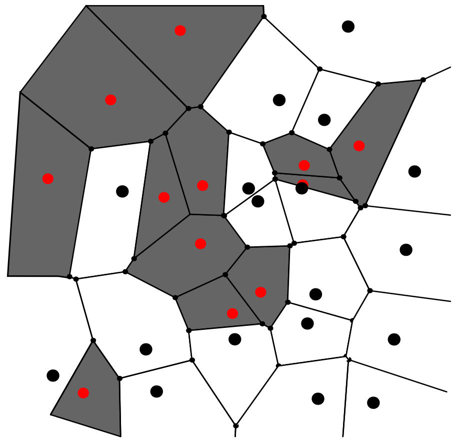
University of Cambridge
Engineering Part IIB

Paper 4F10: Statistical Pattern Processing

Handout 11: Non-Parametric Techniques



Mark Gales
mjfg@eng.cam.ac.uk
Michaelmas 2013

Introduction

Various forms of classifiers, such as Gaussian distributions and logistic regression, have been examined. These start by assuming the nature of the decision boundary, or class-conditional PDFs and then estimate the parameters from training data.

An alternative approach is to use the training data to determine the form of the decision boundary or class-conditional PDFs. These approaches are called non-parametric. We have already seen a couple of schemes that are based on the training examples, support vector machines and Gaussian process classification. Gaussian mixtures are sometimes described as a semi-parametric approach due its ability to approximate many different functional forms.

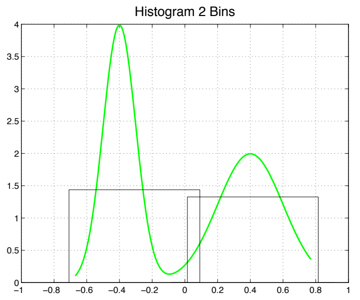
In this lecture techniques for calculating class-conditional density estimates, and building classifiers, without assuming a particular form for the density in advance will be examined. First we we will look in more detail at the density estimation problem and then classification. In particular:

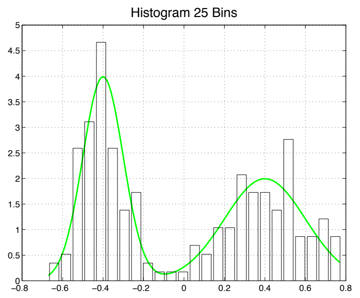
- kernel estimation (Parzen Windows);
- k-nearest neighbour method;
- nearest neighbour classifier.

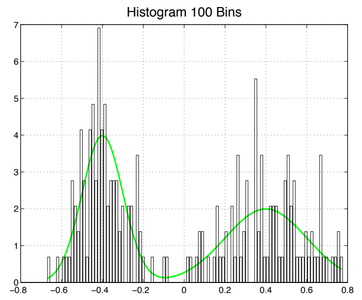
By default these approaches are not sparse (like Gaussian processes). A simple scheme for making a nearest neighbour scheme sparser will be described.

Histograms

Simplest form of non-parametric estimation is the histogram







- Histograms (in 1-D) require bin size determination
- Too large bins cause over-smoothing and loss of detail
- Too small bin cause noisy estimation
- Problem rapidly gets worse as dimension increases : number of bins is each side a constant length rises exponentially with dimension (the curse of dimensionality)

Non-Parametric Density Estimation

The histogram can be generalised to other forms of region. The probability that a new vector $\boldsymbol{x}$ drawn from the unknown density $p(\boldsymbol{x})$ will fall into a region $\mathcal{R}$ is

$$P = \int_{\mathcal{R}} p(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}}$$

If n samples are drawn independently from $p(\boldsymbol{x})$ then

$$P\left[k \text{ of these will fall in region } \mathcal{R}\right] = \left(\begin{array}{c} n \\ k \end{array}\right) P^k \left(1-P\right)^{(n-k)}$$

- This is a binomial distribution with $\mathcal{E}\left\{k\right\} = nP$;
- Can use $\frac{k}{n}$ as an estimate of P
- Variance of this estimate about the "true value" P
$$\mathcal{E}\left\{\left(\frac{k}{n} - P\right)^2\right\} = \frac{1}{n^2}\mathcal{E}\left\{\left(k - nP\right)^2\right\} = \frac{P(1-P)}{n}$$
- Estimate peaks sharply around the mean as the number of samples increases ($n \rightarrow \infty$)

If $p(\boldsymbol{x})$ is continuous and smooth in the region $\mathcal{R}$, the probability can be approximated as

$$P = \int_{\mathcal{R}} p(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \approx p(\boldsymbol{x})V$$

where V is the volume of $\mathcal{R}$ and $\boldsymbol{x}$ is some point lying inside it. Hence

$$p(\boldsymbol{x}) \approx \frac{k/n}{V}$$

Region Selection

How good is the estimate

$$p(\boldsymbol{x}) \approx \frac{k/n}{V} = \tilde{p}(\boldsymbol{x})$$

For this estimate, $\tilde{p}(\boldsymbol{x})$, to be reasonable:

- $\mathcal{R}$ large enough so that binomial distribution is sharply peaked
- $\mathcal{R}$ small enough so that using a constant approximation in a region for $p(\boldsymbol{x})$ is accurate

There are two basic choices in determining the volume, V of the region $\mathcal{R}$:

- use a fixed volume. This is normally achieved by selecting a particular window function which is used for all points.
- allow the volume to vary. The volume depends on the number of points in a particular region of space

Having obtained an estimate of the class-conditional PDF using the data from each of the classes and class priors estimated, a generative model has been estimated, so classification can be performed using Bayes' decision rule where

$$P(\omega_j|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_j)P(\omega_j)}{\sum_{i=1}^{k} p(\boldsymbol{x}|\omega_i)P(\omega_i)} \approx \frac{\tilde{p}(\boldsymbol{x}|\omega_j)P(\omega_j)}{\sum_{i=1}^{k} \tilde{p}(\boldsymbol{x}|\omega_i)P(\omega_i)}$$

Hypercube Regions

Let $\mathcal{R}$ be a hypercube of dimension d and edge length h. Then volume of this is

$$V \;=\; h^d$$

Need to count the number of points that lie within this region.

The number of points may be specified by defining an appropriate kernel, or window, function. One appropriate form is

$$\phi(\boldsymbol{u}) \;=\; \begin{cases} 1 & |u_j| \;\leq\; 0.5 \\ 0 & \text{otherwise} \end{cases}$$

where

$$\boldsymbol{u} = \frac{(\boldsymbol{x} - \boldsymbol{x}_i)}{h}$$

This is a unit hypercube, with centre at $\boldsymbol{x}$. If the sample $\boldsymbol{x}_i$ is within this hypercube the value is one and zero otherwise. The total number of samples contained within the hypercube can be expressed as

$$k \;=\; \sum_{i=1}^{n} \phi\left(\frac{\boldsymbol{x} \;-\; \boldsymbol{x}_i}{h}\right)$$

and a corresponding density estimate is

$$\tilde{p}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d}\phi\left(\frac{\boldsymbol{x} \;-\; \boldsymbol{x}_i}{h}\right)$$

Similar to the histogram except cell locations (centres) defined by data points rather than fixed.

Parzen Windows

The use of hypercubes to estimate PDFs is an example of a general class called Parzen Window estimates.

It is possible to use any form of window function that satisfy

$$\phi(\boldsymbol{u}) \geq 0; \qquad \int \phi(\boldsymbol{u})d\boldsymbol{u} = 1$$

These constraints ensures that the final density estimate will be a valid probability density.

One often used window function is the normal distribution.

$$\phi(\boldsymbol{u}) = \frac{1}{(2\pi h^2)^{d/2}} \exp\left(-\frac{||\boldsymbol{u}||^2}{2h^2}\right)$$

The density estimate is given by

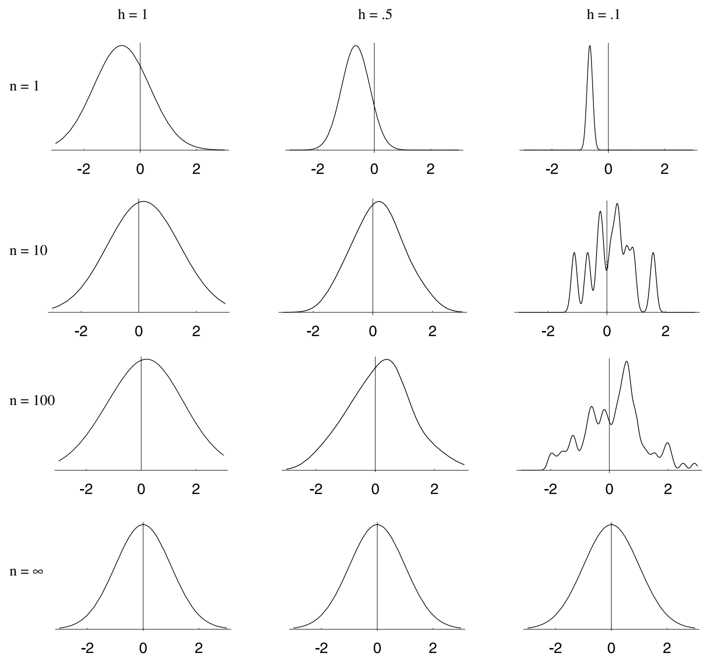
$$\tilde{p}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{(2\pi h^2)^{d/2}} \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}_i||^2}{2h^2}\right) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

where $\alpha_i = 1/n$ and $k(\boldsymbol{x}, \boldsymbol{x}_i) = \phi(\boldsymbol{u})$. The similarity to the decision boundary in SVMs and prediction mean in Gaussian processes is clear.

The window width, h, is a free parameter associated with this estimate and influences the nature of the density estimate (compare to length scale in GPs and kernel width in SVMs). It is commonly made a function of the number of samples, n. One form is $h_n = h/\sqrt{n}$

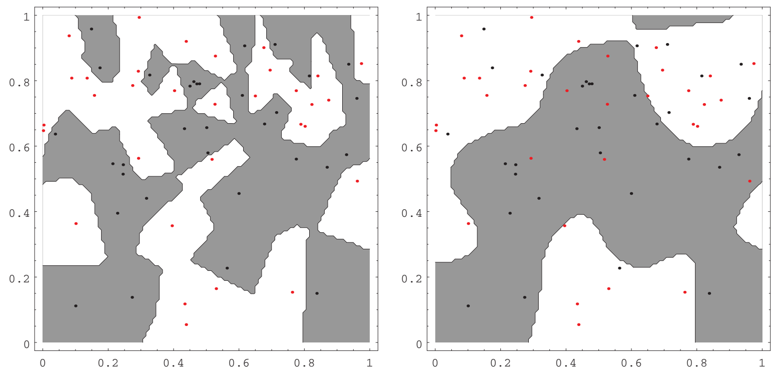
Parzen Window Example

Below are examples of the Parzen Window estimates of a univariate Gaussian PDF. The window width is set using h which is defined for each case. The values of the estimate for different values of n and h are shown below (from DHS).



Parzen Window Classification

The value of h that results from these forms of density estimation will also affect the decision boundaries in a classification task. For the left diagram a small value of h is used, a larger value for the right diagram (figs from DHS).



There are a number of issues with this type of approach:

- Requires all data points to be stored, and evaluation of PDF may be slow (not sparse)
- Need to choose window function and widths.
- Can be shown that window in fact gives a biased estimate of the true PDF (see examples paper)

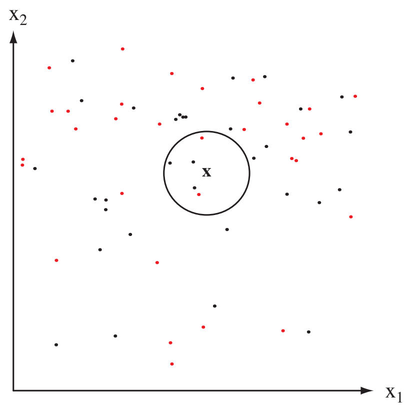
An alternative approach is to adapt the width of the windows according to the data available.

k-Nearest Neighbours Density

Rather than fixing the volume, it can be made to vary according to the density of points in a particular region of space. This is the k-nearest neighbour approach.

The basic procedure for estimating the density at a point $\boldsymbol{x}$

1. Select the value of k.
2. Centre a cell around $\boldsymbol{x}$ and allow it to grow until it captures k nearest neighbours. (From DHS)



Compute the volume of the cell, V.

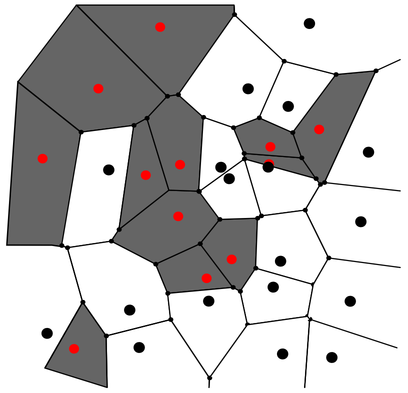
3. Density estimate given by (n total training samples)

$$\tilde{p}(\boldsymbol{x}) = \frac{k/n}{V}$$

Nearest Neighbour Classifier

Rather than using non-parametric techniques to estimate PDFs which may then be used, if required, for classification, non-parametric techniques may be directly used for classification.

For nearest neighbour classifier the space is partitioned into regions with a particular class label. This is sometimes referred to as the Voronoi tessellation and is illustrated below for 2D data.



Note that in the above we are assuming a Euclidean distance measure is used to measure "nearest", and it may be necessary to pre-process the data to make this appropriate.

- Easy to implement;
- No training of a model required

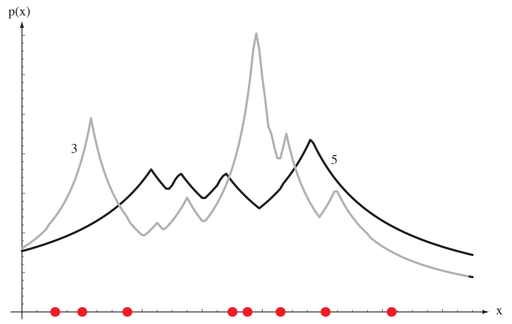
k-NN Classification

An extension to NN is to use k-NN for classification

- Assume that there are n_i points from class ω_i in the data set so $\sum_i n_i = n$. Construct a hypersphere of volume V around $\boldsymbol{x}$ so that k samples are contained of which k_i are from ω_i
- The class-conditional density and prior are given by

$$p(\boldsymbol{x}|\omega_i) \approx \tilde{p}(\boldsymbol{x}|\omega_i) = \frac{k_i/n_i}{V}, \quad P(\omega_i) = \frac{n_i}{n}$$

  The density estimate for two different values of k for a simple 1-D example is shown below (fig from DHS)



- The posterior probability is given by

$$P(\omega_j|\boldsymbol{x}) \approx \frac{\tilde{p}(\boldsymbol{x}|\omega_j)P(\omega_j)}{\sum_{i=1}^{k}\tilde{p}(\boldsymbol{x}|\omega_i)P(\omega_i)} = \frac{k_j}{k}$$

Computational Cost

Consider n labelled training points in d dimensional space. For a direct implementation of the NN classifier:

- each distance measure takes $\mathcal{O}(d)$;
- each of n stored point is compared to the test sample, total cost $\mathcal{O}(nd)$.

As n get large this becomes impractical, so techniques have been developed to reduce the load. The set of distances between training points could be pre-computed and stored - requires storing $n(n-1)/2$ distances. Alternatively:

- partial distances: the distance is computed on some subset r of the parameters, $r < d$. This distance is used to reduce the search. For example if the partial distance is greater than the current closest distance, terminate search.
- search trees: structure the search so that at each level only a subset of the training samples are evaluated. This determines which branch of the tree is then searched. This process typically does not guarantee that the nearest neighbour is found, but can dramatically reduce the computational load.
- editing: eliminate “useless” prototypes, samples, during training. For example remove prototypes that are surrounded by training points of the same category.

Nearest Neighbour Editing

A simple algorithm for allowing editing is for training data $\mathcal{D}$:

```
Initialise $j = 0$;
Construct the full set of neighbours for $\mathcal{D}$ :
    do j=j+1;
    examine the neighbours of $\boldsymbol{x}_j$
    if any neighbour is not from the same class as $\boldsymbol{x}_j$
        mark $\boldsymbol{x}_j$
    until $j = n$ Discard all points that are not marked.
```

This process will leave all the decision boundaries unchanged, hence the performance of the algorithm will be the same as that of the original set of complete points. [It’s non-trivial to get the full set of neighbours in high-dimensional space!]

This algorithm does not guarantee that the minimal set of points is found. However, it can dramatically reduce the number of points.

It is possible to combine all 3 schemes for reducing the cost. Editing may be used to reduce the number of prototypes. These may then be placed in a search tree. Finally partial distances may be used in the evaluation.

Non-Parametric Bayesian Schemes (ref)

In recent years there has also been interest in looking at schemes where, for example, the number of components in a Gaussian mixture model (GMM)

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$$

First consider rather than using a single value of the parameters, $\boldsymbol{\theta}$, a Bayesian approach can be adopted. Here for an M component GMM

$$p(\boldsymbol{x}|\mathcal{D}, M) = \int p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}, M) d\boldsymbol{\theta}$$

where $p(\boldsymbol{\theta}|\mathcal{D}, M)$ determines a GMM parameters

- component prior: $c_1, \ldots, c_M$, prior for each component
- component distribution: $\{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\}, \ldots, \{\boldsymbol{\mu}_M, \boldsymbol{\Sigma}_M\}$

What happens as $M \rightarrow \infty$?

This is an infinite GMM. The change to the likelihood is:

$$p(\boldsymbol{x}|\mathcal{D}) = \int p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}$$

where $p(\boldsymbol{\theta}|\mathcal{D})$ will also specify the number of components.

Non-Parametric Bayesian Schemes (ref)

Unfortunately not possible to get a parametric form for $p(\boldsymbol{\theta}|\mathcal{D})$. How to train the system?

Possible to use a sample-based approach (as with RBM)

$$\begin{aligned} p(\boldsymbol{x}|\mathcal{D}) &= \int p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \\ &\approx \frac{1}{J} \sum_{j=1}^{J} p(\boldsymbol{x}|\boldsymbol{\theta}_j) \end{aligned}$$

where $\boldsymbol{\theta}_j$ is a sample of possible GMM parameters drawn from $p(\boldsymbol{\theta}|\mathcal{D})$.

This has modified the problem from training the distribution, to drawing samples from $p(\boldsymbol{\theta}|\mathcal{D})$ - this is possible!

It is possible to apply the same sort of approaches to

- mixtures of experts: infinite mixture of experts
- hidden Markov models: infinite HMMs