NEURAL NETWORKS & PATTERN RECOGNITION

An Introduction

Stephen Roberts

LECTURE 1,2: INTRODUCTORY MATERIAL

Books

There are a number of good books on pattern recognition. I would, however, suggest a mixture of the following:

1. Fukunaga, *Introduction to Statistical Pattern Recognition*.
2. Ripley, *Pattern Recognition and Neural Networks*.
3. Bishop, *Neural Networks for Pattern Recognition*.
4. Schalkoff, *Pattern Recognition: Statistic, Structural and Neural Approaches*.
5. Kittler, *Pattern Recognition Theory and Practices*.
6. Devijver, *Pattern Recognition: A Statistical Approach*.

These will cover almost all the work on this course in combination, and for most of them in isolation.

Background

We, as human beings, can perform a large number of pattern recognition tasks with ease; consider listening to a conversation in a crowded room:

- How do we separate a single source (speaker) from all the others?
- How do we process the words?

Consider further recognising a face in a crowded scene:

- How do we identify faces?
- How do we identify a particular face?

In a medical context:

- How is a tumour recognised in an MRI image?
- How is a prognosis arrived at from patient information?

The last few of these tasks can tax even the best clinicians; can machines ever hope to compete with humans?

Historical Perspective

- Symbols of scenes (neolithic)
- Mapping
- Occam's razor
- Perspective respresentations in art
- Calculus of probabilities (Thomas Bayes)
- Axiomatic foundations of probability theory
- Digital imaging for space program
- Machine learning / AI
- Structural and syntactic approaches
- Robotics / machine vision
- Neurally-inspired methods
- Applied statistical theory

Current trends in computers have a wide-scale effect on pattern analysis

- Speed
- Parallelism
- Novel architectures

Classification and Regression

We will treat both techniques as *mapping* problems, such that some input variable, $\mathbf{x}$, generates (is mapped to) some output variable $\mathbf{y}$ via a mapping function $\mathcal{F}$, i.e.

$$\mathcal{F} : \mathbf{x} \mapsto \mathbf{y}$$

Any problem may be reduced then to finding an optimal mapping $\mathcal{F}$. In the case of *supervised* methods this mapping is 'learned' or optimised through a *training* data set which consists of input-output pairs $(\mathbf{x}, \mathbf{t})$ where each $\mathbf{t}$ represents a *target* or desired output for the corresponding input $\mathbf{x}$. It should be noted that $\mathcal{F}$ is *conditional* on the training data set and if the latter is poor then we cannot expect the mapping to be anything other than poor also.

Classification

A classification problem is one in which we desire to assign each input to one of a closed set of output *classes*. Our data set will therefore contain targets, $\mathbf{t}$, which will code the class label of each input. The usual format for each target is 'one-of-n' coding such that, for example in a two-class problem, $\mathbf{t} = (1, 0)$ for class 1 and $\mathbf{t} = (0, 1)$ for class 2.

As we will see, the *minimum risk* statistical classifier ascribes an unknown input to the class with the highest Bayes' *a posteriori* probability. Bayes' theorem, a belief update theorem, links a prior belief (the *a priori* probability) to the *a posteriori* belief given a new piece of information.

Regression

In the case of regression (or prediction) each input is mapped, not to a probability space, but onto another continuous number sequence. Again, a mapping function may be 'learned' using a set consisting of input-output pairs, $(\mathbf{x}, \mathbf{t})$ where $\mathbf{t}$ may, for example in the case of prediction, be the value of a data series some number of samples in the future (we need off-line data, of course, to construct such a training data set). Given a set of examples, then, what is the 'best' a system can do? It turns out that, given a training set, our best guess for the output in response to an input $\mathbf{x}$ is the *conditional average* of $\mathbf{t}$ evaluated over the training set. We denote this as

$$\mathbf{y}^{best}(\mathbf{x}) = \langle \mathbf{t} \mid \mathbf{x} \rangle$$

and note that this expression codes a common-sense interpretation; if we are given an input, $\mathbf{x}$, we look at what the target responses were for inputs close to $\mathbf{x}$ in the training set and our 'best' output response is the average of all these values. There is one word of caution, however. Refer to Figure 1; plot (a) shows a target distribution in which the conditional average gives a correct result as the distribution is unimodal given any arbitrary input, $\mathbf{x}_o$ say. Plot (b), however, has a multimodal distribution. In this case the conditional average gives a poor representation. To tackle this form of data a much more sophisticated methodology is required.

Supervised and unsupervised

Given a space in which to perform a classification, unsupervised classification relies on the fact that *data belonging to the same object tend to have similar feature vectors* and *vice versa*. This means that segmentation of the data relies on finding the 'blobs' or 'clumps' which dominate feature space (we hope). Supervised partitioning relies on a set of *training* data from which we can estimate a mapping function, $\mathcal{F}$, which

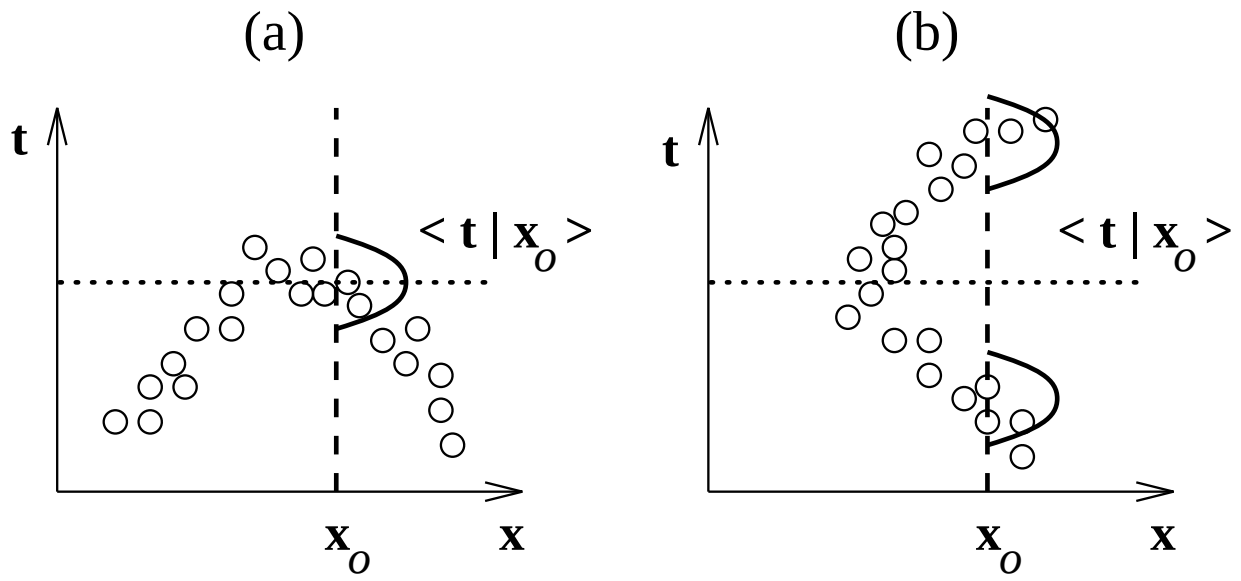

Figure 1: (a) Unimodal distribution - correct results obtained for regression. (b) Multimodal distribution – the basic regression problem becomes intractable and more sophisticated techniques are required.

maps a vector in the feature space to a *classification* space i.e.

$$\mathcal{F} : \mathrm{f} \mapsto \mathbf{C}$$

'hard' segmentation into class r, say, of the data set is then achieved by allocating all data whose feature vectors were mapped to C_r. Both unsupervised and supervised segmentations have problems, the former assumes that each class only has one 'blob' in the feature space and the latter is only as good as the training set (often labouriously created by hand). Both of these methods are intimately related to probability mappings and to *Bayes' theorem* which we look at later in the lectures. The supervised case is clear :

> *If we have any system which we wish to partition into N components, then each component of the system, $\mathbf{x}$, has membership of each of the N classes given by its posterior probability on that class.*

$$\mathbf{x} \mapsto \text{ Class } r \text{ iff } P(r \mid \mathbf{x}) = \max_c \{P(c \mid \mathbf{x})\}$$

If we can estimate the posteriors then we can estimate the partition function.
In the case of unsupervised partioning :

> *Each node/cluster/blob is taken to be a class. Hence if we search for the positions and shapes of K clusters, the data set will be segmented into K classes. Generally, we do not know K ahead of time, and methods for assessing the probable number of classes in a data set are a subject of active research.*

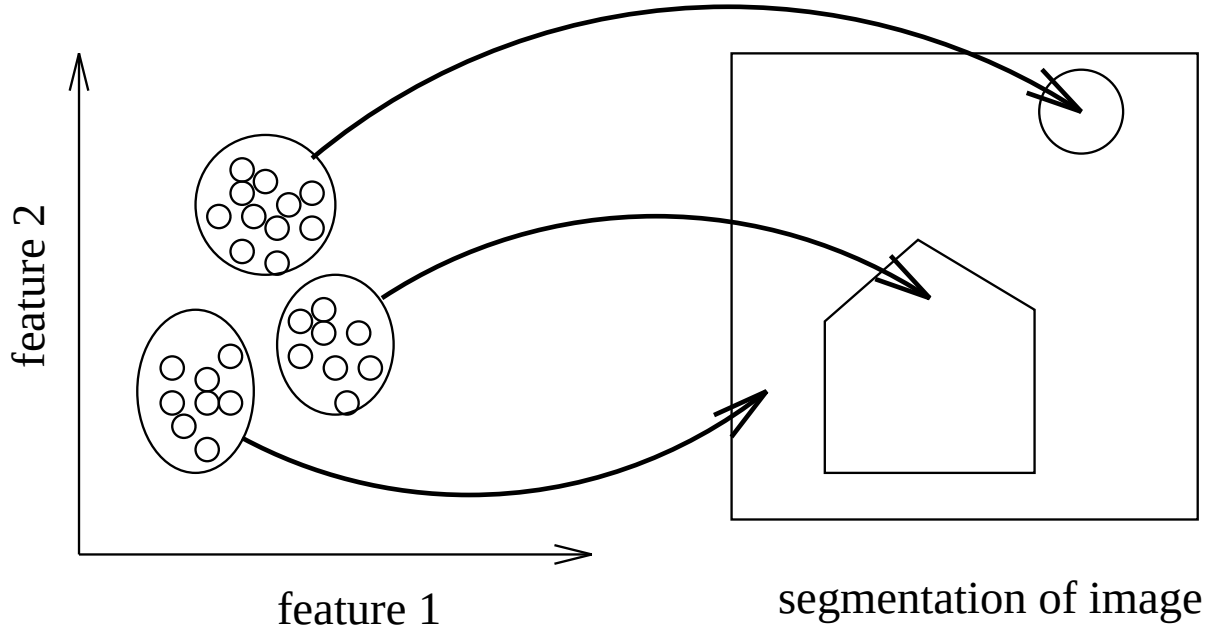

Figure 2: Blobs or clumps in a feature space – unsupervised classification.

Generalisation

One of the perennial problems of optimising a pattern classifier using a training set of data is that the apparent performance of the system improves with the number of tunable parameters in the system. Such a system runs the risk of *over-fitting* to the training data set and would thus perform poorly on new data. What is required is a system, obtained from a training set, which is capable of *generalising* to new data. This is a classic model-fitting problem and the pattern recognition community have adopted several methods to improve generalisation performance.

Validation – one way of doing it:

One of the favoured methods of ensuring decent generalisation performance is that of splitting the training data to provide a separate *validation* set as well as a (new and smaller) training set. One successful strategy is therefore to successively increase the *complexity* of the analyser and re-optimise each time on the *training set*. The performance of the analyser is thence evaluated on both the training *and* validation set. A typical set of results gives rise to curves such as in figure (5) in which the training-set error decreases with increasing complexity whilst the validation error reaches a minimum and thence *increases*. We may perform several 'runs' of such a validation procedure by randomly re-splitting the original data into many different training and validation sets (this is referred to as *n-fold validation*. We may then choose the analyser of complexity which, on average, generalises best (i.e. has lowest validation-set error). We must still, however, test the performance of our optimal system on a *test* set which is independent from either training or validation sets and is unused in the

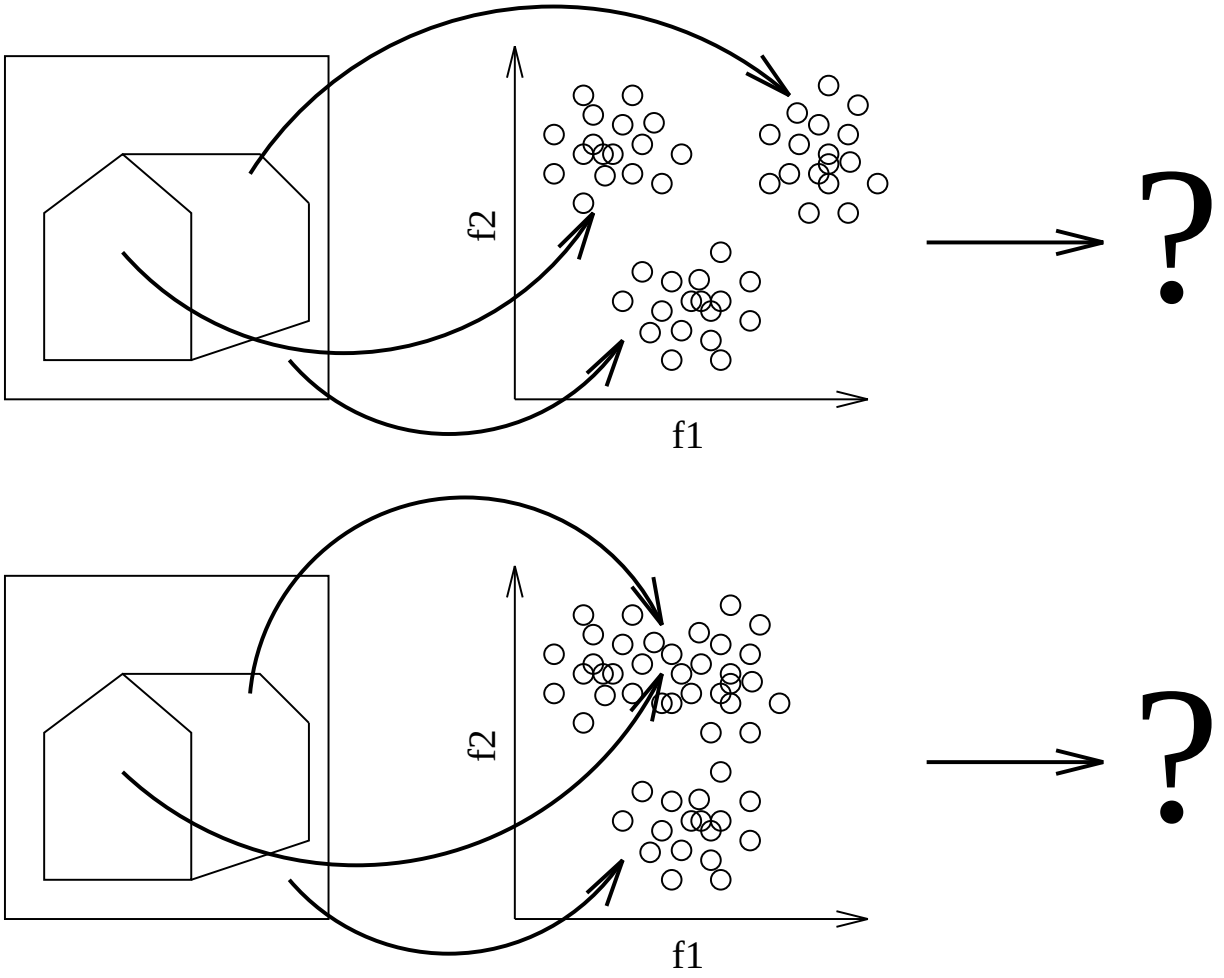

Figure 3: Unsupervised classification has its problems.

development procedure. In many 'real-world' problems, however, the available data set may be small. In this situation Bayesian learning methods may be superior (see later) as these allow comparison of different analysers (of differing complexities, for example) using only the training data set. What do we mean by small, however? Some studies address the issue of the number of pieces of information in the training set as a ratio to the number of adaptive parameters in the analyser. The results suggest that a factor of three or more is required. A general rule of thumb is to be even more pessimistic, however, and a factor of 5-10 is often suggested.

Typical application procedure

The most commonly used (validation) approach gives rise to a simple methodology with which an analyser may be trained and tested :

1. A *labelled* set of data is formed. This set consists of *input-output* pairs $(\mathbf{x}, \mathbf{t})$. We will furthermore assume that each component of the input set (x_1 etc.) has a similar numerical magnitude. This avoids numerically large components being given unfair weighting over those (possibly more useful) components whose

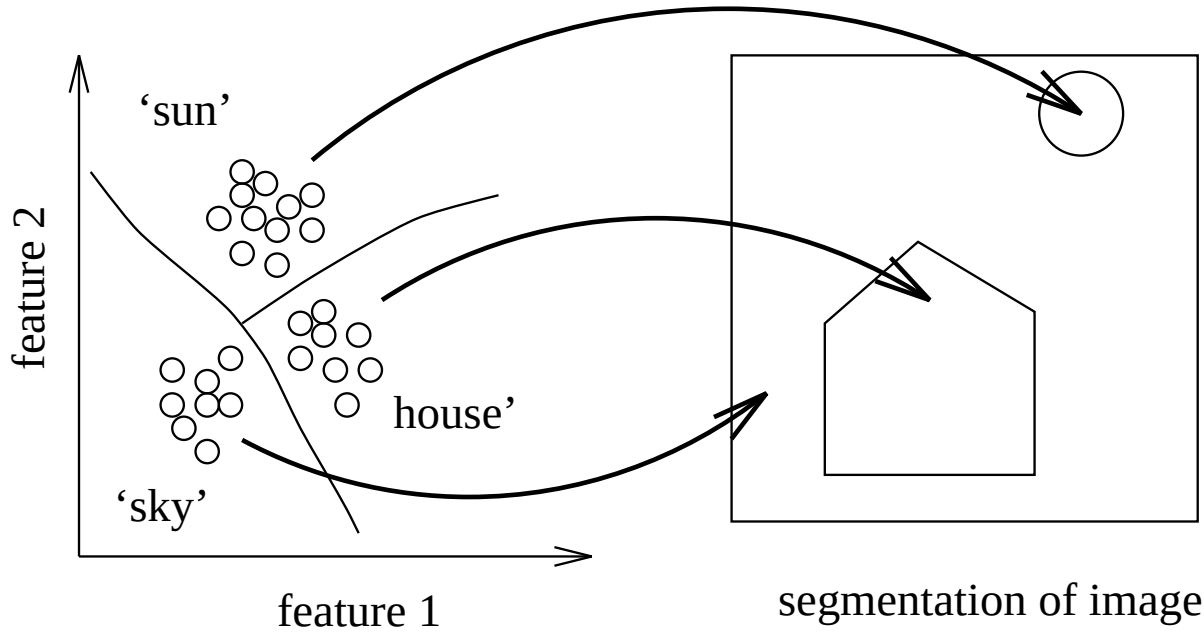

Figure 4: Partitions of a feature space – supervised classification.

magnitudes happen to be smaller. We will also assume that, in the *classification* case, this set is *balanced* i.e. equal numbers of examples from each class are contained in it.

2. The data is randomly split into, typically, *three* data sets: *training*, *validation* and *test*.

3. **a.** Starting with a ‘simple’ analyser (low number of free parameters) optimise its free parameters (weights) to minimise the error on the *training set*.

   b. Obtain performance measures for the analyser on the *validation* set.

   c. Increase the complexity of the anlayser (e.g. more parameters).

   d. Repeat these steps until a levelling or minimum in the validation error vs number of parameters (e.g.) curve is passed.

4. Repeat steps 2 and 3 with different random splits of the original data set. This may be performed ten times, say.

5. Find the analyser complexity that gives the best average (over the ten runs, say) performance on the *validation* set. Use an analyser with this configuration to assess performance on the (hitherto unseen) *test* set.

Generalisation: a quick example

This example is of the (in)famous ‘noisy sine wave’. 100 samples with 30% Gaussian noise are used in training and another 100 as a ‘test’ set. Figure 6 shows the regression using increasing numbers of parameters (actually a number of spline functions).

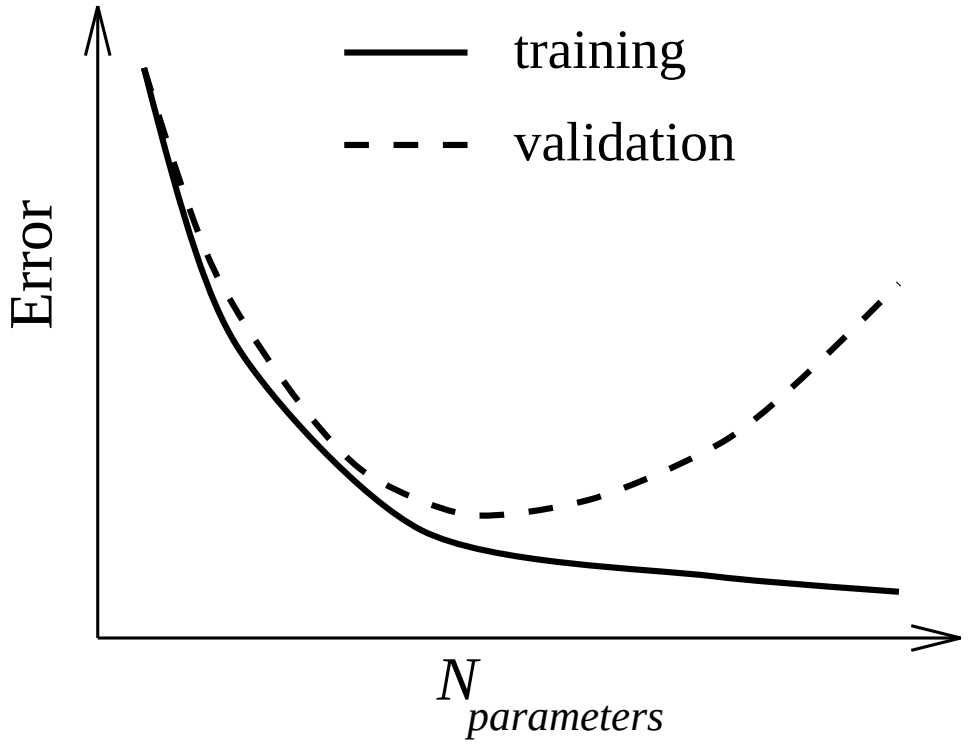

Figure 5: Training set error decreases, on average, monotonically with analyser complexity, whereas validation set error reaches a minimum at the ‘optimal’ analyser complexity.

Note that 1 is not enough, 5 is about OK (it captures the basic sine wave pattern) and 100 captures all the noise as well! Figure 7 shows these same analysers applied to the test set (which has different noise on it remember).

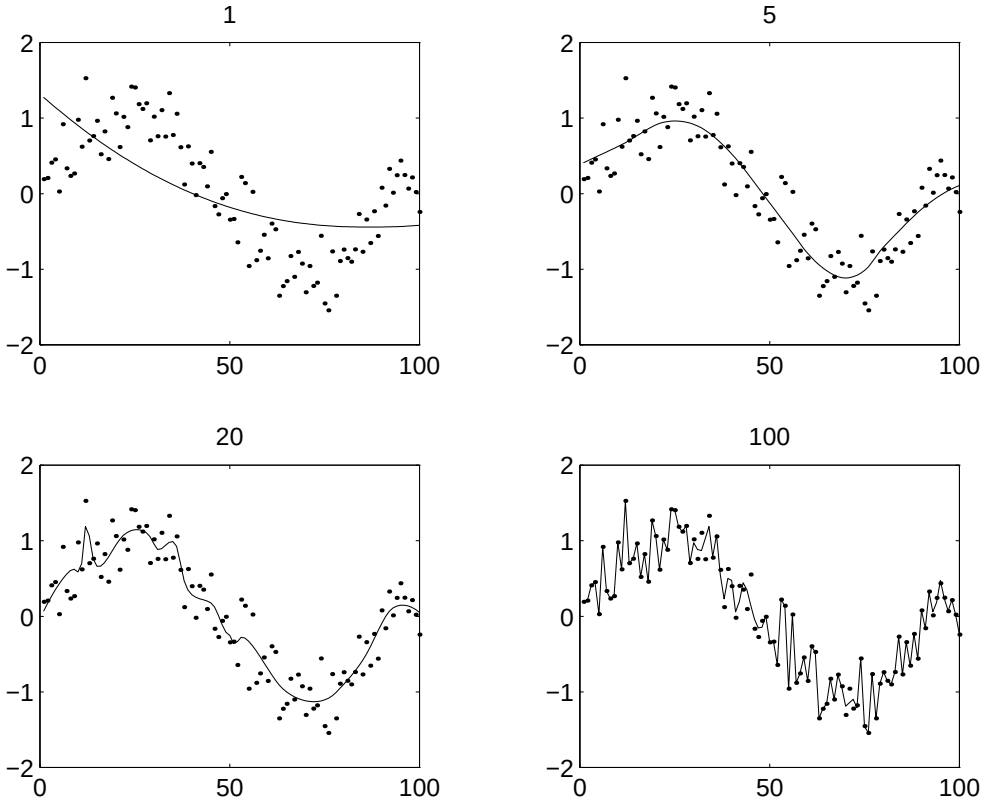

Figure 6: Generalisation: training set.

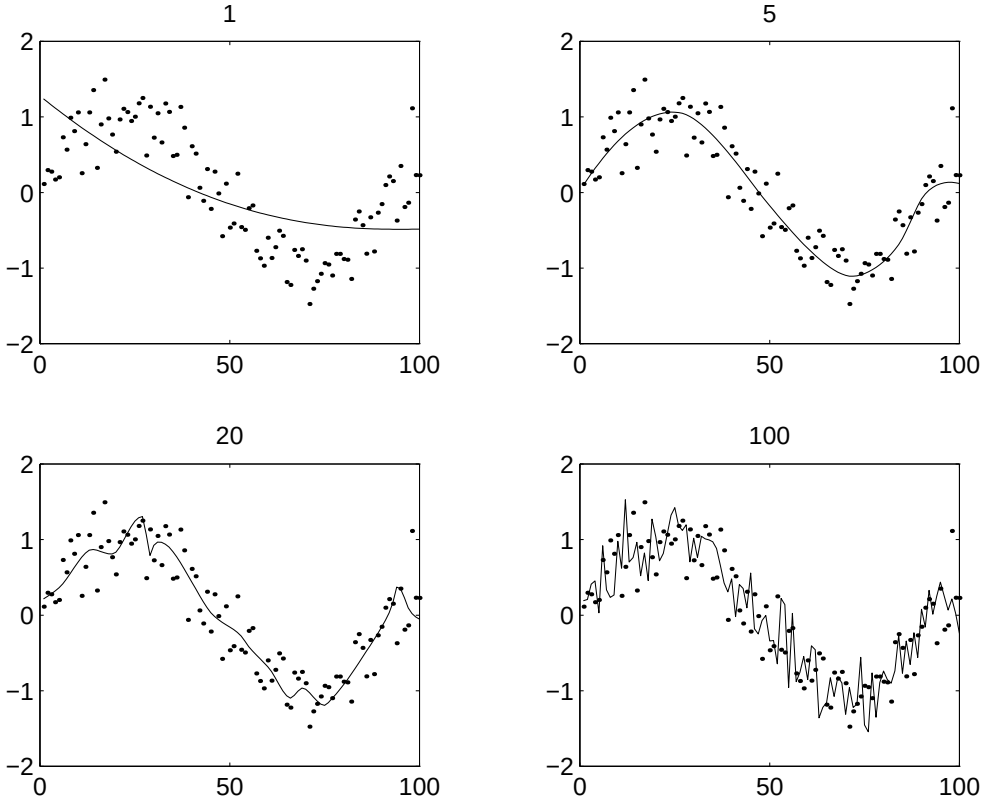

Figure 7: Generalisation: test set.

Some examples

In this section I introduce some simple data sets which, along with some others, are made available on line (see later).

Tremor analysis

This data was collected as part of a study trying to identify patients with muscle tremor. The features are auto-regressive coefficients which detail hand tremor resonances. The data set consists of equal numbers of data from patients and a control group. Figure 8 shows the data. Using committees of flexible classifiers (such as

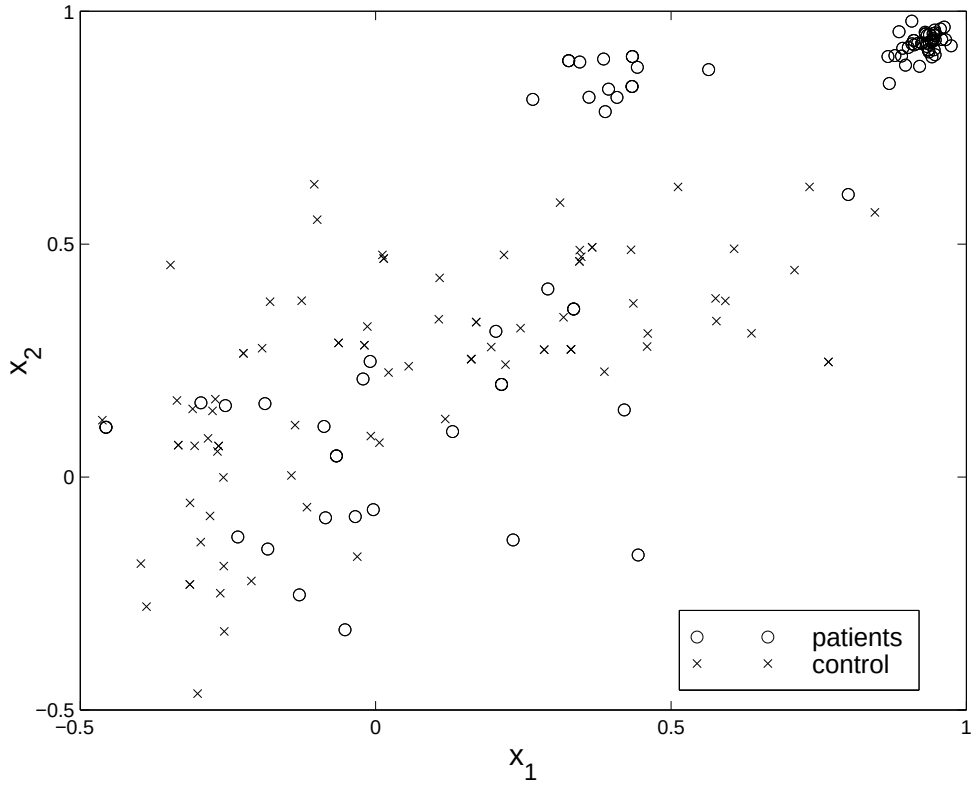

Figure 8: Tremor data set.

'neural' networks) we can classify this data with about 85% accuracy.

Ionosphere radar returns

This is a 33-D data set with redundancy in its features (they are non-independent). The label is 0 or 1 corresponding to to 'good' or 'bad' radar bounce from the ionosphere. The first two principle components are plotted in figure 9.

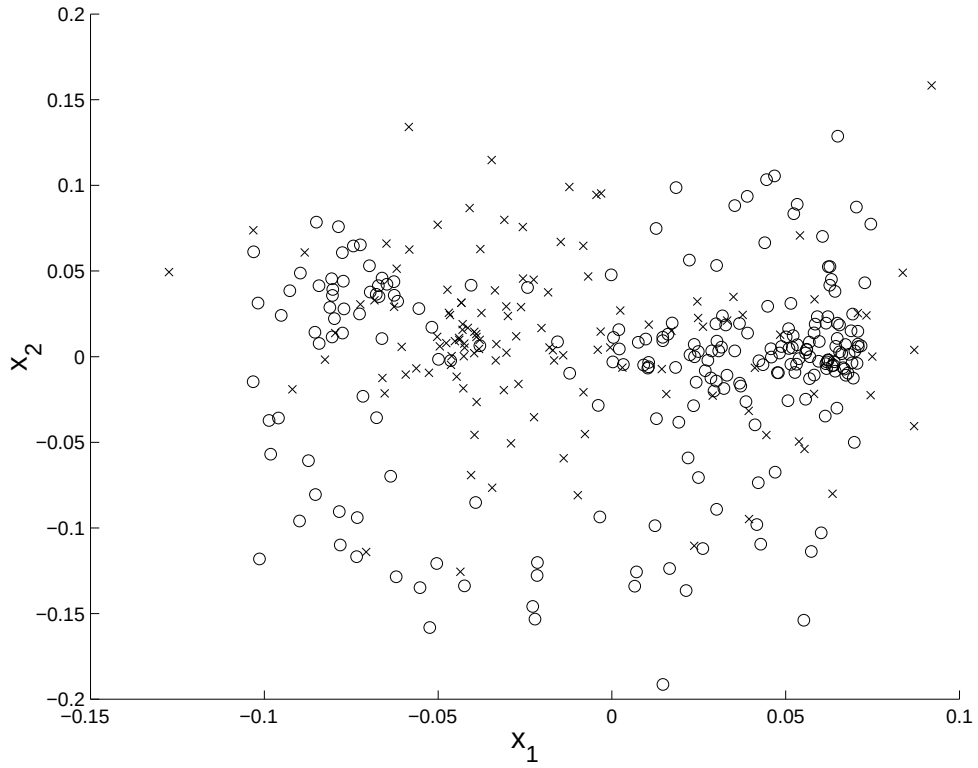

Figure 9: Ionosphere data set.

Predicting the future: chaotic time series

Figure 10 shows the Mackey-Glas chaotic time series. The data has been looked at as a prediction problem by using past samples from the data to try to predict into the future. It is fairly difficult to do it well! We can get (Fig. 11) to a 5% error level fairly easily.

Wine recognition

The data set consists of 178 13-dimensional exemplars which are a set of chemical analyses of three types of wine. Figure 12 shows the projection of the data onto its first two principal (eigen) components. The data, if we use the labels and know there are three kinds of wine, can be classified with 100% accuracy. If we are not allowed to use the labels, we can infer that there are three wines and classify with about 98% accuracy - not bad!

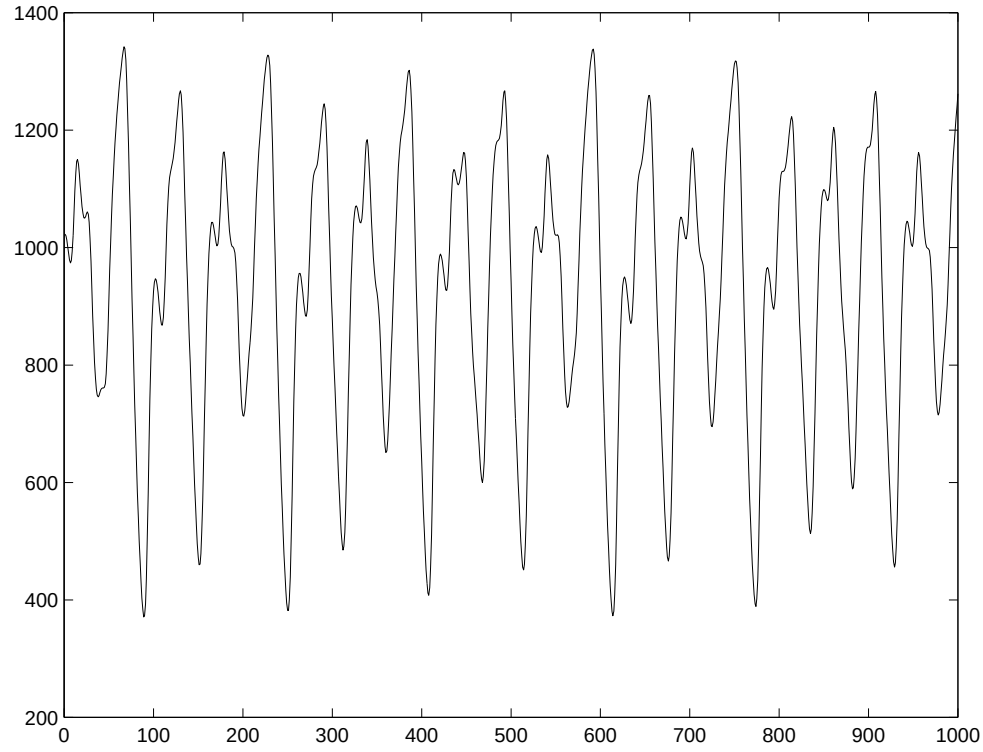

Figure 10: Chaotic time series data.

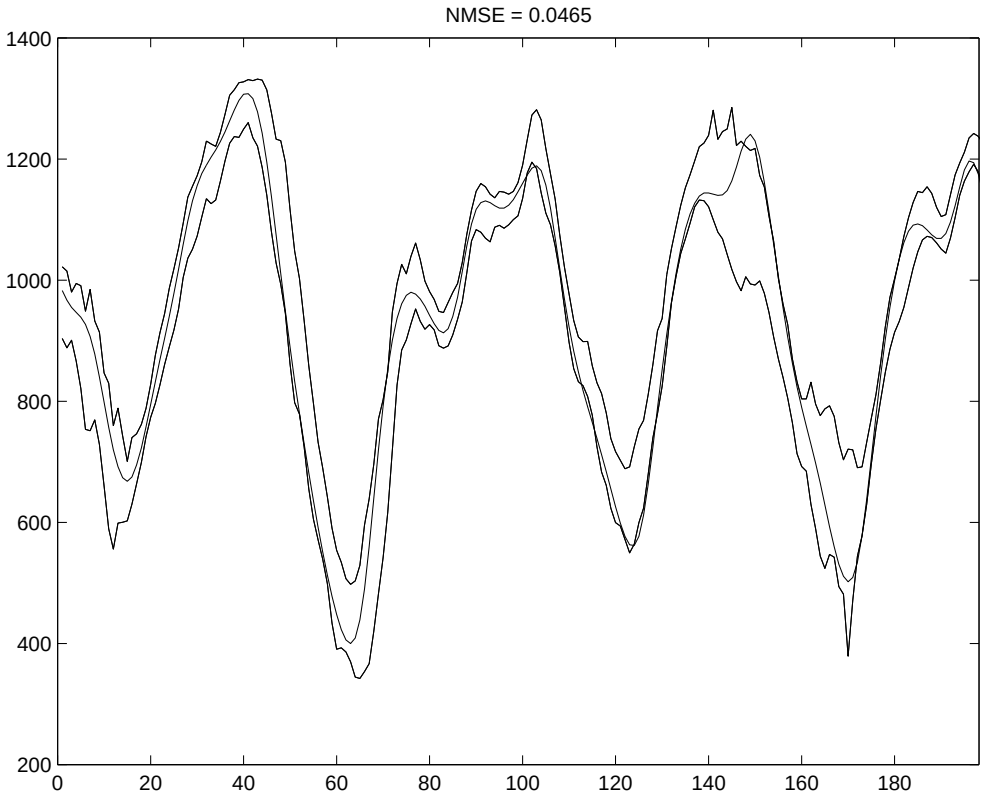

Figure 11: Chaotic time series data - predictions and error bars..

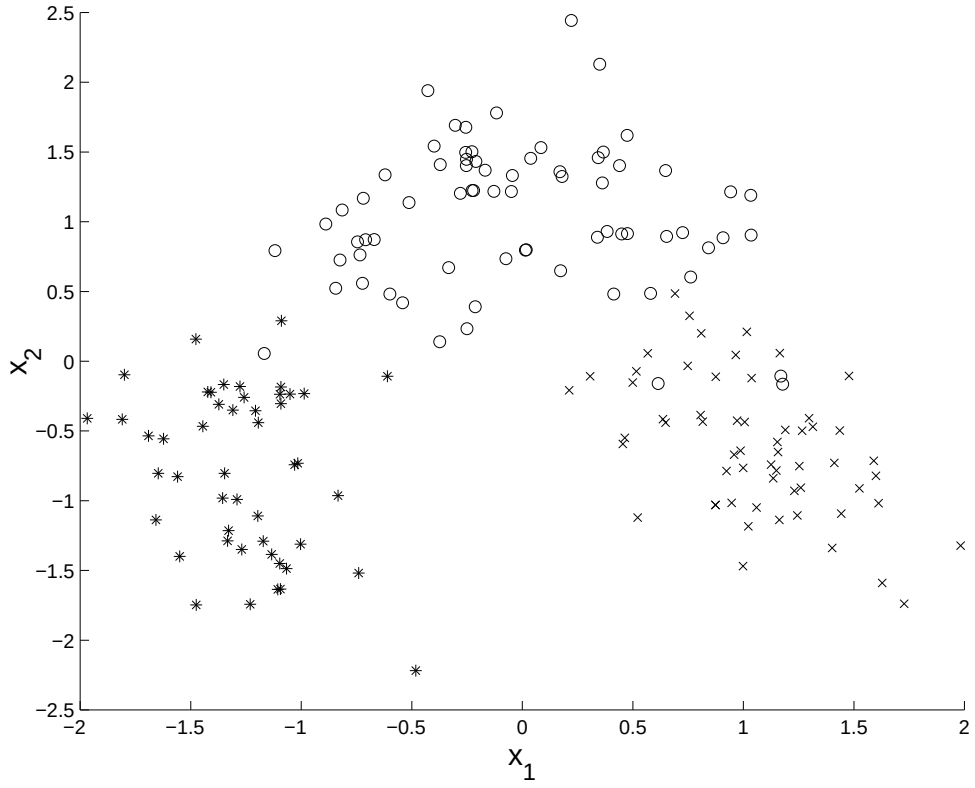

Figure 12: Wine data set.

Resources

Journals

- There is a good web site with journal information at
  `www.ph.tn.tudelft.nl/PRInfo/journals.home.html`

Machine learning archive

- There is a good machine learning and pattern recognition URL at
  `www.aic.nrl.navy.mil/~aha/research/machine-learning.html`

The link to the UCI data base is useful - also lots of free software to download.

Data, software for this course

Some example data sets may be downloaded via my web site:
`http://www.robots.ox.ac.uk/~sjrob/teaching.html`
and following the links to pattern recognition. I have made links to some of our group software archives also. Data sets (more extensive) are also available via the machine learning URL (in particular the UCI data repository) for those who are keen.

Bayes Theorem - the calculus of probabilities: a brief review

You will all, no doubt, be familiar with the following terms:

- Priors, likelihoods, posteriors and evidences,

which describe the components to quantities in the Rev. Thomas Bayes' famous and important theorem (mid 18th century).

Bayes' theorem relies upon the notion of *conditional probabilities*, such as "the chance that I will finish these lecture notes *given* my other tasks is 0.9". What this implies is that the chance (probability) of one event happenning is *conditional* on another. Note also that statements like this code a belief rather than a probability that is obtained by multiple trials (the so-called frequentist approach). It is hence possible to allow singular (non-repeatable) events to have a probability. Cox (1946) showed that so long as these beliefs obey some consistency rules (they sum to unity, are strictly non-negative etc.) known as the *Cox axioms* then these beliefs can be handled just as probabilities. I will come up front and say that *I believe that Bayesian statistics is correct*, although there are some who would disagree (they are gradually proved wrong though), so these lectures make this tacit assumption.

Just to refresh the memory...
Note: I will use capital 'P' to denote a probability, whether this is a 'degree of belief' (bounded in the closed interval [0,1]) or a *probability density function* (pdf - which is unbounded above, is non-negative and must integrate to unity). If the argument is *discrete*, such as $P(k)$ where k is the k-th class of a finite set, then we have belief. If the argument is *continuous*, such as $P(x)$, where x is some variable, then we have a pdf.

Formally we write "probability of A *given* E" as $P(A \mid E)$. What Bayes' theorem states is that:

$$P(A\&E) = P(A \mid E)P(E) = P(E \mid A)P(A)$$

Re-arranging the above gives:

$$P(A \mid E) = \frac{P(E \mid A)P(A)}{P(E)}$$

which is a basic formulation of Bayes' theorem. We can hence evaluate the chance of A occurring *given* E *has occurred* if we know the chance of E occurring *given* A

has occurred and the chances that A and E occur on their own.

$$\text{posterior } = \frac{\text{likelihood } \times \text{prior}}{\text{evidence}}$$

So we have 'converted' some *prior* estimate about the chance of A into a better estimate *given* information about E. This notion of using information in E to refine our belief about A is very elegant.

Say we had a series of hypotheses, $A_1, A_2, ..., A_n$ each dependent upon E then Bayes' theorem for *mixtures* states that

$$P(E) = \sum_{i=1}^{n} P(E \mid A_i) P(A_i)$$

we may combine this with the previous equation to obtain a set of *posterior* probabilities, one for each hypothesis A_i. It may be shown that, if we wish to *decide* the most probable outcome to event E happenning, that outcome is the A_i with the *largest posterior probability*.

$$\text{Most likely outcome if } E \text{ has occurred } = \text{ argmax}\{P(A_i|E)\}$$

An example

Take a simple, 1-D example. Let E be the gaining of information regarding a datum, x, and let there be two hypotheses, or classes, conditional on x, called A and B. Hence

$$P(A \mid x) = \frac{P(x \mid A) P(A)}{P(x)}$$

$$P(B \mid x) = \frac{P(x \mid B) P(B)}{P(x)}$$

where, from Bayes' mixture theorem

$$P(x) = P(x \mid A) P(A) + P(x \mid B) P(B)$$

The *likelihood* terms, e.g. $P(x \mid A)$ define how *likely* it is to generate x *given* class A etc. If we take a simple case, where A, B are taken to be Gaussian (normally) distributed classes, then this example may be drawn simply (see figure 13).

- $P(A \mid x) = 1 - P(B \mid x)$
- If $P(A) = P(B)$ (no prior preference) then choosing class with maximum *posterior* is the same as choosing the *maximum likelihood* class (as the evidence is common to all classes).

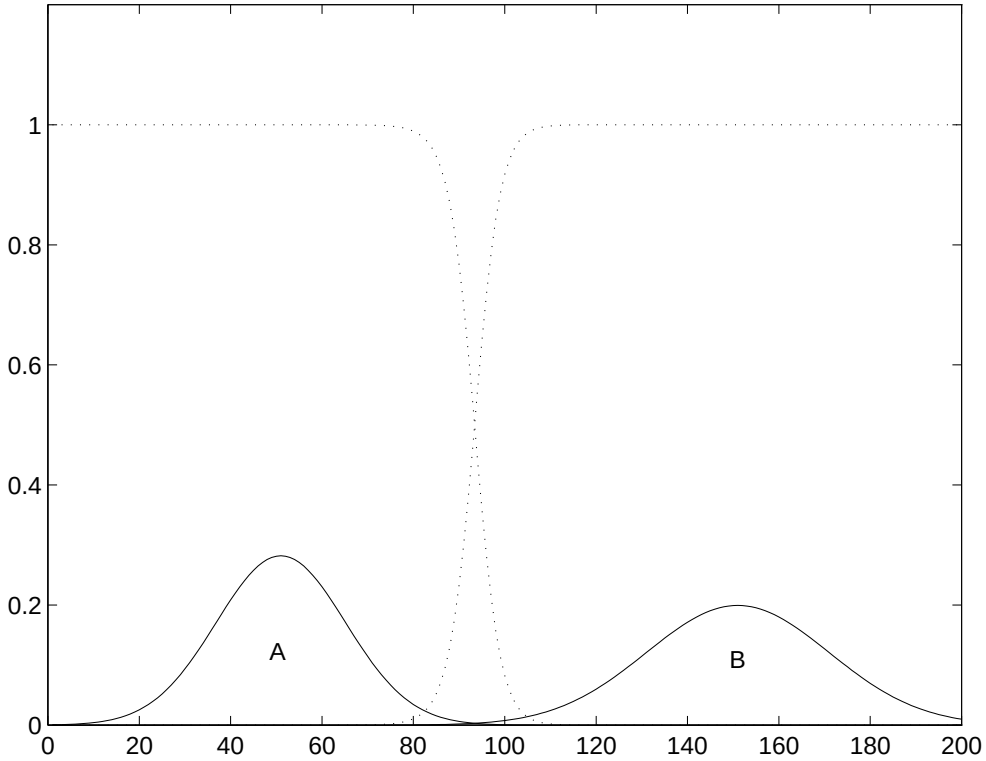

Figure 13: Posteriors and likelihoods.

To give another example, from image processing, let us say that x represents the value at some pixel from an edge detection operation. A and B can then be thought of as the chances of that pixel belonging to an edge or not.

$$P(\text{edge} \mid x) = \frac{P(x \mid \text{edge})P(\text{edge})}{P(x)}$$

and if $P(\text{edge}) = P(\text{no edge})$ then

$$P(\text{edge} \mid x) = \frac{P(x \mid \text{edge})}{P(x \mid \text{edge}) + P(x \mid \text{no edge})}$$

The figure shows some edge detection using such a scheme. I have not said yet how to get the likelihoods or how to go about performing classification *in practice*.

Classfication: in more detail

Minimum risk classifiers

We consider a set of regions such that R_k corresponds to class C_k, so if there are c classes we have $R_1 ... R_c$.

The total probability of getting a correct classification is

$$P(correct) \;=\; \sum_{k=1}^{c} P(x \in R_k, C_k) \tag{1}$$


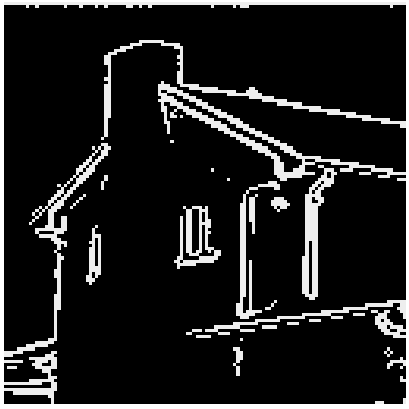

Figure 14: House image and the 'significant' edges ($P > 0.95$).

$$= \sum_k P(x \in R_k \mid C_k) P(C_k)$$
$$= \sum_k \int_{R_k} P(x \mid C_k) P(C_k)$$

this is maximised by always choosing to classify x to the class with the largest value of $P(x \mid C_k)P(C_k)$ which is just the posterior probability via Bayes' theorem.

The **loss matrix** enables a different costs to be associated with different wrong decisions. L_{kj} is loss when x is classified to C_j when $x \in C_k$. The expected loss over C_k

$$loss_k = \sum_{j=1}^{c} L_{kj} \int_{R_j} P(x \mid C_k) dx$$

as

$$loss_{total} = \sum_{k=1}^{c} loss_k P(C_k)$$

so

$$loss_{total} = \sum_{j=1}^{c} \int_{R_j} \left\{ \sum_{k=1}^{c} L_{kj} P(x \mid C_k) P(C_k) \right\} dx$$

this is minimised if the integrand is minimised at each x. We will thus choose to classify to C_j when

$$\sum_{k=1}^{c} L_{kj} P(x \mid C_k) P(C_k) < \sum_{k=1}^{c} L_{ki} P(x \mid C_k) P(C_k)$$

for all $i \neq j$. Consider a vector of output probabilities, $\mathbf{p}$, then classify using $\mathbf{p}' = \mathbf{Lp}$.

Note that the expected loss under this rule (the Bayes' rule) is just $E[L_{pmax}(1 - p'_{max})]$. This is known as the *Bayes' loss (or error)* and is the lower bound to the loss

for the data set. You can never do better than this (on average). If we don't follow the Bayes rule then this expectation is just called the expected loss or error.

The reject option

Let's make things a little simpler by assigning no loss to getting the right answer $L_{kk} = 0, \forall k$, and the same loss to getting it wrong, $L_{kj} = 1, k \neq j$. Now introduce another class, the 'doubt' class. We will classify to this class when we are not certain which other class to classify into. Let the loss for this be $L_{k,doubt} = d, \forall k$. Remember that the expected loss (for this loss matrix) is just:

$$loss = E[1 - P(C_{chosen} \mid x)]$$

to minimise this we must choose to classify to the class with the minimum

$$\{1 - P(C_1 \mid x), 1 - P(C_2 \mid x), ..., 1 - P(C_K \mid x), d\}$$

we see that we still choose the class with the largest $P_{max} = P(C \mid x)$ unless $P_{max} < 1 - d$ in which case we reject to the 'doubt' class.

By varying d we can look at the trade-off characteristics between accuracy and the fraction of data not rejected.

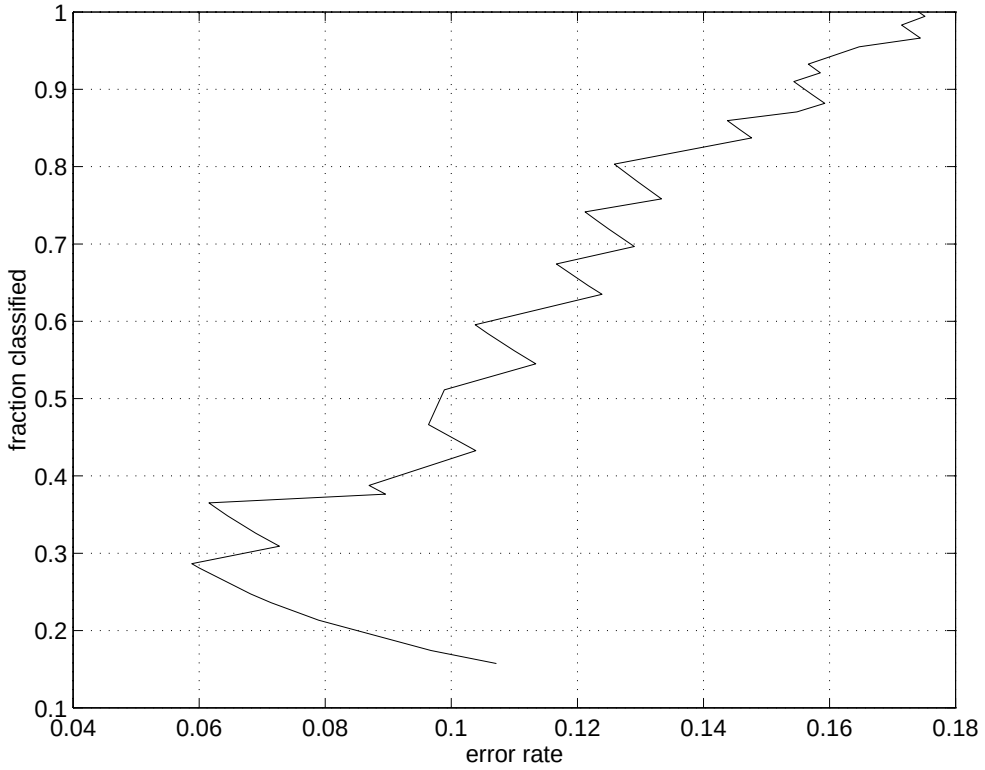

Figure 15: Reject option curve (tremor data set).

Receiver operating characteristics (ROC)

The ROC curve is much used in tha assessment of systems for use in variable cost domains i.e. where the elements of the loss matrix may vary in time. An example might be the a medical diagnosis problem in which the cost of falsely diagnosing a disease reduces as the disease becomes benign.

The ROC curve is a plot of true positive rate for a class against false positive rate. Consider a decision in a two class system, in which a decision is made to class C_1 if

$$P(C_1 \mid x) > t$$

By varying the threshold t we obtain a set of classifiers with differing characteristics, and differering true and false positive rates.

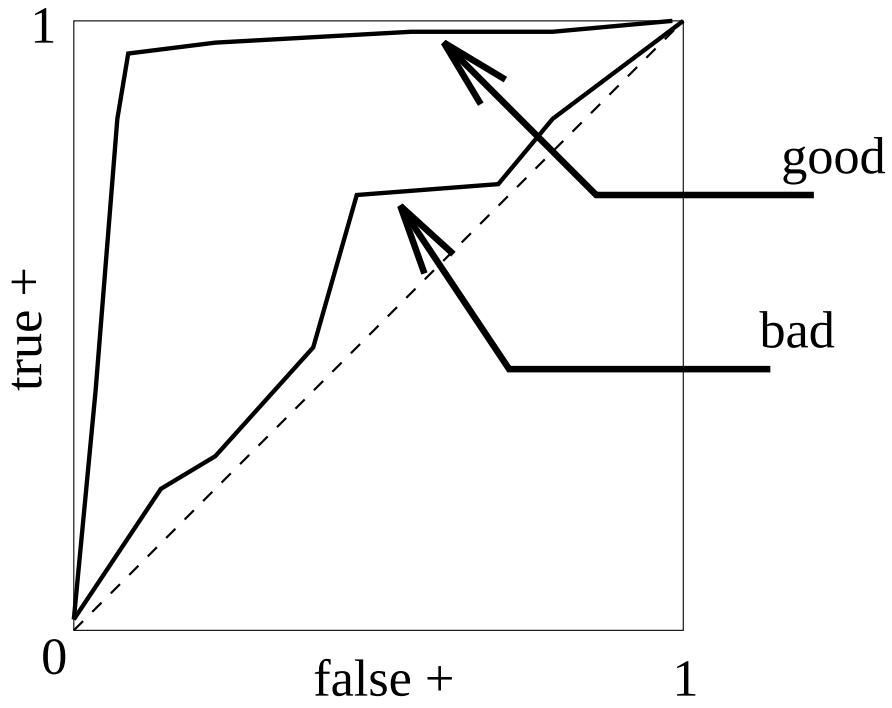

Figure 16: ROC curve, showing good and bad classifiers.

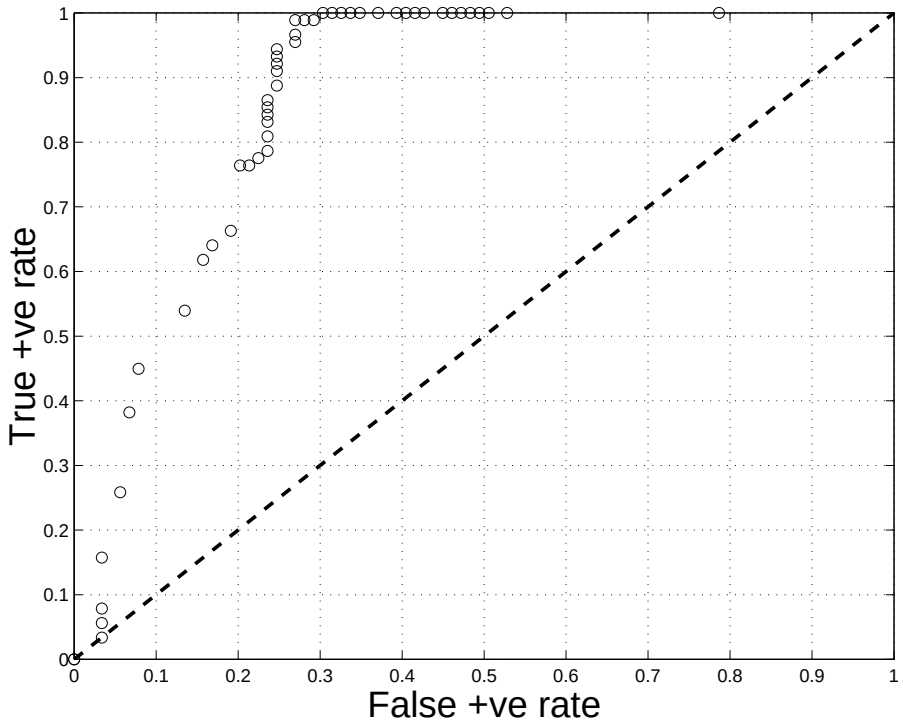

Figure 17: ROC curve for the tremor data set.

Uncertainty and information gain

Note that the statistic $\mathcal{U} = 1 - \max_k\{P(C_k \mid x)\}$ naturally denotes the uncertainty in the decision to classify to the class with the largest posterior probability. We can see this quantity as a measure of reliability in the decision. It makes sense that we use the reject or doubt option when this quantity is low.

If a datum provides clear information then $\mathcal{U}$ is small. Imagine we have no information provided by x, the posteriors just lie at the priors. If x is informative then the uncertainty (inherent typically in the priors) collapses and the posteriors diverge from the priors.

As an aside The strict measure of uncertainty is the entropy, H of some random source. Entropy (in bits) is defined, over a continous space, as:

$$H(x) = -\int P(x)\log_2 P(x)dx$$

and in discrete space as:

$$H(k) = -\sum_{k'} P(k')\log_2 P(k')$$

For a Gaussian source the entropy is $\propto \log_2 \sigma$ hence large entropies correspond to large variances and *vice versa*.

Consider now the observation of some datum x. We know that if we classify to class k then $P(k \mid x)$ was the maximum posterior. We consider the measure

$$r = \log_2\left(\frac{P(k \mid x)}{P(k)}\right)$$

By Bayes' therom this is also equal to:

$$r = \log_2\left(\frac{P(x \mid k)}{P(x)}\right)$$

for $P(x \mid k) > P(x)$ then the region that class k models must be a subset of the entire data space. This means that we have localised x. The tighter this localisation, with reference to the original data set, then the larger the measure r, hence the lower the uncertainty in the classification decision. Figure 18 shows this for the tremor data set using different classifiers. Note that the gain is bounded above by unity (1 bit = ability to classifiy into one of two classes) and below by 0. The latter lies along the decision boundary. A datum on this boundary is impossible to classify, so seeing it gives no information regarding its class.

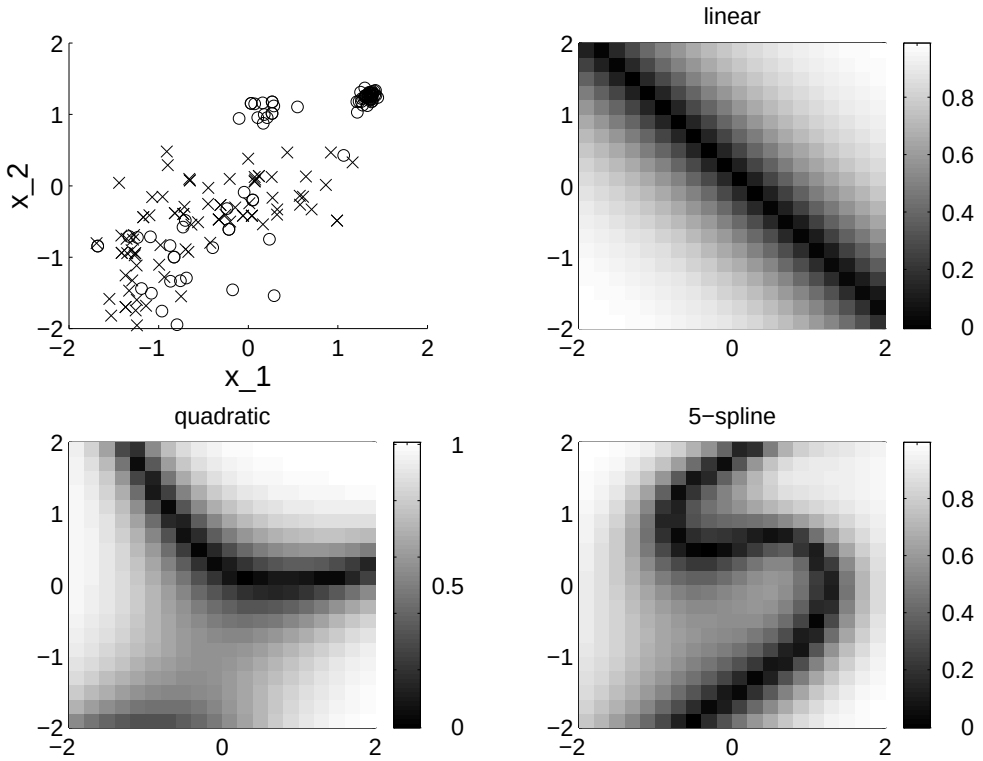

Figure 18: Information gain as $log_2\left(\frac{P(x|k)}{P(x)}\right)$.

The outlier option

The reject option offers the option to disregard a datum if the subsequent classification has high uncertainty. If we look again at figure 13 we see that the posteriors are high (they asymptote to unity) as we go far away from the data (likelihoods). This *assumes that the classes represent a complete hypothesis space,* i.e. there truly are no other classification options. If we are faced with data which lie far away from all known classes, we may wish to assign it to an outlier class under the belief that it is unlikely to have been truly generated by our hypothesis set. This approach is also referred to as *novelty detection.* Figure 19 depicts this.

Regression: in more detail

Let us consider a data set which consists of the pairs $(x, t) \in \Re$. We will consider our errors to be Gaussian distributed, such that the output of the regressor given input x is $y(x)$.

$$P(y \mid x) \propto \exp\left\{\frac{\beta}{2}[y(x) - t(x)]^2\right\}$$

as an error functional can be formed from the negative log error pdf so

$$E(x) \propto [y(x) - t(x)]^2$$

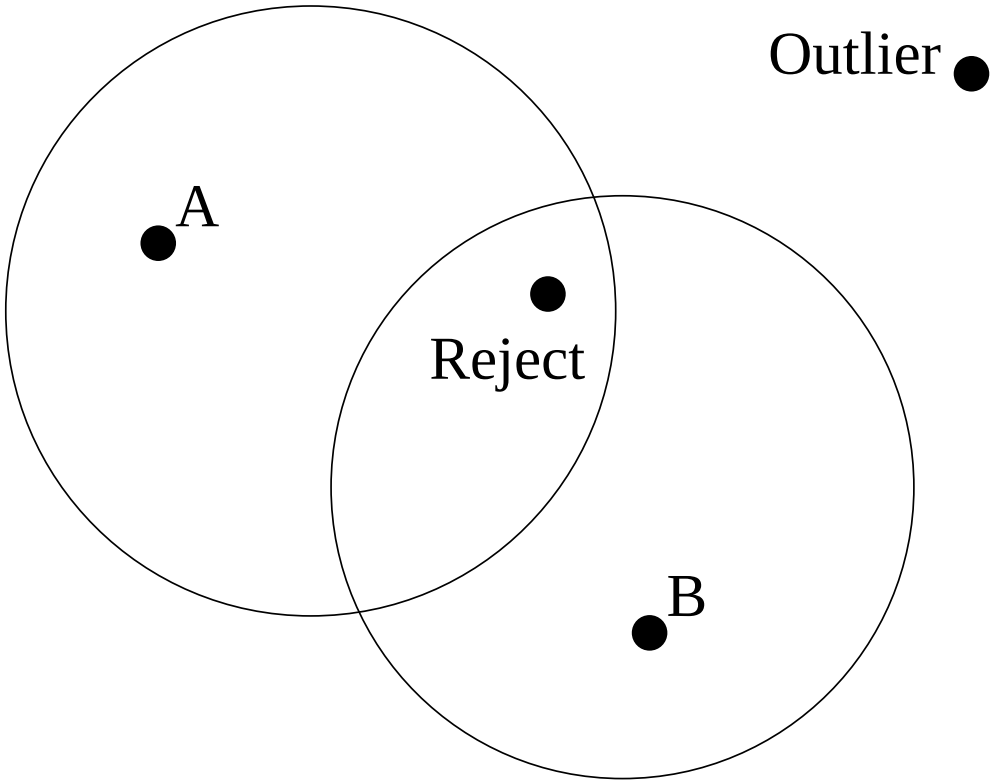

Figure 19: Rejects and outliers.

Integrating this over the target pdf gives

$$\langle E(x) \rangle = \int P(t \mid x)[y(x) - t(x)]^2 dt(x)$$

expanding and completing the square (noting that $y \neq y(t)$) gives

$$\langle E(x) \rangle \propto [y(x) - \langle t \mid x \rangle]^2 + \text{var}\,(t \mid x)$$

where $\langle t \mid x \rangle = \int P(t \mid x) t(x) dt(x)$. Given that the last term depends on the data set, the best we can do is to, for each x, get a regressor $y(x) = \langle t \mid x \rangle$, the *conditional average of t given x*.

Error bars

On of the things you might have noticed is that, for regression, the argument of our probability is a continuous variable, y, rather than a member of a discrete set of classes. Just as a full (most informative) description of a class decision is $P(C_k \mid x)$ so the most informative regression measure is the *density function* $P(y \mid x)$. Note that the latter is a pdf, whilst $P(C_k \mid x)$ is just a single number representing a degree of belief.

How do we represent a density? One way is to assume that the density is normal and then just give its mean and variance. If we do this then regressors must output

the mean, which we expect to be $y(x) = \langle t \mid x \rangle$ along with an *error bar*. How to get that error bar is dealt with later on in the course as it requires an understanding of Bayesian learning methods.

Just to make another thing clear; *posterior probabilities (beliefs) do not have error bars!*