

L7: Kernel density estimation

Non-parametric density estimation

Histograms

Parzen windows

Smooth kernels

Product kernel density estimation

The naïve Bayes classifier

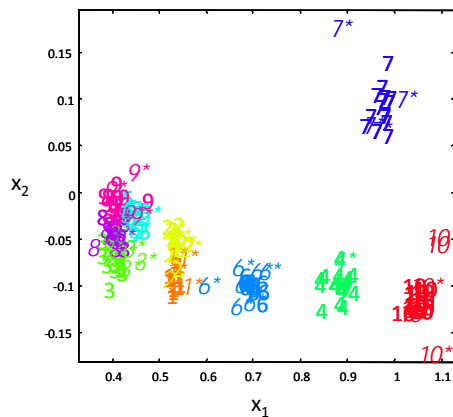
Non-parametric density estimation

In the previous two lectures we have assumed that either

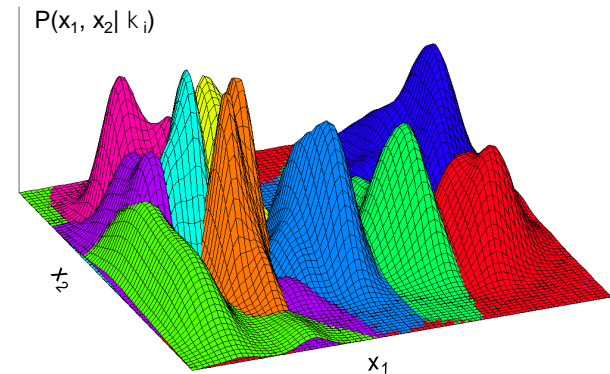
- The likelihoods were known (LRT), or
- At least their parametric form was known (parameter estimation)

The methods that will be presented in the next two lectures do not afford such luxuries

- Instead, they attempt to estimate the density directly from the data without assuming a particular form for the underlying distribution
- Sounds challenging? You bet!



non-parametric density estimation



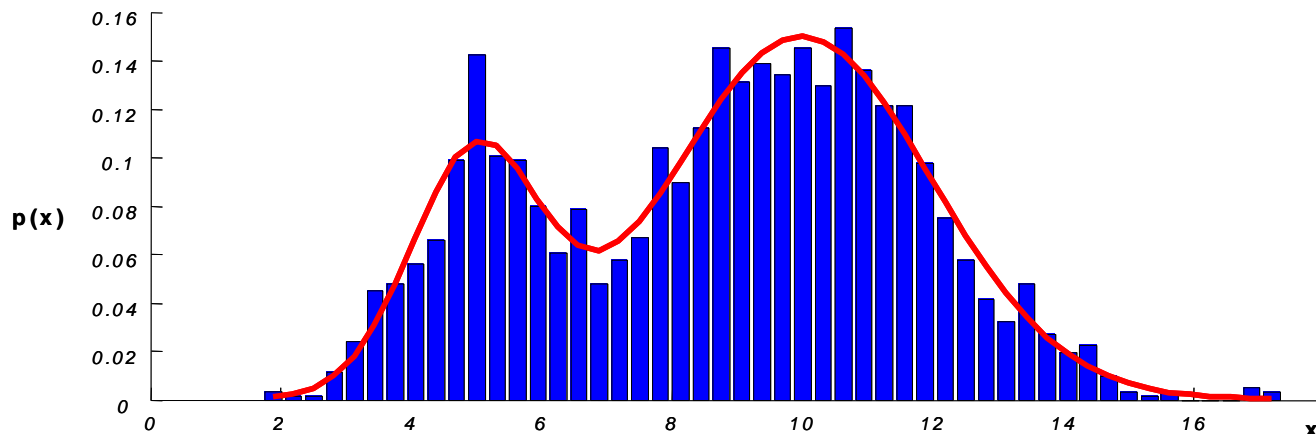
The histogram

The simplest form of non-parametric DE is the histogram

- Divide the sample space into a number of bins and approximate the density at the center of each bin by the fraction of points in the training data that fall into the corresponding bin

$$f(x) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{[x_i, x_i + \Delta x)}(x)$$

- The histogram requires two “parameters” to be defined: bin width and starting position of the first bin



The histogram is a very simple form of density estimation, but has several drawbacks

- The density estimate depends on the starting position of the bins
 - “ For multivariate data, the density estimate is also affected by the orientation of the bins
- The discontinuities of the estimate are not due to the underlying density; they are only an artifact of the chosen bin locations
 - “ These discontinuities make it very difficult (to the naïve analyst) to grasp the structure of the data
- A much more serious problem is the curse of dimensionality, since the number of bins grows exponentially with the number of dimensions
 - “ In high dimensions we would require a very large number of examples or else most of the bins would be empty
- These issues make the histogram unsuitable for most practical applications except for quick visualizations in one or two dimensions
- Therefore, we will not spend more time looking at the histogram

Non-parametric DE, general formulation

Let us return to the basic definition of probability to get a solid idea of what we are trying to accomplish

- The probability that a vector \mathbf{x} , drawn from a distribution $p(\mathbf{x})$, will fall in a given region R of the sample space is

$$\int_R p(\mathbf{x}) d\mathbf{x}$$

- Suppose now that N vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ are drawn from the distribution; the probability that k of these N vectors fall in R is given by the binomial distribution

$$\binom{N}{k} p^k (1-p)^{N-k}$$

- It can be shown (from the properties of the binomial p.m.f.) that the mean and variance of the ratio $\frac{k}{N}$ are

$$\left[\frac{k}{N} \right] = p \quad \text{and} \quad \left[\frac{k}{N} \right]^2 = \frac{p(1-p)}{N}$$

- Therefore, as $N \rightarrow \infty$ the distribution becomes sharper (the variance gets smaller), so we can expect that a good estimate of the probability p can be obtained from the mean fraction of the points that fall within

—

[Bishop, 1995]

- On the other hand, if we assume that ρ is so small that ρ does not vary appreciably within it, then

$$\frac{1}{V} \int_V \rho(\mathbf{x}) dV \approx \rho(\mathbf{x}_i)$$

“ where V is the volume enclosed by region

- Merging with the previous result we obtain

$$\frac{1}{V} \int_V \rho(\mathbf{x}) dV \approx \rho(\mathbf{x}_i) \quad \text{---}$$

- This estimate becomes more accurate as we increase the number of sample points N and shrink the volume

In practice the total number of examples is fixed

- To improve the accuracy of the estimate $\rho(\mathbf{x}_i)$ we could let V approach zero but then N would become so small that it would enclose no examples
- This means that, in practice, we will have to find a compromise for
 - “ Large enough to include enough examples within
 - “ Small enough to support the assumption that ρ is constant within

- In conclusion, the general expression for non-parametric density estimation becomes

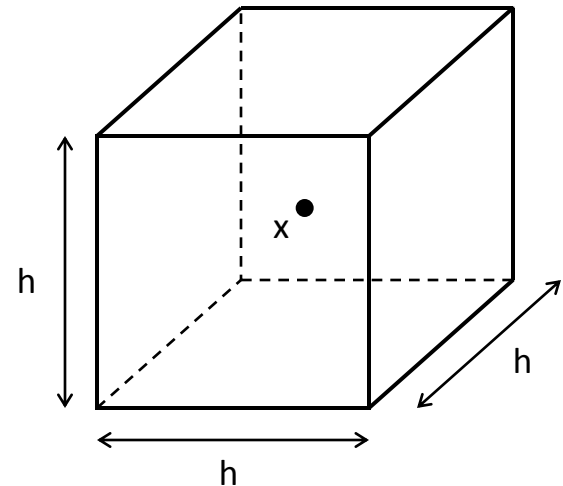
$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \quad \text{— where } K(x) = \begin{cases} \frac{1}{2} & \text{if } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- When applying this result to practical density estimation problems, two basic approaches can be adopted
 - “ We can fix h and determine n from the data. This leads to kernel density estimation (KDE), the subject of this lecture
 - “ We can fix n and determine h from the data. This gives rise to the k-nearest-neighbor (kNN) approach, which we cover in the next lecture
- It can be shown that both kNN and KDE converge to the true probability density as $n \rightarrow \infty$, provided that h shrinks with n , and that nh grows with n appropriately

Parzen windows

Problem formulation

- Assume that the region that encloses the examples is a hypercube with sides of length h centered at x
 - Then its volume is given by $V = h^d$, where d is the number of dimensions
- To find the number of examples that fall within this region we define a kernel function



$$K(x) = \begin{cases} 1 & \text{if } x \text{ is inside a hypercube of side } h \text{ centered on } x \\ 0 & \text{otherwise} \end{cases}$$

- This kernel, which corresponds to a unit hypercube centered at the origin, is known as a Parzen window or the naïve estimator
- The quantity $K(x)$ is then equal to unity if x is inside a hypercube of side h centered on x , and zero otherwise

[Bishop, 1995]

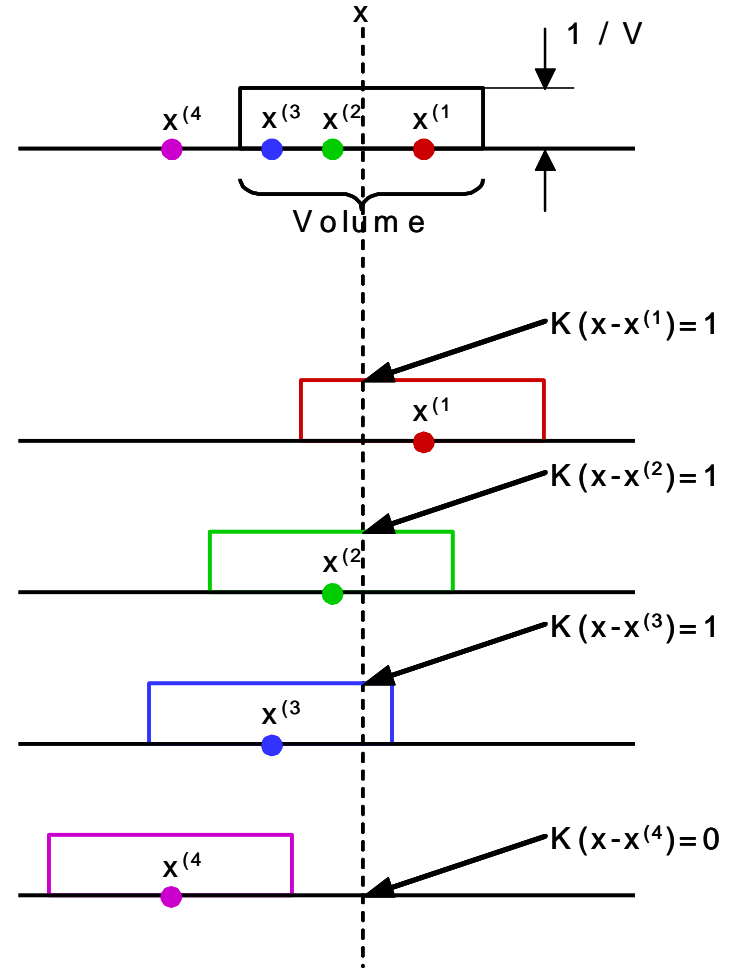
- The total number of points inside the hypercube is then

$$\left(\frac{\text{Volume}}{V} \right)$$

Substituting back into the expression for the density estimate

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{[x^{(i)}, x^{(i+1)})}(x)$$

- Notice how the Parzen window estimate resembles the histogram, with the exception that the bin locations are determined by the data



- To understand the role of the kernel function we compute the expectation of the estimate $\hat{f}(x)$

$$\dots \dots \dots \int \left[\frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \right] f(x) dx$$

“ where we have assumed that vectors x_i are drawn independently from the true density $f(x)$

- We can see that the expectation of $\hat{f}(x)$ is a convolution of the true density $f(x)$ with the kernel function $K(\cdot)$

“ Thus, the kernel width h plays the role of a smoothing parameter: the wider h is, the smoother the estimate $\hat{f}(x)$

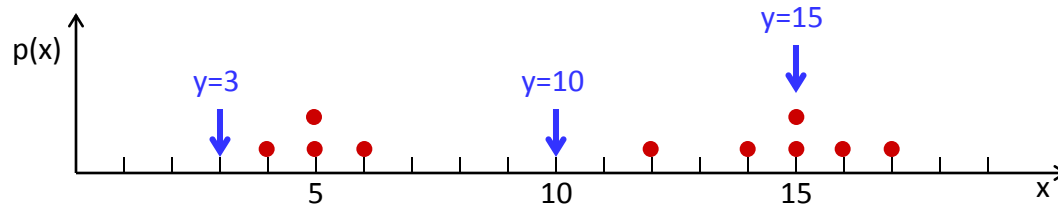
- For $h \rightarrow 0$, the kernel approaches a Dirac delta function and $\hat{f}(x)$ approaches the true density $f(x)$

“ However, in practice we have a finite number of points, so h cannot be made arbitrarily small, since the density estimate $\hat{f}(x)$ would then degenerate to a set of impulses located at the training data points

Exercise

- Given dataset $\{ \dots \}$, use Parzen windows to estimate the density at \dots ; use
- Solution

“ Let’s first draw the dataset to get an idea of the data



“ Let’s now estimate

$$\left(\quad \right) \text{ --- } \left(\text{---} \right) \text{ --- } \left[\left(\text{---} \right) \left(\text{---} \right) \left(\text{---} \right) \right]$$

“ Similarly

$$\left(\quad \right) \text{ --- } [\quad]$$

$$\left(\quad \right) \text{ --- } [\quad]$$

Smooth kernels

The Parzen window has several drawbacks

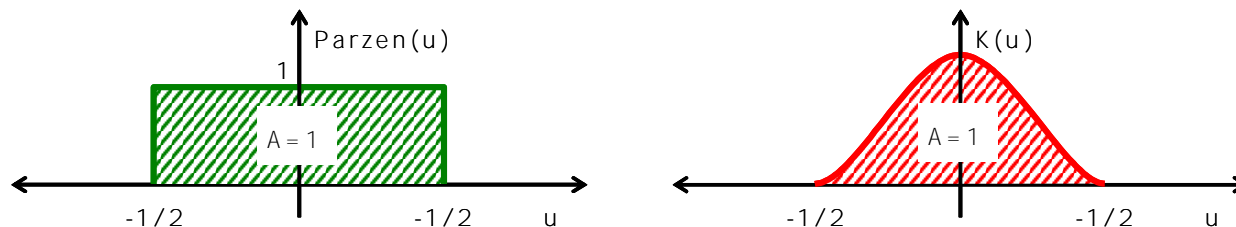
- It yields density estimates that have discontinuities
- It weights equally all points, regardless of their distance to the estimation point

For these reasons, the Parzen window is commonly replaced with a smooth kernel function

()

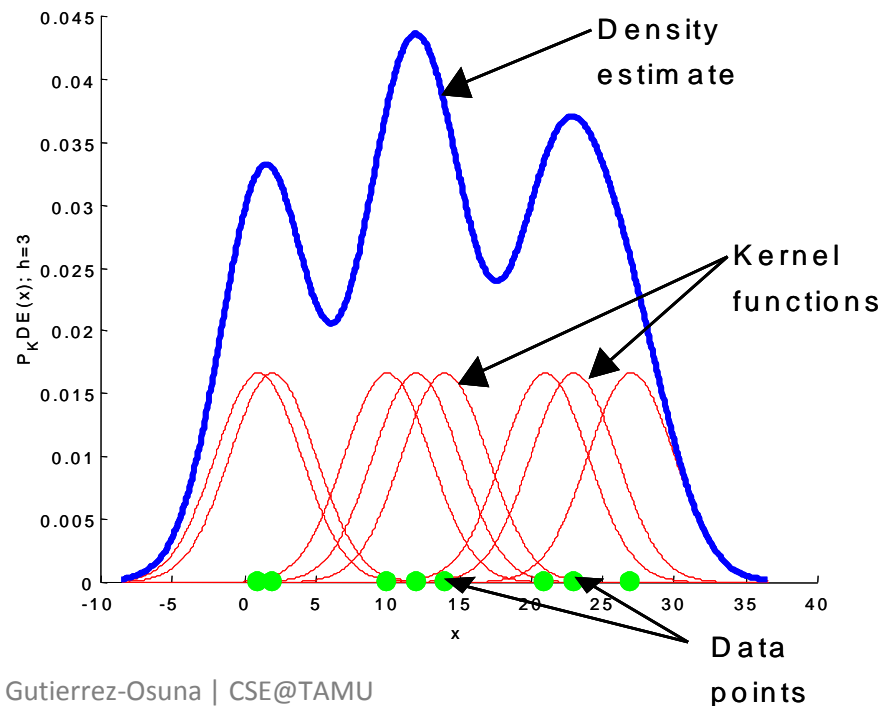
- Usually, but not always, will be a radially symmetric and unimodal pdf, such as the Gaussian () () -
- Which leads to the density estimate

() — (—)



Interpretation

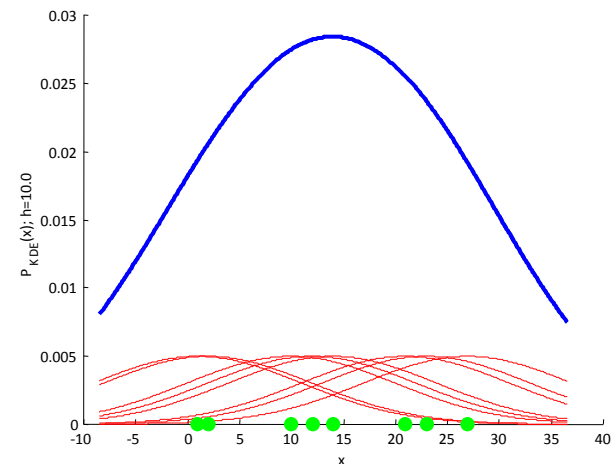
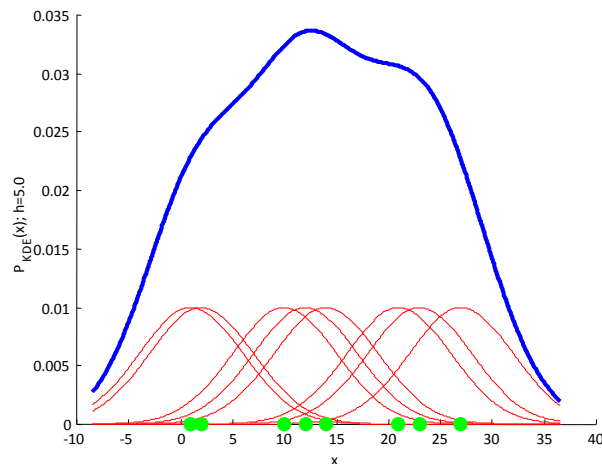
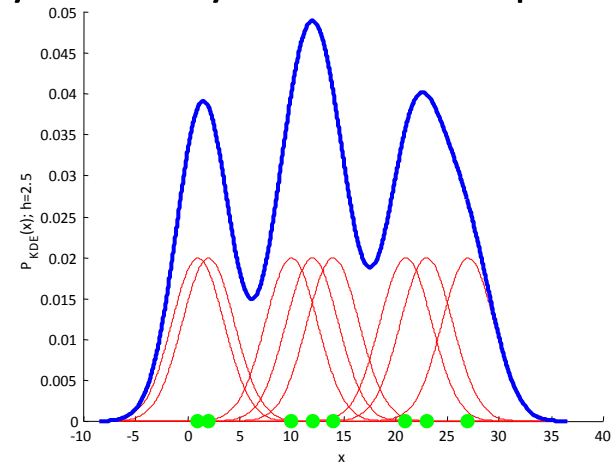
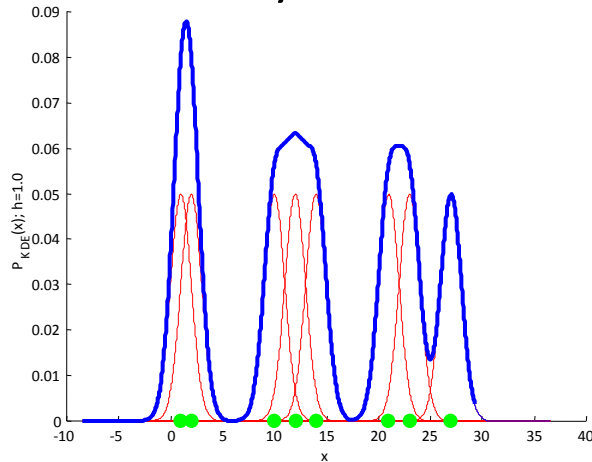
- Just as the Parzen window estimate can be seen as a sum of boxes centered at the data, the smooth kernel estimate is a sum of “bumps”
- The kernel function determines the shape of the bumps
- The parameter h , also called the smoothing parameter or bandwidth, determines their width



Bandwidth selection

The problem of choosing h is crucial in density estimation

- A large h will over-smooth the DE and mask the structure of the data
- A small h will yield a DE that is spiky and very hard to interpret



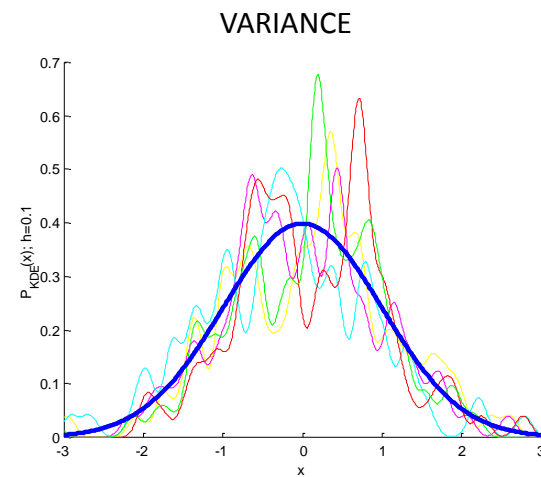
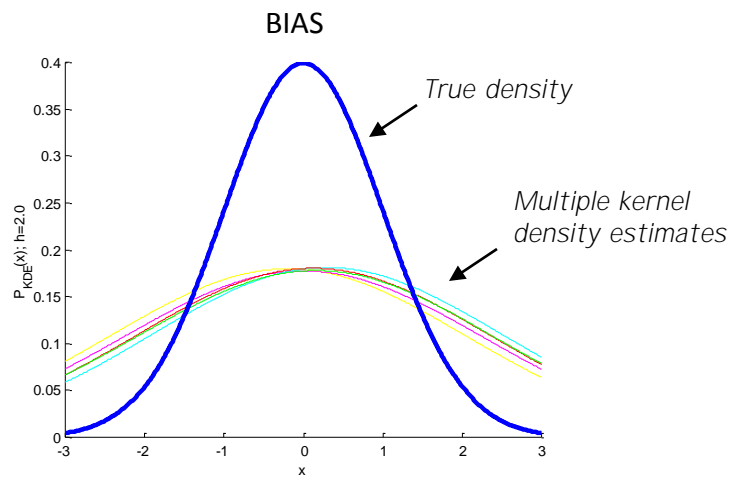
- We would like to find a value of $\hat{\theta}$ that minimizes the error between the estimated density and the true density

“ A natural measure is the MSE at the estimation point $\hat{\theta}$, defined by

$$[(\hat{\theta} - \theta)^2] = \underbrace{[\hat{\theta} - \theta]^2}_{\text{Bias}} + \underbrace{[\theta - \theta]^2}_{\text{Variance}}$$

- This expression is an example of the bias-variance tradeoff that we saw in an earlier lecture: the bias can be reduced at the expense of the variance, and vice versa
 - “ The bias of an estimate is the systematic error incurred in the estimation
 - “ The variance of an estimate is the random error incurred in the estimation

- The bias-variance dilemma applied to bandwidth selection simply means that
 - “ A large bandwidth will reduce the differences among the estimates of $\hat{P}_{KDE}(x)$ for different data sets (the variance), but it will increase the bias of $\hat{P}_{KDE}(x)$ with respect to the true density
 - “ A small bandwidth will reduce the bias of $\hat{P}_{KDE}(x)$, at the expense of a larger variance in the estimates $\hat{P}_{KDE}(x)$



Bandwidth selection methods, univariate case

Subjective choice

- The natural way for choosing \hat{f}_h is to plot out several curves and choose the estimate that best matches one's prior (subjective) ideas
- However, this method is not practical in pattern recognition since we typically have high-dimensional data

Reference to a standard distribution

- Assume a standard density function and find the value of the bandwidth that minimizes the integral of the square error (MISE)

$$\hat{h}_{MISE} = \arg \min_h \int_{-\infty}^{\infty} \left(\hat{f}_h(x) - f(x) \right)^2 dx$$

- If we assume that the true distribution is Gaussian and we use a Gaussian kernel, it can be shown that the optimal value of \hat{h}_{MISE} is

“ where $\hat{\sigma}$ is the sample standard deviation and n is the number of training examples

- Better results can be obtained by
 - “ Using a robust measure of the spread instead of the sample variance, and
 - “ Reducing the coefficient c to better cope with multimodal densities
 - “ The optimal bandwidth then becomes

where $c = \left(\frac{4.69}{n} \right)^{1/5}$

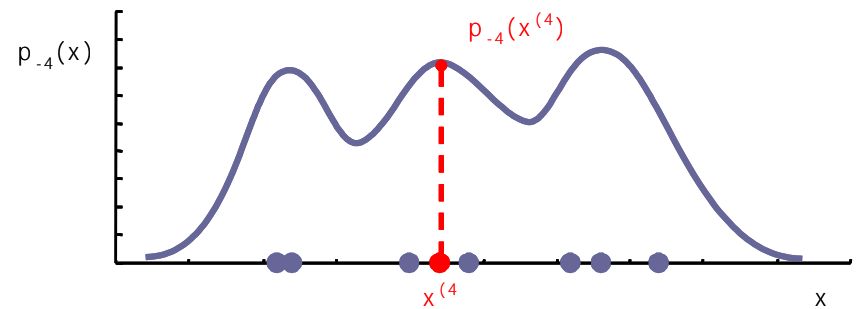
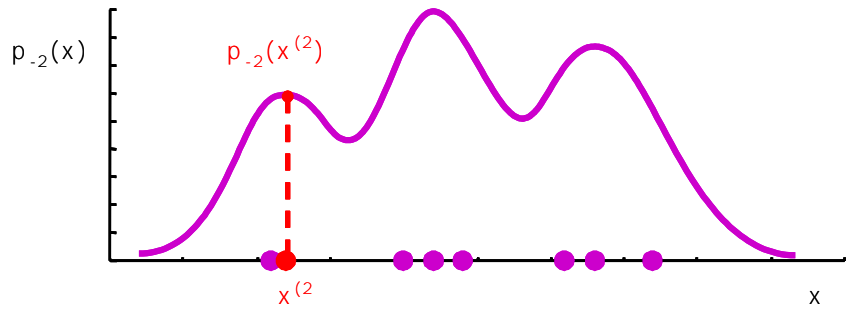
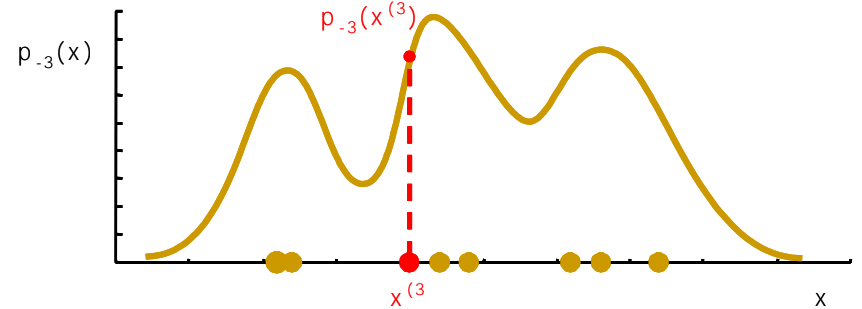
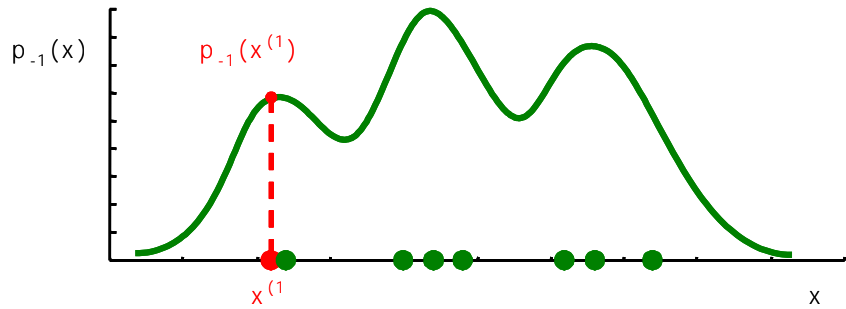
- IQR is the interquartile range, a robust estimate of the spread
 - “ IQR is the difference between the 75th percentile (Q_3) and the 25th percentile (Q_1):
 - “ A percentile rank is the proportion of examples in a distribution that a specific example is greater than or equal to

Maximum likelihood cross-validation

- The ML estimate of \hat{f} is degenerate since it yields $\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^n \delta(x - x_i)$, a density estimate with Dirac delta functions at each training data point
- A practical alternative is to maximize the “pseudo-likelihood” computed using leave-one-out cross-validation

$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^n \delta(x - x_i)$$

[Silverman, 1986]



Multivariate density estimation

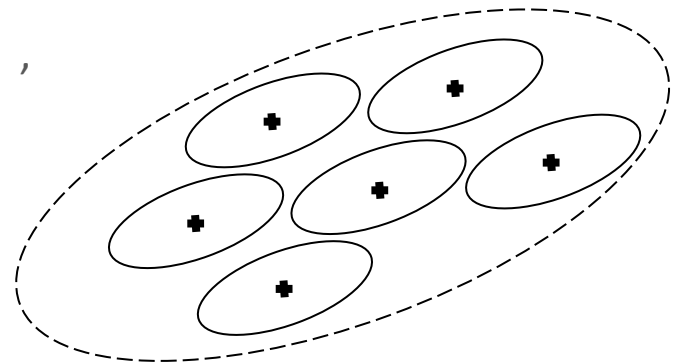
For the multivariate case, the KDE is

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \frac{1}{h^d} K\left(\frac{x - x_i}{h}\right)$$

- Notice that the bandwidth h is the same for all the axes, so this density estimate will be weight all the axis equally
- If one or several of the features has larger spread than the others, we should use a vector of smoothing parameters or even a full covariance matrix, which complicates the procedure

There are two basic alternatives to solve the scaling problem without having to use a more general KDE

- Pre-scaling each axis (normalize to unit variance, for instance)
- Pre-whitening the data (linearly transform so $\Sigma^{-1/2}(x - \mu)$, estimate the density, and then transform back [Fukunaga]
 - “ The whitening transform is $\Sigma^{-1/2}(x - \mu)$ where λ_i and v_i are the eigenvalue and eigenvector matrices of Σ
 - “ Fukunaga’s method is equivalent to using a hyper-ellipsoidal kernel



Product kernels

A good alternative for multivariate KDE is the product kernel

$$\hat{f}(\mathbf{x}) = \prod_{d=1}^D \hat{f}_d(x_d)$$

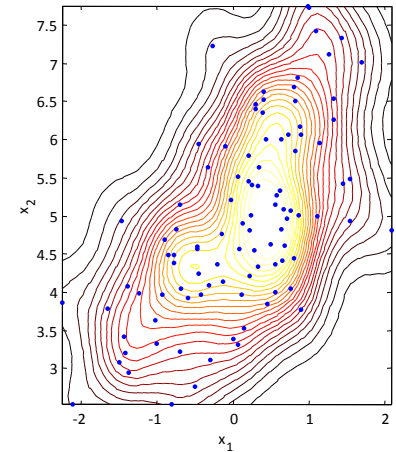
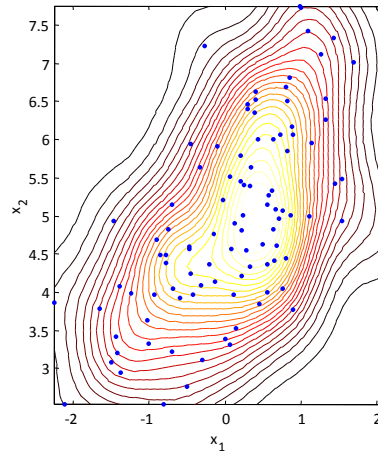
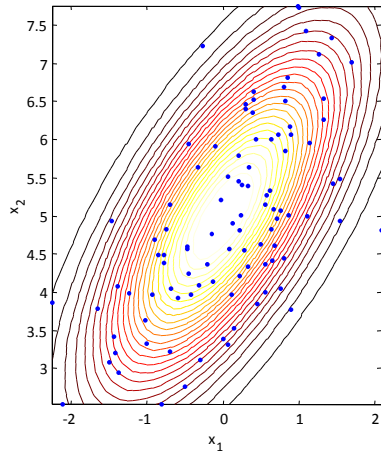
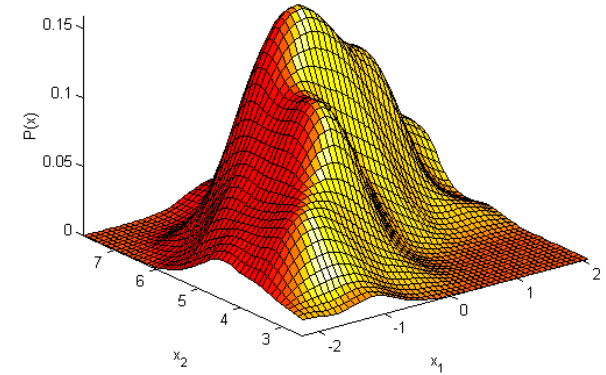
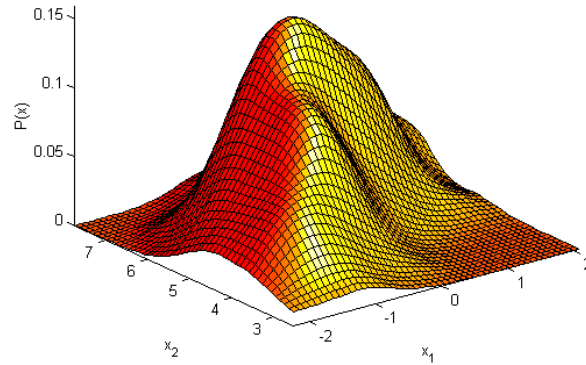
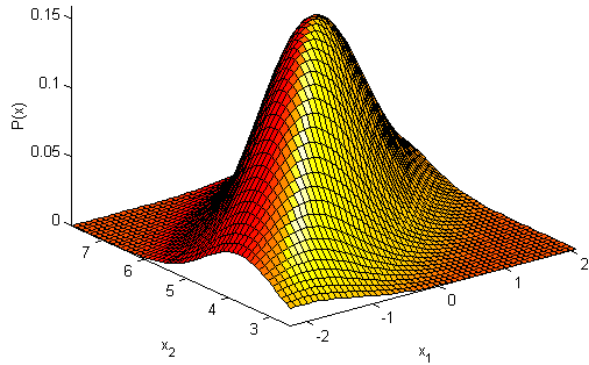
- The product kernel consists of the product of one-dimensional kernels
 - “ Typically the same kernel function is used in each dimension (), and only the bandwidths are allowed to differ
 - “ Bandwidth selection can then be performed with any of the methods presented for univariate density estimation
- Note that although () uses kernel independence, this does not imply we assume the features are independent
 - “ If we assumed feature independence, the DE would have the expression

$$\hat{f}(\mathbf{x}) = \sum_{\mathbf{x}'} \prod_{d=1}^D \hat{f}_d(x_d')$$

- “ Notice how the order of the summation and product are reversed compared to the product kernel

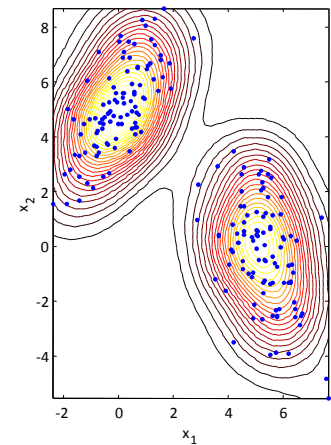
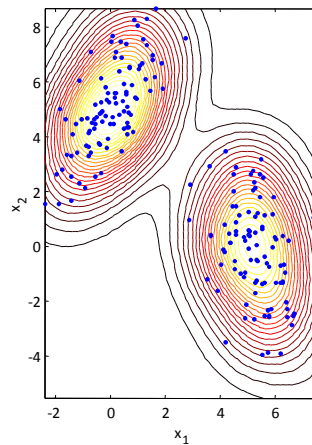
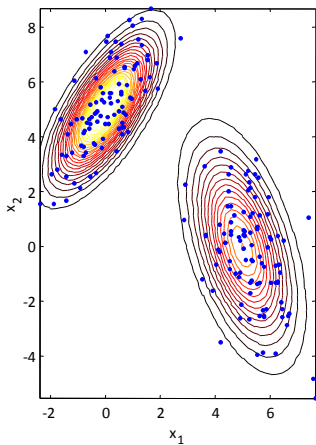
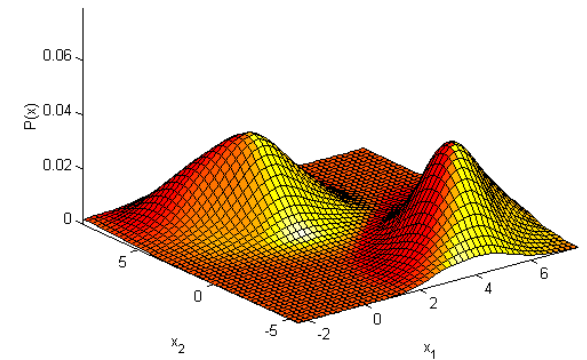
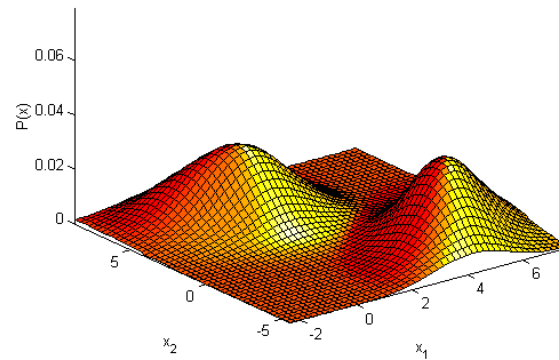
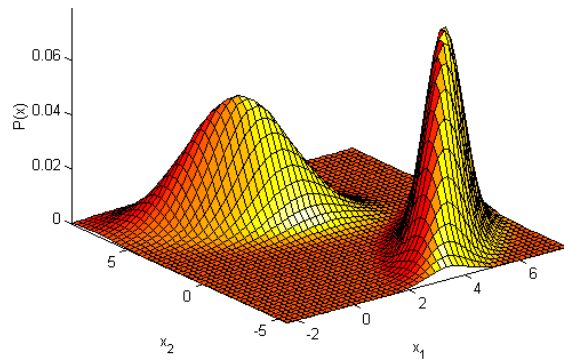
Example I

- This example shows the product KDE of a bivariate unimodal Gaussian
 - ” 100 data points were drawn from the distribution
 - ” The figures show the true density (left) and the estimates using (middle) and (right)



Example II

- This example shows the product KDE of a bivariate bimodal Gaussian
 - ” 100 data points were drawn from the distribution
 - ” The figures show the true density (left) and the estimates using (middle) and (right)



Naïve Bayes classifier

Recall that the Bayes classifier is given by the following family of DFs

$$f(x) = \frac{P(x|c)}{P(x)}$$

- Using Bayes rule, these discriminant functions can be expressed as

$$f(x) = \frac{P(c) \prod_{i=1}^n P(x_i|c)}{P(x)}$$

where $P(c)$ is our prior knowledge and $P(x)$ is obtained through DE

- Although the DE methods presented in this lecture allow us to estimate the multivariate likelihood, the curse of dimensionality makes it a very tough problem!

One highly practical simplification is the Naïve Bayes classifier

- The Naïve Bayes classifier assumes that features are class-conditionally independent

$$P(x_1, x_2, \dots, x_n | c) = \prod_{i=1}^n P(x_i | c)$$

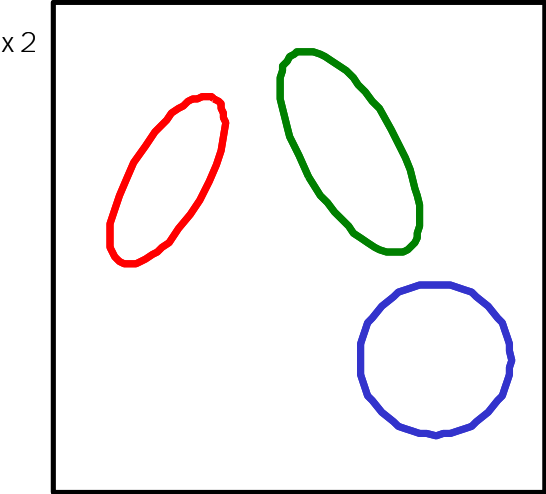
This assumption is not as rigid as assuming independent features

- Merging this expression into the DF yields the decision rule for the Naïve Bayes classifier

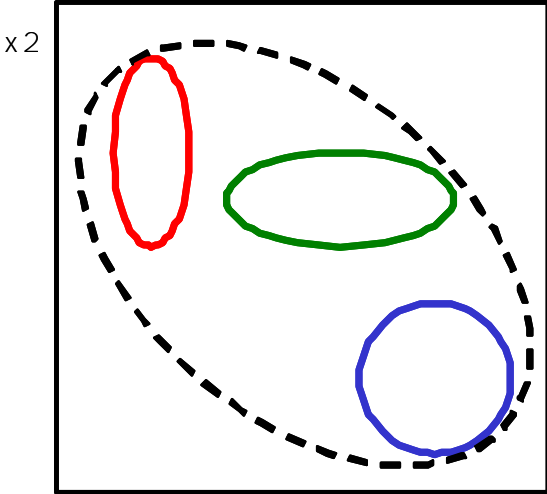
$$f(x) = \frac{P(c) \prod_{i=1}^n P(x_i | c)}{P(x)}$$

- The main advantage of the NB classifier is that we only need to compute the univariate $P(x_i | c)$, which is much easier than estimating the multivariate $P(x | c)$
- Despite its simplicity, the Naïve Bayes has been shown to have comparable performance to artificial neural networks and decision tree learning in some domains

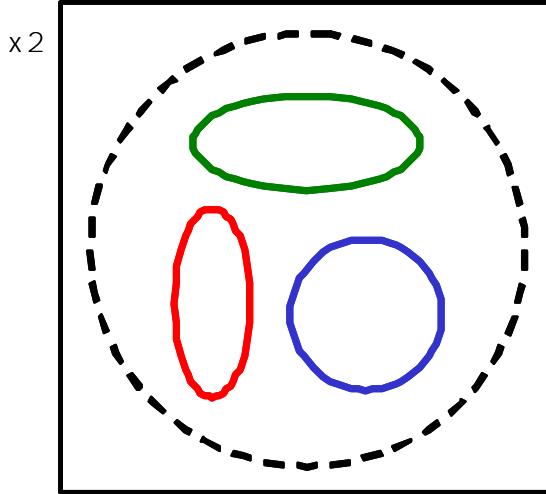
Class-conditional independence vs. independence



() ()



() ()
 () ()



() ()
 () ()