common of which are treated in the following sections.[49]

[49] For statistical context methods see P.H. Swain, S.B. Varderman and J.C. Tilton, Contextual classification of multispectral image data, *Pattern Recognition*, vol. 13, 1981, pp. 429–441, and N. Khazenie and M.M. Crawford, A spatial–temporal autocorrelation model for contextual classification, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, July 1990, pp. 529–539.

8.20.2 Context Classification by Image Pre-processing

Perhaps the simplest method for exploiting spatial context is to process the image data before classification in order to modify or enhance its spatial properties.[50] A median filter (Sect. 5.3.2) will help in reducing salt and pepper noise that would lead to inconsistent class labels with a point classifier, if not removed first. The application of simple averaging filters, possibly with edge preserving thresholds, can be used to impose a degree of homogeneity among the brightness values of adjacent pixels, thereby increasing the chance that neighbouring pixels may be given the same label.

An alternative is to generate a separate channel of data that associates spatial properties with pixels. For example, a texture channel could be added and classification carried out on the combined spectral and texture channels.

One of the more useful spatial pre-processing techniques is the ECHO (Extraction and Classification of Homogeneous Objects) classification methodology[51] in which regions of similar spectral properties are "grown" before classification is performed. Several region growing techniques are available, possibility the simplest of which is to aggregate pixels into small regions by comparing their brightnesses in each channel; smaller regions are then combined into bigger regions in a similar manner. When that is done ECHO classifies the regions as single objects. It only resorts to standard maximum likelihood classification when it has to treat individual pixels that could not be put into regions. ECHO is included in the *Multispec* image analysis software package.[52]

8.20.3 Post Classification Filtering

If a thematic map has been generated using a simple point classifier, a degree of spatial context can be developed by logically filtering the map.[53] If the map is examined in, say, 3×3 windows, the label at the centre can be changed to that most represented in the window. Clearly this must be done carefully, with the user controlling the minimum size region of a given cover type that is acceptable in the filtered image product.

[50] See P. Atkinson, J.L. Cushnie, J.R. Townshend and A. Wilson, Improving thematic map land cover classification using filtered data, *Int. J. Remote Sensing*, vol. 6, 1985, pp. 955–961.

[51] R.L. Kettig and D.A. Landgrebe, Classification of multispectral image data by extraction and classification of homogeneous objects, *IEEE Transactions on Geoscience Electronics*, vol. GE-14, no. 1, 1976, pp. 19–26.

[52] http://dynamo.ecn.purdue/-biehl/Multispec/.

[53] F.E. Townsend, The enhancement of computer classifications by logical smoothing, *Photogrammetric Engineering and Remote Sensing*, vol. 52, 1986, pp. 213–221.

8.20.4 Probabilistic Relaxation Labelling

Spatial consistency in a classified image product can be improved using the process of label relaxation. While it has little theoretical foundation, and is more complex than the methods outlined in the previous sections, it does allow the spatial properties of a region to be carried into the classification process in a logical manner.

8.20.4.1 The Algorithm

The process commences by assuming that a classification has already been carried out, based on spectral data alone. Therefore, for each pixel a set of posterior probabilities is available that describe the likelihoods that the pixel belongs to each of the possible ground cover classes under consideration. That set could be computed from (8.6) or (8.7) if maximum likelihood classification had been used. If another classification method had been employed, some other assignment process will be required, which could be as simple as allocating a high probability to the most favoured class label and lower probabilities to the others.

Although, in principle, we should perhaps use the posterior probabilities $p(\omega_i|\mathbf{x})$ in what is to follow, for notational simplicity we instead adopt the following expression for the set of label probabilities on a pixel m :

$$p_m(\omega_i)\ i = 1\ldots M \tag{8.67}$$

where M is the total number of classes; $p_m(\omega_i)$ should be read as the probability that ω_i is the correct class for pixel m. As posteriors, the full set of $p_m(\omega_i)$ for the pixel sum to unity:

$$\sum_i p_m(\omega_i) = 1$$

Suppose now that a neighbourhood is defined surrounding pixel m. This can be of any size and, in principle, should be large enough to ensure that all the pixels considered to have any spatial correlation with m are included. For high resolution imagery this is not practical and a simple neighbourhood such as that shown in Fig. 8.27 is often adopted.

Now assume that a *neighbourhood function* $Q_m(\omega_i)$ can be found, by means to be described below, through which the pixels in the prescribed neighbourhood can influence the classification of pixel m. This influence is exerted by multiplying the label probabilities in (8.67) by the $Q_m(\omega_i)$. The results can be turned into a new set of label probabilities for the pixel by dividing by their sum:

$$p'_m(\omega_i) = \frac{p_m(\omega_i)Q_m(\omega_i)}{\sum_i p_m(\omega_i)Q_m(\omega_i)} \tag{8.68}$$

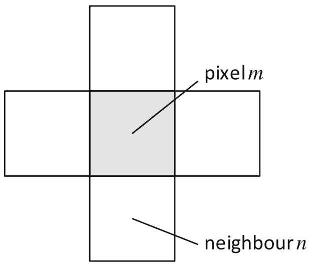


Fig. 8.27 A neighbourhood about pixel m

The modification is made to the set of label probabilities for all pixels. In the following it will be seen that $Q_m(\omega_i)$ depends on the label probabilities of the neighbouring pixels, so that if all pixel probabilities are modified in the manner just described then the neighbours of any given pixel have also been altered. Consequently (8.68) should be applied again to give newer estimates still of the label probabilities. Indeed, (8.68) is applied as many times as necessary to ensure that the $p'_m(\omega_i)$ have stabilised—i.e. that they do not change with further iteration. It is assumed that the $p'_m(\omega_i)$ then represent the correct set of label probabilities for the pixel, having taken account of both spectral data, in the initial determination of label probabilities, and spatial context, via the neighbourhood functions. Since the process is iterative, (8.68) is usually written as an explicit iteration formula:

$$p_m^{k+1}(\omega_i) = \frac{p_m^k(\omega_i)Q_m^k(\omega_i)}{\sum_i p_m^k(\omega_i)Q_m^k(\omega_i)} \tag{8.69}$$

where k is the iteration counter. Depending on the size of the image and its spatial complexity, the number of iterations required to stabilise the label probabilities may be quite large. However, most change in the probabilities occurs in the first few iterations and there is good reason to believe that proceeding beyond say 5 to 10 iterations may not be necessary in most cases (see Sect. 8.20.4.4).

8.20.4.2 The Neighbourhood Function

Consider just one of the neighbours of pixel m in Fig. 8.27—call it pixel n. Suppose there is available some measure of compatibility of the current labelling on pixel m and its neighbouring pixel n. Let $r_{mn}(\omega_i, \omega_j)$ describe numerically how compatible it is to have pixel m classified as ω_i and neighbouring pixel n classified as ω_j. It would be expected, for example, that this measure would be high if the adjoining pixels are both wheat in an agricultural region, but low if one of the neighbours was snow. There are several ways these *compatibility coefficients*, as they are called, can be defined. An intuitively appealing definition is based on conditional probabilities. The compatibility measure $p_{mn}(\omega_i|\omega_j)$ is the probability that ω_i is the correct label for pixel m if ω_j is the correct label for pixel n. A small piece of evidence in favour of ω_i being correct for pixel m is $p_{mn}(\omega_i|\omega_j)p_n(\omega_j)$—i.e. the probability that ω_i is correct for pixel m if ω_j is correct for pixel n multiplied by the probability that ω_j is correct for pixel n. This is the joint probability of pixel m being labelled ω_i **and** pixel n being labelled ω_j.

Since probabilities for all possible labels on pixel n are available (even though some might be very small) the total evidence from pixel n in favour of ω_i being the correct class for pixel m will be the sum of the contributions from all pixel n's labelling possibilities, viz.

$$\sum_j p_{mn}(\omega_i|\omega_j)p_n(\omega_j)$$

Consider now the full neighbourhood of the pixel m. All the neighbours contribute evidence in favour of labelling pixel m as belonging to class ω_i. These contributions are added[54] via the use of *neighbour weights* d_n that recognise that some neighbours may be more influential than others. Thus, at the kth iteration, the total neighbourhood support for pixel m being classified as ω_i is:

$$Q_m^k(\omega_i) = \sum_n d_n \sum_j p_{mn}(\omega_i|\omega_j)p_n^k(\omega_j) \tag{8.70}$$

This is the definition of the neighbourhood function. In (8.69) and (8.70) it is common to include pixel m in its own neighbourhood so that the modification process is not entirely dominated by the neighbours, particularly if the number of iterations is so large as to take the process quite a long way from its starting point. Unless there is good reason to do otherwise the neighbour weights are generally chosen all to be the same.

8.20.4.3 Determining the Compatibility Coefficients

Several methods are possible for determining values for the compatibility coefficients $p_{mn}(\omega_i|\omega_j)$. One is to have available a spatial model for the region under consideration, derived from some other data source. In an agricultural area, for example, some general idea of field sizes, along with a knowledge of the pixel size of the sensor, should make it possible to estimate how often one particular class occurs simultaneously with a given class on an adjacent pixel. Another approach is to compute values for the compatibility coefficients from ground truth pixels, although the ground truth needs to be in the form of training regions that contain heterogeneous and spatially representative cover types.

[54] An alternative way of handling the full neighbourhood is to take the geometric mean of the neighbourhood contributions.

8.20.4.4 Stopping the Process

While the relaxation process operates on label probabilities, the user is interested in the actual labels themselves. At the completion of relaxation, or at any intervening stage, each of the pixels can be classified according to the highest label probability. Thought has to be given as to how and when the iterations should be terminated. As suggested earlier, the process can be allowed to go to a natural completion at which further iteration leads to no further changes in the label probabilities for all pixels. That however presents two difficulties. First, up to several hundred iterations may be involved. Secondly, it is observed in practice that classification accuracy improves in the first few iterations but often deteriorates later in the process.[55] If the procedure is not terminated, the thematic map, after a large number of iterations, can be worse than before the technique was applied. To avoid those difficulties, a stopping rule or some other controlling mechanism is needed. As seen in the example following, stopping after just a few iterations may allow most of the benefit to be drawn from the process. Alternatively, the labelling errors remaining at each iteration can be checked against ground truth, if available, and the iterations terminated when the labelling error is seen to be minimised.[56]

Another approach is to control the propagation of contextual information as iteration proceeds.[57] In the first iteration, only the immediate neighbours of a pixel have an influence on its labelling. In the second iteration the neighbours two away will now have an influence via the intermediary of the intervening pixels. As iterations proceed, information from neighbours further out is propagated into the pixel of interest to modify its label probabilities. If the user has a view of the separation between neighbours at which the spatial correlation has dropped to negligible levels, then the appropriate number of iterations should be able to be estimated at which to terminate the process without unduly sacrificing any further improvement in labelling accuracy. Noting also that the nearest neighbours should be most influential, with those further out being less important, a useful variation is to reduce the values of the neighbour weights d_n as iteration proceeds so that after, say, 5–10 iterations they have been brought to zero.[58] Further iterations will have no effect, and degradation in labelling accuracy cannot occur.

[55] J.A. Richards, D.A. Landgrebe and P.H. Swain, On the accuracy of pixel relaxation labelling, *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-11, 1981, pp. 303–309.

[56] P. Gong and P.J. Howarth, Performance analyses of probabilistic relaxation methods for land-cover classification, *Remote Sensing of Environment*, vol. 30, 1989, pp. 33–42.

[57] T. Lee, Multisource context classification methods in remote sensing, *PhD Thesis*, The University of New South Wales, Kensington, Australia, 1984.

[58] T. Lee and J.A. Richards, Pixel relaxation labelling using a diminishing neighbourhood effect, *Proc. Int. Geoscience and Remote Sensing Symposium, IGARSS89*, Vancouver, 1989, pp. 634–637.

8.20.4.5 Examples

Figure 8.28 shows a simple application of relaxation labelling,[59] in which a hypothetical image of 100 pixels has been classified into just two classes—grey and white. The ground truth for the region is shown, along with the thematic map assumed to have been generated from a point classifier, such as the maximum likelihood rule. That map functions as the "initial labelling." The compatibility coefficients are shown as conditional probabilities, computed from the ground truth map. Label probabilities were assumed to be 0.9 for the favoured label in the initial labelling and 0.1 for the less likely label. The initial labelling, by comparison with the ground truth, can be seen to have an accuracy of 88%—there are 12 pixels in error. The labelling at significant stages during iteration, selected on the basis of the largest current label probability, is shown, illustrating the reduction in classification error owing to the incorporation of spatial information into the process. After 15 iterations all initial labelling errors have been removed, leading to a thematic map 100% in agreement with the ground truth. In this case the relaxation process was allowed to proceed to completion and there have been no ill effects. This is an exception and stopping rules have to be applied in most cases.[60]

As a second example, the leftmost 82 × 100 pixels of the agricultural image shown in Fig. 5.11 have been chosen[61] to demonstrate the benefit of diminishing the neighbourhood contribution with iteration. A maximum likelihood classification of the four Landsat multispectral scanner bands was carried out to label the image into 7 classes and to initialise the relaxation process. The accuracy achieved was 65.6%.

Conditional probabilities were not used as compatibility coefficients. Instead, a slightly different set of compatibilities was adopted.[62] To control the propagation of context information, and thereby avoid the deleterious effect of allowing the relaxation process to proceed unconstrained, the neighbourhood weights were diminished with iteration count according to

$$d_n(k) = d_n(1)e^{-\alpha(k-1)}$$

in which α controls how the neighbour weights change with iteration. If $\alpha = 0$ there is no reduction and normal relaxation applies. For α large the weights drop quickly with iteration. The central pixel was not included in the neighbourhood definition in this example.

Table 8.7 shows how the relaxation performance depends on α. Irrespective of the value chosen the optimal result is achieved after about 4 iterations, giving an accuracy of 72.2%. The table also shows the result achieved if relaxation is left to

[59] Other simple examples will be found in Richards, Landgrebe and Swain, *loc. cit.*

[60] *ibid.*

[61] Full details of this example will be found in T. Lee and J.A. Richards, *loc. cit.*

[62] S. Peleg and A. Rosenfeld, A new probabilistic relaxation procedure, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, 1980, pp. 362–369.

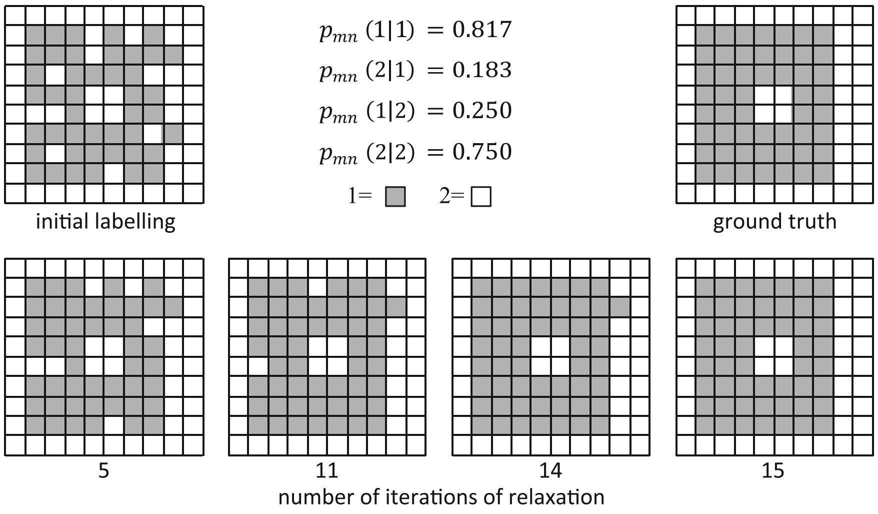


Fig. 8.28 Simple example of pixel relaxation labelling

Table 8.7 Performance of relaxation labelling with a diminishing neighbourhood influence

	Optimal result		Final result	
α	Accuracy	At iteration	Accuracy	At iteration
0.0	72.2	4	70.6	32
1.0	72.2	4	71.4	17
1.8	72.2	4	72.1	10
2.0	72.2	4	72.2	9
2.2	72.2	4	72.2	8
2.5	72.2	5	72.2	7
3.0	72.2	4	72.2	6

run for more iterations (final result). As seen, without diminishing the neighbour weights or without diminishing them sufficiently, the final result is worse than the initial classification error. However, for values of α in the vicinity of 2 the result is fixed at 72.2% from iteration 4 onwards.

8.20.5 Handling Spatial Context by Markov Random Fields

Another method for incorporating the effect of spatial context is to use the construct of the Markov Random Field (MRF). When developing this approach it is useful to commence by considering the whole image, rather than just a local neighbourhood. We will restrict our attention to a neighbourhood once we have established some fundamental concepts.

Suppose there is a total of M pixels in the image, with measurement vectors $\mathbf{x}_m, m = 1\ldots M$ in which $m = (i,j)$. We describe the full set of vectors by $\mathbf{X} = \{\mathbf{x}_1\ldots\mathbf{x}_M\}$.

Let the labels on each of the M pixels, derived from a classification, be represented by the set $\Omega = \{\omega_{c1}\ldots\omega_{cM}\}$ in which ω_{cm} is the label on pixel m, drawn from a set of $c = 1\ldots C$ available classes. We refer to Ω as the *scene labelling*, because it looks at the classification of every pixel in the scene.

We want to find the scene labelling that best matches the ground truth—the actual classes of the pixels on the earth's surface—which we represent by Ω^*. There will be a probability distribution $p\{\Omega\}$ associated with a labelling Ω of the scene, which describes the likelihood of finding that distribution of labels over the image. Ω is sometimes referred to as a *random field*. In principle, what we want to do is find the scene labelling $\hat{\Omega}$ that maximises the global posterior probability $p(\Omega|\mathbf{X})$, and thus that best matches Ω^*. $p(\Omega|\mathbf{X})$ is the probability that Ω is the correct overall scene labelling given that the full set of measurement vectors for the scene is $\mathbf{X}$. By using Bayes' theorem we can express it as

$$\hat{\Omega} = \underbrace{\text{argmax}}_{\Omega}\{p(\mathbf{X}|\Omega)p(\Omega)\} \tag{8.71}$$

in which the argmax function says that we choose the value of Ω that maximises its argument. The distribution $p(\Omega)$ is the prior probability of the scene labelling.

What we need to do now is to perform the maximisation in (8.71) recognising, however, that the pixels are contextually dependent; in other words noting that there is spatial correlation among them. To render the problem tractable we now confine our attention to the neighbourhood immediately surrounding a pixel of interest. Our task is to find the class c that maximises the conditional posterior probability $p(\omega_{cm}|\mathbf{x}_m, \omega_{\partial m})$, where $\omega_{\partial m}$ is the labelling on the pixels in the neighbourhood of pixel m. A possible neighbourhood is that shown in Fig. 8.27, although often the immediately diagonal neighbours about m are also included. Now we note

$$\begin{aligned}
p(\omega_{cm}|\mathbf{x}_m, \omega_{\partial m}) &= p(\mathbf{x}_m, \omega_{\partial m}, \omega_{cm})/p(\mathbf{x}_m, \omega_{\partial m})\\
&= p(\mathbf{x}_m|\omega_{\partial m}, \omega_{cm})p(\omega_{\partial m}, \omega_{cm})/p(\mathbf{x}_m, \omega_{\partial m})\\
&= p(\mathbf{x}_m|\omega_{\partial m}, \omega_{cm})p(\omega_{cm}|\omega_{\partial m})p(\omega_{\partial m})/p(\mathbf{x}_m, \omega_{\partial m})
\end{aligned}$$

The first term on the right hand side is a class conditional distribution function, conditional also on the labelling of the neighbouring pixels. While we do not know that distribution, it is reasonable to assume that the probability of finding a pixel with measurement vector $\mathbf{x}_m$ from class ω_{cm} is not dependent on the class membership of spatially adjacent pixels so that $p(\mathbf{x}_m|\omega_{\partial m}, \omega_{cm}) = p(\mathbf{x}_m|\omega_{cm})$, the simple class conditional distribution function compiled for class ω_{cm} pixels from the remote sensing measurements. We also assume that the measurement vector $\mathbf{x}_m$ and the neighbourhood labelling are independent, so that $p(\mathbf{x}_m, \omega_{\partial m}) = p(\mathbf{x}_m)p(\omega_{\partial m})$. Substituting these simplifications into the last expression above gives

$$\begin{aligned} p(\omega_{cm}|\mathbf{x}_m, \omega_{\partial m}) &= p(\mathbf{x}_m|\omega_{cm})p(\omega_{cm}|\omega_{\partial m})p(\omega_{\partial m})/p(\mathbf{x}_m)p(\omega_{\partial m}) \\ &= p(\mathbf{x}_m|\omega_{cm})p(\omega_{cm}|\omega_{\partial m})/p(\mathbf{x}_m) \end{aligned}$$

Since $p(\mathbf{x}_m)$ is not class dependent it does not contribute any discriminating information so that the maximum posterior probability rule of (8.71) at the local neighbourhood level becomes

$$\begin{aligned} \hat{\omega}_{\mathrm{cm}} &= \underbrace{\mathrm{argmax}}_{\omega_{\mathrm{cm}}} p(\omega_{cm}|\mathbf{x}_m, \omega_{\partial m}) \\ &= \underbrace{\mathrm{argmax}}_{\omega_{\mathrm{cm}}} p(\mathbf{x}_m|\omega_{cm})p(\omega_{cm}|\omega_{\partial m}) \end{aligned}$$

or, expressed in the form of a discriminant function for class m,

$$g_{\mathrm{cm}}(\mathbf{x}_m) = \ln p(\mathbf{x}_m|\omega_{cm}) + \ln p(\omega_{cm}|\omega_{\partial m}) \tag{8.72}$$

We now need to consider the meaning of the conditional probability $p(\omega_{cm}|\omega_{\partial m})$. It is the probability that the correct class is c for pixel m given the labelling of the neighbouring pixels. It is analogous to the neighbourhood function in probabilistic relaxation given in (8.70). Because it describes the labelling on pixel m, conditional on the neighbourhood labels, the random field of labels we are looking for is called a Markov Random Field (MRF).

How do we determine a value for $p(\omega_{cm}|\omega_{\partial m})$? It can be expressed in the form of a Gibbs distribution[63]

$$p(\omega_{cm}|\omega_{\partial m}) = \frac{1}{Z}\exp\{-U(\omega_{cm})\} \tag{8.73}$$

in which $U(\omega_{cm})$ is called an *energy function* and Z is a normalising constant, referred to as the *partition function,* which ensures that the sum of (8.73) over all classes c on pixel m is unity. To this point we have said nothing special about the style of the neighbourhood or how the individual neighbours enter into the computation in (8.73). In relaxation labelling they entered through the compatibility coefficients. In the MRF approach the neighbour influence is encapsulated in the definition of sets of cliques of neighbours. The cliques of a given neighbourhood are sets of adjacent pixels that are linked (or related in some sense of being correlated), as shown in Fig. 8.29 for the two neighbourhoods given.[64] Once a neighbourhood definition has been chosen the energy function is evaluated as a sum of clique *potentials* $V_{\mathcal{C}}(\omega_{cm})$, each one of which refers to a particular clique $\mathcal{C}$ in the neighbourhood:

[63] Although a little complex in view of the level of treatment here, see S. German and D. German, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, 1984, pp. 721–741.

[64] For the simple first order (four neighbour) neighbourhood the concept of cliques is not important, since there are only the four neighbourhood relationships.

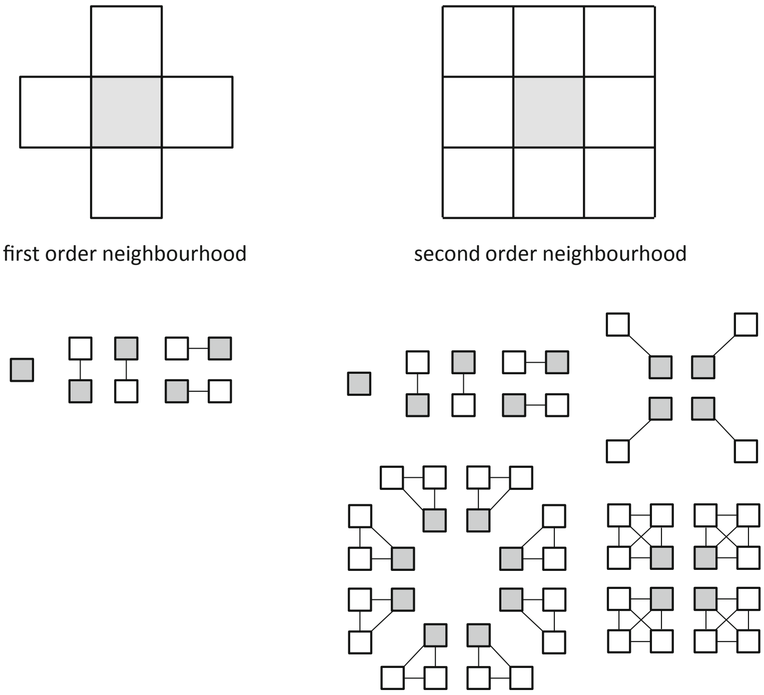


Fig. 8.29 First and second order neighbourhoods and their cliques

$$U(\omega_{cm}) = \sum_{\mathcal{C}} V_{\mathcal{C}}(\omega_{cm}) \tag{8.74}$$

The neighbourhood most often chosen is that of Fig. 8.27 for which the cliques are just the neighbour pairs vertically and horizontally, and the pixel itself, as seen in the left hand side of Fig. 8.29. That leads to the so-called Ising model for which

$$V_{\mathcal{C}}(\omega_{cm}) = \beta[1 - \delta(\omega_{cm}, \omega_{\mathcal{C}m})]$$

where $\delta(\omega_{cm},\, \omega_{\mathcal{C}m})$ is the Kronecker delta; it is unity when the arguments are equal and zero otherwise. $\omega_{\mathcal{C}m}$ is the labelling on the member of the binary clique other than the pixel m itself. β is a parameter to be chosen by the user, as below. Thus $U(\omega_{cm})$ is found by summing over the neighbourhood:

$$U(\omega_{cm}) = \sum_{\mathcal{C}} \beta[1 - \delta(\omega_{cm}, \omega_{\mathcal{C}m})] \tag{8.75}$$

which is now substituted into (8.73) and (8.72) to give, assuming a multivariate normal class model,

$$\begin{aligned} g_{cm}(\mathbf{x}_m) = & -{}^1\!/\!_2 \ln|\mathbf{C}_c| - {}^1\!/\!_2(\mathbf{x}_m - \mathbf{m}_c)^{\mathrm{T}}\mathbf{C}_c^{-1}(\mathbf{x}_m - \mathbf{m}_c) \\ & - \sum_{\mathcal{C}} \beta[1 - \delta(\omega_{cm}, \omega_{\mathcal{C}m})] \end{aligned} \tag{8.76}$$

where the constant Z has been removed as non-discriminating. It is interesting to understand the structure of this last equation. Whenever a neighbour has the same label as the central pixel the spatial term is zero and thus the spectral evidence in favour of the labelling on the central pixel, via the class conditional distribution function, is unaffected. If a neighbour has a different label the discriminant function for the currently favoured class on the central pixel is reduced, thus making that class slightly less likely.

To use (8.76) there needs to be an allocation of classes over the scene before the last term can be computed. Accordingly, an initial classification would be performed, say with the maximum likelihood classifier of Sect. 8.3. Equation (8.76) is then used to modify the labels on the individual pixels to incorporate the effect of context. However, in so doing some (initially, many) of the labels on the pixels will be modified. The process is then run again, and indeed as many times presumably until there are no further changes. This process is referred to as *iterated conditional modes.*[65]

8.21 Bibliography on Supervised Classification Techniques

The machine learning techniques that underpin thematic mapping in remote sensing are covered in many texts at different levels of mathematical complexity. Although it does not have a remote sensing focus, the standard treatment for many researchers, which covers the basis of much of the material treated in this chapter, is

R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification*, 2nd ed., John Wiley & Sons, N.Y., 2001.

A more recent, comprehensive, and slightly more mathematical, coverage containing good examples, but again without a remote sensing focus, is

C.M. Bishop, *Pattern Recognition and Machine Learning,* Springer Science + Business Media LLC, N.Y., 2006.

Although its treatment of the more standard procedures is not as easily assimilated by those with a limited mathematical background, it is nevertheless an important work, especially for some of the more advanced topics covered in this chapter. A book at a comparable level of mathematical detail and which is a very good complement to those above is

[65] See J. Besag, On the statistical analysis of dirty pictures, *J. Royal Statistical Society B*, vol. 48, no. 3, 1986, pp. 259–302.

T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer Science + Business Media LLC, N.Y., 2009.

For a focus on the classification challenges of high dimensional (hyperspectral) image data see

D.A. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*, John Wiley & Sons, Hoboken, N.J., 2003.

This book is written for the remote sensing practitioner, student and researcher and keeps its mathematical detail at the level needed for understanding algorithms and techniques. It is a very practical guide on how high dimensional image analysis should be performed.

Although now largely of historical interest

N.J. Nilsson, *Learning Machines*, McGraw-Hill, N.Y., 1965

and its successor

N.J. Nilsson, *The Mathematical Foundations of Learning Machines*, Morgan Kaufmann, San Francisco, 1990.

are helpful to consult on the foundations of linear classifiers. Nilsson also has good sections on the maximum likelihood rule and the committee (layered) classifier concept.

Neural networks are covered in many machine learning texts, including Duda et al., Bishop and Hastie et al., above. One of the earlier standard texts on neural networks, which is very readable, is

Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, Mass., 1989

while the following paper is a good introduction

R.P. Lippman, An introduction to computing with neural nets, *IEEE ASSP Magazine*, April 1987, pp. 4–22.

Readable accounts of the performance of neural networks in remote sensing image classification are found in

J.A. Benediktsson, P.H. Swain and O.K. Ersoy, Neural network approaches versus statistical methods in classification of multisource remote sensing data, *IEEE Transactions on Geoscience and Remote Sensing,* vol. 28, no. 4, July 1990, pp. 540–552,

J.D. Paola and R.A. Schowengerdt, A review and analysis of backpropagation neural networks for classification of remotely sensed multispectral imagery, *Int. J. Remote Sensing*, vol. 16, 1995, pp. 3033–3058, and

J.D. Paola and R.A. Schowengerdt, A detailed comparison of backpropagation neural network and maximum likelihood classifiers for urban land use classification, *IEEE Transactions on Geoscience and Remote Sensing,* vol. 33, no. 4, July 1995, pp. 981–996.

One cannot escape the Hughes phenomenon when considering the classification of remotely sensed data, also called the curse of dimensionality in machine learning treatments. The original paper is

G. F Hughes, On the mean accuracy of statistical pattern recognizers, *IEEE Transactions on Information Theory*, vol. IT-14, no.1, 1968, pp. 55–63,

while Landgrebe above also contains a good general discussion of the topic. Its influence on the support vector machine is examined in

M. Pal and G.F. Foody, Feature selection for classification of hyperspectral data for SVM, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 5, May 2010, pp. 2297–2307.

The development of kernel transformation methods, and particularly their application to remote sensing, has been rapid over the past two decades. The standard, non-remote sensing treatment, which includes coverage of support vector machines, is

B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularisation, Optimisation and Beyond*, MIT Press, Cambridge, 2002.

For a concentration on remote sensing topics see

G. Camps-Valls and L. Bruzzone, *Kernel Methods for Remote Sensing Data Analysis*, John Wiley & Sons, Chichester, U.K., 2009.

A short review will be found in

L. Gómez-Chove, J Muñoz-Marí, V. Laparra, J. Malo-López and G. Camps-Valls, A review of kernel methods in remote sensing data analysis, in S. Prasad, L.M. Bruce and J. Chanussot, eds., *Optical Remote Sensing: Advances in Signal Processing and Exploitation Techniques*, Springer, Berlin, 2011.

The formative paper for support vector machines in remote sensing is

J.A. Gualtieri and R.F. Cromp, Support vector machines for hyperspectral remote sensing classification, *Proc. SPIE*, vol. 3584, 1998, pp. 221–232

while a good tutorial treatment is

C.T.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, vol. 2, 1998, pp. 121–167.

Papers which demonstrate the application of kernel methods and support vector classifiers to remote sensing problems are

F. Melgani and L. Bruzzone, Classification of hyperspectral remote sensing images with support vector machines, *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, no. 8, August 2004, pp. 1778–1790,

G. Camps-Valls and L. Bruzzone, Kernel-based methods for hyperspectral image classification, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 6, June 2005, pp. 1351–1362.

An review of the suitability of support vector classifiers in land cover classification is

C. Huang, L.S. Davis and J.R.G. Townshend, An assessment of support vector machines for land cover classification, *Int. J. Remote Sensing*, vol. 23, no. 4, 2002, pp. 725–749, while

G. Mountrakis, J. Im and C. Ogole, Support vector machines in remote sensing: a review, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, 2011, pp. 247–259

is an extensive, up to date critical literature review, although with some omissions. Material on Markov Random Fields (MRF) can be found in Bishop above, but the fundamental papers, both of which are quite detailed, are

S. German and D. German, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, 1984, pp. 721–741, and

J. Besag, On the statistical analysis of dirty pictures, *J. Royal Statistical Society B*, vol. 48, no. 3, 1986, pp. 259–302.

One of the earliest applications of the MRF in remote sensing is in

B. Jeon and D.A. Landgrebe, Classification with spatio-temporal interpixel class dependency contexts, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, 1992, pp. 663–672.

Other remote sensing applications that could be consulted are

A.H.S. Solberg, T. Taxt and A.K. Jain, A Markov Random Field model for classification of multisource satellite imagery, *IEEE Transactions on Geoscience and Remote Sensing,* vol. 34, no. 1, January 1996, pp. 100–113, and

Y. Jung and P.H. Swain, Bayesian contextual classification based on modelling M-estimates and Markov Random Fields, *IEEE Transactions on Geoscience and Remote Sensing,* vol. 34, no. 1, January 1996, pp. 67–75.

Development of the fundamental probabilistic label relaxation algorithm is given in

A. Rosenfeld, R. Hummel and S. Zuker, Scene labeling by relaxation algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-6, 1976, pp. 420–433.

8.22 Problems

8.1 Suppose you have been given the training data in Table 8.8 for three spectral classes, in which each pixel is characterised by only two spectral components λ_1 and λ_2. Develop the discriminant functions for a maximum likelihood classifier and use them to classify the patterns

Table 8.8 Training data for three classes, each with two measurement dimensions (wavebands)

Class 1		Class 2		Class 3	
λ_1	λ_2	λ_1	λ_2	λ_1	λ_2
16	13	8	8	19	6
18	13	9	7	19	3
20	13	6	7	17	8
11	12	8	6	17	1
17	12	5	5	16	4
8	11	7	5	14	5
14	11	4	4	13	8
10	10	6	3	13	1
4	9	4	2	11	6
7	9	3	2	11	3

$$\mathbf{x}_1 = \begin{bmatrix} 5 \\ 9 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 9 \\ 8 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 15 \\ 9 \end{bmatrix}$$

under the assumption of equal prior probabilities.

8.2 Repeat the problem 8.1 but with the prior probabilities

$$p(1) = 0.048$$

$$p(2) = 0.042$$

$$P(3) = 0.910$$

8.3 Using the data of problem 8.1 develop the discriminant functions for a minimum distance classifier and use them to classify the patterns $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$.

8.4 Develop a parallelepiped classifier from the training data given in problem 8.1 and compare its classifications with those of the maximum likelihood classifier for the patterns $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ and the new pattern

$$\mathbf{x}_4 = \begin{bmatrix} 3 \\ 7 \end{bmatrix}$$

At the conclusion of the tests in problems 8.1, 8.3 and 8.4, it would be worthwhile sketching a spectral space and locating in it the positions of the training data. Use this to form a subjective impression of the performance of each classifier in problems 8.1, 8.3 and 8.4.

8.5 The training data in Table 8.9 represents a subset of that in problem 8.1 for just two of the classes. Develop discriminant functions for both the maximum likelihood and minimum distance classifiers and use them to classify the patterns

Table 8.9 Training data for two classes, each with two measurement dimensions (wavebands)

Class 1		Class 3	
λ_1	λ_2	λ_1	λ_2
11	12	17	8
10	10	16	4
14	11	14	5
		13	1

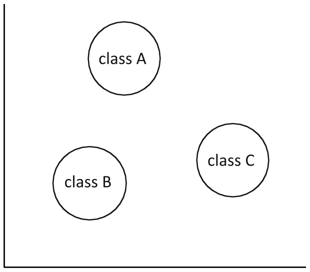


Fig. 8.30 Three linearly separable classes

$$\mathbf{x}_5 = \begin{bmatrix} 14 \\ 7 \end{bmatrix} \quad \mathbf{x}_6 = \begin{bmatrix} 20 \\ 13 \end{bmatrix}$$

Also classify these patterns using the minimum distance and maximum likelihood classifiers developed on the full training sets of problem 8.1 and compare the results.

8.6 Suppose a particular scene consists of just water and soil, and that a classification into these cover types is to be carried out on the basis of near infrared data using the maximum likelihood rule. When the thematic map is produced it is noticed that some water pixels are erroneously labelled as soil. How can that happen, and what steps could be taken to avoid it? It may help to sketch some typical one dimensional normal distributions to represent the soil and water in infrared data, noting that soil would have a very large variance while that for water would be small. Remember the mathematical distribution functions extend to infinity.

8.7 Figure 8.30 shows 3 classes of data, that are linearly separable. Draw three linear separating surfaces on the diagram, each of which separates just two of the classes, in a maximum margin sense. Then demonstrate that the one-against-one multiclass process in Sect. 8.17 will generate the correct results. Extend the example to 4 and 5 classes.

8.8 Compare the properties of probabilistic relaxation and Markov Random Fields based on iterated conditional modes, as methods for context classification in remote sensing.

8.9 If you had to use a simple method for embedding spatial context would you choose pre-classification filtering of the data or post-classification filtering of the thematic map?

8.10 Compare the steps you would need to take to perform thematic mapping based on the maximum likelihood rule, a neural network and a support vector machine with kernel.

8.11 Would kernels be of value in minimum distance classification?

8.12 (This is a bit complex) With probabilistic relaxation treated in Sect. 8.20.4 the results can improve with early iterations and then deteriorate. That is because processes such as simple averaging take over when relaxation nears what is called a fixed point, at which individual labels are strongly favoured for the pixels, but the iterations are then allowed to continue. See J.A. Richards, P.H. Swain and D.A. Landgrebe, On the accuracy of pixel relaxation labelling, *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-11, 1981, pp. 303–309. Would you expect the same to happen with the iterated conditional modes of the MRF approach?

8.13 In two dimensions the linear surface of (8.29) is written $w_1x_1 + w_2x_2 + w_3 = 0$. Show that its perpendicular distance to the origin is $w_3/\sqrt{w_1^2 + w_2^2}$. Can you generalise that expression to any number of dimensions?