REMOTE SENSING

Digital Image Processing

OVERVIEW

Introduction
Image Quality Assessment: Basic Statistics and Histogram Analysis
Histogram
Image Enhancement
Image Preprocessing
Radiometric Correction • Geometric Correction
Sensor Models Image Matching Automated Photogrammetric Mapping Interior Orientation Relative Orientation
Aerial Triangulation
Principal Component Analysis
Spatial Filtering
Band Rationing and Vegetation Indices
Band Ratio • Vegetation Index • Simple Ratio • Normalized Difference Vegetation Index • Enhanced Vegetation Index 1 and 2 • Wide Dynamic Range Vegetation Index • Three-Band Model
Image Classification

PRINCIPAL COMPONENT ANALYSIS

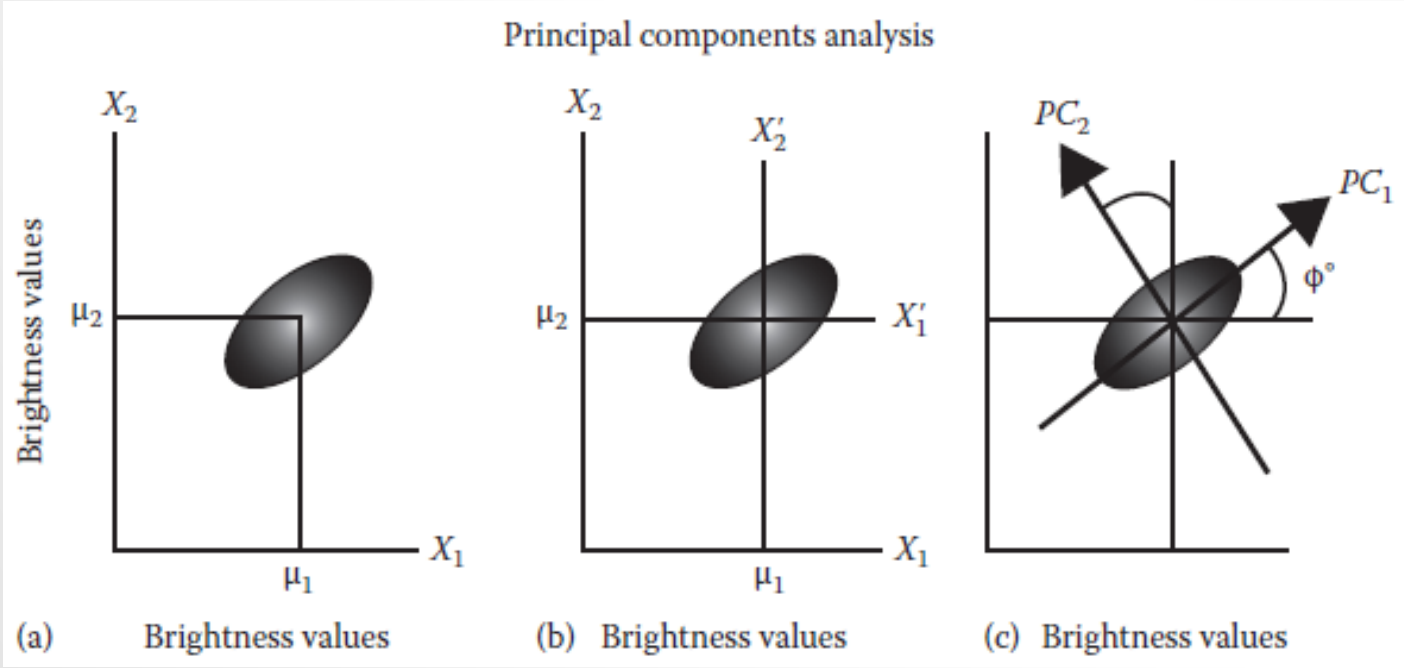
Image transformation techniques based on complex processing of the statistical characteristics of multiband datasets can be used to reduce this redundancy and correlation between bands.

The new bands that result from this statistical procedure are called **components**.
The process attempts to statistically maximize the amount of information (or variance) from the original data into the least number of useful **new** components.

PCA transforms the axes of the multispectral space such that it coincides with the directions of greatest correlation.
Each of these new axes is orthogonal to one another; that is, they are at right angles, and the component images are arranged such that the greatest amount of variance (or information) within the original dataset is contained within each component and the amount of variance decreases with each component

PRINCIPAL COMPONENT ANALYSIS

Transformation of original data on X1 and X2 axes onto PC1 and PC2 axes requires transformation coefficients that can be applied in a linear fashion to original pixel values. These new axes are called the first PC. The second PC is perpendicular (orthogonal) to PC1. Subsequent components contain decreasing amounts of the variance found in the dataset.



(a) the cluster of BVs from two bands of an image, (b) a new coordinate system defined by the X′, (c) the PCA transformation that occurs by rotating to the new axis, which is orthogonal to the original X′ axis. The new axes are no longer the bands of the original image, but derivative components from those data.

PRINCIPAL COMPONENT ANALYSIS

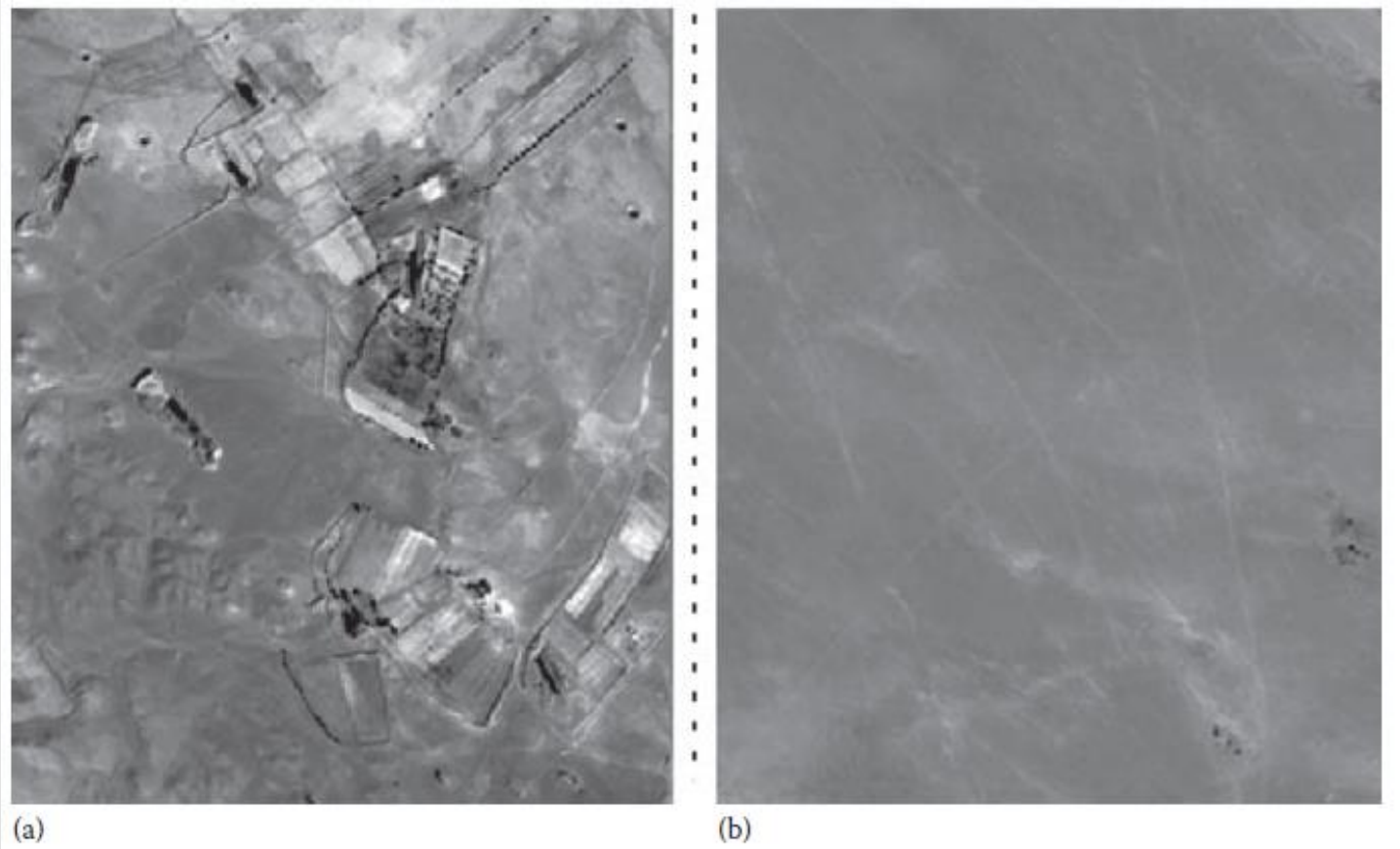
By computing the correlation between each band and each PC, it is possible to determine how each band ***loads*** or is associated with each PC.

A linear combination of original BV and factor scores (eigenvectors) produces the new BV for each pixel of every PC.
It is often the case that the majority of the information contained in a multispectral dataset can be represented by the first three or four PCA components.

Higher-order components may be associated with noise in the original dataset.

SPATIAL FILTERING

The number of changes in BVs per unit distance for any particular part of the image is called **spatial frequency**—that is, the **roughness** of the tonal variations occurring in an image. Figure demonstrates the differences between low-frequency (less roughness) and high-frequency (more roughness) images. In a low-frequency area, the changes in BVs are subtle over the given area, while the opposite is true in a high-frequency image.

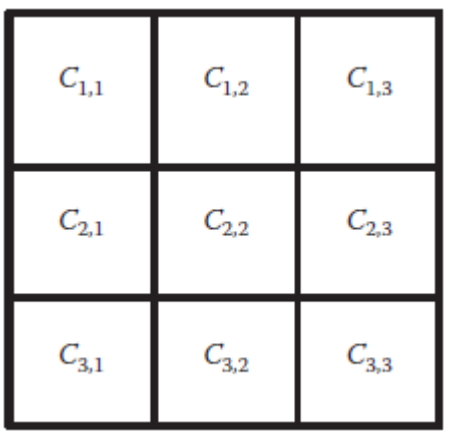


(a) (b)

Local operations are performed (spatial filtering) to extract quantitative information, and the BV of a given pixel is modified based on the values of neighbouring pixels.

SPATIAL FILTERING

A filter (or a convolution mask/kernel) is a moving window function that defines a small sub-window with a dimension of 3 × 3 or larger and usually with odd-numbered dimensions (e.g., 3 × 3, 5 × 5, and 7 × 7). Pixel C2,2 in the window is the center pixel, and odd-numbered window sizes ensure that there is always a center pixel in the sub-window.



Filtering involves computing a weighted average of the pixels in the **moving window**. The choice of **weights** determines how the filter affects the image. A window of weight values is called a **convolution kernel**. Multiplying each pixel in the moving window by its weight and summing all the products yield a new value for the center pixel. The values used in a convolution kernel define whether the filter is **low pass** or **high pass**.

SPATIAL FILTERING

13	39	42
28	55	85
73	84	87

(a) Original image

×

1	1	1
1	1	1
1	1	1

(b) Convolution kernel

=

13	39	42
28	56	85
73	84	87

(c) Resulting image

$$BV_{out} = int\left(\frac{\sum_{i=0}^{9} c_i \times BV_i}{n}\right)$$

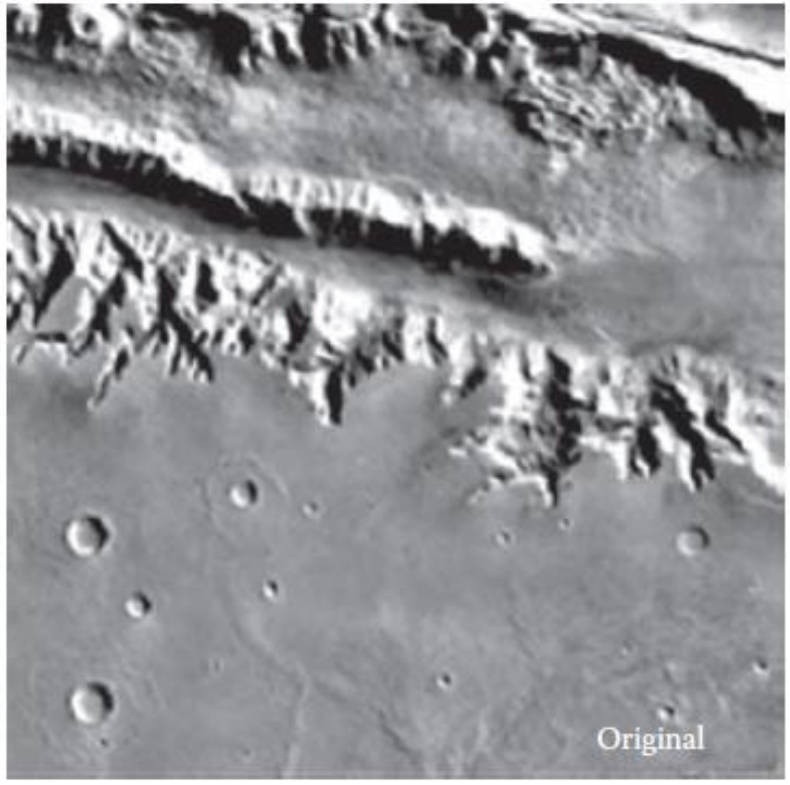
$$BV_{out} = 13+39+42+28+55+85+73+84+87 = 506$$

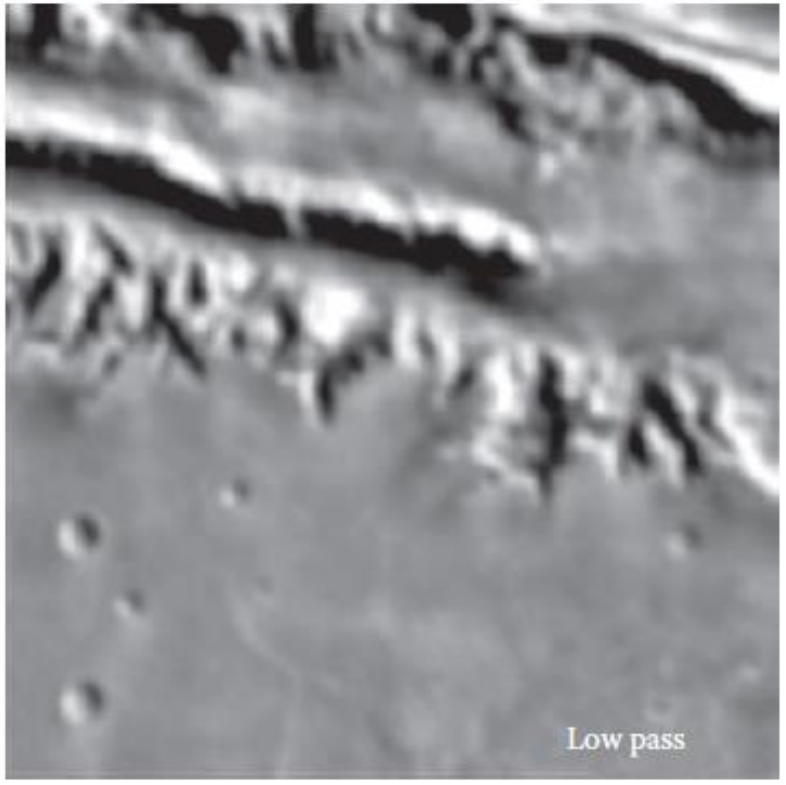
$$BV_{out} = 506/9 \qquad BV_{out} = 56.22$$

SPATIAL FILTERING

Low-pass filters are designed to emphasize **low-frequency** features and de-emphasize the high-frequency components of an image. Thus, information changing very fast across a landscape (e.g., in an urban area) will be subdued, while low-frequency information (e.g., grassland, water) is preserved. Low-pass filters are excellent for retaining low-frequency information and are useful for removing noise (such as speckle) in an image.

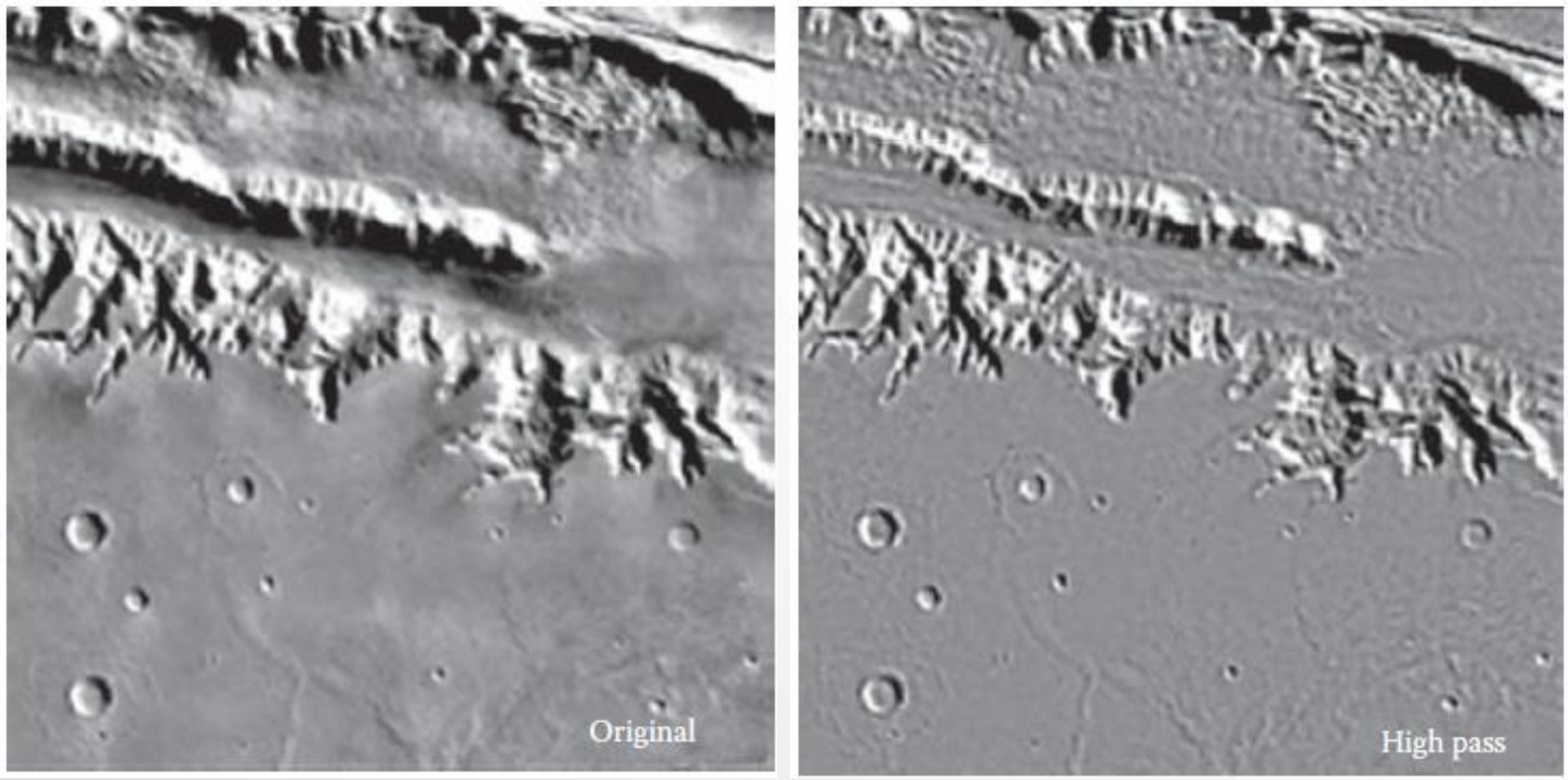
Low-pass filters make similar cover areas appear uniform and can be useful for boundary detection. Conversely, low-pass filters do not preserve edges, and larger window sizes lead to greater smoothing.





SPATIAL FILTERING

High-pass filters emphasize the detailed high-frequency components of an image and deemphasize the more general low-frequency information. They enhance image details (infrequent information) and are useful where lower-frequency information tends to **hide** parts of the scene of interest, for example, roads in an urban scene. When building a high-pass filter, the center pixel of the kernel is given more weight. Consequently, if it is an edge, then the pixel will be greatly enhanced because edges have higher pixel values.



SPATIAL FILTERING

Spatial filtering methods can also be used to remove noise in the data (e.g., striping or speckles). De-striping of an image can also be done by running a low-pass and a high-pass filter on an image and then adding the filter outputs.

Filtering Technique	What It Does	Filter Examples
High frequency	Allows high-frequency information to pass through	Enhancing structural details
	Suppresses low-frequency information	Bring out boundaries and edges
	Edges are sharp and small features stand out	
	Large features look suppressed	
Low frequency	Allows low-frequency information to pass through	Highlight larger features
	Suppresses high-frequency information	Bring out information in larger features
	Edges get subdued	
	Larger features are enhanced	
	Smaller features begin to get smoothed	
Edge enhancement	Detect edges/boundaries between features	Aid in automated feature extraction
		Useful for geologic information, urban areas, boundaries, etc.
Frequency domain	Converts data from spatial to frequency domain	Enhancement, compression
		Noise removal, image restoration
		Textural classification, quality assessment

BAND RATIOING AND VEGETATION INDICES

Another useful image-processing technique exploits the relationships among the BVs of different bands of image features. Mathematical expressions are applied to image bands in order to extract thematic information. These expressions may be **simple ratio (SR)** or complex equations and are generally developed to target a specific feature of interest. Many such algorithms have been developed to highlight characteristics of land cover, such as vegetation, soil, water, and urban areas, and the information extracted can be applied to a wide range of analyses.

BAND RATIOING AND VEGETATION INDICES

Index	Formula	Source
Simple ratio	$SR = \frac{\rho_{red}}{\rho_{nir}}$	Birth and McVey (1968)
Normalized Difference Vegetation Index	$NDVI = \frac{(\rho_{nir} - \rho_{red})}{(\rho_{nir} + \rho_{red})}$	Rouse et al. (1974)
Soil-adjusted Vegetation Index	$SAVI = \frac{(\rho_{nir} - \rho_{red})}{(\rho_{nir} + \rho_{red} + a) \times (1 + a)}$	Huete (1988)
Green Normalized Difference Vegetation Index	$GNDVI = \frac{(\rho_{nir} - \rho_{green})}{(\rho_{nir} + \rho_{green})}$	Buschmann and Nagel (1993)
Green Atmospherically Resistant Vegetation Index	$GARI = \frac{\rho_{nir} - [\rho_{green} - (\rho_{blue} - \rho_{red})]}{\rho_{nir} - [\rho_{green} + (\rho_{blue} - \rho_{red})]}$	Gitelson et al. (1996)
Enhanced Vegetation Index	$EVI = 2.5 \times \frac{(\rho_{nir} - \rho_{red})}{(\rho_{nir} + 6(\rho_{red}) - 7.5(\rho_{blue}) + 1)}$	Huete et al. (1996)
Visible Atmospherically Resistant Index	$VARI = \frac{(\rho_{green} - \rho_{red})}{(\rho_{green} + \rho_{red} - \rho_{blue})}$	Gitelson et al. (2002)
Wide Dynamic Range Vegetation Index	$WDRVI = \frac{a \times (\rho_{nir} - \rho_{red})}{a \times (\rho_{nir} + \rho_{red})}$	Gitelson (2004)
Three-band Model	$TbM = [\rho(\lambda_1)^{-1} - \rho(\lambda_2)^{-1}] \times \rho(\lambda_3)$	Gitelson et al. (2006)
Enhanced Vegetation Index 2	$EVI2 = 2.5 \times \frac{(\rho_{nir} - \rho_{red})}{(\rho_{nir} + 2.4(\rho_{red}) + 1)}$	Jiang et al. (2008)

BAND RATIOING AND VEGETATION INDICES

Band Ratio

BVs of specific targets of interest vary from image to image depending on environmental factors, including topography, the slope of the target surface, aspect ratio, solar angle, seasonal changes, atmospheric conditions, water content, substrate conditions, or shadowing.

This may make complex image analysis functions such as classification, feature discrimination, and change detection difficult to perform. However, certain ratio transformations applied to two or more spectral bands can minimize such effects.

In addition, these ratios may generate unique information not otherwise attainable, through visual image analysis techniques.

$$\mathrm{BR}, P_x = \frac{\mathrm{BV}, P_x B_x}{\mathrm{BV}, P_x B_y}$$

where BR, Px is the output value for a pixel (Px) using the BVs of two bands: band x (Bx) and band y (By). One obvious problem becomes clear that BR, Px = 0 is a possible outcome. There are several methods to address this, however, including assigning a value of 1 to any BV with a value of 0 or adding a small value to the denominator if it equals zero (such as 0.1).

BAND RATIOING AND VEGETATION INDICES

Vegetation Index

Simple Ration, one of the first documented VIs that provides a simple formula for measuring the ratio of red reflectance (ρ_{red} in% or dimensionless) to NIR reflectance (ρ_{nir}):

$$SR = \frac{\rho_{red}}{\rho_{nir}}$$

Green vegetation strongly reflects incident irradiation in the NIR region (40%–60%) while absorbing up to 97% in the red region. As vegetation greenness declines, red reflectance increases and NIR reflectance decreases. By computing the ratio of red to NIR, this relationship can be quantified.

Normalized Difference Vegetation Index

NDVI is functionally equivalent to SR, and comparison plots reveal no scatter between SR and NDVI.

$$NDVI = \frac{(\rho_{nir} - \rho_{red})}{(\rho_{nir} + \rho_{red})}$$

NDVI is widely applied to spectral and image data for monitoring, analyzing, and mapping VBVs.

BAND RATIOING AND VEGETATION INDICES

Vegetation Index

There are several characteristics of NDVI that contribute to its utility and continuing popularity among vegetation experts:

- Seasonal and phonological changes in vegetation can be monitored.
- Normalized data make comparisons more reliable.
- Ratioing reduces some cases of multiplicative noise caused by differences in solar angle, shadows, and topographic variations.

Conversely, a major disadvantage to NDVI is the nonlinear nature of the relationship between NDVI values and many VBVs. The index becomes saturated at high levels, and as VBVs increase, NDVI shows little variation.

Enhanced Vegetation Index 1 and 2

Several VIs are tailored to specific sensors and may be tuned to maximize the results of analysis at specific resolution characteristics. An example of this is the **Enhanced Vegetation Index (EVI)** developed specifically for application to MODIS data. EVI is similar to NDVI; however, it includes several coefficients in the equation to account for atmospheric scattering and to reduce the saturation effects of NDVI at high values.

$$EVI = 2.5 \times \frac{(\rho_{nir} - \rho_{red})}{(\rho_{nir} + 6(\rho_{red}) - 7.5(\rho_{blue}) + 1)}$$

BAND RATIOING AND VEGETATION INDICES

Enhanced Vegetation Index 2 (EVI2), was developed for use with datasets that did not have sensitivity in the blue region of the spectrum.

$$EVI2 = 2.5 \times \frac{(\rho_{nir} - \rho_{red})}{(\rho_{nir} + 2.4(\rho_{red}) + 1)}$$

Wide Dynamic Range Vegetation Index

A simple adjustment to NDVI to compensate for the high-end saturation. The Wide Dynamic Range Vegetation Index (WDRVI) applies a weighted coefficient (a) to NDVI with a value of 0.1–0.2 to linearize the index relationship to VBVs.

$$WNDVI = \frac{a \times (\rho_{nir} - \rho_{red})}{a \times (\rho_{nir} + \rho_{red})}$$

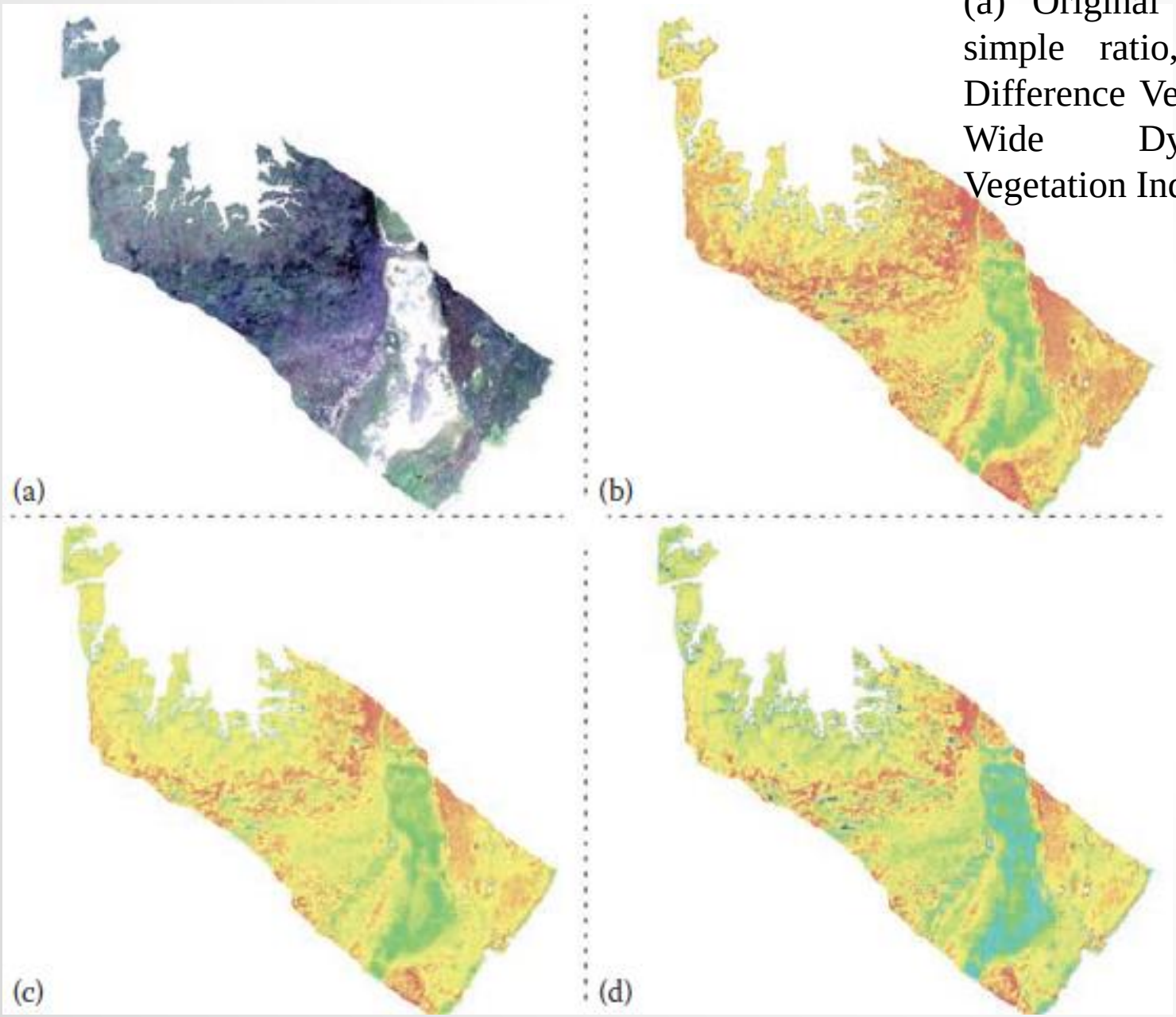
Three Band Model

Index that may be optimizable for other pigments and potentially other features of interest. Three-band model (TbM) requires the use of three spectral bands that must be identified as follows:

- Band 1 ($\lambda 1$): The band that is most sensitive to changes in VBV.
- Band 2 ($\lambda 2$): The band that is the most insensitive to changes in VBV.
- Band 3 ($\lambda 3$): The band that accounts for backscattering/noise among samples.

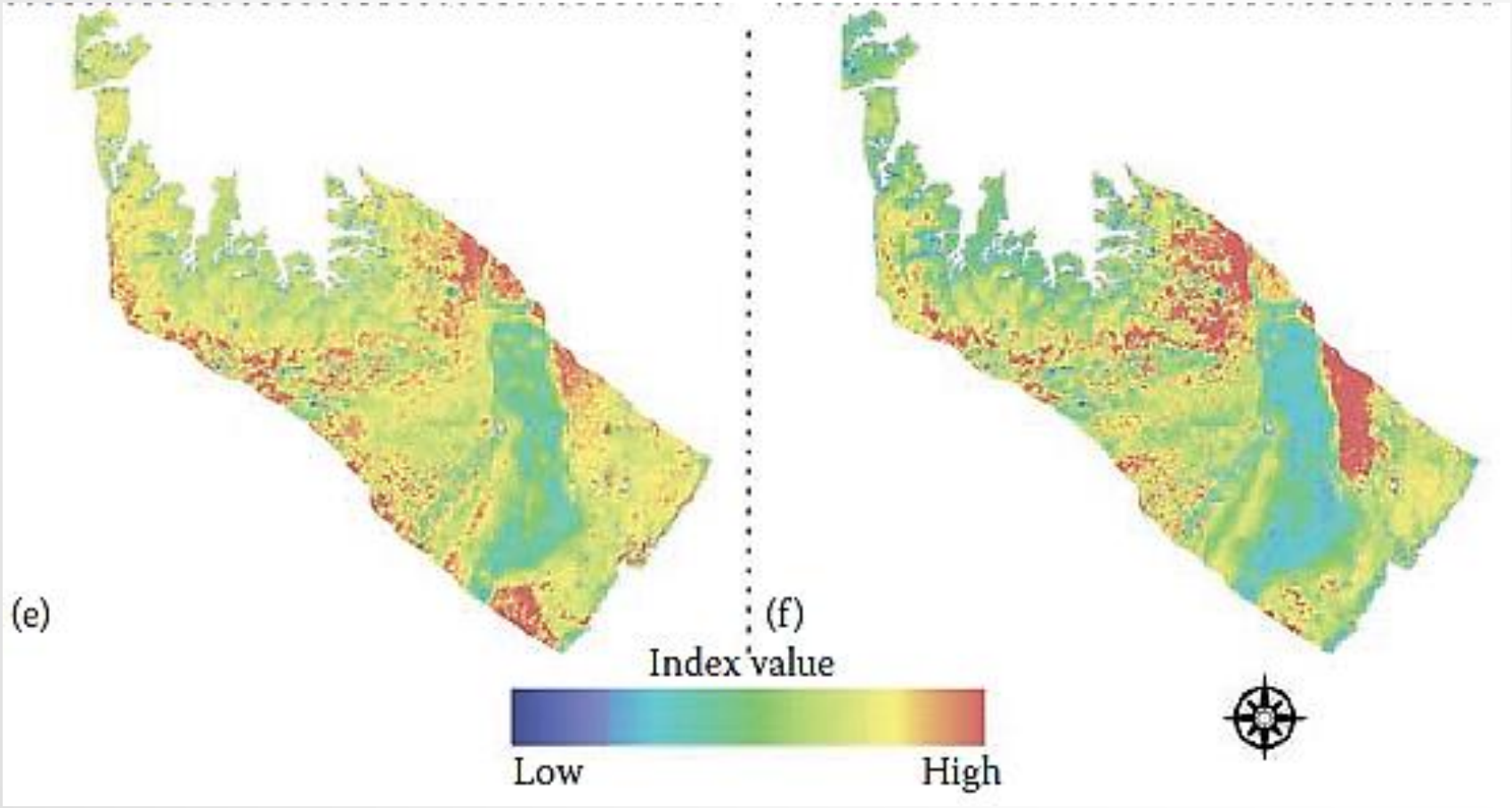
$$TbM = \left[\rho(\lambda_1)^{-1} - \rho(\lambda_2)^{-1}\right] \times \rho(\lambda_3)$$

BAND RATIOING AND VEGETATION INDICES



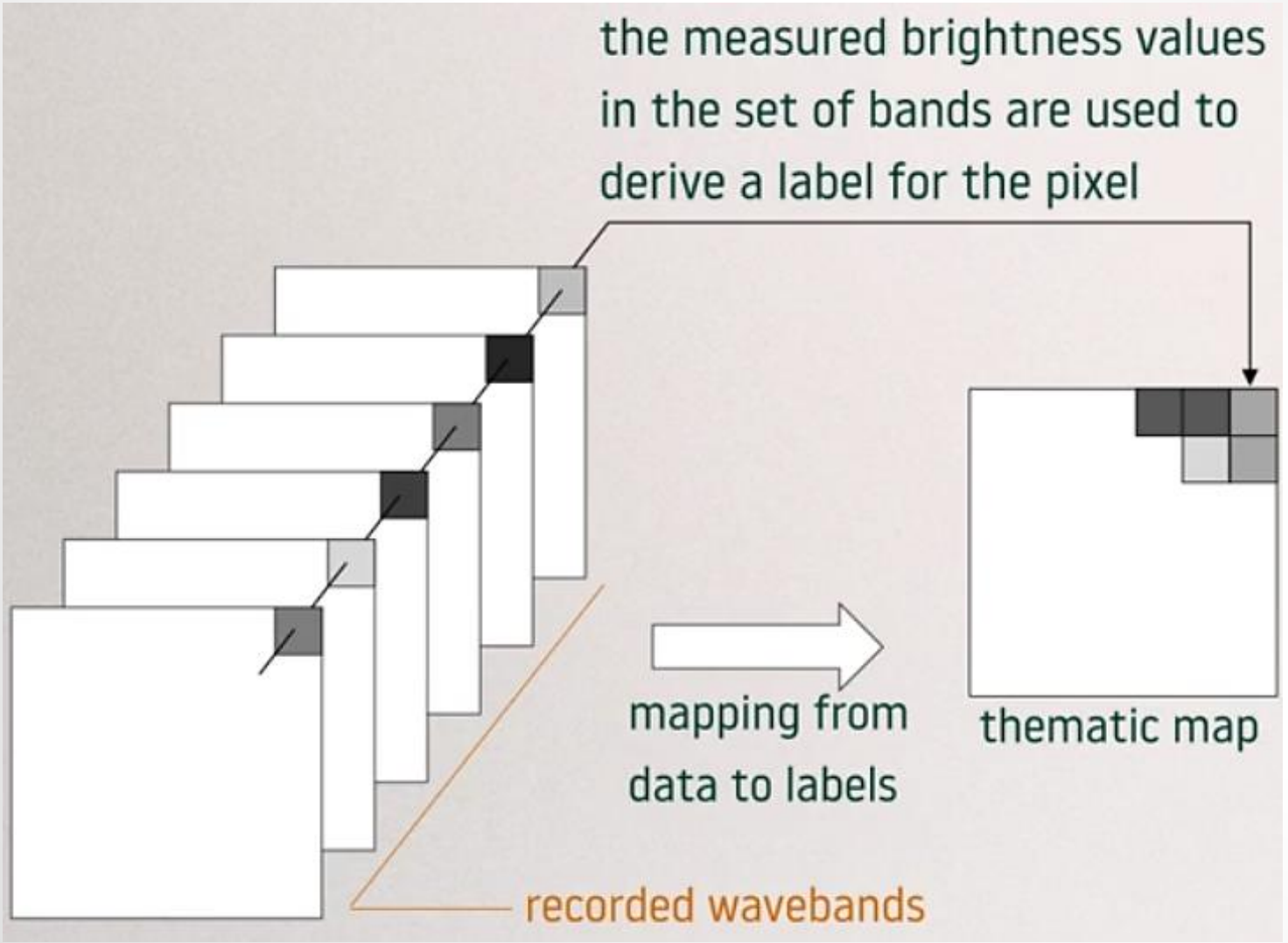
(a) Original aerial image, (b) simple ratio, (c) Normalized Difference Vegetation Index, (d) Wide Dynamic Range Vegetation Index

BAND RATIOING AND VEGETATION INDICES



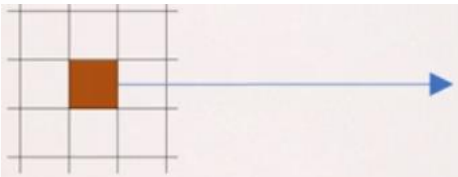
(e) Enhanced Vegetation Index (f) three-band model

CLASSIFICATION



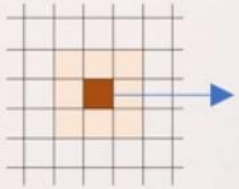
CLASSIFICATION CASES

1. At the individual pixel level, as before.



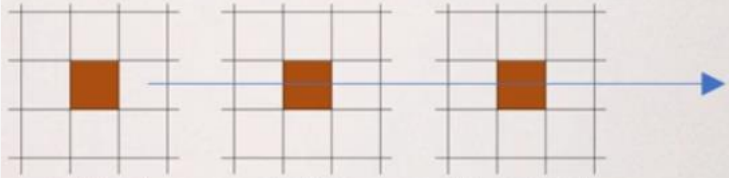
Pixel is labelled based on just its spectral properties—i.e. the pixel vector.

2. At the individual pixel level, but in the context of neighbouring pixels.



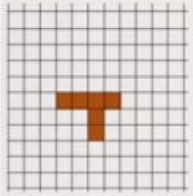
Pixel is labelled based on its spectral properties and the neighbouring labels.

3. At the individual pixel level, but involving several different data types.



Pixel is labelled taking account of all available data types.

4. For the detection and labelling of objects made up of sets of pixels.



Objects such as buildings and vehicles are identified and labelled.

CLASSIFICATION CASES

TRAINING

Sets of known pixels, with ground cover labels corresponding to those of interest to the analyst, are used to train the specific classification algorithm being used.

CLASSIFICATION, LABELLING or GENERALISATION

The trained classifier is used to add the user-specified labels to all the unknown pixels in the scene.

TESTING

A set of labelled test pixels, usually different from those employed for training, is used to test how well the classification has worked—in other words, what percentage of the test pixels is correctly labelled?

CLASSIFICATION

The maximum likelihood classifier

This was the mainstay classifier in remote sensing for many decades; it is easy to understand and its operation and results align well with the spectral reflectance characteristics of the cover types of interest. It is a good classifier when the spectral dimensionality of the data set is not high, but has limitations with hyperspectral data. We start with this technique since it provides background ideas that are important for what is to follow.

The minimum distance classifier

This is a very simple approach to classification. We use it because it can be helpful in its own right but, more importantly, it provides a good introduction to the final two techniques that we will treat.

The support vector machine

This is more complex mathematically. It has been widely used over the past two decades or so, largely because it can be used with hyperspectral data sets.

The neural network and deep learning

The neural network has been used in remote sensing for about 30 years. Although it had limited success in its original form, recent variations that lead to the convolutional neural network and deep learning have made it very popular, especially for complex problems.

CLASSIFICATION. SPECTRAL AND INFORMATION CLASSES

Most often the class labels we use in remote sensing are human constructs such as the ground cover classes of corn, wheat, sand, clay, scrubland, pine forests, diseased crops, and so on.

But the best that the recorded remote sensing image data can tell us is whether there are naturally occurring groups of pixels, or regions of the spectral space, that we hope in some way or other map well or approximately to the cover classes of interest to us.

Any identifiable groups of data in the spectral domain are called spectral classes or data classes.

The ground cover class labels we prescribe and search for in the data are called information classes.

The challenge in operational remote sensing is to gather information about the information classes in which we are interested though a discovery of the spectral class structure of the data. We actually undertake a mapping:

recorded spectral data → spectral classes → information classes

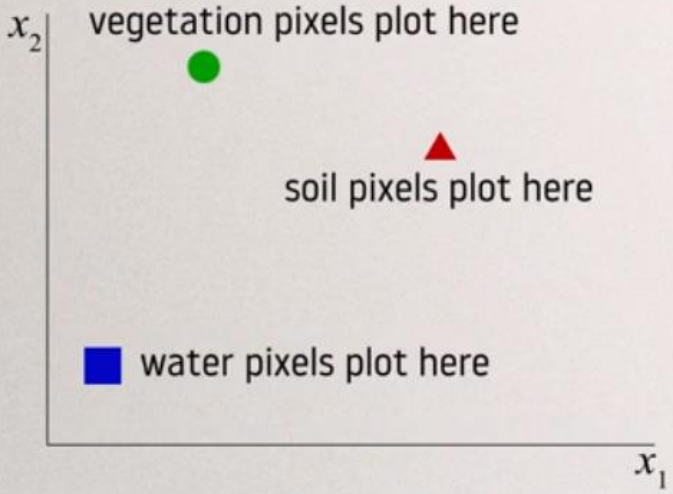
There can be several spectral classes per information class. In simple exercises we assume that the information and spectral classes are the same, which we will do for most of this module, unless otherwise indicated.

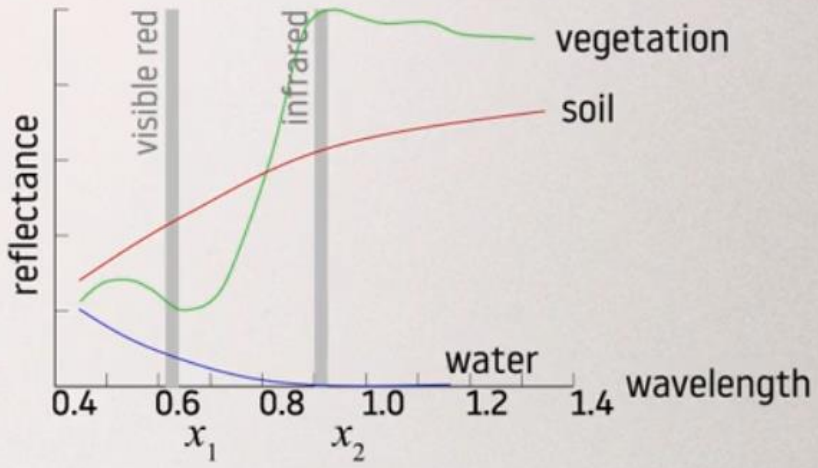
CLASSIFICATION

- Classification can be carried out on the basis of
  - The spectral properties of pixels alone (point classifiers)
  - The spectral properties of pixels and their neighbours
  - The spectral and equivalent properties of pixels represented by different data types
  - Groups of pixels defining objects
- Practical classification involves three general steps: training, labelling unknown pixels and accuracy testing.
- When training using labelled data is involved the process is called supervised classification. When there is no training the process is called unsupervised classification.
- Although we define information class labels based on a particular practical exercise, classifier algorithms actually work with classes within the data itself, called spectral classes. We have to discover the bridge between the two.

MAXIMUM LIKELIHOOD CLASSIFIER

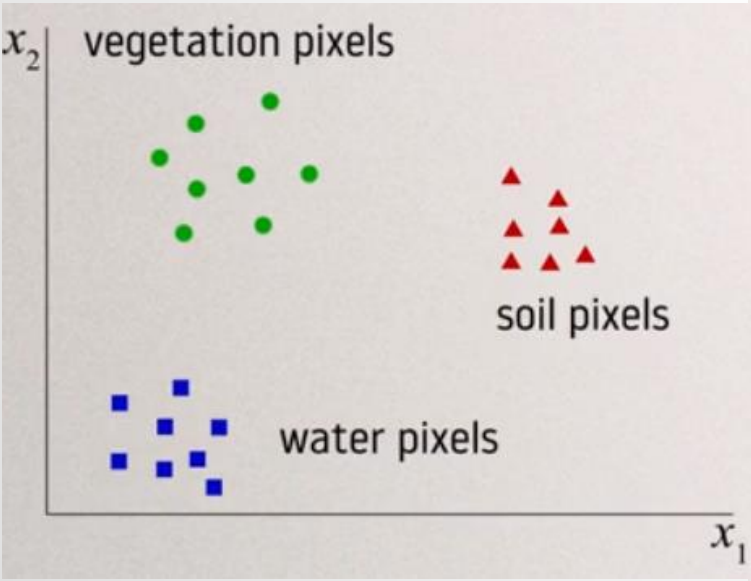
To start us on the road to developing the maximum likelihood classifier, consider the simple two dimensional spectral space shown below, based on bands which measure in the visible red and near infrared parts of the spectrum. Using our knowledge of the reflectance characteristics of the cover types of interest to us—in this example vegetation, soil and water—we can see where each pixel type will be located in the space.



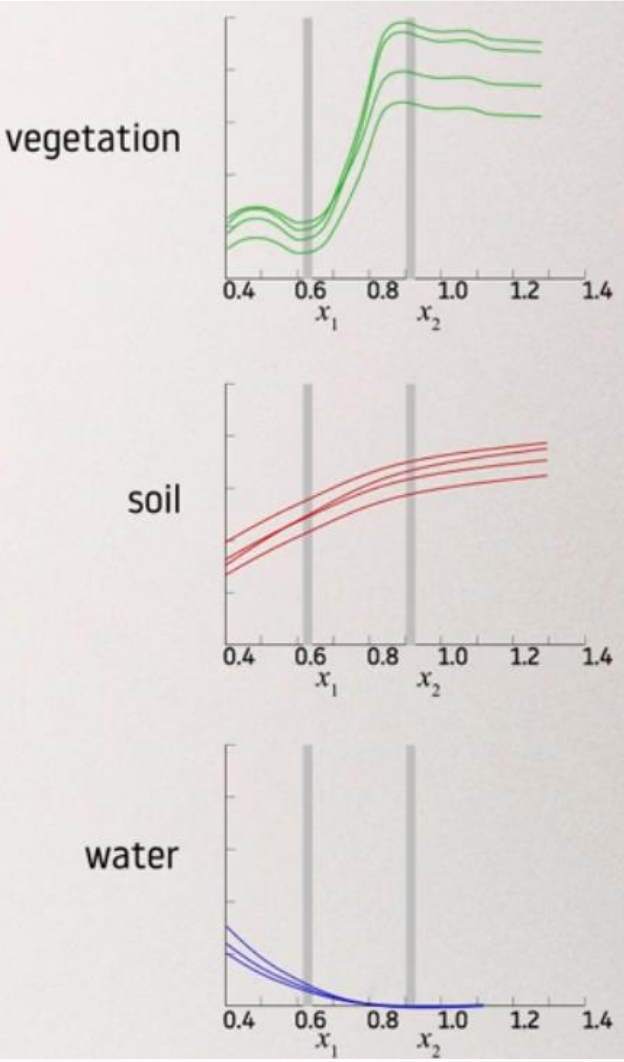


In a real image, there will be variations in each of the three spectral reflectance curves, caused by natural differences. The situation is more like on the next slide . . .

MAXIMUM LIKELIHOOD CLASSIFIER

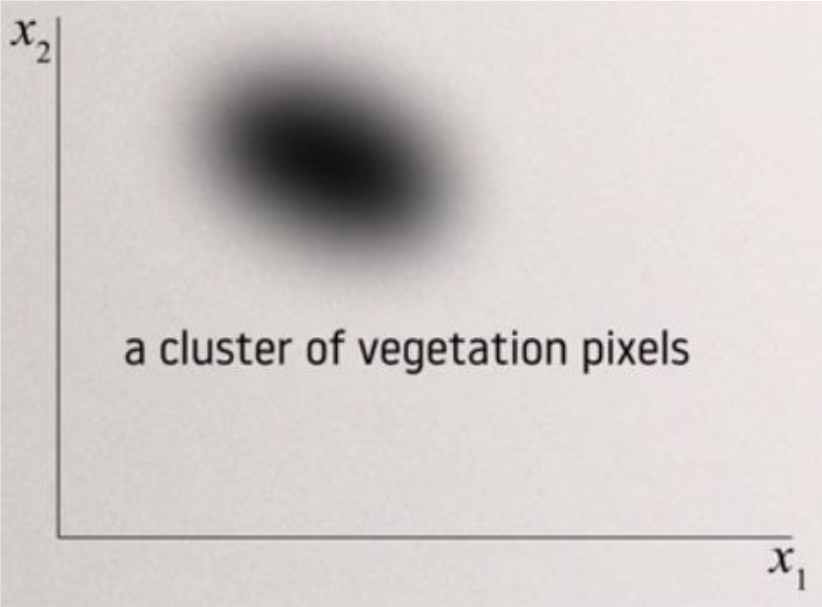


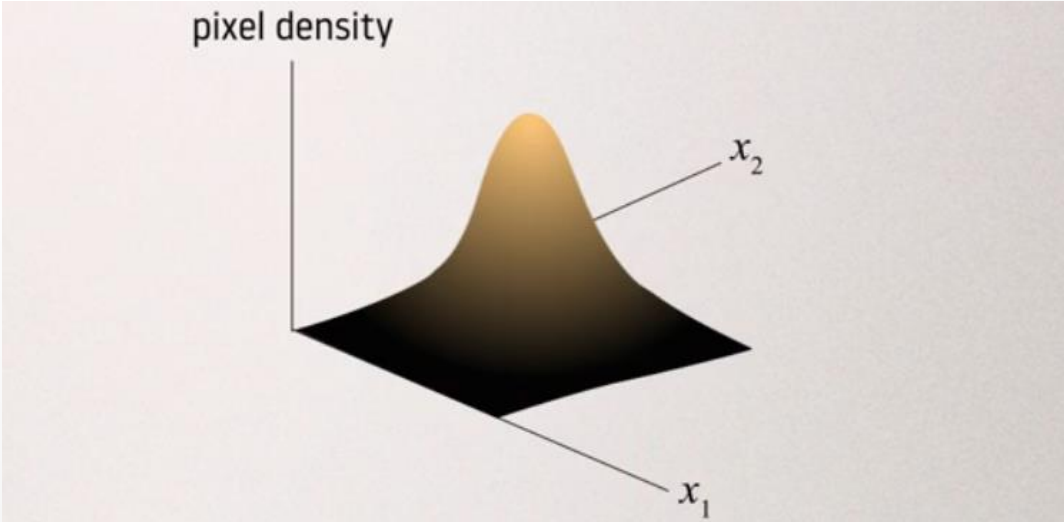
Variations in the spectral reflectance curves lead to groups of pixels, which tend to cluster in regions of the spectral space.



MAXIMUM LIKELIHOOD CLASSIFIER

In the maximum likelihood classifier we assume that the clusters of pixels are densest towards their centres, with the density dropping as we move away from the cluster centre. In general, that is a reasonable assumption.

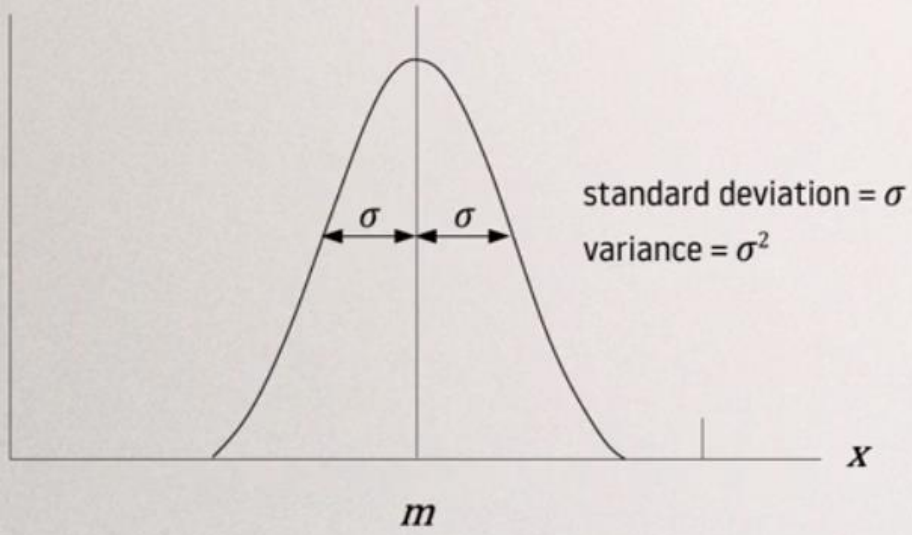


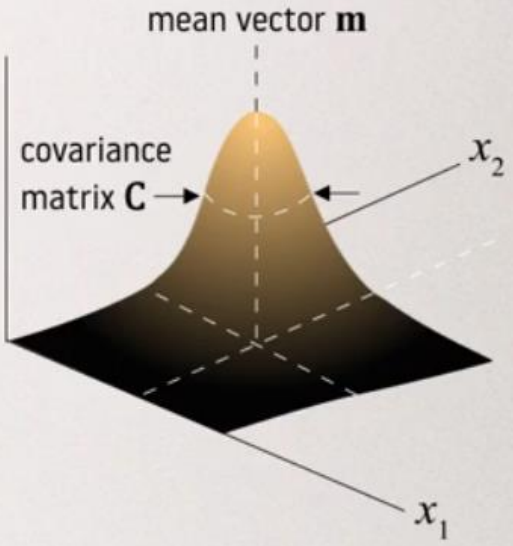


We then make the assumption that the spread of pixels within a cluster can be represented by a probability distribution, most commonly a normal (or Gaussian) distribution as shown here.

MAXIMUM LIKELIHOOD CLASSIFIER

Having assumed that the pixels within a given class can be represented by a normal distribution, we can use the properties of that distribution to help us develop a classifier algorithm. Recall that a normal distribution is defined by its mean and variance (or standard deviation). In the case of a multidimensional normal distribution the parameters are the mean vector and the covariance matrix.

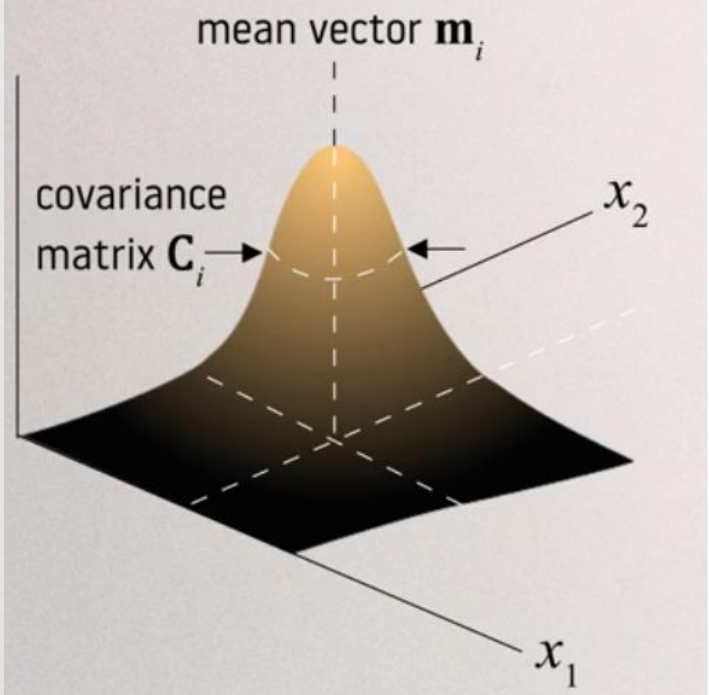




$$p(x)=(2\pi)^{-½}\,\sigma^{-1}\exp\{-½(x-m)^2/\sigma^2\}$$

$$p(\mathbf{x})=(2\pi)^{-N/2}\,|\mathbf{C}|^{-0.5}\exp\{-½(\mathbf{x}-\mathbf{m})^{\mathrm{T}}\,\mathbf{C}^{-1}\,(\mathbf{x}-\mathbf{m})\}$$

MAXIMUM LIKELIHOOD CLASSIFIER



the distribution function for class ω_i

We refer to a specific class - class i - by the symbol ω_i

We then write the equation for its probability distribution as

$$p(\mathbf{x} \mid \omega_i) = (2\pi)^{-N/2} |\mathbf{C}_i|^{-0.5} \exp\{-\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T \mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)\}$$

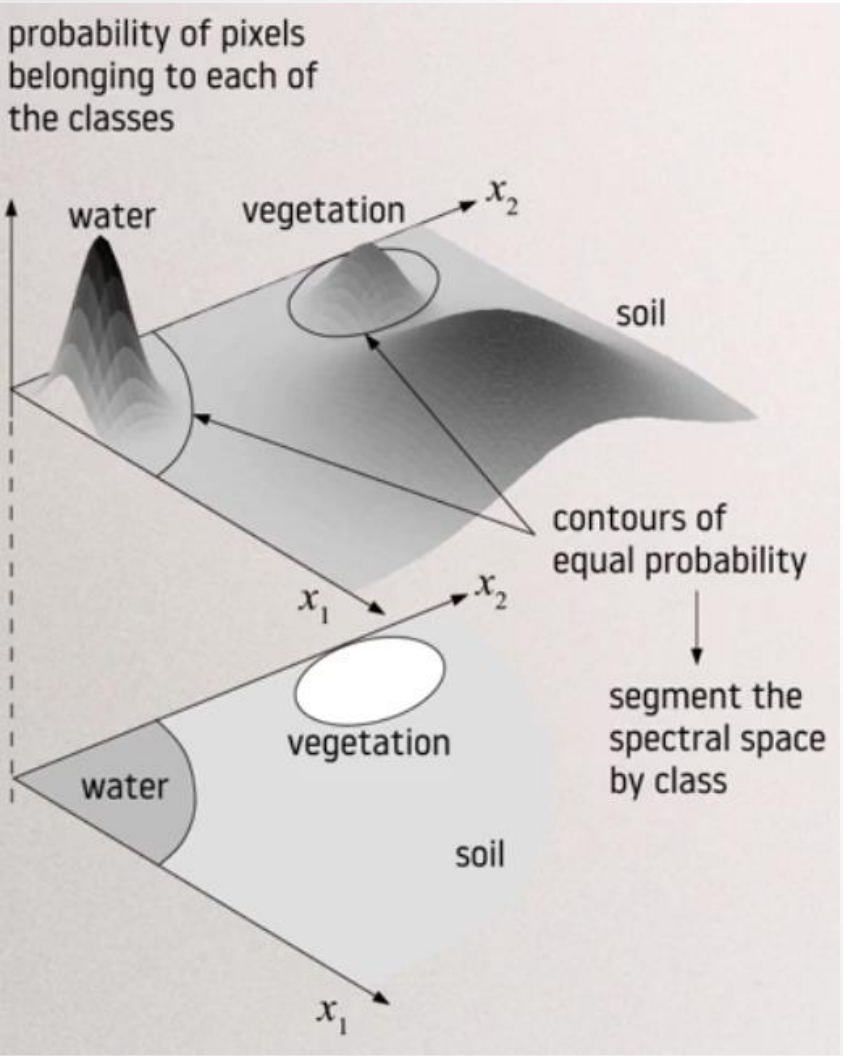
where the mean and covariance for that class

$$\mathbf{m}_i, \mathbf{C}_i$$

completely define the class distribution and, together, are sometimes called the class signature

$\mathbf{m}_i, \mathbf{C}_i$ are computed from the available training pixels for the class.

MAXIMUM LIKELIHOOD CLASSIFIER



Once we have estimated the signatures for each class using the available training data, we could choose the correct class for a pixel, with spectral vector $\mathbf{x}$, by comparing the class probabilities

$p(\mathbf{x}|\text{vegetation})$

$p(\mathbf{x}|\text{soil})$

$p(\mathbf{x}|\text{water})$

These are properly called class conditional probabilities

and choosing to allocate the pixel to the class with the highest probability.

This is shown diagrammatically opposite, which also demonstrates how contours of equal probability effectively segment the spectral space into classes.

There is however a better approach ...

MAXIMUM LIKELIHOOD CLASSIFIER

When we write the class distribution as below we are saying that the pixels of interest come from class ω_i and that this expression gives us the probability (likelihood) of finding a pixel at position $\mathbf{x}$ (i.e. with pixel vector $\mathbf{x}$) from that class.

$$p(\mathbf{x} \mid \omega_i)$$

It is a conditional probability in that it gives the likelihood of $\mathbf{x}$ occurring conditional on the class of interest being ω_i.

What about this expression? $p(\omega_i \mid \mathbf{x})$

It says that, given we are examining a pixel $\mathbf{x}$, the probability that ω_i is the correct class for that pixel is $p(\omega_i \mid \mathbf{x})$

MAXIMUM LIKELIHOOD CLASSIFIER

If we knew the full set of $p(\omega_i \mid \mathbf{x})$ we could choose the correct class for a pixel by allocating it to the ω_i corresponding to the largest $p(\omega_i \mid \mathbf{x})$. Stating this mathematically we say:

$\mathbf{x} \in \omega_i$ if $p(\omega_i \mid \mathbf{x}) > p(\omega_j \mid \mathbf{x})$ for all $j \neq i$

where the symbol ∈ means "belongs to" or "is a member of".

This equation is called a **decision rule** because it allows us to decide the correct class for a previously unseen pixel vector $\mathbf{x}$ if we know the set of $p(\omega_i \mid \mathbf{x})$. Those probabilities are called **posterior probabilities** for reasons which will become clear soon.

The problem is that we do not know values for the posterior probabilities. We know the class distribution functions $p(\omega_i \mid \mathbf{x})$ through having estimated their mean vectors and covariance matrices from the available training data, but we do not know the $p(\omega_i \mid \mathbf{x})$.

MAXIMUM LIKELIHOOD CLASSIFIER

Fortunately, there is a very famous theorem in probability theory that allows us to handle the problem of the unknown $p(\omega_i \mid \mathbf{x})$. Called **Bayes' Theorem**, it says:

$$p(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)p(\omega_i)}{p(\mathbf{x})}$$

Using Bayes' Theorem in our decision rule we have, noting that the denominator $p(\mathbf{x})$ appears on both sides and can be removed, that

$\mathbf{x} \in \omega_i$ if $p(\mathbf{x} \mid \omega_i)p(\omega_i) > p(\mathbf{x} \mid \omega_j)p(\omega_j)$ for all $j \neq i$

Thus, the decision rule is now stated in terms of the known (from training data) distributions functions and a set of new probabilities of the form $p(\omega_i)$. What are they?

MAXIMUM LIKELIHOOD CLASSIFIER

Effectively, $p(\omega_i)$ is the probability that the pixels from class ω_i appear somewhere in the image. They are not computed from anything but, instead, are a property of the scene itself. For example, if the scene being imaged consisted of 35% of water pixels, 55% of soil pixels and 10% of vegetation pixels, then $p(water) = 0.35$, $p(soil) = 0.55$, $p(vegetation) = 0.1$.

They are called prior probabilties because they are the probabilities (if we know them) with which we could guess the class membership of a pixel ***prior*** to any analysis.

In contrast the $p(\omega_i \mid \mathbf{x})$ are called posterior probabilities because they are the probabilities with which we assess the class membership of a pixel ***after*** we have carried out our analysis using the information provided by the measurement vector $\mathbf{x}$ for the pixel.

In order to use our decision rule we need to find or estimate values for the prior probabilities. In remote sensing that is usually done through some prior knowledge of the area being imaged. For example, if it was largely vegetated then a high value of $p(vegetation)$ would be chosen, and so on. If we had absolutely no way of estimating the priors, then it is common to assume they are all equal, in which case our decision rule just reverts to where we started - i.e. choosing the class for a pixel based just on the class distribution functions.

MAXIMUM LIKELIHOOD CLASSIFIER

Recall that our decision rule for deciding the class membership of an unseen pixel vector is:

$\mathbf{x} \in \omega_i$ if $p(\mathbf{x} \mid \omega_i)p(\omega_i) > p(\mathbf{x} \mid \omega_j)p(\omega_j)$ for all $j \neq i$

in which the $p(\mathbf{x} \mid \omega_i)$ are multivariate normal distributions. Thus, to apply the decision rule in that form requires evaluating the normal distribution function each time we want to allocate a label to an unknown pixel. We can generate a mathematically more convenient form if we take the natural logarithms of both sides. That will not alter the decision.

We now define the discriminant function for the class ω_i

$$g_i(\mathbf{x}) = \ln\{p(\mathbf{x} \mid \omega_i)p(\omega_i)\} = \ln p(\mathbf{x} \mid \omega_i) + \ln p(\omega_i)$$

Noting $p(\mathbf{x} \mid \omega_i) = (2\pi)^{-N/2} |\mathbf{C}_i|^{-0.5} \exp\{-½(\mathbf{x}-\mathbf{m}_i)^T \mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)\}$ this becomes

$$g_i(\mathbf{x}) = -½N\ln 2\pi - ½\ln|\mathbf{C}_i| - ½(\mathbf{x}-\mathbf{m}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}-\mathbf{m}_i) + \ln p(\omega_i)$$

MAXIMUM LIKELIHOOD CLASSIFIER

$$g_i(\mathbf{x}) = -½N\ln 2\pi - ½\ln|\mathbf{C}_i| - ½(\mathbf{x}-\mathbf{m}_i)^T\,\mathbf{C}_i^{-1}\,(\mathbf{x}-\mathbf{m}_i) + \ln p(\omega_i)$$

The first term contains no discriminating information and can be removed, leaving as the discriminant function for the Gaussian maximum likelihood rule

$$g_i(\mathbf{x}) = \ln p(\omega_i) - ½\ln|\mathbf{C}_i| - ½(\mathbf{x}-\mathbf{m}_i)^T\,\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)$$

and the decision rule is

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

MAXIMUM LIKELIHOOD CLASSIFIER

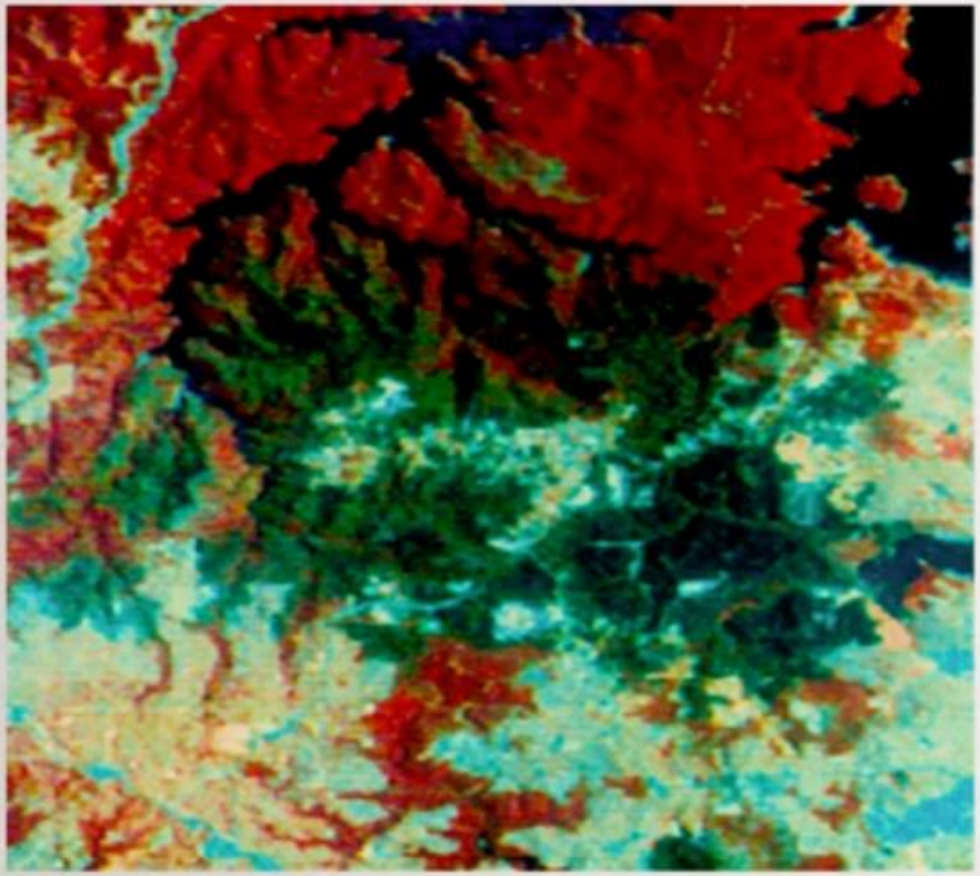


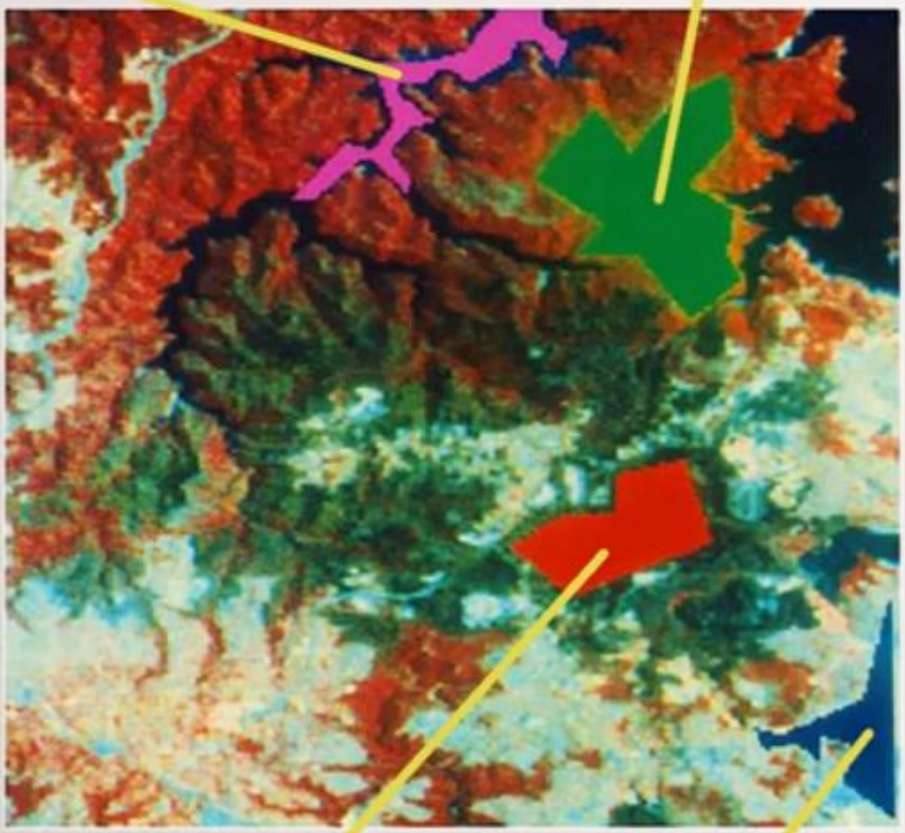
Image to be labelled or classified

1979 Landsat MSS segment of 256x276 pixels; four bands

Apparently four dominant cover types:

- vegetation
- burned vegetation (fire burn)
- urban
- water

MAXIMUM LIKELIHOOD CLASSIFIER



Select training pixels for each class

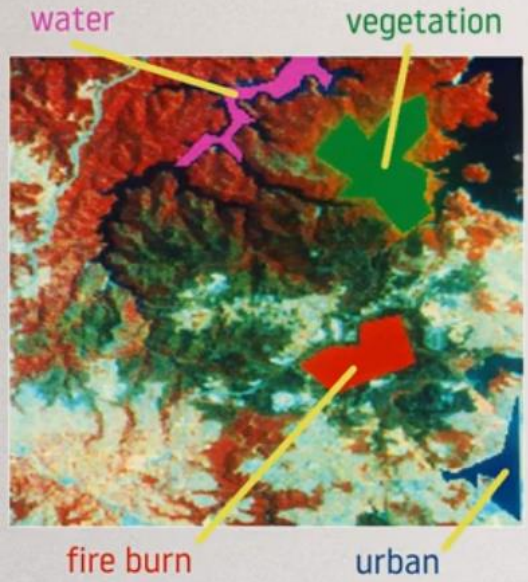
Choosing training data is simple in this case for such broadly defined classes.

Numbers of training pixels:

water	847
fire burn	1293
vegetation	2347
urban	781

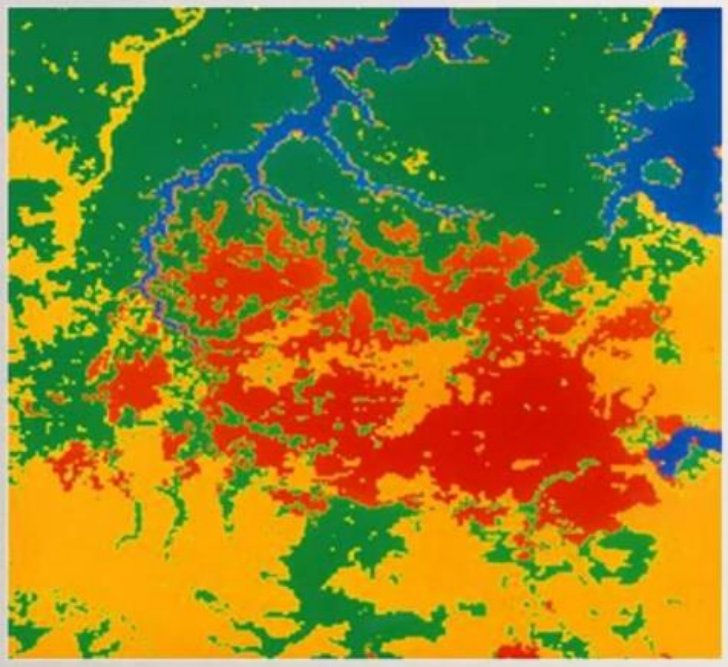
MAXIMUM LIKELIHOOD CLASSIFIER

Training involves estimating the class signatures from the available training samples.



class	mean vector	covariance matrix			
water	44.27 28.82 22.77 13.89	14.36 9.55 4.49 1.19	9.55 10.51 3.71 1.11	4.49 3.71 6.95 4.05	1.19 1.11 4.05 7.65
fire burn	42.85 35.02 35.96 29.04	9.30 10.51 12.30 11.00	10.51 20.29 22.10 20.62	12.30 22.10 32.68 27.78	11.00 20.62 27.78 30.23
vegetation	40.46 30.92 57.50 57.68	5.56 3.91 2.04 1.43	3.91 7.46 1.96 0.56	2.04 1.96 19.75 19.71	1.43 0.56 19.71 29.27
urban	64.14 60.44 81.84 72.25	43.58 46.42 7.99 -14.86	46.42 60.57 17.38 -9.09	7.99 17.38 67.41 67.57	-14.86 -9.09 67.57 94.27

MAXIMUM LIKELIHOOD CLASSIFIER



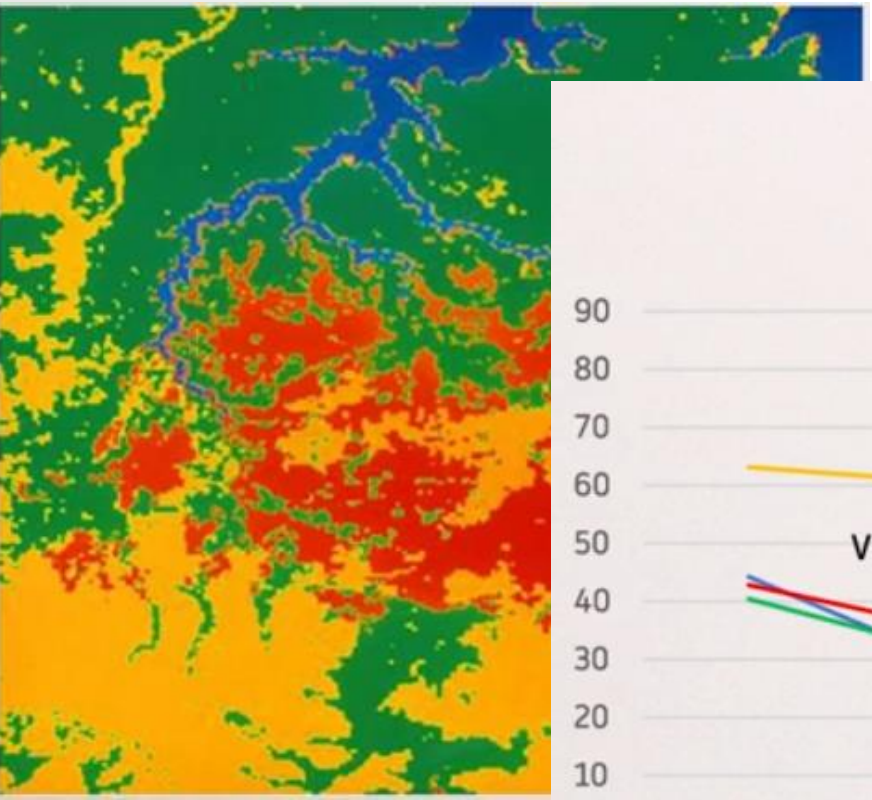
Knowing the pixel size (0.4424ha) we can also estimate the class areas:

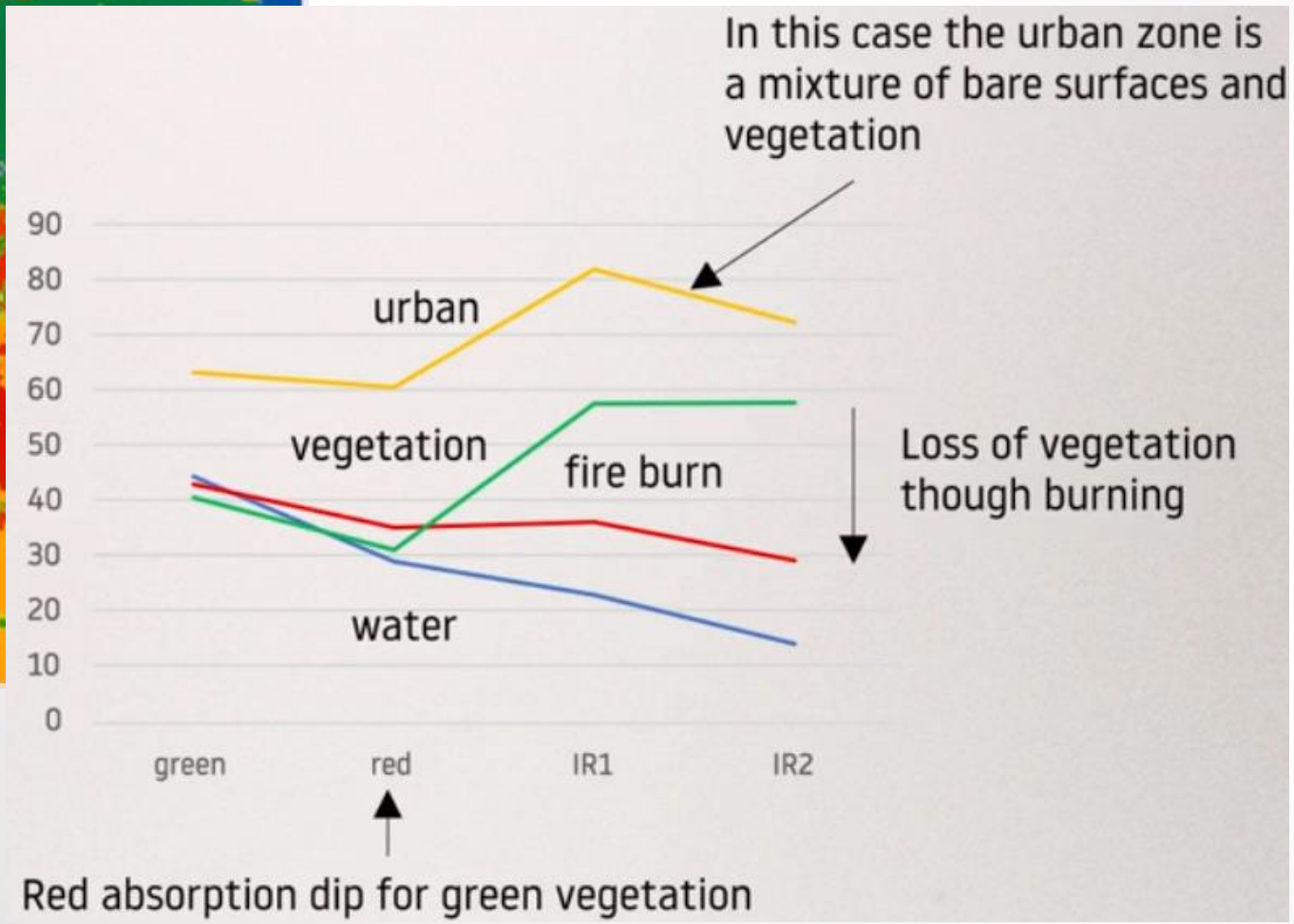
water	2,137h
fire burn	6,274ha
vegetation	12,765ha
urban	10,083ha

We labelled (classified) 70,656 pixels by using a total of 5268 training pixels. Thus, by spending time (and often field work) to find the correct labels in this case for only 7.5% of the pixels we can then label the full image. This is a good return on the time spent on training.

MAXIMUM LIKELIHOOD CLASSIFIER

Note that the elements of the mean vectors (plotted below) are consistent with what we know about the spectral reflectance curves of the cover types.





MAXIMUM LIKELIHOOD CLASSIFIER

First, how many training pixels per class are needed?

For an N dimensional spectral space the mean vector has N elements. The covariance matrix is symmetric of size NxN; it has $\frac{1}{2}N(N+1)$ 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 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. That was the case for the example just considered.

MAXIMUM LIKELIHOOD CLASSIFIER

For data with low dimensionality, say up to 10 bands, $10N$ to $100N$ 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. A hyperspectral sensor with 220 bands for example, would seem to require between 2,000 and 20,000 pixels per training class for good training. Given that, in practice, substantial field work and use of ancillary data sources might be needed to obtain acceptable training data, these numbers are bordering on the impractical.

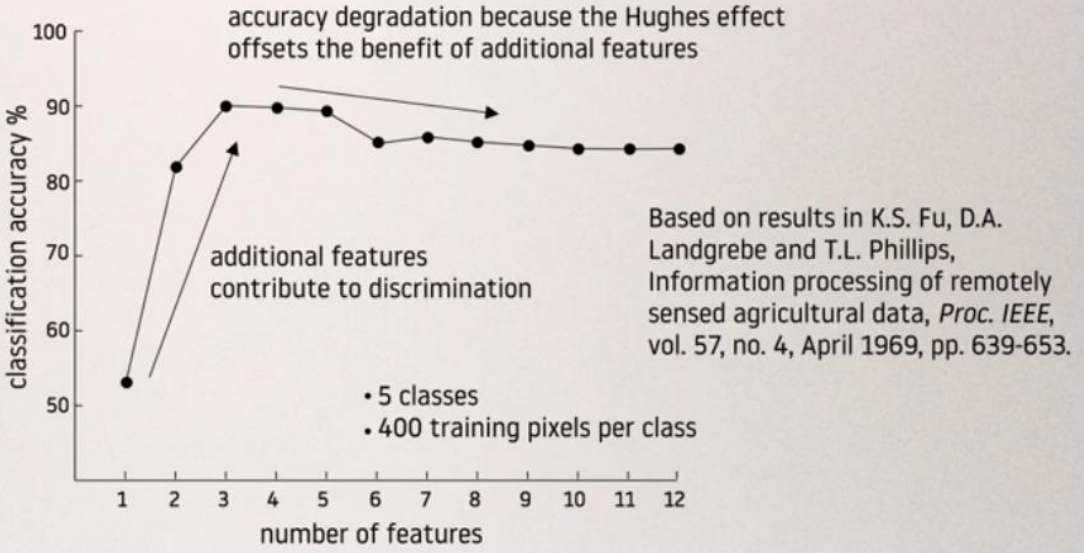
For data with high dimensionality, either methods have to found to reduce the number of bands to use without sacrificing classifier performance, or other classifiers need to be found that do not suffer the limitations of the maximum likelihood algorithm.

MAXIMUM LIKELIHOOD CLASSIFIER

The problem of not having enough samples (pixels) to estimate classifier parameters reliably with increasing dimensionality of the spectral space goes by several names. In remote sensing, it is called the Hughes phenomenon. In the general field of machine learning it is called the **Curse of Dimensionality**, because increasing dimensionality does not always lead to better results, as might first be thought.

This is disappointing. With improving technology, offering more bands of data per image, one would expect better results. But unless the numbers of training pixels per class can also be increased, poorer estimates of the covariance matrix can lead to a loss of performance. This is seen in the simple example opposite.

Although not as severe as with the maximum likelihood rule, other classifiers are also affected by the **Hughes phenomenon**.



MAXIMUM LIKELIHOOD CLASSIFIER

Now consider the shapes of the decision surfaces between classes.

Although not assured, the shapes of the decision surfaces do give us some prior qualitative assessment of the power of a classification algorithm. Higher order surfaces can fit between classes with complicated distributions better than surfaces of lower order.

For the maximum likelihood classifier the boundary — i.e. the decision surface — between the i^{th} and j^{th} classes is the locus of points $\mathbf{x}$ for which the two discriminant functions are equal:

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

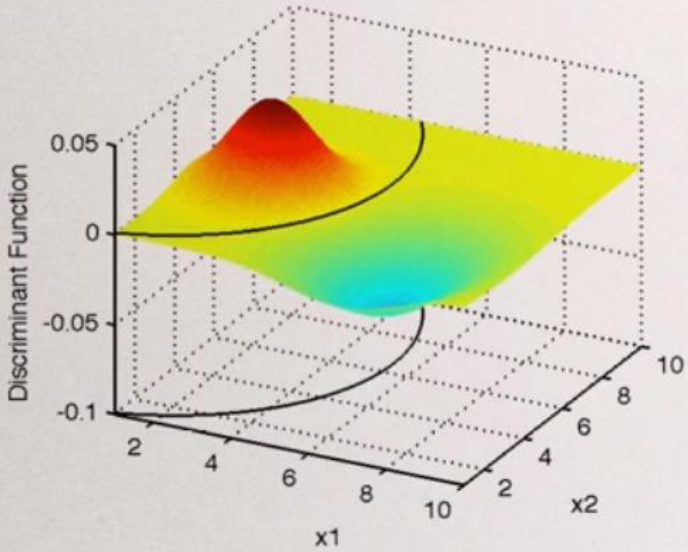
so that

$$\underbrace{\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_j)^T\,\mathbf{C}_j^{-1}(\mathbf{x}-\mathbf{m}_j) - \tfrac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T\,\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{m}_i)}_{\text{this is the difference between two second order functions, which itself will be second order (spheroidal, ellipsoidal, paraboloidal, etc.)}} + \ln p(\omega_i) - \ln p(\omega_j) - \tfrac{1}{2}\ln|\mathbf{C}_i| + \tfrac{1}{2}\ln|\mathbf{C}_j| = 0$$

MAXIMUM LIKELIHOOD CLASSIFIER

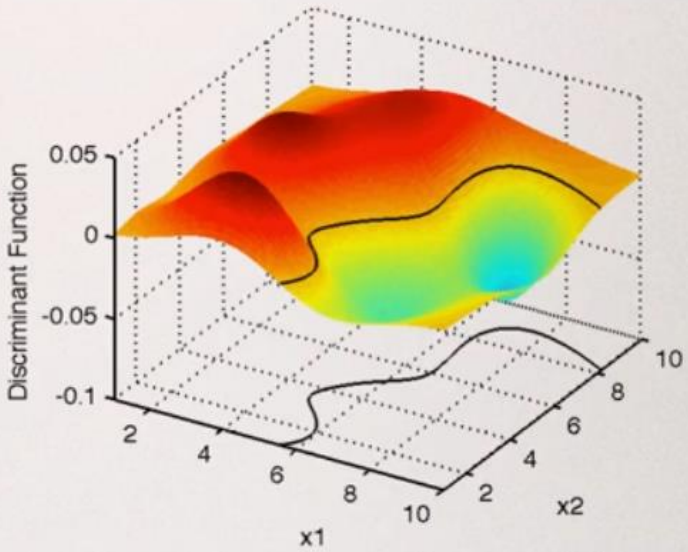
More complex decision surfaces can be generated if we use more than a single normal (Gaussian) distribution for each class. For clarity, the distributions for the second class in the example below are shown plotted negatively, but that does not affect the separating boundary. We will show how to use this approach in Module 3.

class 1: single Gaussian



class 2: single Gaussian

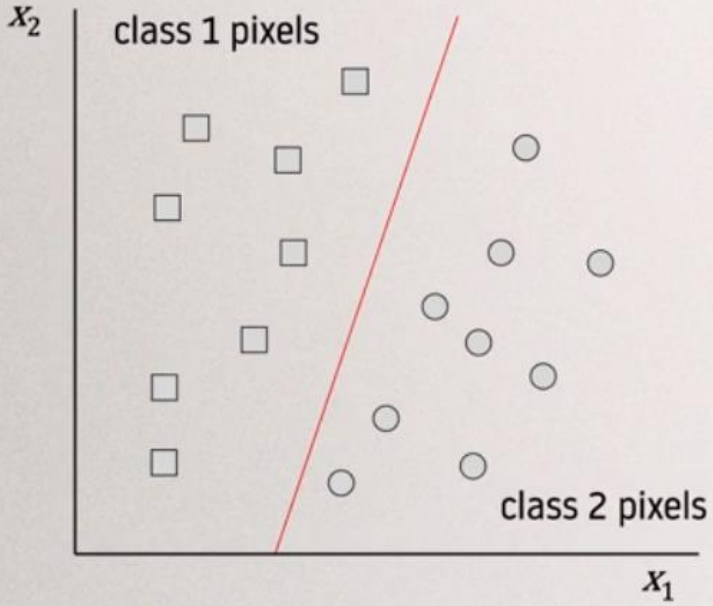
class 1: three Gaussians—these could be spectral classes



class 2: two Gaussians —these could be spectral classes

SIMPLE CLASSIFIER

To start us on the road to developing other classification algorithms, consider the following two-dimensional, two class spectral space. Although simple, it will enable us to develop all of the important concepts for what is to follow.



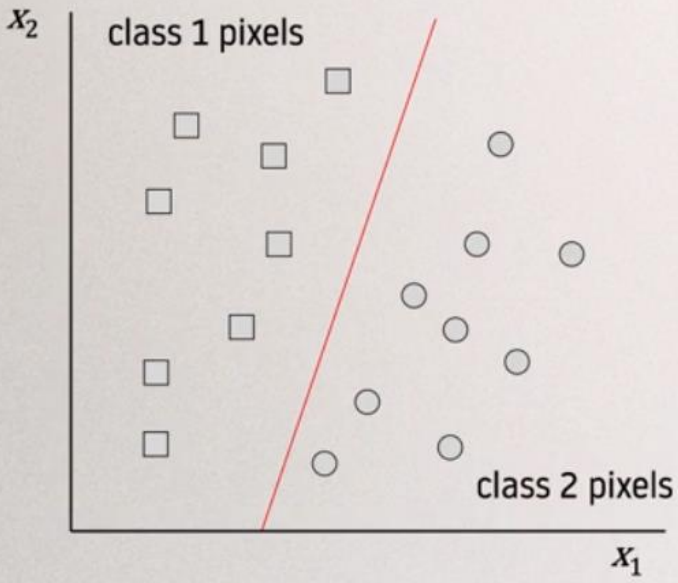
For this example a straight line can be used to separate the two classes.

We could set up a simple **decision rule** which says that pixels to the left of the line belong to class 1 whereas those to the right belong to class 2.

How can we express that mathematically?

SIMPLE CLASSIFIER

We start by setting up an equation for the line. As we know from simple geometry the line will have an equation of the form $x_2 = mx_1 + c$ or $x_2 - mx_1 - c = 0$, where m is the slope and c the axis intercept.



We can generalise the equation to:

$$w_1 x_1 + w_2 x_2 + w_3 = 0$$

in which we refer to the w_i as weights, or weighting coefficients.

The above is for the simple case of two dimensions. In N dimensions the line becomes a ***hyperlane*** and has the equation

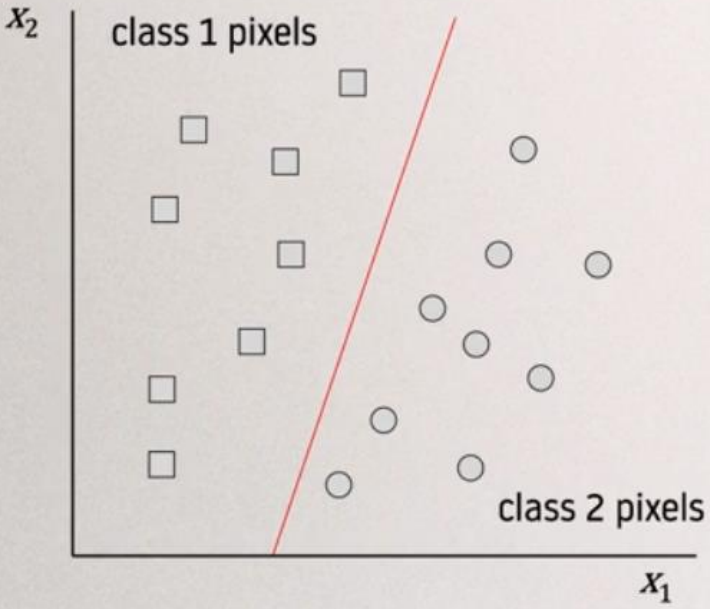
$$w_1 x_1 + w_2 x_2 + \ldots w_N x_N + w_{N+1} = 0$$

SIMPLE CLASSIFIER

The equation $w_1 x_1 + w_2 x_2 + \ldots w_N x_N + w_{N+1} = 0$ can be expressed in vector form:

$\mathbf{w}^T\mathbf{x} + w_{N+1} = 0$ or $\boldsymbol{w}.\mathbf{x} + w_{N+1} = 0$

where $\mathbf{x}$ is the pixel column vector and $\mathbf{w}$ is a column vector of the weights. If you apply the rules of vector algebra you will see that the top equation is generated by these two compact forms.



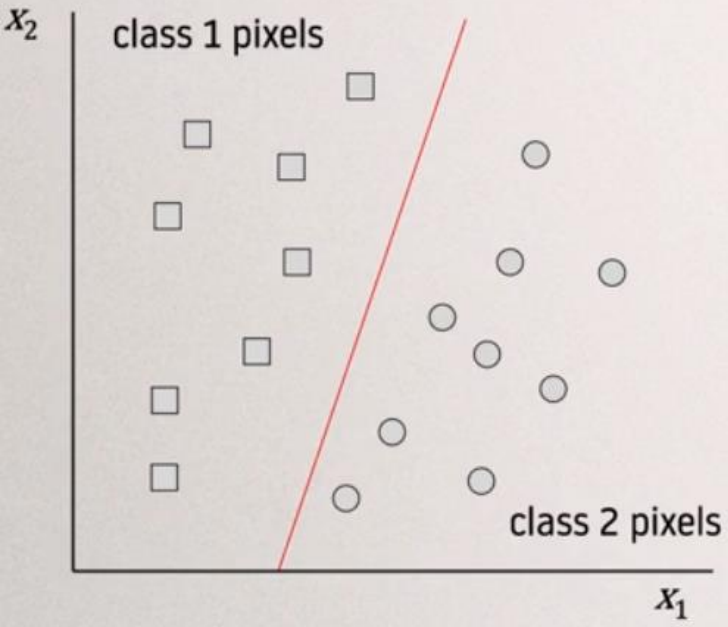
If values for $\mathbf{x}$ to the left of the separating line are substituted into the expression

$$\mathbf{w}^T\mathbf{x} + w_{N+1}$$

the result will be positive. For the values of $\mathbf{x}$ to the right of the separating line the result will be negative.

We thus have a decision rule . . .

DECISION RULE



Noting the symbol ∈ means "belongs to," we can write our decision rule for deciding the correct class label for a pixel, with vector $\mathbf{x}$, in mathematical form as:

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

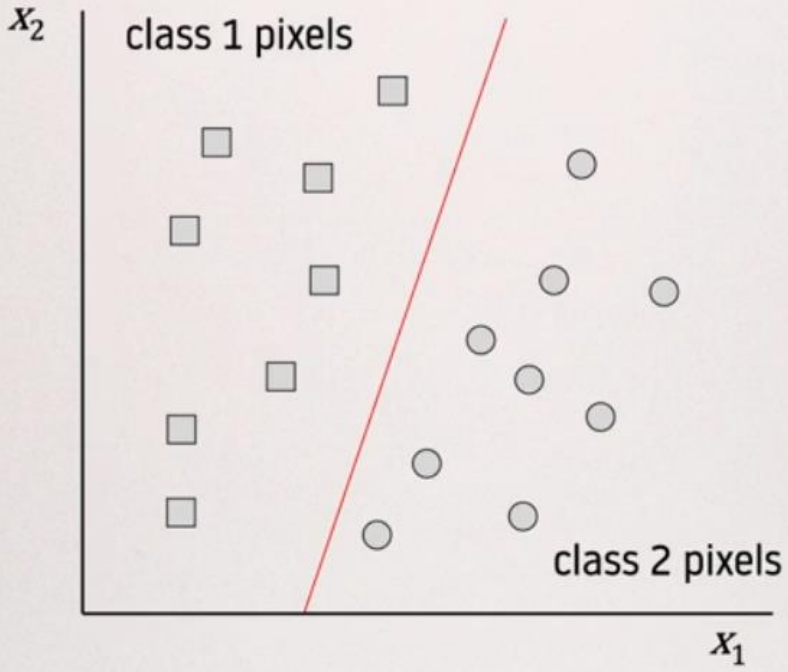
$$\mathbf{x} \in \text{class 2 if } \mathbf{w}^T\mathbf{x} + w_{N+1} < 0$$

This rule is particularly important and we will re-visit it several times during this course.

To be able to use this rule we need to know the values of w_i i.e. the weights, or the coefficients in the equation or the hyperlane (line in two dimensions).

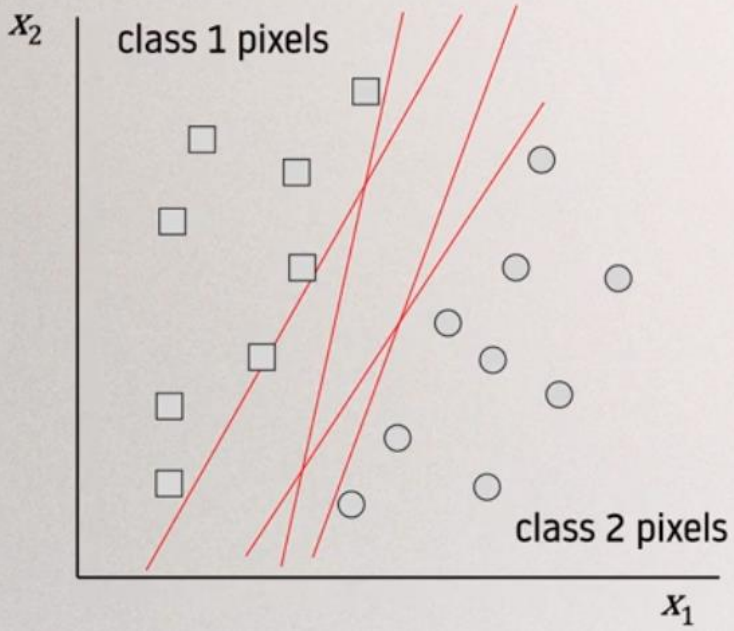
TRAINING A LINEAR CLASSIFIER

The data set we are dealing with is called “linearly separable” because a straight line (or hyperplane in general) can be used to separate the classes. Often, this is quite restrictive, but we will see later how to overcome that problem. Our objective now is to find a way of training the classifier—i.e. finding the equation of the hyperplane through determining values for the weights.



TRAINING A LINEAR CLASSIFIER

One of the earliest methods, going back to the 1960s, involves choosing an arbitrary initial separating line, which we will now call a **decision surface**, and then iterating it into position by repeated reference to each training pixel in turn. It is not used any more, but for anyone interested in this approach it will be found in N.J. Nilsson, *Learning Machines*, McGraw-Hill, NY., 1965



One of the problems with this approach is that a large number of solutions is possible, as seen in the diagram. We don't know which is the optimal solution. We will see later how to find that.

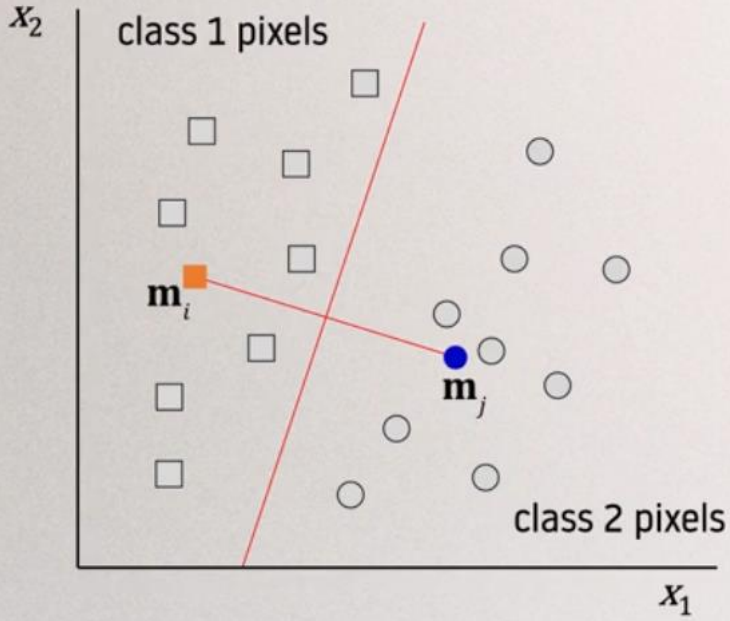
There are two other problems we must keep in mind and which we will have to consider eventually.

First, real data is often not linearly separable.

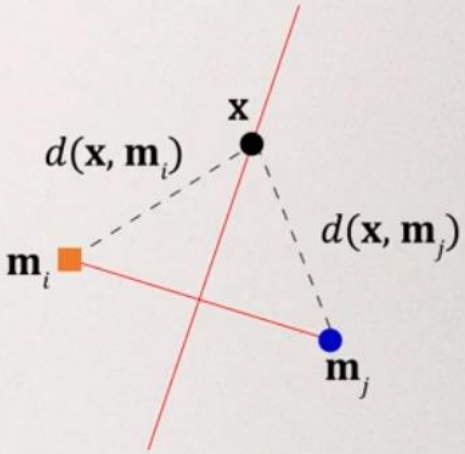
Secondly, real data usually involves more than two classes.

MINIMUM DISTANCE CLASSIFIER

Another simple approach to training is to choose as the decision surface the hyperplane which is the perpendicular bisector of the line between the means of the training classes.



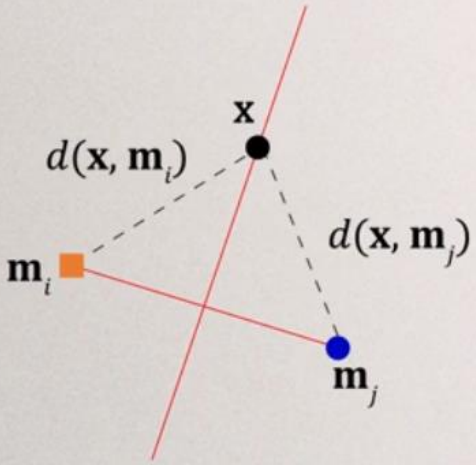
The easiest way to find the equation of the line is to find the locus of the point $\mathbf{x}$ which is equidistant from the two means



we require $d(\mathbf{x}, \mathbf{m}_i) = d(\mathbf{x}, \mathbf{m}_j)$

MINIMUM DISTANCE CLASSIFIER

If the distances are to be equal, then so will be their squares, so that $d(\mathbf{x}, \mathbf{m}_i)^2 = d(\mathbf{x}, \mathbf{m}_j)^2$



Now, from vector Euclidean geometry the squared magnitudes of the distances are

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

$$d(\mathbf{x}, \mathbf{m}_j)^2 = (\mathbf{x}-\mathbf{m}_j)^T (\mathbf{x}-\mathbf{m}_j)$$

giving

$$\mathbf{x}^T\mathbf{x} - \mathbf{m}_i^T\mathbf{x} - \mathbf{x}^T\mathbf{m}_i + \mathbf{m}_i^T\mathbf{m}_i = \mathbf{x}^T\mathbf{x} - \mathbf{m}_j^T\mathbf{x} - \mathbf{x}^T\mathbf{m}_j + \mathbf{m}_j^T\mathbf{m}_j$$

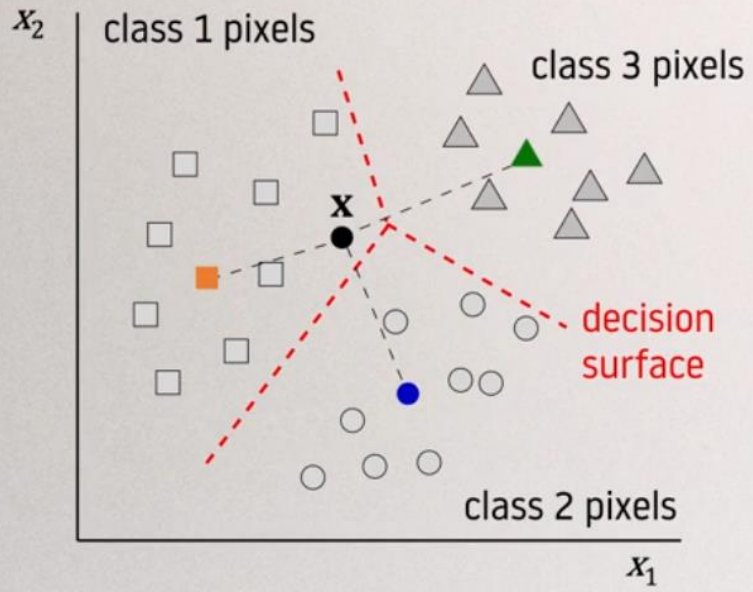
so that, noting, $\mathbf{y}^T\mathbf{z} = \mathbf{z}^T\mathbf{y}$ and $(\mathbf{y}^T+\mathbf{z}^T) = (\mathbf{y}+\mathbf{z})^T$

$$2(\mathbf{m}_j-\mathbf{m}_i)^T \mathbf{x} + (\mathbf{m}_i^T\mathbf{m}_i-\mathbf{m}_j^T\mathbf{m}_j) = 0$$

which is the equation of the separating hyperplane (linear)

MINIMUM DISTANCE CLASSIFIER

Although we have the equation for the linear decision surface, the decision rule applied in the minimum distance classifier actually makes the class allocation for a pixel based on comparing its distance from each class. It is also not just restricted to two classes. The diagram below shows the case for three, but it can be used for any number required in a given exercise. The decision surface is still linear, but in a piecewise fashion.

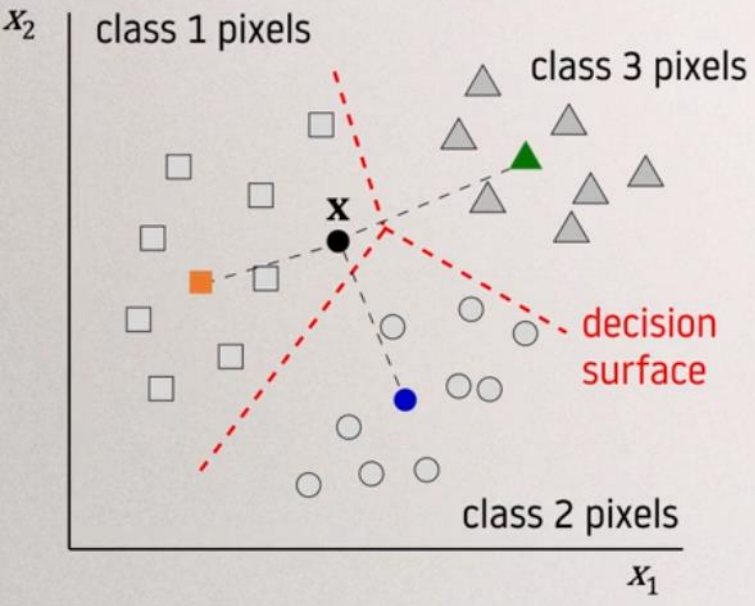


The usual decision rule for the minimum distance classifier is

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

Note that the square of the distance is used. There is no need to compute actual distance, which involves a square root operation, since the smallest distance is also the smallest distance squared.

MINIMUM DISTANCE CLASSIFIER



It is a linear classifier, in that the decision surfaces are hyperplanes.

Training data is used to find the class means.

Unknown pixels are allocated to the class of the closest mean.

It is a multi-class classifier, not just binary.

It is simple and fast, both in training and classification.

Although not as potentially powerful as some of the later algorithms we will consider, it should not be overlooked for simple exercises, especially when used together with clustering, as we will see later in the course

MAXIMUM LIKELIHOOD CLASSIFIER VS MINIMUM DISTANCE CLASSIFIER

Steps to be followed in undertaking thematic mapping in remote sensing

Maximum likelihood classifier	Minimum distance classifier
Gather labelled training data and labelled testing data.	
Use training data to estimate statistics for class: $\mathbf{m}_i, \mathbf{C}_i$ Classifier is now trained and ready to use.	Use training data to estimate statistics for class: $\mathbf{m}_i$ Classifier is now trained and ready to use.
Apply the decision rule to each pixel in the image to obtain the most likely class label for that pixel based on class conditional or posterior probabilities.	Apply the decision rule to each pixel in the image to obtain the most likely class label for that pixel based on the class means.
Produce a thematic map and a table of area estimates.	

Use labelled testing data to evaluate the accuracy of the thematic map. If not acceptable, refine the classification by examining which classes are most in error and seeing whether additional spectral classes (per information class) are needed.

SELECTING CLASSIFIER

Remember Occam's Razor, but re-stated here (with license!) as:

In remote sensing, don't necessarily use a complicated classifier when a simple one will do the job just as well

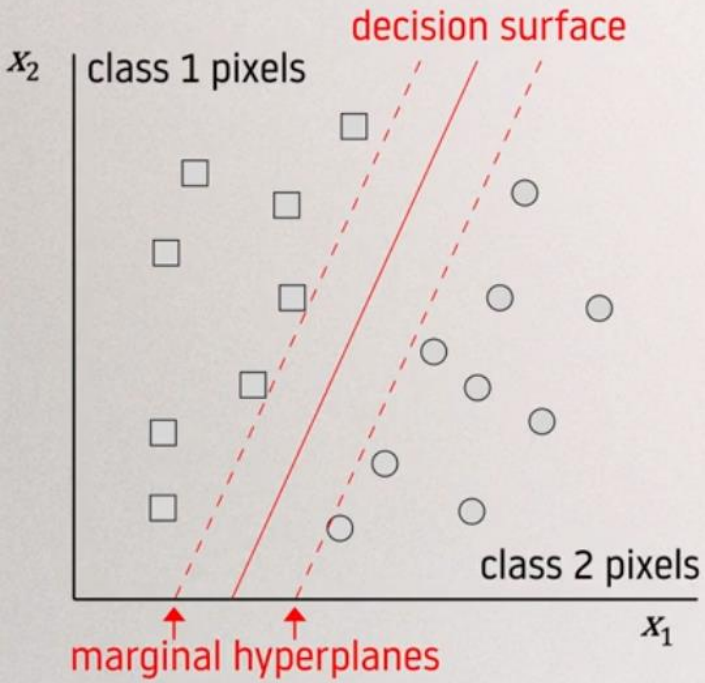
There will be occasions in remote sensing, particularly when the number of bands is small, when classifiers as simple as the minimum distance rule will do the job perfectly well.

Other simple methods are the nearest neighbour classifier, the parallelepiped classifier, the table look-up classifier and the spectral angle mapper; we will not consider them in this course, but you should find them easy to understand.

William of Ockham 1285-1347

SUPPORT VECTOR MACHINE

The support vector classifier is based on the concept that the optimum decision surface is that which lies mid-way between the two classes of pixel. In that objective it is not too different from the minimum distance classifier that we looked at in the last lecture, except that the definition of "mid-way" is chosen differently.



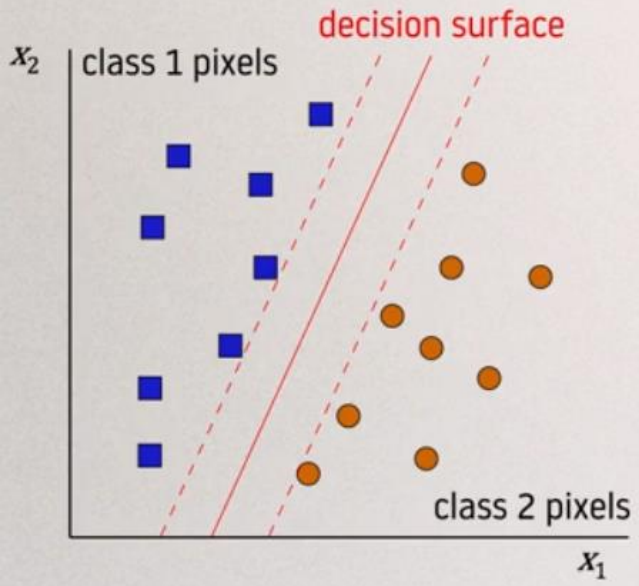
Instead of being located between the class means, the decision surface, or separating hyperplane, is located mid-way between the pixels from each class that are closest to the separating hyperplane. Our objective is to find the location and orientation of that hyperplane which actually maximises the separation between the classes.

We start this search by drawing two "marginal" hyperplanes that are parallel to the decision surface and pass through the nearest pixels from each class, as shown on the figure opposite.

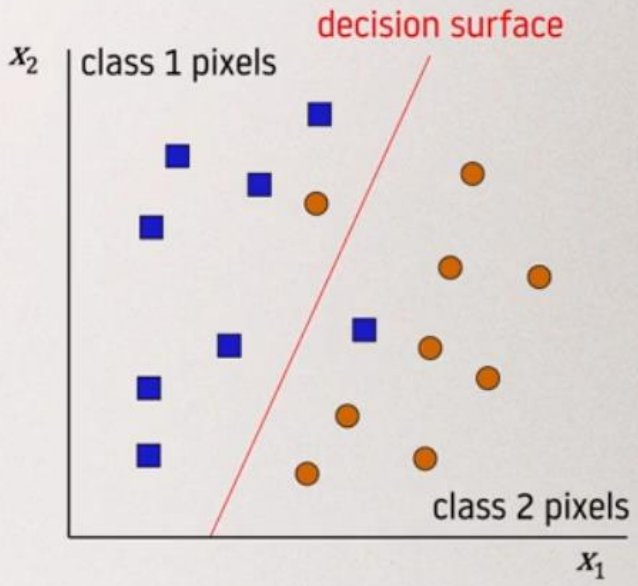
SUPPORT VECTOR MACHINE

The development of support vector classifier theory is a bit complicated. We will do it in four stages:

Stage 1 Finding the decision surface that maximises the separation between the marginal hyperplanes for a linearly separable data set.

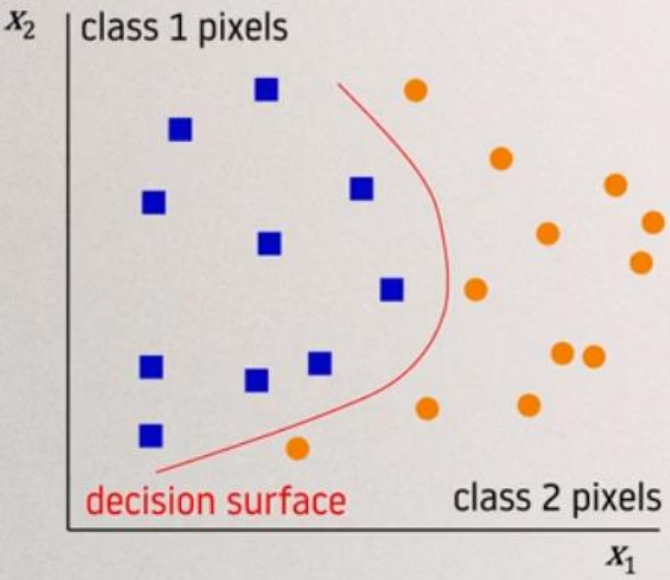


Stage 2 Accounting for overlapping classes, which is most likely to occur in practice.

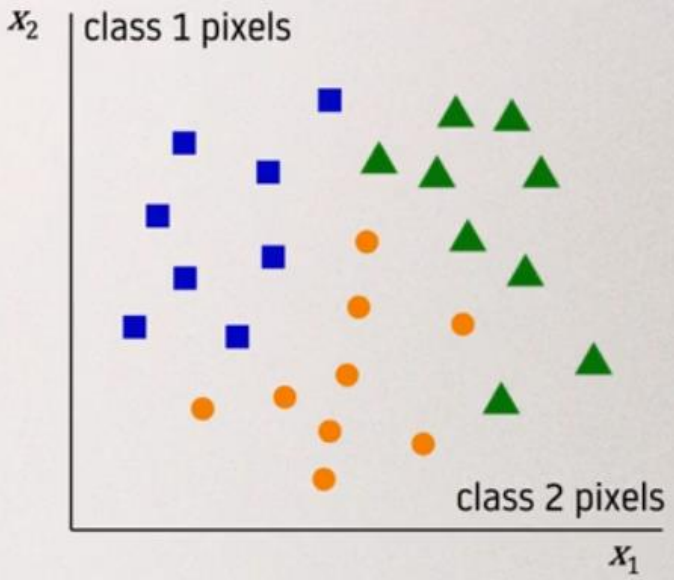


SUPPORT VECTOR MACHINE

Stage 3 Handling non-linearly separable data sets.



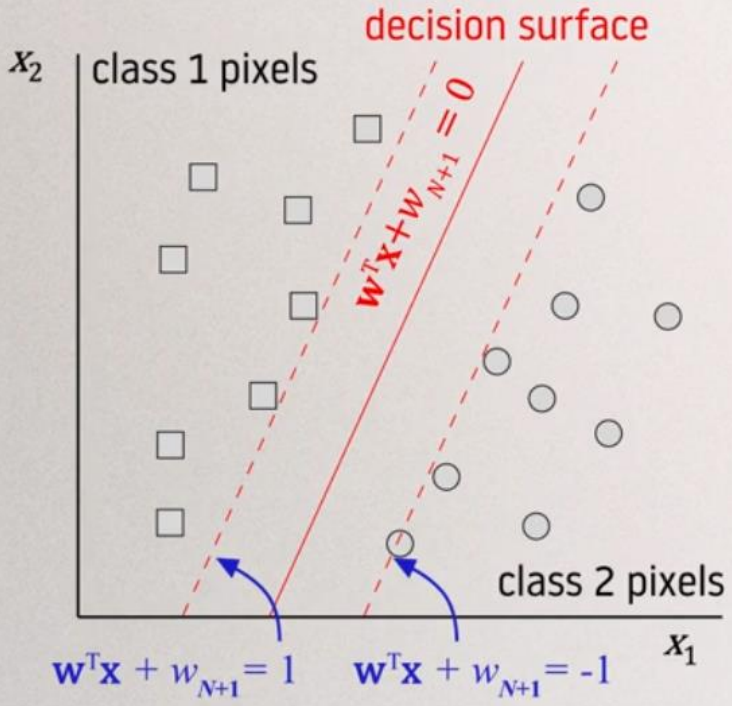
Stage 4 Handling more than two classes of pixels.



SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE: STEP 1, MAXIMISING THE MARGIN FOR LINEAR SEPARABILITY

The equations for linear decision surfaces are given by the expression we derived in the first lecture of this module: $w_1x_1 + w_2x_2 + \ldots w_Nx_N + w_{N+1} = 0$ which in vector form is $\mathbf{w}^T\mathbf{x} + w_{N+1} = 0$.



As with the minimum distance rule:

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

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

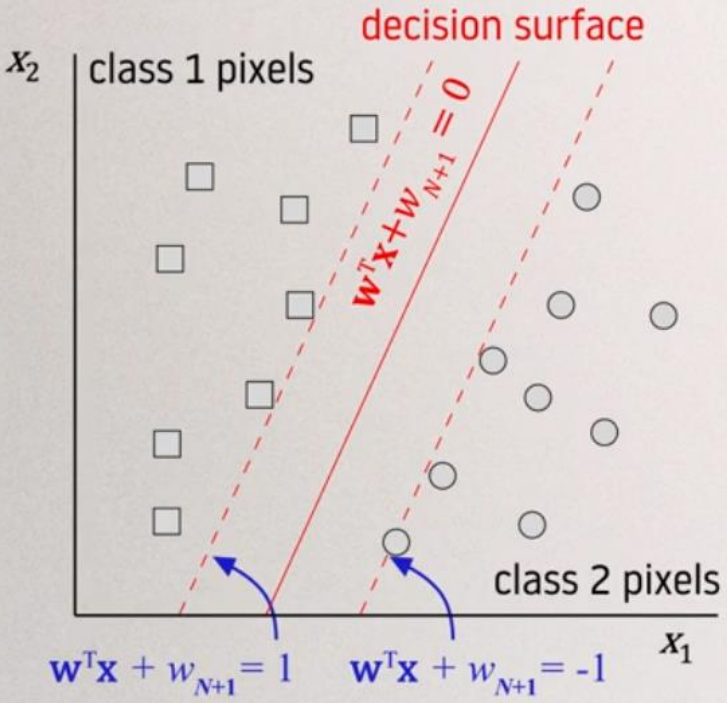
The weights in these expressions are not unique. Provided we keep their values in the same relative proportions we can scale them up and down without affecting the equation itself.

Importantly, we can scale the weights in such a manner that the equations of the marginal hyperplanes are as shown on the figure.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE: STEP 1, MAXIMISING THE MARGIN FOR LINEAR SEPARABILITY

Our immediate objective now is to find the weights such that the gap between the marginal hyperplanes is as large as possible.



We call that gap the "margin". On the next slide we see that its value is

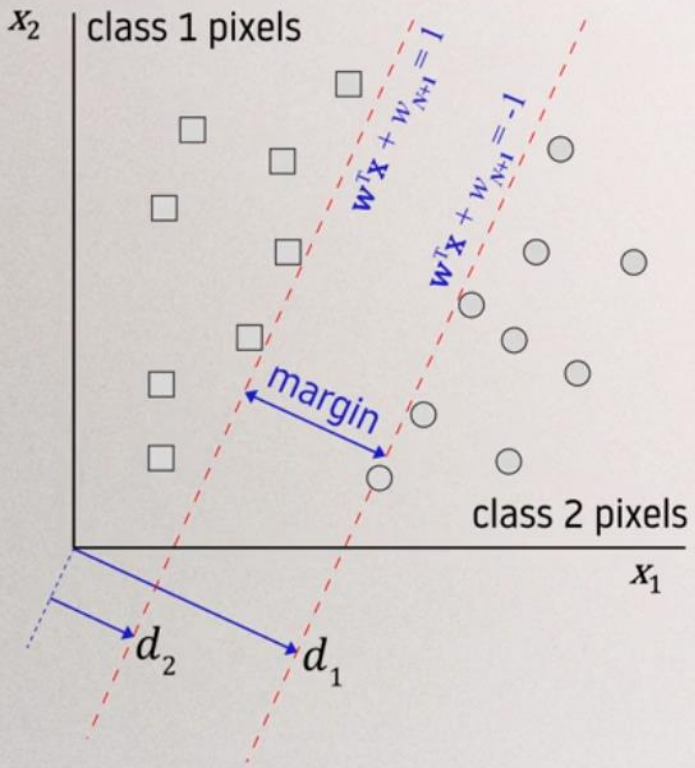
$$\text{margin} = {}^{2}/_{\|\mathbf{w}\|}$$

where $\|\mathbf{w}\|$ is called the norm of the vector. It is the Euclidean length, given as the square root of the sum of the squares of the individual weights.

Strictly, the margin is defined as the distance of the closest training sample to the decision surface, but the choice here is also acceptable for what is to follow.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —CALCULATING THE MARGIN



The perpendicular distance from a point $\mathbf{x}_0$ to a plane, in general, is

$$\frac{|\mathbf{w}^T\mathbf{x}_0 + w_{N+1}|}{\|\mathbf{w}\|}$$

If $\mathbf{x}_0 = 0$, the origin, and the plane is the right hand marginal hyperplane in the diagram, then the distance d_1 is given by:

$$\frac{|1 + w_{N+1}|}{\|\mathbf{w}\|}$$

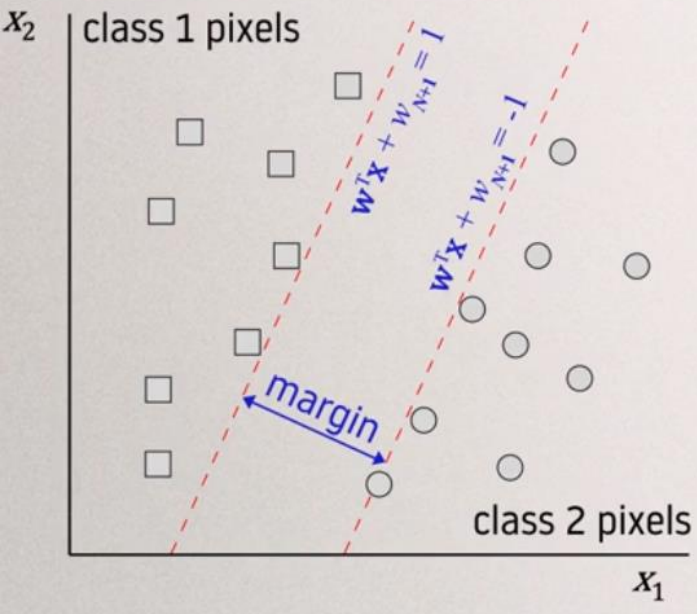
Likewise the distance d_2 is given by:

$$\frac{|-1 + w_{N+1}|}{\|\mathbf{w}\|}$$

So that the margin is $d_1 - d_2 = {}^2/_{\|\mathbf{w}\|}$

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —MAXIMISING THE MARGIN



We want to maximise the margin $^2/_{\|\mathbf{w}\|}$ which is equivalent to minimizing the norm of the weight vector.

However, in seeking to minimise $\|\mathbf{w}\|$ we want to ensure that the resulting marginal hyperplanes are, by definition, placed such that the training pixels are on their correct sides, as depicted in the diagram. This represents a constraint on the minimization of the weight vector.

The way to handle that constrained minimisation is to use the technique of Lagrange multipliers*. That entails setting up a function called the Lagrangian $\mathcal{L}$ which consists of what we want to minimize, from which is subtracted a proportion of each constraint. The proportions are the Lagrange multipliers.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —MAXIMISING THE MARGIN

We choose as the Lagrangian, the following expression. The half and square of the weight vector norm are chosen for later convenience and don't alter what we are doing.

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i f_i$$

The $\alpha_i \geq 0$ are a set of Lagrange multipliers, and the f_i are conditions, one for each training pixel, that ensure the pixels are on the correct side of their respective marginal hyperplane. What can we use for those conditions?

Consider a new set of binary variables y_i one for each of the i^{th} training pixels.

y_i has the value +1 for class 1 pixels and -1 for class 2 pixels.

Now for all pixels $\mathbf{x}_i$ in class 1 beyond the corresponding marginal hyperplane $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1} \geq 1.$ (A)

So for those class 1 pixels $y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}) \geq 1$ or $y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1})$ -1 $\geq 0.$

Exactly the same condition applies for class 2 pixels because the signs of y_i and equation (A) both change.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —MAXIMISING THE MARGIN

Thus the equation for the Lagrangian from the previous slide now becomes

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \{y_i(\mathbf{w}^T\mathbf{x}_i + w_{N+1}) - 1\}$$ where we have used $$f_i = y_i(\mathbf{w}^T\mathbf{x}_i + w_{N+1}) - 1$$ (B)

What we need to do now is minimise with $\mathcal{L}$ respect to the weights. While we are attempting to do that the second term in the Lagrangian is trying to make $\mathcal{L}$ bigger for pixels that are on the wrong side of their marginal hyperlane, because in that case the entry in the curly brackets $\{\}$ will be negative; recall the α_i are by definition non-negative.

The mathematics now becomes a little tedious but leads to some remarkable and important results. If you choose not to follow the detail, we will still summarise the important results at the end.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —MAXIMISING THE MARGIN

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \{y_i(\mathbf{w}^T\mathbf{x}_i + w_{N+1}) - 1\}$$ where we have used $f_i = y_i(\mathbf{w}^T\mathbf{x}_i + w_{N+1}) - 1$ (B)

To minimise $\mathcal{L}$ with respect to the weights we equate its partial derivative to zero, in the usual way. Noting that $\|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ we find.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \text{ so that } \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \quad \text{(C)}$$

Thus we know the weights if we know the α_i.

This means that we can set up the equation of the decision surface from the (full) set of training pixels, knowing also which class each pixel belongs to – the y_i. In a sense, this is what happens with all classifier training methods: the training pixels are used to find the decision surface. But this will soon become simpler.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —MAXIMISING THE MARGIN

Minimising $\mathcal{L}$ with respect w_{N+1} we find

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{N+1}} = -\sum_i \alpha_i y_i = 0$$ But we don't yet know the value of w_{N+1}. That will come later. (D)

We can now use (C) and (D) to simplify the original Lagrangian expression at (B).

First, using (C) we note that

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

Substituting this into the Lagrangian formula, and using (D) we find we can express $\mathcal{L}$ in the so-called dual form

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i \quad \text{(E)}$$

By minimizing the Lagrangian with respect to the weights we now have it expressed in terms of the Lagrange multipliers. Remember their role is to try to make $\mathcal{L}$ big, so we can now seek to maximise (E) with respect to the α_i.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —MAXIMISING THE MARGIN

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i \qquad \text{(E)}$$

Maximising this expression yields values for the α_i. In practice, that is a complicated procedure so it is usually carried out numerically.

There is now an important additional constraint we haven't yet met. It comes from the application of the Lagrange multiplier technique, and is one of the so-called Karush-Kuhn-Tucker (KKT) conditions:

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

This is an amazing constraint. It says that either α_i is zero or the term in the $\{\}$ brackets is zero. The latter is only true for pixels lying on one of the marginal hyperplanes, in which case α_i is non-zero. The α_i corresponding to all other training pixels is zero, so those pixels become unimportant in the training process.

The pixels lying on the marginal hyperplanes are now called support vectors, because they are the only ones needed to support training.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —MAXIMISING THE MARGIN

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_i \qquad \text{(E)}$$

Assume we have now solved (E) under these circumstances. That gives us the remaining unknowns in the expression for the weight vector we derived earlier: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$, so we can now completely specify the vector and thus the decision surface (although we still need a value for w_{N+1}). Because of the KKT condition of the previous slide we can write this expression for the weight vector in terms of just those pixels lying on the marginal hyperplanes. In a sense, that is quite logical because it is those pixels which define the margin!

Thus $\mathbf{w} = \sum_{i \in \mathcal{S}} \alpha_i y_i \mathbf{x}_i$ where $\mathcal{S}$ is the set of support vectors, which is much smaller than the original set of training pixels vectors.

Thus only a small subset of the training pixels is required to find the decision surface.

Apart from w_{N+1} we have now trained the support vector classifier. In the next lecture we will look at how it is used.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —THE CLASSIFICATION STEP

From the previous lecture we have the weight vector for the decision surface defined just in terms of the support vectors—i.e. just those training patterns that lie on the marginal hyperplanes.

$$\mathbf{w} = \sum_{i \in \mathcal{S}} \alpha_i y_i \mathbf{x}_i$$

Recall our decision rule is

$\mathbf{x} \in$ class 1 if $\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{N+1} > 0$ or $\mathbf{x} \in$ class 1 if $z > 0$

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

Using the above form for $\mathbf{w}$ $$z = sgn\{\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{N+1}\} = sgn\left\{ \sum_{i \in \mathcal{S}} \alpha_i y_i \mathbf{x}_i{}^{\mathrm{T}}\mathbf{x} + w^{N+1} \right\}$$

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —THE CLASSIFICATION STEP

Thus decisions about the class membership of the unknown pixel $\mathbf{x}$ are made based on the set support vectors.

$$z = sgn\{\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_{N+1}\} = sgn\left\{\sum_{i\in\mathcal{S}} \alpha_i\, y_i\, \mathbf{x}_i^{\mathrm{T}}\mathbf{x} + w_{N+1}\right\} \qquad \text{(F)}$$

Now what about the value for w_{N+1}? The simplest approach is to choose a support vector from each class — call them $\mathbf{x}(1)$ and $\mathbf{x}(-1)$ respectively — and substitute them into their respective marginal hyperplane equations, so that we have

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

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

giving $w_{N+1} = -\frac{1}{2}\,\mathbf{w}^{\mathrm{T}}[\mathbf{x}(-1) + \mathbf{x}(1)]$ A variation on this averages over groups of support vector pairs

We now have all the parameters in (F) specified, so that we can now label unknown pixels.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —THREE OTHER STAGES

There are three practical limitations in the material presented so far that have to be overcome before the support vector classifier can be used in remote sensing.

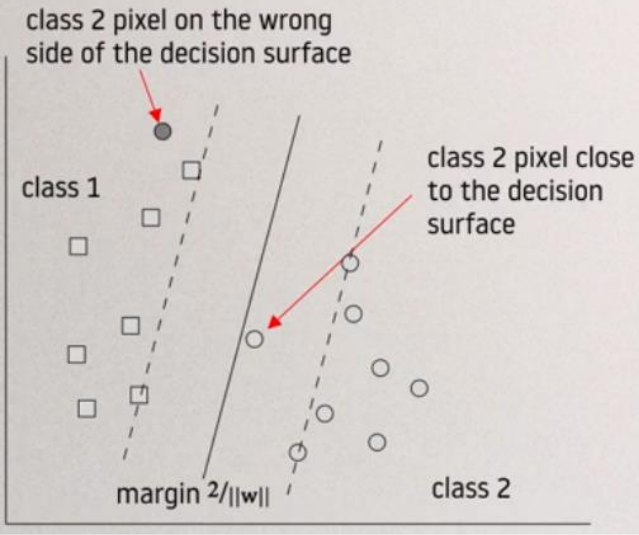
- It assumes the two classes are completely separable
- It is a linear classifier.
- It is binary, in that it only separates two classes of data

We now turn our attention to overcoming those limitations.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —OVERLAPPING CLASSES

It is unrealistic to think that the two classes of data, in practice, will be completely separate as in our previous diagrams. Instead, the situation will be more like that below, in which there will be some pixel vectors on the wrong side of the decision surface. In this illustration we have only shown one such pixel but, in practice there could be several. We have also shown a pixel close to the decision surface; the reason for that will become clear soon.



The support vector approach developed so far cannot handle the situations shown in the diagram and requires modification. That is done by relaxing the requirement of finding a maximum margin solution.

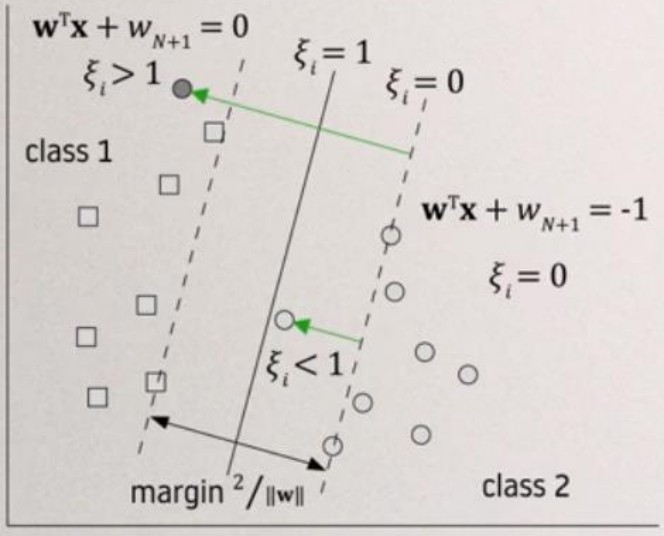
Instead, we agree that such a goal is not possible for all training pixels and accept that some will be in error during training.

We therefore, introduce some **"slackness"** into training.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —OVERLAPPING CLASSES

To handle this situation we introduce a set of positive "slack variables" ξ_i, one for each training pixel, which we use to modify the original decision rule.



Thus, instead of requiring

$$y_i\,(\mathbf{w}^T\mathbf{x}_i + w_{N+1}) \geq 1$$

for a pixel to be on its right side of the decision surface and thus correctly classified, we modify the rule to

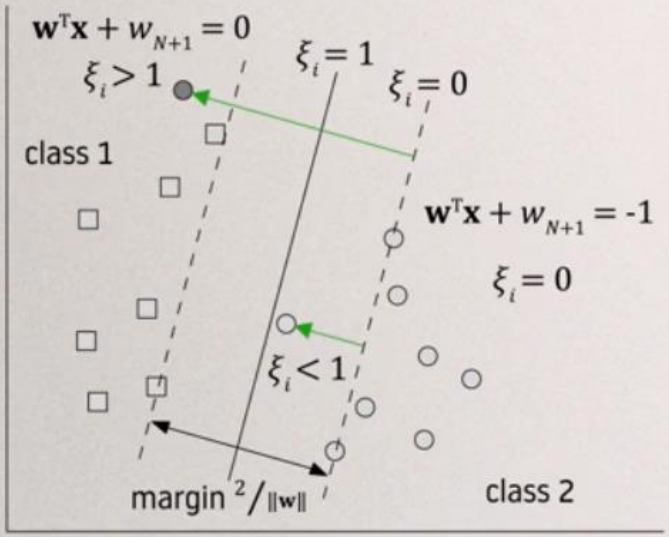
$$y_i\,(\mathbf{w}^T\mathbf{x}_i + w_{N+1}) \geq 1 - \xi_i$$

What values does ξ_i take, and how do they affect the rule?

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —OVERLAPPING CLASSES

$y_i(\mathbf{w}^T\mathbf{x}_i + w_{N+1}) \geq 1 - \xi_i$ What values does ξ_i take?



$\xi_i = 0$ for training pixels that are on, or on the correct side of, the marginal hyperplane — our usual decision rule

$\xi_i = 1$ for a pixel on the separating hyperplane — the decision surface — because $\mathbf{w}^T\mathbf{x}_i + w_{N+1} = 0$ and $|y_i| = 1$.

$\xi_i > 1$ 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.

$\xi_i = |y_i - (\mathbf{w}^T\mathbf{x}_i + w_{N+1})| < 1$ for all other training pixels.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —OVERLAPPING CLASSES

From the previous slide we note that ξ_i is zero for correctly located training pixels, whereas it is positive for those which are either in the wrong class or in the region between the marginal hyperplanes and the decision surface. Therefore their sum will be an indication of the total error incurred by the poorly located training pixes — in terms of where the boundaries have been placed.

What we want to do now is maximise the margin, as before, but also minimise the error caused by those poorly located pixels. We now have a decision to make. Do we want to give maximising the margin priority over minimising the training error caused by placing the decision surface so that some pixels are in error, or do we want to minimise the latter?

What we, in fact, do is set up a measure that allows us to strike a compromise between two objectives. Remember that maximising the margin is the same as minimising the norm of the weight vector $\|\mathbf{w}\|$. The following measure provides the compromise; the parameter C (called the regularization parameter) is a user-chosen weight that allows us to trade off minimizing the weight vector norm with the total error caused be poorly located training pixels.

$$\frac{1}{2}\|\mathbf{w}\| + C\sum_i \xi_i$$

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —OVERLAPPING CLASSES

$$\frac{1}{2}\|\mathbf{w}\| + C\sum_i \xi_i$$

We need to minimise this term subject to the constraints:

1. Since $y_i\,(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}) \geq 1 - \xi_i$ or $y_i\,(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}) - 1 + \xi_i \geq 0$

   we want to ensure that the argument $y_i\,(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_{N+1}) - 1 + \xi_i$ remains positive.

2. We must also ensure that all the ξ_i remains positive.

Thus the Lagrangian to be minimised is

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

this is to be minimised

one set of Lagrange multipliers

another set of Lagrange multipliers

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —OVERLAPPING CLASSES

We will not proceed with the theory any further at this stage, because it parallels what we did with the case where there were no overlapping classes; there are, however, more constraints to take into account here because of the additional Lagrange multipliers*. Once the Lagrangian has been optimised, including a numerical solution to find the Lagrange multipliers, the following results are obtained:

Again $\alpha_i = 0$ for many of the training pixels, meaning they will not contribute in the classification step (i.e. the step of labelling unknown pixels). Only those training pixels for which $\alpha_i \neq 0$ take part — again they are the support vectors.

The decision rule turns out to be the same as before (the μ_i drop out in the optimisation)

$\mathbf{x} \in$ class 1 if $z > 0$

$\mathbf{x} \in$ class 2 if $z < 0$

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

Note that the user has to find a value for C beforehand; that will influence the values of the Lagrange multipliers

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —THE CLASSIFICATION STEP

Consider our decision rule again, in which the two classes are defined by the sign being positive or negative:

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

We note that the central operation is a scalar or dot product of the form $\mathbf{w}^{\mathrm{T}}\mathbf{x}$ or $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}$

We now assume we can transform the set of pixel vectors so that they become linearly separable. We will see an example shortly, but for now just represent the transformation as $\phi(\mathbf{x})$ so that the scalar products above take the form:

$$\phi(\mathbf{x}_i)^{\mathrm{T}}\,\phi(\mathbf{x}) \rightarrow k(\mathbf{x}_i, \mathbf{x})$$

This is called a kernel function.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —USE OF KERNELS

Using the kernel function in the decision rule we have

$$z = sgn\{k(\mathbf{w},\mathbf{x}) + w_{N+1}\} = sgn\left\{\sum_{i\in S} \alpha_i\, y_i\, k(\mathbf{x}_i, \mathbf{x}) + w_{N+1}\right\}$$

The interesting thing about this expression is that we don't need to know the function ϕ provided we can choose an appropriate kernel k.

What functions can be used as kernels? Any that is decomposable (in principle) into a scalar product is suitable. We will not go into that detail, but the kernels shown on the next slide are commonly used.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —USE OF KERNELS

Common kernels

square of the scalar product	$k(\mathbf{x}_i, \mathbf{x}) = (\mathbf{x}_i^{\mathrm{T}}\mathbf{x})^2$	of limited value
polynomial	$k(\mathbf{x}_i, \mathbf{x}) = (\mathbf{x}_i^{\mathrm{T}}\mathbf{x} + b)^m$	
Gaussian radial basis function	$k(\mathbf{x}_i, \mathbf{x}) = \exp\{-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2\}$	the most popular
sigmoidal	$k(\mathbf{x}_i, \mathbf{x}) = \tanh(\kappa\mathbf{x}_i^{\mathrm{T}}\mathbf{x} + b)$	

Note that the last three kernals have parameters (b, m, γ, κ) values for which need to be determined.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —USE OF KERNELS

We now consider a very simple example of how kernels work, using a basic quadratic function.

$$k(\mathbf{y}, \mathbf{x}) = (\mathbf{y}^T\mathbf{x})^2$$

So we can see what happens, restrict the data space to two dimensions, thus $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

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

which can be written as $k(\mathbf{y}, \mathbf{x}) = [x_1^2 \quad \sqrt{2}\, x_1 x_2 \quad x_2^2] \begin{bmatrix} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{bmatrix}^T \begin{bmatrix} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{bmatrix}$

scalar product

Thus $(\mathbf{y}^T\mathbf{x})^2$ can be expressed as a scalar product and is an acceptable kernel.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —USE OF KERNELS

Since $k(\mathbf{y},\mathbf{x}) = (\mathbf{y}^{\mathrm{T}}\mathbf{x})^2 = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix}$ we can see that the associated transformation is

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

It is a three dimensional space defined in terms of the squares and cross products of the original dimensions.

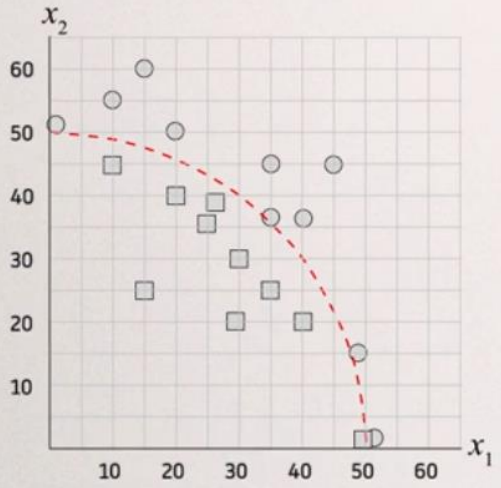
Call the new coordinates $\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$ and apply this to a simple example . . .

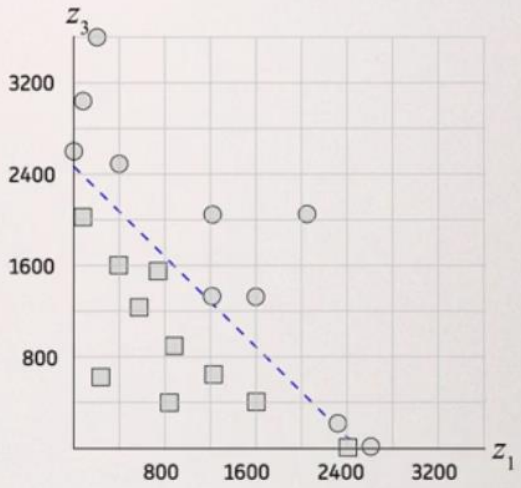
SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE —USE OF KERNELS

Applying this transformation $\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$ to the data set below yields the results shown

Original data is not linearly separable. The classes lie either side of the quadrant of a circle.





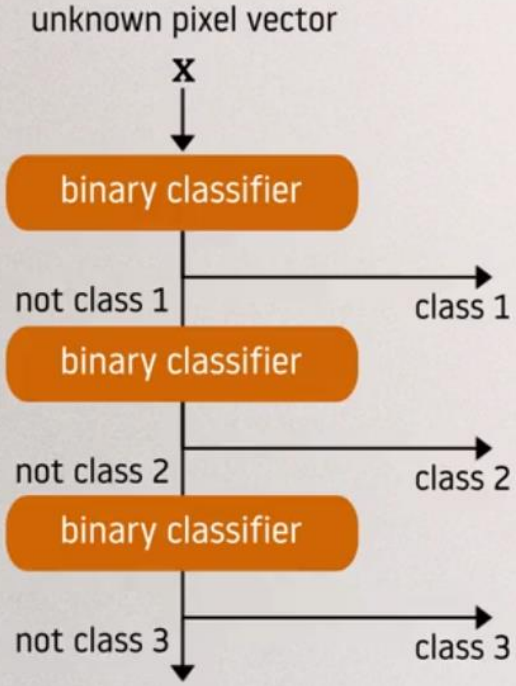
The transformed data is linearly separable. The classes now lie either side of a straight line.

Although a third dimension has been created by the transformation, the second z_2 axis is not used in this example. Linear separability has been achieved in a two-dimensional sub-space.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE — HANDLING MULTIPLE CLASSES

We now come to the final step in using the support vector machine — turning the binary classifier into one that will handle many classes. Several approaches have been adopted. The simplest is the decision tree:



Each SVM classifier is trained to separate one class from the rest, sequentially.

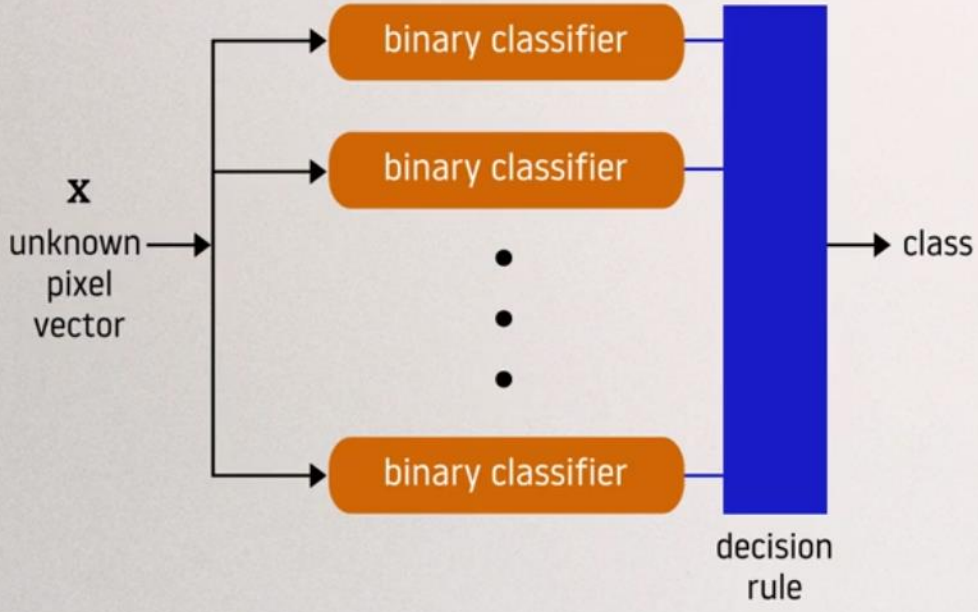
A problem is that the training sets are unbalanced, in that there are many fewer pixels representative of the class being separated than in the aggregated set of all the remaining training pixels.

Also, we don't know the optimal order in which to separate each class.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE — HANDLING MULTIPLE CLASSES

An approach often used in practice is the *one-against-the-rest or one-against-all* (OAA) strategy.



Each classifier is, again, trained to separate one class form the rest, not sequentially as before, but in parallel.

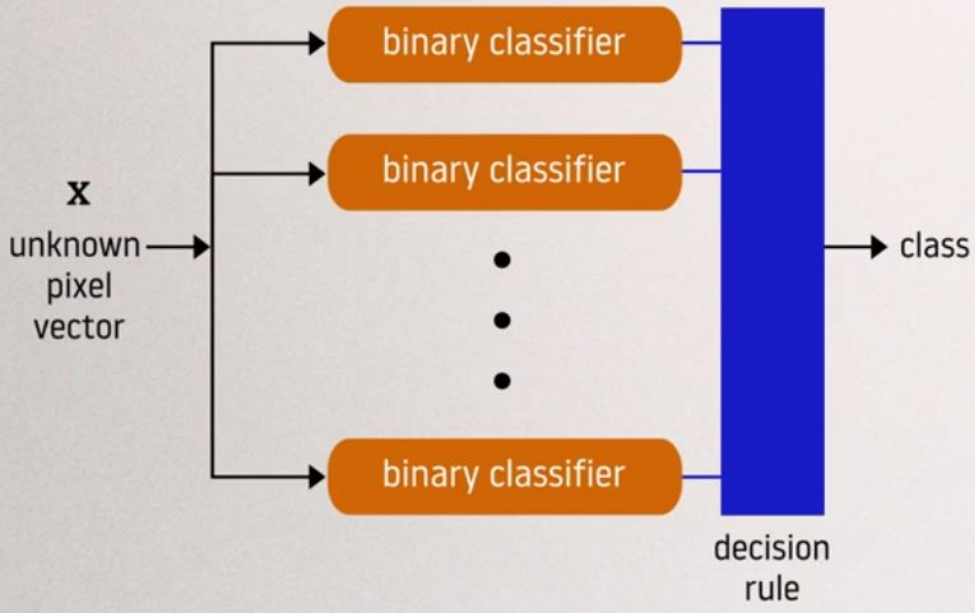
There is one classifier for each of the M classes of pixel.

For an unknown pixel a decision is taken over all the classifier outputs to find the most favoured label. That can be done by choosing the class (classifier) which is associated with the largest value of $\sum_{i\in\mathcal{S}} \alpha_i y_i \mathbf{x}_i^{\mathrm{T}}\mathbf{x} + w_{N+1}$, which is the argument of the binary classifier decision rule we saw previously with the SVM.

SUPPORT VECTOR MACHINE

THE SUPPORT VECTOR MACHINE — HANDLING MULTIPLE CLASSES

Yet another approach is the *one-against-one* (OAO) strategy, which uses the same parallel arrangement.



A set of classifiers is trained, in which each classifier is designed to separate just two classes. For M classes there will, therefore, be $M(M\text{-}1)/2$ separate classifiers.

Each class will appear explicitly $M\text{-}1$ times among the binary classifiers, but all classifiers will respond to all unknown pixels.

An unknown pixel is placed into the class which has the greatest number of recommendations in favour of it among the $M(M\text{-}1)/2$ decisions.

This works well but training can take a long time because of the need to develop $M(M\text{-}1)/2$ classifiers; this will be a large number even for a practical number of classes.

SUPPORT VECTOR MACHINE

We now have all the tools in place for using the SVM.

- We know how a maximum margin decision rule can be developed on linearly separable data.
- We know how to extend that to the case where there are overlapping classes.
- We have seen how kernels can be used to allow data which is not linearly separable to be handled.
- We have seen that there are methods to undertake multi-category classification based on networks of binary (SVM) classifiers.

SUPPORT VECTOR MACHINE

Before we apply the SVM to a real problem there are several initial steps we have to take:

- We need to choose which kernel function to use — the Gaussian radial basis function is most common.
- We have to choose values for any parameters in the kernel, in this case γ.
- We have to find a value for the regularisation parameter C.

The last two steps are important because the values chosen for those parameters have to be optimal in terms of reducing classification error — i.e. achieving best performance.

- And, we have to choose our multi-class strategy — OAO is often used.

These steps are now arranged in the sequence followed in practice, using our previous tabular form . . .

SUPPORT VECTOR MACHINE

Steps to be followed in undertaking thematic mapping in remote sensing
Support vector classifier
Choose the kernel and multiclass strategy to use.
Gather labelled training data and labelled testing data.
Use training data and grid search procedures to find the kernel and regularization parameters, and the support vectors.
Apply the decision rule based on the support vectors to each pixel in the image to obtain the most likely class label for that pixel.
Produce a thematic map and a table of area estimates.
Use labelled testing data to evaluate the accuracy of the thematic map. If not acceptable refine the classification by examining which classes are most in error and seeing whether additional spectral classes are needed or whether the parameter estimates need refining.

SUPPORT VECTOR MACHINE

Finally, we need access to SVM software.

- It is possible to write your own software, based on material presented in these lectures. There are, however, a number of public domain and commercial routines available.
- One of the most popular is LibSVM available from the National Taiwan University. See https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Matlab also includes an SVM toolbox, as does ENVI. See https://www.harrisgeospatial.com/docs/SupportVectorMachine.html

SUPPORT VECTOR MACHINE

The image is a segment of a **Quickbird 2** image taken over the city of Boumerdès in Algeria, located on the coast of the Mediterranean sea. It was acquired on 22 April 2002 and consists of 500x600 pixels, with spatial resolution of 0.6m, achieved through pan-sharpening the 2.4m multispectral bands. The Quickbird 2 sensor has four bands:

blue 450-520nm

green 520-600nm

red 630-690nm

NIR 760-890nm

SUPPORT VECTOR MACHINE

There are several cover types in the image, the most notable being:

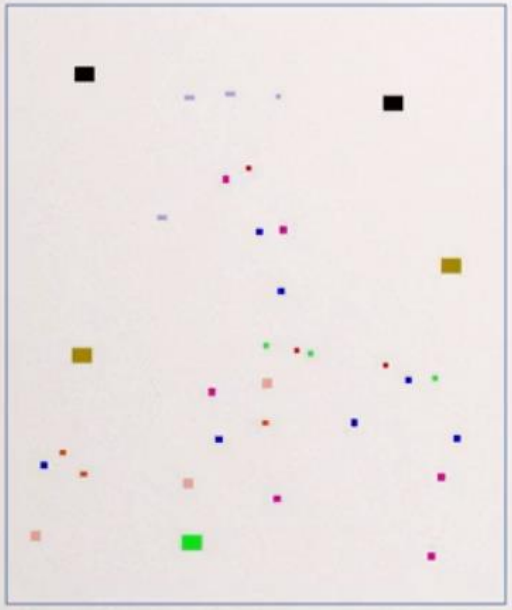
- water
- sand
- tree
- asphalt
  - traffic streets—called asphalt 1
  - pavements—called asphalt 2
- rock
- roof
  - tiles—called roof 1
  - cement—called roof 2
- bare soil

These are the classes in which we are interested. We call them information classes.

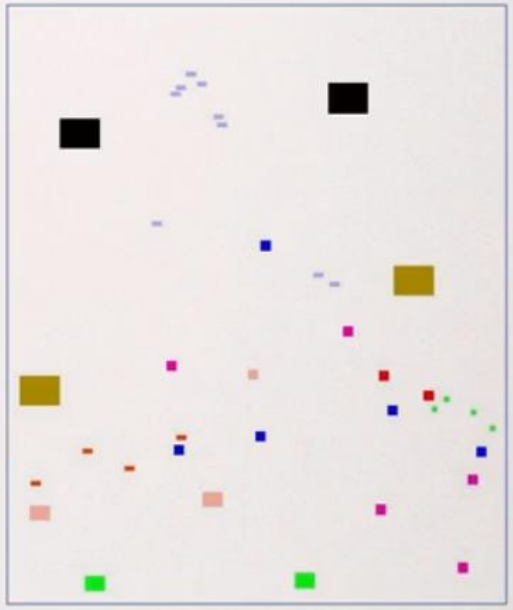
Two of them have two sub-classes each. We call these spectral classes. Often, to get good class separation we may have to use several spectral classes per information class.

SUPPORT VECTOR MACHINE

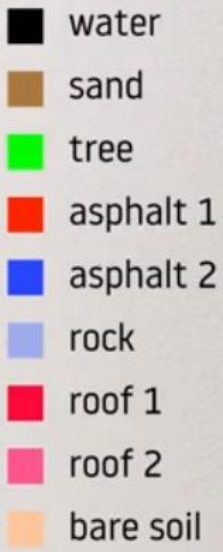
We now need to choose labelled pixels for training and testing. The maps below show those used in this exercise.



training pixels



testing pixels



SUPPORT VECTOR MACHINE

Some important data:

	number of samples	
	training	testing
water	600	2400
sand	600	2400
tree	375	700
asphalt 1	105	200
asphalt 2	343	500
rock	175	450
roof 1	75	200
roof 2	294	500
bare soil	300	700
	2867	**8050**

SUPPORT VECTOR MACHINE

The next step is to design the classifier.

We have to choose (1) the kernel function to use, and then determine the "optimal" value of its parameters

(2) the multiclass strategy to use

(3) the value of the regularization parameter C.

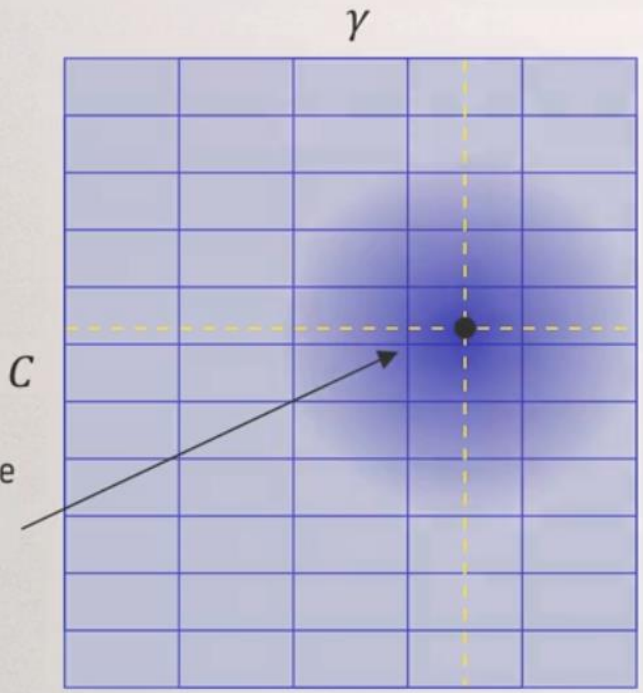
This exercise was carried out using the Gaussian radial basis function kernel $k(\mathbf{x}_i, \mathbf{x}) = \exp\{-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2\}$ which has just a single parameter γ.

We show how a value can be found for this on the next slide, in conjuction with finding a value C.

The one against one (OAO) multiclass strategy was used, resulting in 36 separate classifiers, since there are 9 spectral classes.

SUPPORT VECTOR MACHINE

Parameter determination is usually carried out by grid searching. Ranges of values of C and γ are searched to find the combination which minimizes the training error (shown here in blue shading).



Trying to find the grid intersection closest to the minimum error.

In this exercise, based on the experience of the authors, the following slightly simpler procedure was adopted:

γ was chosen initially as 0.25.

C was then varied from 25 to 200 in steps of 25.

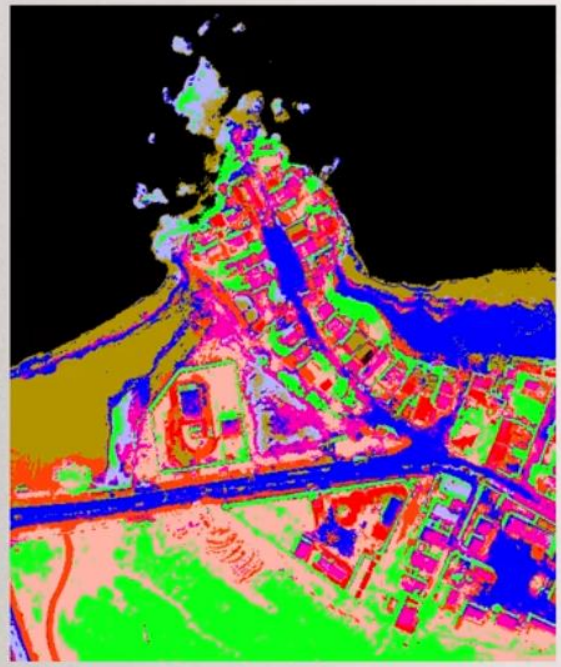
Using the best value of C from the previous step, γ was then varied from 0.25 to 2 in steps of 0.25.

Although not as comprehensive as a full grid search, this simpler approach nevertheless finds values for the parameters that work well enough in this case.

Note that the same parameter values were used for all 36 binary classifiers. They were $C = 200$ and $\gamma = 2$.

SUPPORT VECTOR MACHINE

The results



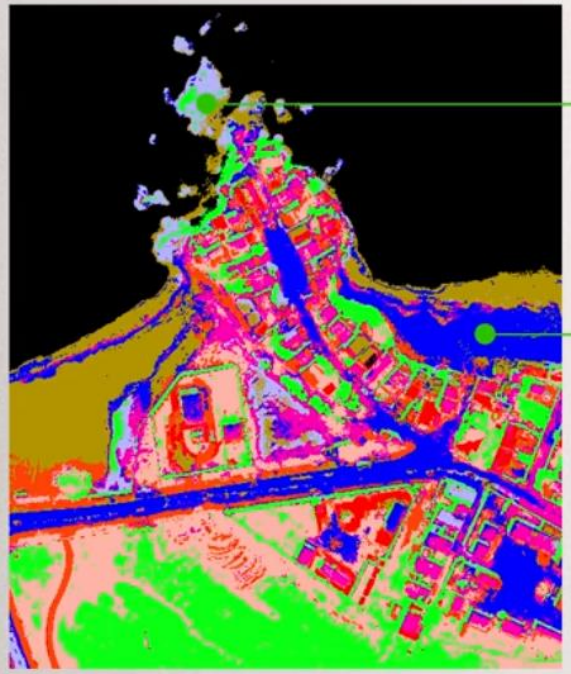
6192 out of the 8050 testing pixels were correctly classified, representing an overall accuracy of 76.9%. The table below shows the performance by class.

water	100%
sand	65.7%
tree	95.6%
asphalt 1	63.5%
asphalt 2	85.4%
rock	44.0%
roof 1	62.5%
roof 2	72.0%
bare soil	44.1%

Accuracy was assessed by a process called 2-fold cross validation, which we will treat when considering classification accuracy in detail in module 3 of the course.

SUPPORT VECTOR MACHINE

The results



thematic map

Note classification errors

Some pixels of rock have been classified as tree

Many pixels of bare soil have been classified as asphalt

We need more sophisticated error measures in cases like this. Again, we will meet them in Module 3.

SUPPORT VECTOR MACHINE

The previous simple example does not demonstrate one of the key benefits of the support vector machine — the ability to handle image data of high dimensionality, such as hyperspectral imagery.

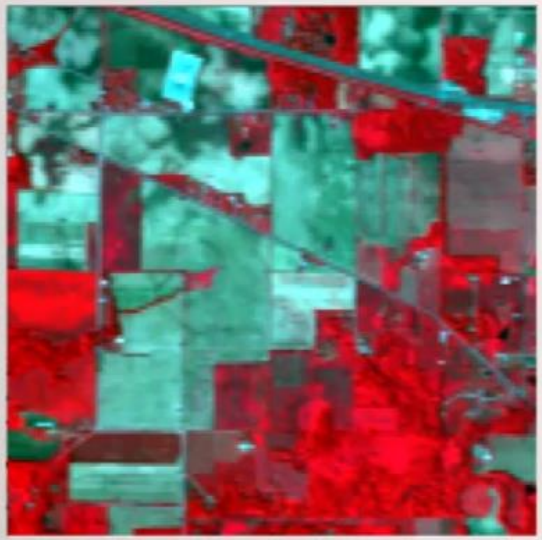
Consider now an exercise involving 200 spectral bands. This is drawn from F. Melgani and L. Bruzonne, Classification of hyperspectral images with support vector machines, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, No. 2, August 2004, pp. 1778-1790.

The data set, called Indian Pines*, was recorded by AVIRIS (Airborne Visible and Infrared Sensor) in 1992. AVIRIS records 224 bands over the spectral range 0.4-2.5μm. At the time of this experiment it recorded 220 bands with a 10 bit radiometric resolution.

* Baumgardner, M. F., Biehl, L. L., Landgrebe, D. A. (2015). 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3. Purdue University Research Repository. doi:10.4231/R7RX991C

SUPPORT VECTOR MACHINE

The Indian Pines Data Set:



20 bands discarded because of atmospheric problems with the data

red — channel 50
green — channel 27
blue — channel 17

7 of 16 classes discarded because not enough training samples are available

SUPPORT VECTOR MACHINE

Experiment design

One against all (OAA) multiclass strategy

Gaussian radial basis function kernel

Through searching, chose $C = 40$ and $\gamma = 0.25$

	number of samples	
	training	testing
corn-no till	742	692
corn min-till	442	392
grass/pasture	260	237
grass/trees	389	358
hay-windrowed	236	253
soybean-no till	487	481
soybean-min till	1245	1223
soybean-clean till	305	309
woods	651	643
	4757	**4588**

SUPPORT VECTOR MACHINE

Results

corn-no till	91.5%
corn min-till	87.9%
grass/pasture	94.9%
grass/trees	98.9%
hay-windrowed	100.0%
soybean-no till	88.6%
soybean-min till	91.3%
soybean-clean till	95.8%
woods	99.4%
overall accuracy	**93.4%**

This is a remarkably good result, especially with data of such high dimensionality.

Sensitivity to values of C and γ

To check the importance of having precise values for the regulation and kernel parameters, the authors carried out the following sensitivity tests, in which one was held constant while the other was varied over the range shown. They then computed the average performance over the tests, as shown below. The fact that the variance is small, indicates that the parameters do not have to be determined with high preccision in these ranges in order still to get good results. Note though that the original (grid) searching operation is still needed to get the nominal values for C and γ.

range		mean accuracy	variance in accuracy
$C = 1\text{-}100$	$\gamma = \text{-}1$	92.6%	0.84%
$C = 40$	$\gamma = 0.1\text{ -}3$	92.5%	0.50%

UNSUPERVISED CLASSIFICATION

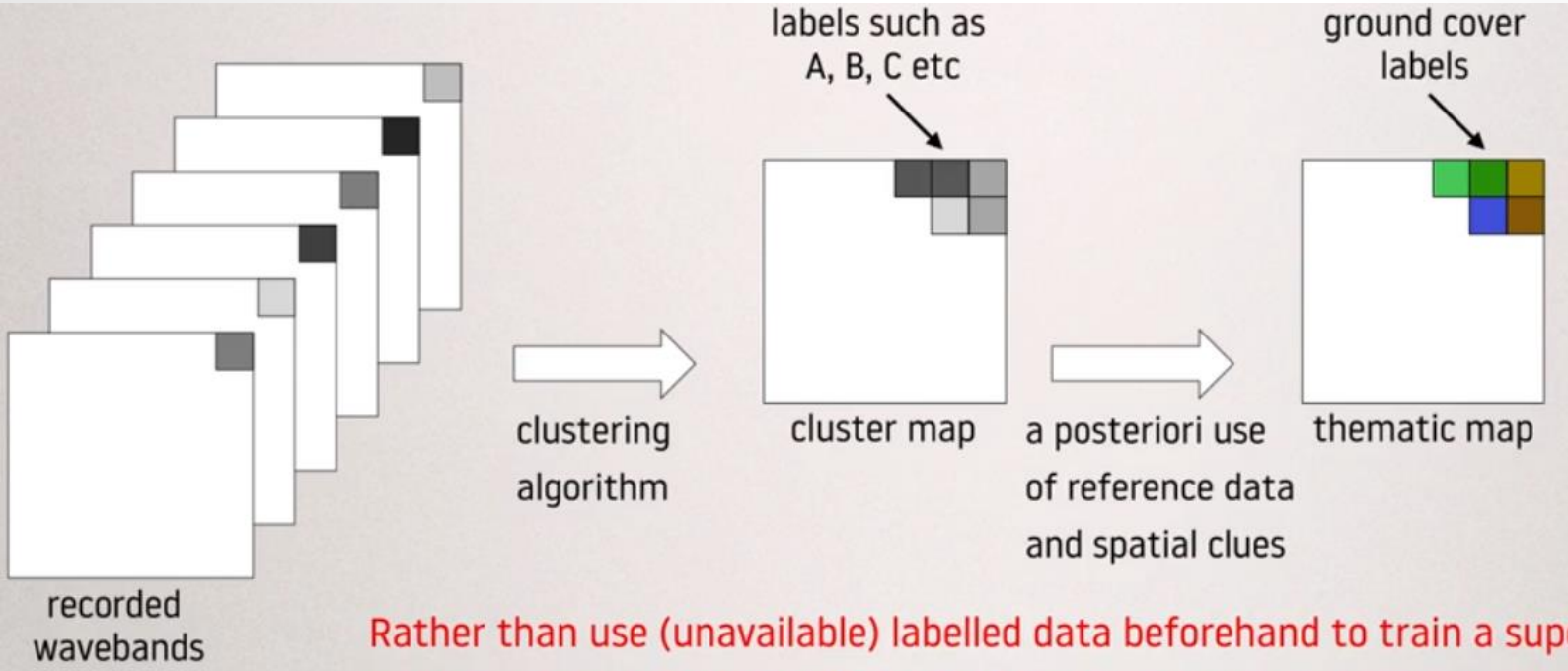
Sometimes labelled training data is unavailable, and yet we might still wish to undertake some form of thematic mapping. We can do that using the process called clustering, which can produce a map of unspecified labels.

As the name implies clustering looks for groups of similar pixels. Similarity is usually assessed on the basis of the spectral properties of pixels, so the groups we search for are clusters in spectral space.

We can often identify the pixel labels produced by clustering through the use of spatial clues in the image itself, and by using the cluster means as surrogates for spectral reflectance information. Also, forms of reference data like maps and air photos give us hints as to what the cluster classes in the thematic map might represent.

UNSUPERVISED CLASSIFICATION

UNSUPERVISED CLASSIFICATION THROUGH CLUSTERING



Rather than use (unavailable) labelled data beforehand to train a supervised algorithm such as the CNN, SVM or MLC classifiers, here we use reference data afterwards in an effort to add meaningful ground cover labels to the labels found by clustering. Once a clustering algorithm has placed a pixel into a particular cluster it can be labelled that way on a cluster map. We need to turn that cluster map into a thematic map

UNSUPERVISED CLASSIFICATION

CLUSTERING: SIMILARITY METRICS

Clustering algorithms place pixels into groups (or clusters) based on their similarity. The most common measure of similarity is based on spectral measurements. Two pixels with very similar measurement vectors are likely to belong to the same class and thus cluster.

In order to quantify similarity in spectral space we need a similarity measure. The most common metrics are based on measuring the spectral distances between pixels.

One logical distance metric is the Euclidean distance between pixels, defined by:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|$$

$$= \{(\mathbf{x}_1 - \mathbf{x}_2)^{\mathrm{T}}(\mathbf{x}_1 - \mathbf{x}_2)\}^{1/2}$$

$$= \left\{\sum_{n=1}^{N}(x_{1n} - x_{2n})^2\right\}^{1/2}$$

N is the number of bands

UNSUPERVISED CLASSIFICATION

CLUSTERING: SIMILARITY METRICS

Another common similarity metric is the city-block distance, defined by:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sum_{n=1}^{N} |x_{1n} - x_{2n}|$$

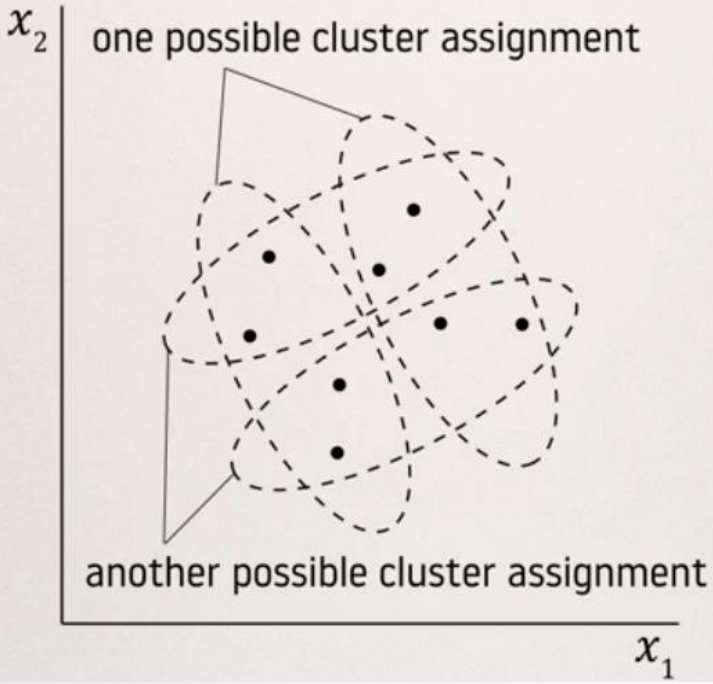
This is just the accumulated difference along each spectral dimension, similar to walking between two locations in a city laid out on a rectangular grid.

There are other distance metrics too, but they are used less frequently than the ones presented here. Furthermore, some clustering algorithms measure similarity not just in terms of spectral distance but also taking into account spatial proximity, similar to our observation of spatial context in previous lectures.

UNSUPERVISED CLASSIFICATION

CLUSTERING: NON-UNIQUENESS

Using the concept of spectral similarity it should be possible to group pixels in a data set. However, a unique grouping of pixels may not always be possible, as seen in the following diagram. Also, how do we know which one is the better clustering?



Two possible clusterings of eight pixel vectors in a two dimensional spectral space

UNSUPERVISED CLASSIFICATION

CLUSTERING: NON-UNIQUENESS

Clearly, we need some way of evaluating whether one cluster assignment is better than another. To do that we need a "quality of clustering" criterion, a common one of which is the sum of squared error measure (SSE).

This checks how far away all the pixels in a given cluster are from the cluster centroid—i.e. the mean—and then sums those distances within the cluster. It does so for all clusters and sums the results.

A good cluster assignment is one that leads to compact clusters and thus to a small SSE.

$$SSE = \sum_{C_i} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = \sum_{C_i} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)^{\mathrm{T}} (\mathbf{x} - \mathbf{m}_i)$$

In which $\mathbf{m}_i$ is the mean vector of the i^{th} cluster C_i and $\mathbf{x} \in C_i$ is a pixel to that cluster.

UNSUPERVISED CLASSIFICATION

CLUSTERING: FINDING AN ACCEPTABLE ALGORITHM

In principle we should be able to develop a clustering algorithm that minimizes the SSE for a given data set. But that turns out to be impractical since it would require an enormous number of candidate clusterings of the available data to find that with the smallest SSE. Instead, some heuristic methods have been developed that work well, two of which we will develop here.

The *k* means (or migrating means) algorithm

The *k* means approach to clustering requires the user to specify beforehand how many clusters to search for, and to specify a set of initial cluster mean vectors. The image pixels are then assigned to the cluster of the closet mean, after which the set of means is re-computed. The pixels are then assigned to the nearest of the new set of means, and so on until the means and assignments do not change. Algorithmically this can be expressed as on the next slide. In the next lecture we will see how the algorithm works.

UNSUPERVISED CLASSIFICATION

CLUSTERING: THE *k* MEANS (OR MIGRATING MEANS) ALGORITHM

1. Select a value for C, the number of clusters into which the pixels are to be grouped.
2. Initialise cluster generation by selecting C points in spectral space to serve as candidate cluster centres.

$$\hat{\mathbf{m}}_c \; c = 1 \ldots C$$

3. Assign each pixel vector $\mathbf{x}$ to the candidate cluster of the nearest mean using an appropriate distance metric. That generates a cluster of pixel vectors about each candidate mean.
4. Compute a new set of cluster means from the groups formed in Step 3.

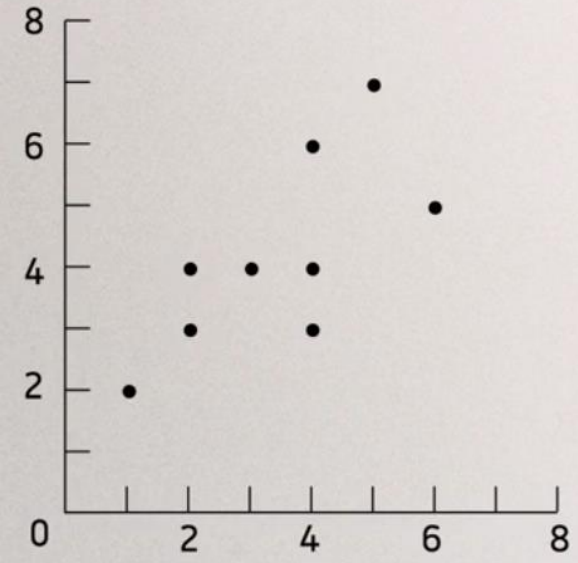
$$\mathbf{m}_c \; c = 1 \ldots C$$

5. If $\mathbf{m}_c = \hat{\mathbf{m}}_c$ for all c then the procedure is complete. Otherwise put $\hat{\mathbf{m}}_c = \mathbf{m}_c$ and return to step 3.

UNSUPERVISED CLASSIFICATION

OPERATION OF THE *k* MEANS CLUSTERING ALGORITHM

We now demonstrate the *k* means algorithm using the two dimensional data set shown here.



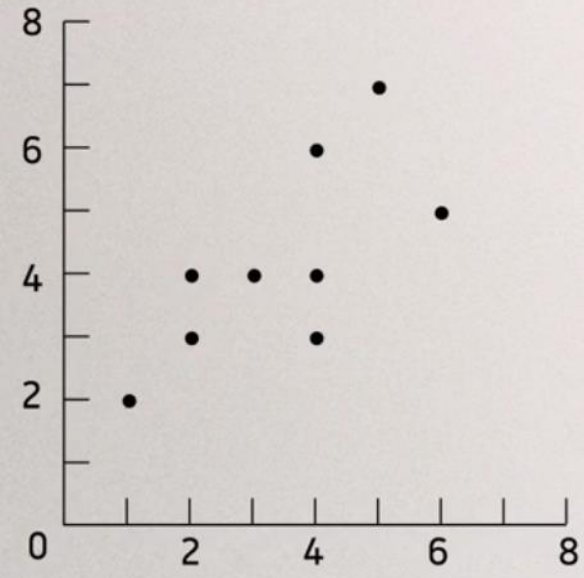
Two decisions have to made before using the algorithm. The first is the number of clusters to be found and the second is where to place the cluster centres initially.

Examining this data set it seems there are two or possibly three clusters. In practice that might not be obvious, so a guideline is needed. Because we can merge clusters later, it is good to estimate on the high side, recognizing however that the more clusters there are the longer the algorithm is likely to take to converge. A guideline in remote sensing is to estimate the number of information (ground cover) classes in a scene and then search for 2 to 3 times that number of clusters.

UNSUPERVISED CLASSIFICATION

OPERATION OF THE *k* MEANS CLUSTERING ALGORITHM

There are several ways to set the initial positions of the clusters:



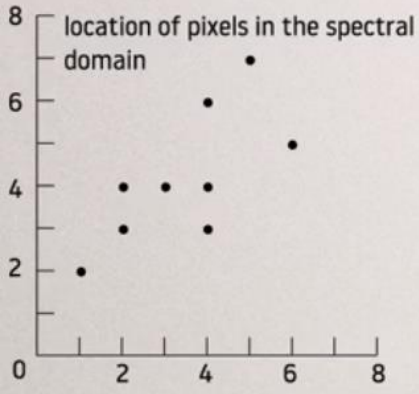
They can be spaced uniformly along the multi-dimensional diagonal of the spectral space. That is a line from the origin to the point of maximum brightness on each axis.

A refinement of that guideline is to choose the multi-dimensional diagonal that joins the actual spectral extremities of the data.

Another approach is to space the initial cluster centres uniformly along the first principal component of the data.

UNSUPERVISED CLASSIFICATION

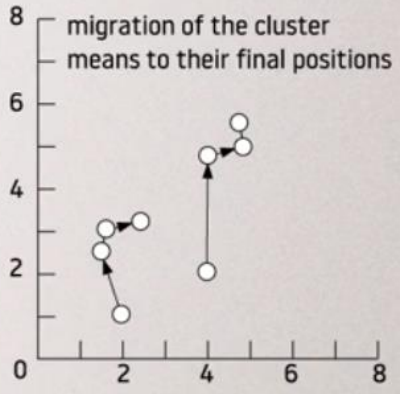
OPERATION OF THE *k* MEANS CLUSTERING ALGORITHM

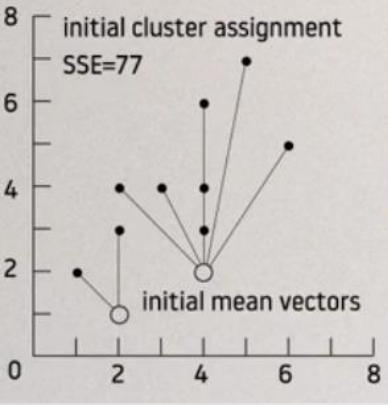


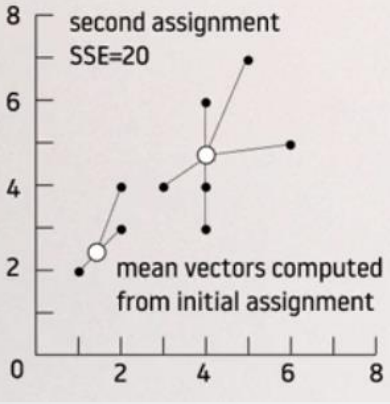
The bottom row shows the operation of the k means algorithm and the evolution of the clusters. Note that there is a progressive reduction in SSE. The diagram to the right shows the migration of the means, by iteration.

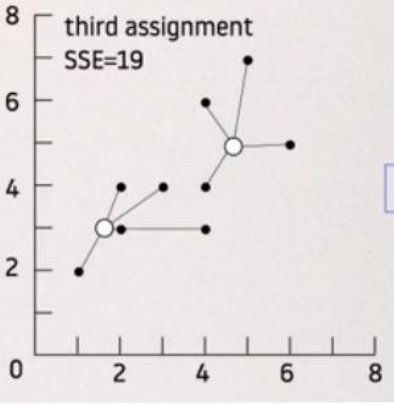
We are looking for two clusters.

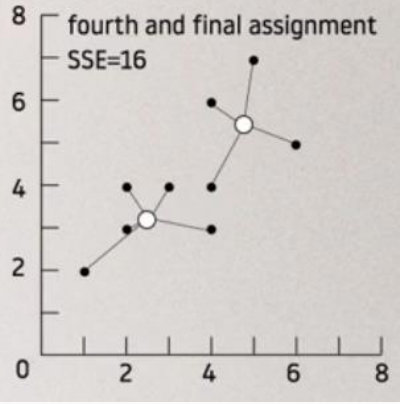
The method also goes by the name of iterative optimization.











UNSUPERVISED CLASSIFICATION

THE ISODATA ALGORITHM

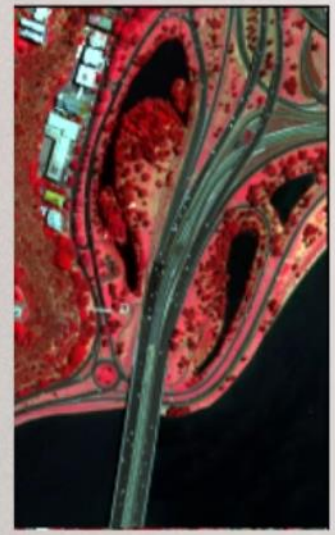
The Isodata algorithm builds on the k means approach by introducing a number of checks on the clusters found either at the end of or at intervals during the process. These checks, and subsequent actions, include:

- Seeing whether any clusters contain so few points as to be meaningless
  If the statistics of clusters are important, say for use in a later maximum likelihood classification, then poor estimates will be obtained if the clusters do not contain a sufficient number of members.
- Seeing whether any pairs of clusters are so close that they should be merged
  In module 3 we will look at similarity measures in the context of classification. They will give an indication of whether classes (and clusters) are too similar spectrally as to be useful.
- Seeing whether some clusters are unreasonably elongated in certain spectral dimensions that it would be sensible to split them
  Elongated clusters are not necessarily a problem, but if they are, then comparison of the standard deviations of the clusters along each spectral dimension will help reveal their elongated nature (at least in the absence of strongly correlated data)

UNSUPERVISED CLASSIFICATION

AN EXAMPLE OF UNSUPERVISED CLASSIFICATION USING *k* MEANS CLUSTERING

We now consider an application of clustering to unsupervised classification, using a five channel data set of an image recorded by the HyMap sensor near the city of Perth in Western Australia in January 2010.



red - channel 29
green - channel 15
blue - channel 7

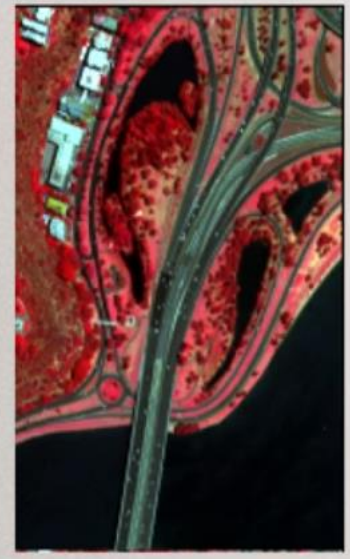
HypMap channels used for supervised classification, by clustering

channel	band centre (nm)	band width (nm)
7 (visible green)	511.3	17.6
15 (visible red)	634.0	16.4
29 (near infrared)	846.7	16.3
80 (middle infrared)	1616.9	14.8
108 (middle infrared)	2152.7	30.2

UNSUPERVISED CLASSIFICATION

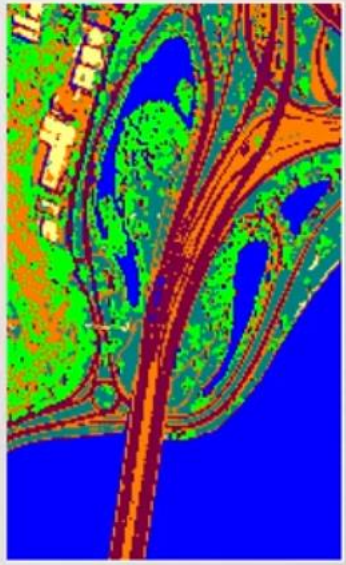
AN EXAMPLE OF UNSUPERVISED CLASSIFICATION USING *k* MEANS CLUSTERING

The algorithm was used to find 6 clusters, since inspection of the scene indicated that there were about 6 distinctive cover types, based on the colours shown in the image. The cluster map generated is shown below.



red - channel 29
green - channel 15
blue - channel 7

We can see that the clusters, represented by different colours, seem to follow the visual patterns of the classes in the image. In particular it is easy to associate the brown and orange clusters with highways, road pavements and bare regions, the yellows with buildings, and the shades of green with various types of vegetation. Clearly the dark blue cluster is water.



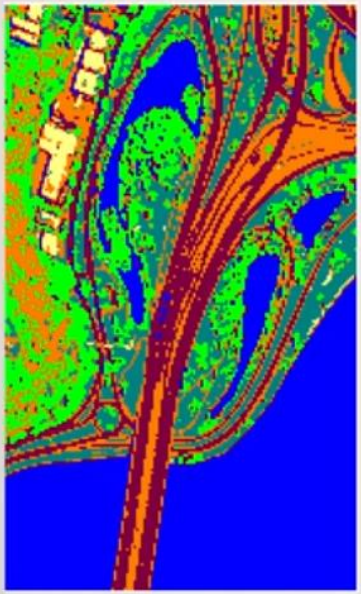
We can add supporting evidence to this interpretation of the colours if we examine the cluster means ...

UNSUPERVISED CLASSIFICATION

AN EXAMPLE OF UNSUPERVISED CLASSIFICATION USING *k* MEANS CLUSTERING

Classes

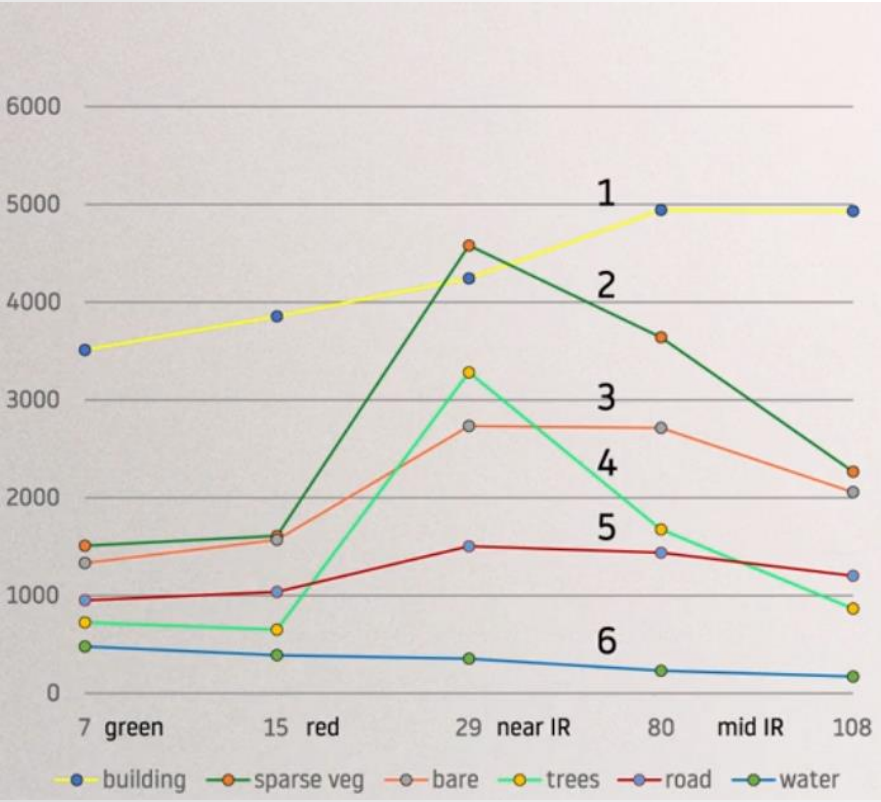
- background
- building
- sparse vegetation
- bare
- trees
- road
- water



		cluster mean vectors (on 16 bit scale)				
cluster	label	channel 7	channel 15	channel 29	channel 80	channel 108
1	building	3511.9	3855.7	4243.7	4944.2	4931.6
2	sparse veg	1509.6	1609.3	4579.5	3641.7	2267.0
3	bare	1333.9	1570.7	2734.3	2715.1	2058.7
4	trees	725.6	650.6	3282.4	1676.2	866.6
5	road	952.3	1037.1	1503.7	1438.5	1202.3
6	water	479.2	391.1	354.8	231.0	171.6

UNSUPERVISED CLASSIFICATION

AN EXAMPLE OF UNSUPERVISED CLASSIFICATION USING *k* MEANS CLUSTERING



Plots of cluster means follow the spectral reflectance curves of the ground cover class labels assigned to the cluster. This is further information that has been used to identify the clusters.

		cluster mean vectors (on 16 bit scale)				
cluster	label	channel 7	channel 15	channel 29	channel 80	channel 108
1	building	3511.9	3855.7	4243.7	4944.2	4931.6
2	sparse veg	1509.6	1609.3	4579.5	3641.7	2267.0
3	bare	1333.9	1570.7	2734.3	2715.1	2058.7
4	trees	725.6	650.6	3282.4	1676.2	866.6
5	road	952.3	1037.1	1503.7	1438.5	1202.3
6	water	479.2	391.1	354.8	231.0	171.6

UNSUPERVISED CLASSIFICATION

AN EXAMPLE OF UNSUPERVISED CLASSIFICATION USING *k* MEANS CLUSTERING

Here we show a NIR versus red scatterplot of the original data along with a bi-spectral plot of the cluster centres. This is also helpful in identifying the clusters. Note that the four classes of water, road, bare and building lie almost in a straight line from low to high brightness in this pair of bands, while vegetation lies to the top left. Such behavior is well-known by remote sensing practitioners.