Simple density estimation techniques

Alberto Paccanaro
EMAp – FGV

www.paccanarolab.org

Material and images in these slides are from (or adapted from):
C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006

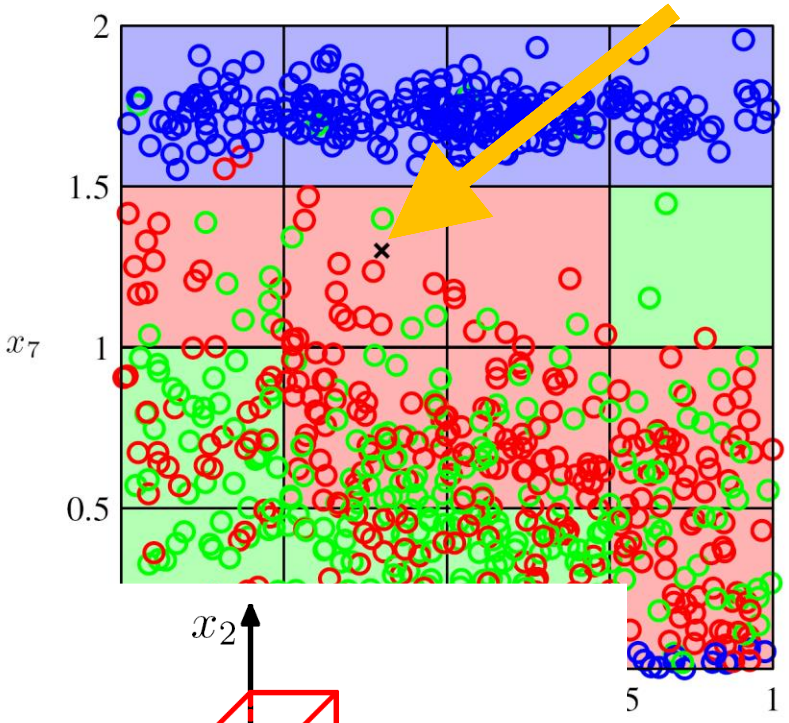
The K-nearest-neighbour algorithm
(a.k.a. K-nearest neighbour, KNN, K-NN, Knn, K-nn)

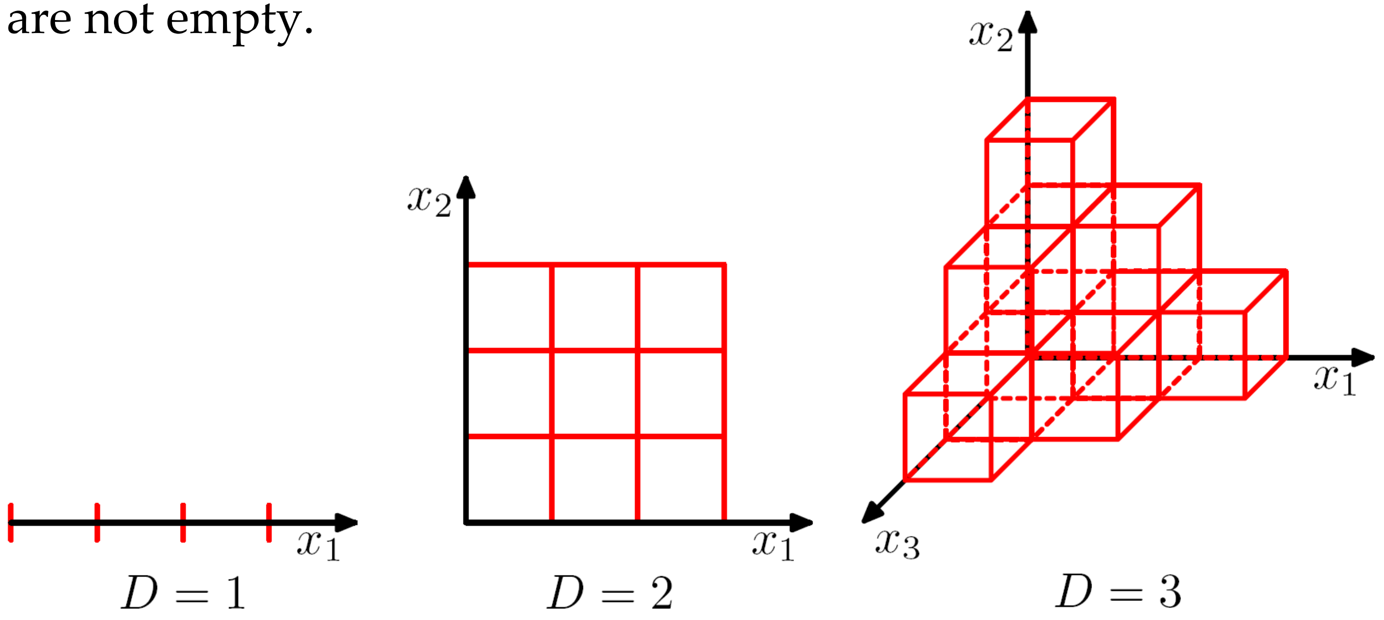
classification

Example: 12 dim points, 3 classes

Idea: determine the identity of the cross using nearby points from the training set.

- The number of cells grows exponentially with the dimensionality of the space.
- We need an exponentially large quantity of training data to ensure cells are not empty.





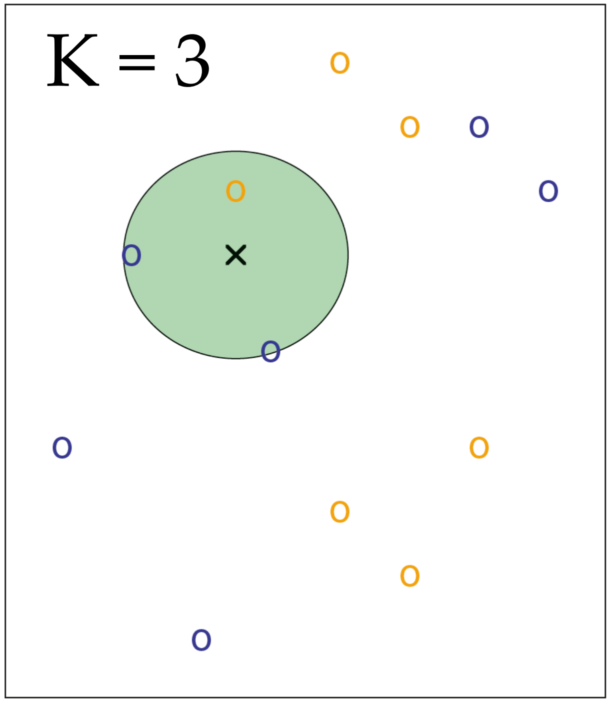
The KNN algorithm

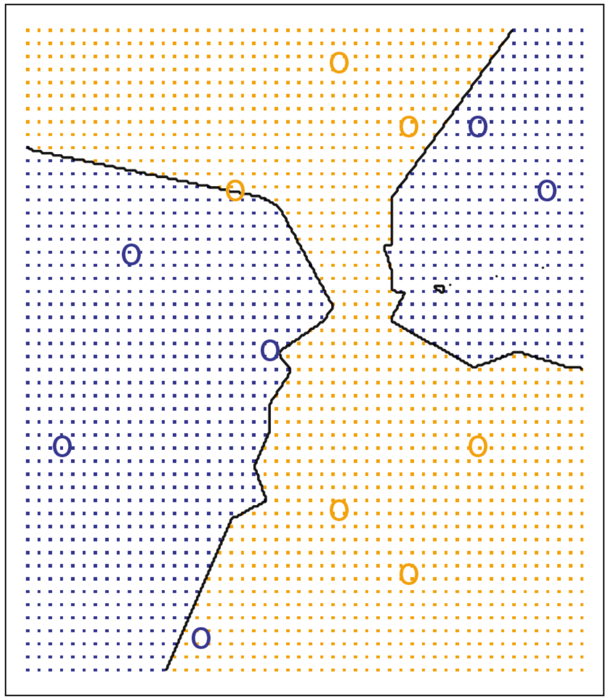
Dataset: N datapoints belonging to L classes

Choose: *K > 0*

To classify a new test observation $\mathbf{x_0}$,

- identify the neighbours *K* points in the training data that are closest to $\mathbf{x}_0$, represented by N_0.
- For each class *j*, calculate the fraction F_j of points in N_0 whose response values equal *j*
- Classify the test observation $\mathbf{x}_0$ to the class with the largest F_j





Questions

1. Is there an **underlying probabilistic model**? Can you explain in terms of probabilities what we are doing when we are applying this algorithm?

2. Which parameter controls the **complexity** of the model? Can the model overfit? How?

Let’s take a step back…

Density estimation

Density estimation

We want to obtain a model p($\mathbf{x}$) of a random variable $\mathbf{x}$, given a finite set $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of observations.

Parametric Methods:

1. we specify a form of the distribution. This is controlled by a few parameters.
2. We learn the parameters, e.g. maximizing the likelihood

Nonparametric Methods:

1. Avoids setting specific distributions.
2. There are parameters, but they only control the complexity of the model

Why is it important? How would I use it?

For example, in **classification problems**.

- For each of class, build p($\mathbf{x}$|C_k).
- Then assign a new input $\mathbf{x}_{new}$ to the class with the highest posterior using Bayes theorem:

$$p(C_k|\mathbf{x}_{new}) = \frac{p(\mathbf{x}_{new}|C_k)\, p(C_k)}{p(\mathbf{x}_{new})}$$

Also, if the model has different components, these can reveal structure in the data and can be interpreted as "clusters", we will see this when we look at unsupervised techniques…

Why is it important?
How would I use it?

Another example, in **outlier detection problems**.

Problem: we want to identify points which lie outside the "normal" behaviour of a system.

An important issue is that often we have many datapoints relating to the "normal" behaviour of the system and only very few related to the "abnormal" behaviour that we wish to identify.

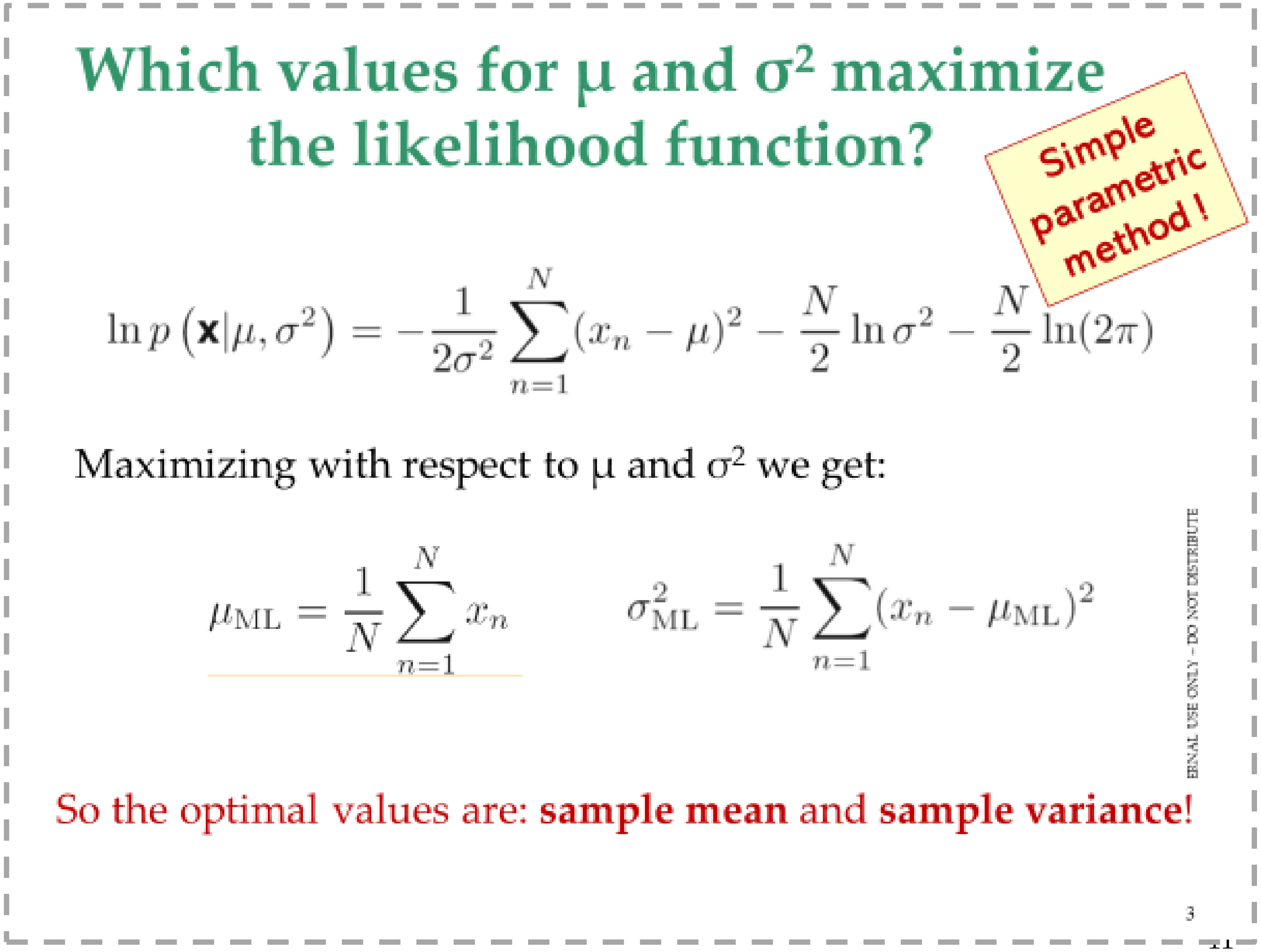
- build p($\mathbf{x}$) using data from the "normal" system behaviour – this is your model
- when a new input $\mathbf{x}_{\text{new}}$ calculate its probability under the model. If it is *too low*, then identify the point as "abnormal" (outlier).

Parametric methods

- Single Gaussian (seen earlier)
- Mixture of Gaussians

We already saw one parametric model…

We saw how to fit a Gaussian distribution to the data…



Which values for μ and σ² maximize the likelihood function?

Simple parametric method !

$$\ln p\left(\mathbf{x}|\mu,\sigma^2\right) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

Maximizing with respect to μ and σ² we get:

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}x_n \qquad \sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{\mathrm{ML}})^2$$

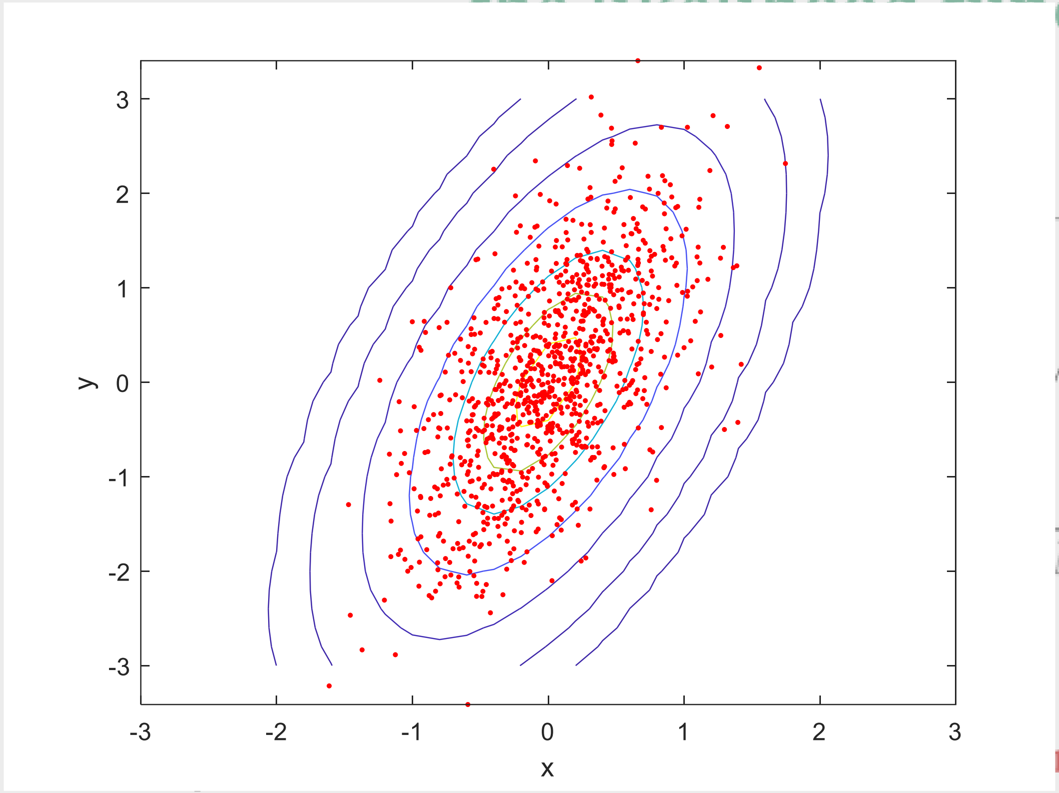
So the optimal values are: **sample mean** and **sample variance**!

We already saw one parametric model…

We saw how to fit a Gaussian distribution to the data…

Which values for μ and σ² maximize the likelihood function?

Simple parametric method !



$$- \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi)$$

we get:

$$\frac{1}{N} \sum_{n=1}^{N} (x_n - \mu_{\text{ML}})^2$$

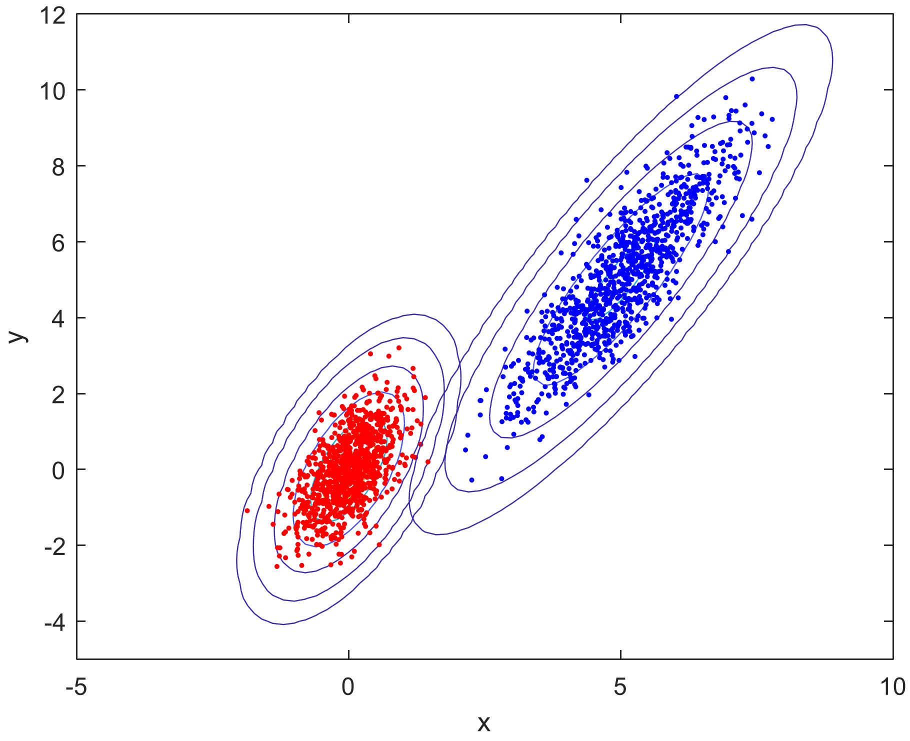
n and sample variance!

For outlier detection:

- For a new input $\mathbf{x}_{new}$ calculate its probability under the model.

For classification:

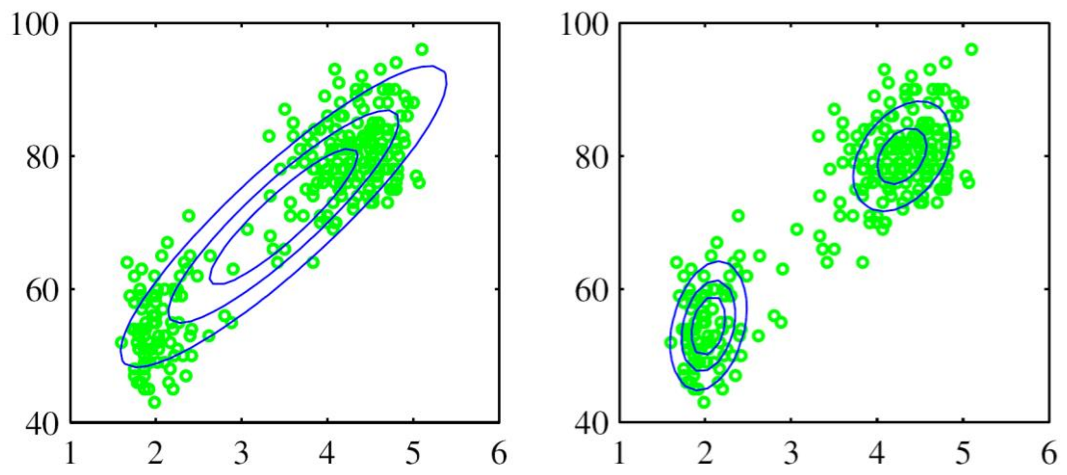
- For each of class, build $p(\mathbf{x}|C_k)$.
- Then assign a new input $\mathbf{x}_{new}$ to the class with the highest posterior using Bayes theorem

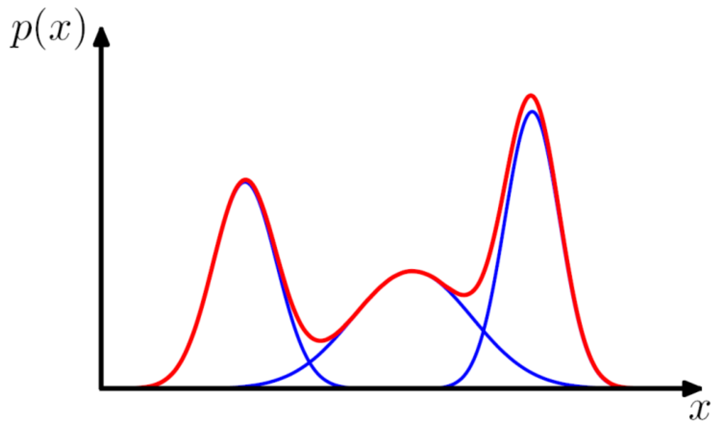


A.PACCANAR

Mixture of Gaussians

A simple Gaussian distribution is limited when modelling real data sets.

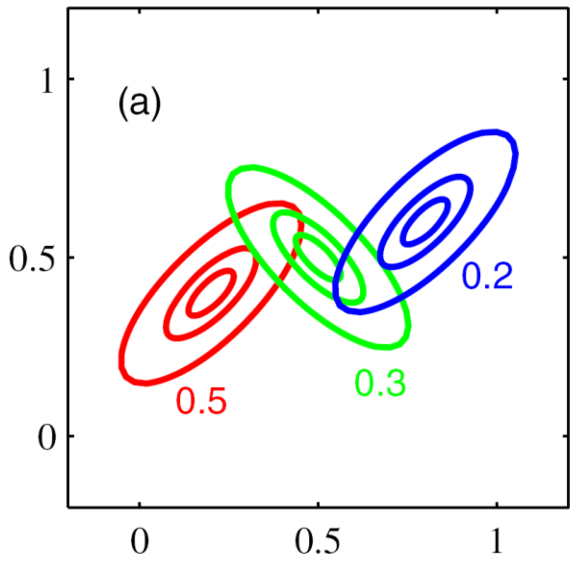




By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination continuous densities can be approximated to arbitrary accuracy.

The model

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



Each Gaussian density $N(\mathbf{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$

- is a *component* of the mixture
- has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$.

π_k are the mixing coefficients $\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$

Also, $p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k)$, so we can view:

- $\pi_k = p(k)$ as the prior probability of picking a component
- $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = p(\mathbf{x}|k)$ as the probability of **x** conditioned on k

Think of it as a *generative model*: to generate a point, pick a Gaussian k with probability π_k and then generate a point according to $N(\mathbf{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$

Posterior probabilities $p(k|\mathbf{x})$, also known as *responsibilities:*

$$\begin{aligned}\gamma_k(\mathbf{x}) &\equiv p(k|\mathbf{x}) \\ &= \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} \\ &= \frac{\pi_k\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_l \pi_l\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)}\end{aligned}$$

The best way to maximize the likelihood function with respect to the parameters is using the EM algorithm…

The Expectation Maximization (EM) algorithm for Mixtures of Gaussians

It alternates between two steps:

1. E-step: Compute the posterior probabilities given our current model - i.e. how much do we believe each Gaussian generates each datapoint.

2. M-step: Assuming that the data was really generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

   where

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}).$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right\}$$

   and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

From C. Bishop book, chapter 9

A few considerations on EM

- Iterative algorithm
- The number of Gaussians (K) controls the complexity of the model
- No learning rate
- Can get stuck in local minima

… we will understand these better later on when we look at gradient descent techniques…

Nonparametric methods

- Histograms
- K-nearest neighbour (☺ !)
- Kernel Density Estimation

Histograms

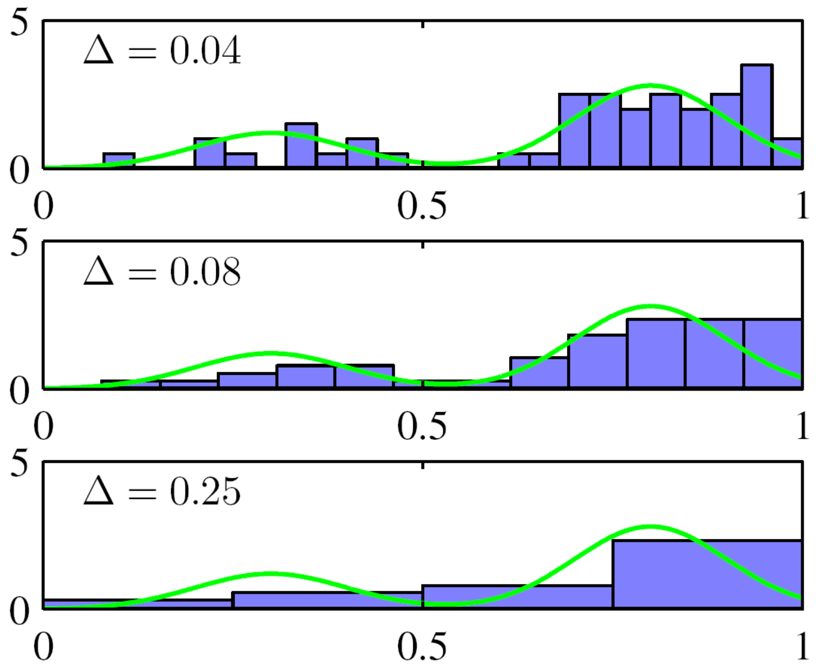
We consider single continuous variable x

- partition x into distinct bins of width Δi
- count the number n_i of observations of x falling in bin i.
- divide by the total number N of observations and by the width Δ_i of the bins

Probability values are: $p_i = \frac{n_i}{N\Delta_i}$

and we have $\int p(x)\,\mathrm{d}x = 1$

- Density $p(x)$ that is constant over the width of each bin.



Δ controls the complexity of the model !

Note: Bins can be have the same width $\Delta_i = \Delta$, or different width

Advantage: once the histogram has been computed, the data can be discarded (good if dataset is large)

Disavantage: scaling with dimensionality

An "approximate" calculation…

N observations are being drawn from some unknown probability density $p(\mathbf{x})$ in some D-dimensional space

→ we wish to estimate the value of $p(\mathbf{x})$

I have "simplified" the math…

small region *R*, of volume *V*

K = number of points in R

Probability P of being within *R*: $P \simeq \frac{K}{N}$

We also have:

$$P \simeq p(\mathbf{x})V$$

Hence:

$$\boxed{p(\mathbf{x}) = \frac{K}{NV}}$$

This eq. gives us 2 methods, depending on whether we fix K or V…

Case 1: we fix K

we obtain the K-nearest-neighbour algorithm for density estimation…

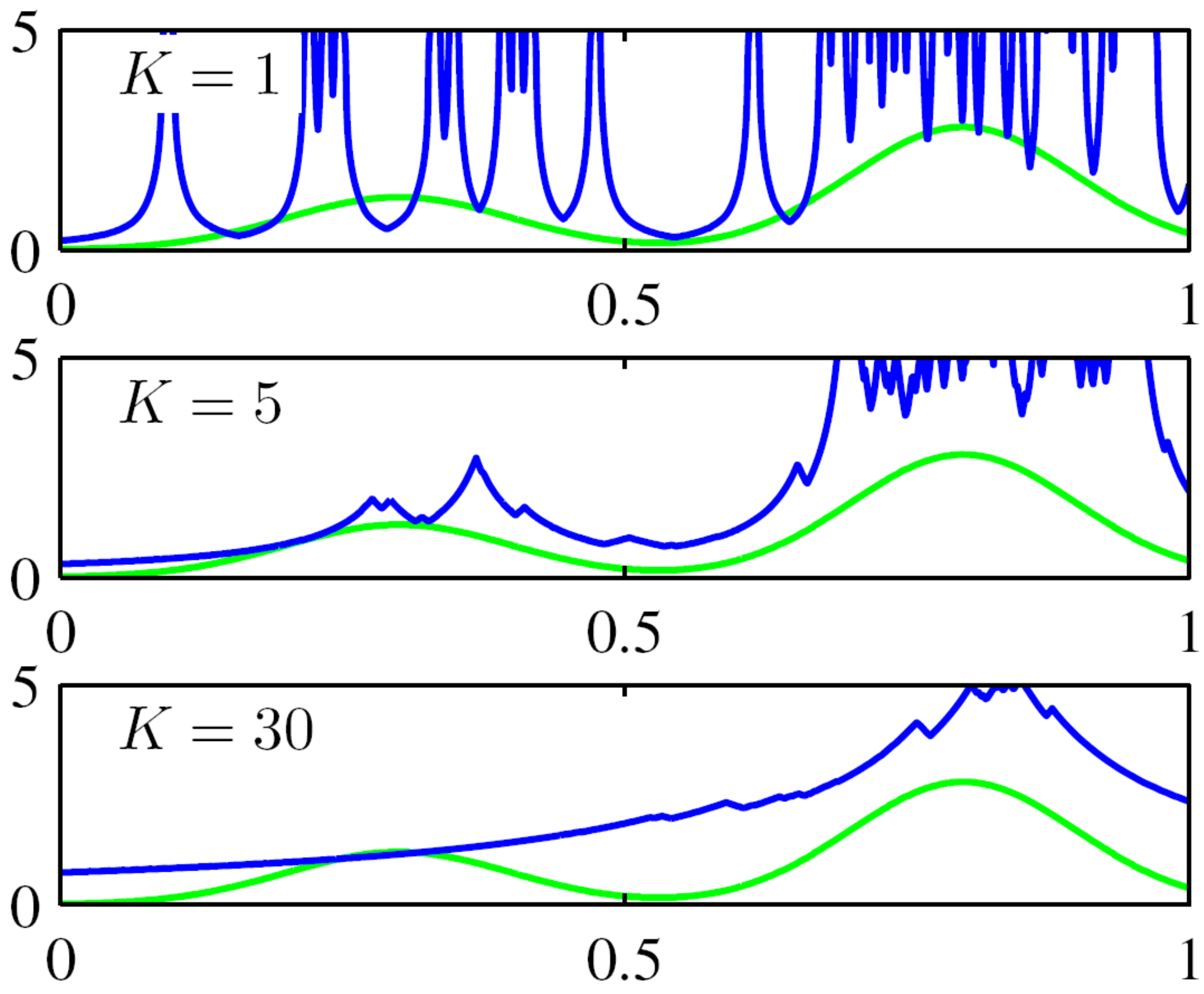
$$\boxed{p(\mathbf{x}) = \frac{K}{NV}}$$

We fix the value of *K* and use the data to find an appropriate value for *V*

- consider a small sphere centred on the point **x** where we want to estimate the density *p*(**x**)
- allow the radius of the sphere to grow until it contains *K* points.
- the estimate of the density *p*(**x**) is then given by

$$p(\mathbf{x}) = \frac{K}{NV}$$

with *V* set to the volume of the resulting sphere



Note that the parameter K governs the degree of smoothing…

So, what is K-nearest-neighbour *classification* doing ?

Data set of N points with N_k points in class C_k with $\sum_k N_k = N$

To classify a new point **x**, we draw a sphere centred on **x** containing K points.

V volume of the sphere.

$$p(\mathbf{x}) = \frac{K}{NV} \qquad p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V} \qquad p(\mathcal{C}_k) = \frac{N_k}{N}$$

We have all the pieces to apply Bayes theorem:

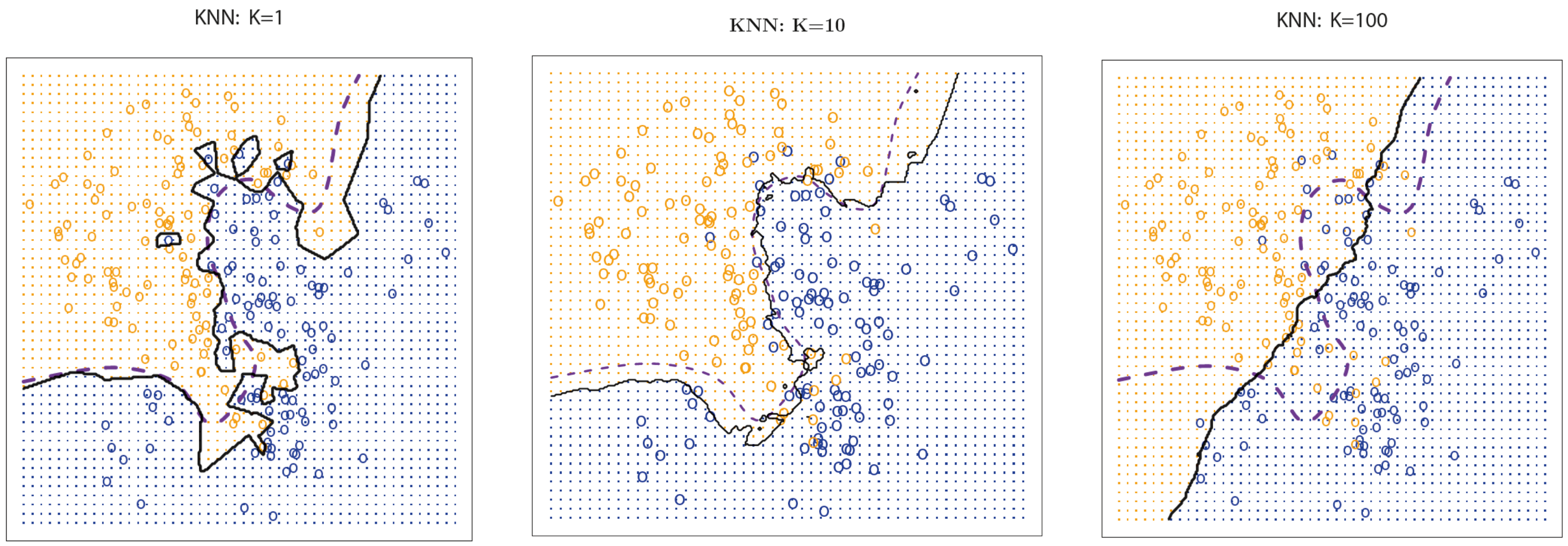
$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}$$

1. Is there an underlying probabilistic model? Can you explain in terms of probabilities what we are doing when we are applying this algorithm?

K-nearest-neighbour is minimizing the probability of misclassification, by assigning point **x** to the class having the largest posterior probability.

Note: it is estimating $p(\mathbf{x})$ in a very simple, nonparametric way.

2. Which parameter controls the complexity of the model? Can the model overfit? How?



When $K = 1$, the decision boundary is overly flexible. As K grows, the method becomes less flexible and produces a decision boundary that is close to linear.

Case 2: we fix V

we obtain the Kernel density estimation…

$$\boxed{p(\mathbf{x}) = \frac{K}{NV}}$$

- Region *R:* a small hypercube centred on **x**
- to count the number *K* of points falling in *R*, define:

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leqslant 1/2, \qquad i = 1, \ldots, D \\ 0, & \text{otherwise} \end{cases}$$

(a unit cube centred on the origin).

$$k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right) = \begin{cases} 1 & \text{if the data point } \mathbf{x}_n \text{ lies inside a cube of side } h \text{ centred on } \mathbf{x} \\ 0 & \text{otherwise} \end{cases}$$

- total number of data points lying inside this cube:

$$K = \sum_{n=1}^{N} k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right)$$

$$\boxed{p(\mathbf{x}) = \frac{K}{NV}}$$

- Volume of a hypercube of side h in D dimensions is $V = h^D$
- The estimated density at **x** is:

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right)$$

(sum over N cubes centred on the N data points $\mathbf{x}_n$).

- $k(\mathbf{u})$ is a *kernel function,* and in this context is also called a *Parzen window.*
- The density model is called a kernel density estimator, or *Parzen* estimator

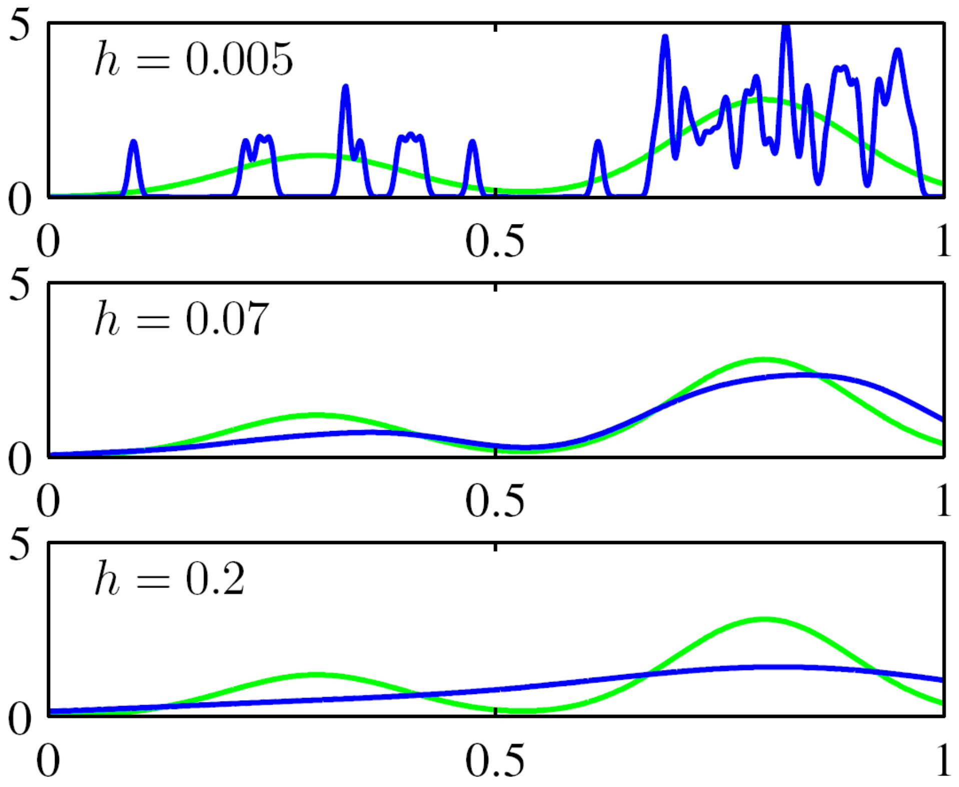
- We can use a smoother kernel function
- For example, using a Gaussian:

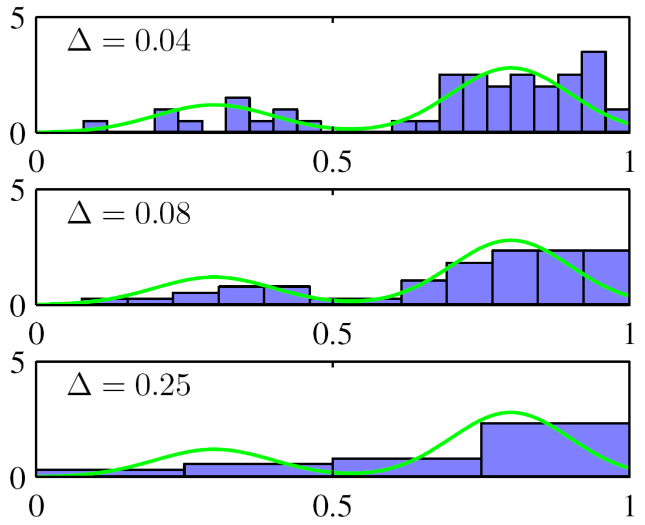
$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{1/2}} \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_n\|^2}{2h^2}\right\}$$

The density model is obtained by:

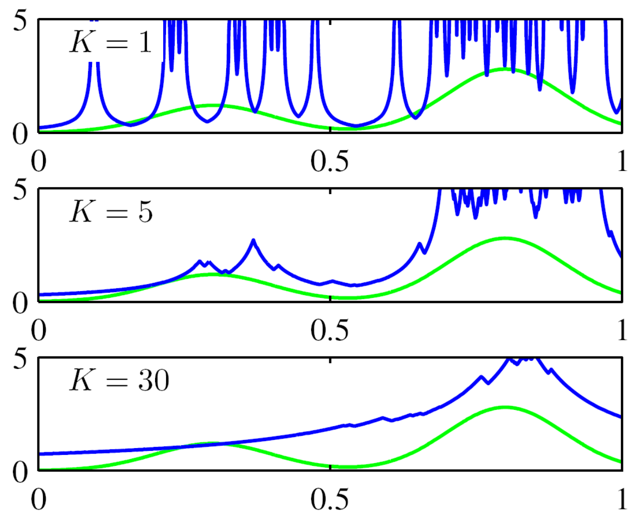
- placing a Gaussian over each data point
- adding up the contributions over the whole data set,
- dividing by *N* to normalize.

h *controls the complexity of the model !*

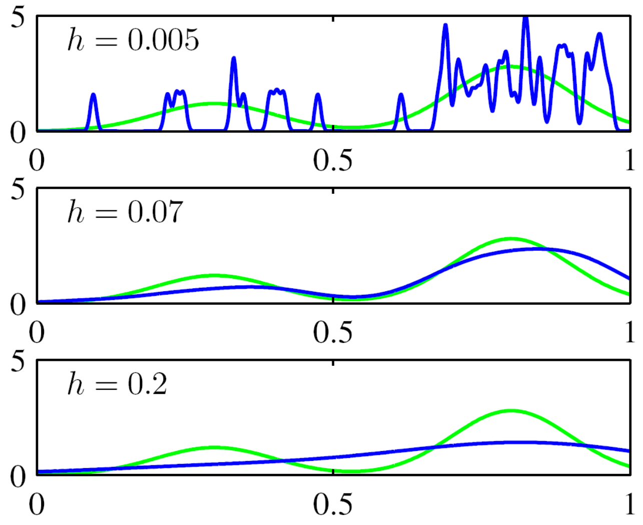




histogram



KNN



Gaussian kernel

Naïve Bayes: a nonparametric method for classification

Two class classification, of a vector $\mathbf{x} = (x_1, \ldots x_n)$ into C_0, C_1

$$P(C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_k) \cdot P(C_k)}{p(\mathbf{x})}$$

***Idea**: make the simplifying assumption that the attribute values are conditionally independent given the target values*

$$\begin{aligned} P(\mathrm{a,b,c}) &= P(\mathrm{a} \mid \mathrm{b,c})P(\mathrm{b,c}) \\ P(\mathrm{b,c}) &= P(\mathrm{b} \mid \mathrm{c})P(\mathrm{c}) \\ P(\mathrm{a,b,c}) &= P(\mathrm{a} \mid \mathrm{b,c})P(\mathrm{b} \mid \mathrm{c})P(\mathrm{c}) \end{aligned}$$

$$\begin{aligned} p(\mathbf{x} \mid C_i) &= p(x_1, x_2, \ldots, x_n \mid C_i) \\ &\approx p(x_1 \mid C_i)p(x_2 \mid C_i) \ldots p(x_n \mid C_i) \\ &= \prod_j p(x_j \mid C_i). \end{aligned}$$

Learning in Naive Bayes consists of estimating the various $P(C_i)$ and various $p(x_j \mid C_i)$ using the equation above based on their frequencies over the training data.

Important

- $p(x_i|C_k)$ can be modelled by a parametric distribution (e.g. Gaussian)
- It works with **missing data** !!!
- Results are easily **interpretable** !!!
- For many problems, it works extremely well ☺