Lecture 10: Density estimation II

- Parzen windows
- Smooth kernels
- Bandwidth selection for univariate data
- Multivariate density estimation
- Product kernels
- Naïve Bayes classifier

KDE: Parzen windows (1)

- **In the previous lecture we found out that the non-parametric density estimate was**

$$P(x) \cong \frac{k}{NV} \quad \text{where} \quad \begin{cases} \text{V is the volume surrounding x} \\ \text{N is the total number of examples} \\ \text{k is the number of examples inside V} \end{cases}$$

- **Suppose that the region $\Re$ that encloses the k examples is a hypercube with sides of length h centered at the estimation point 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 <u>kernel function K(u)</u>**

$$K(u) = \begin{cases} 1 & |u_j| < 1/2 \quad j = 1,..,D \\ 0 & \text{otherwise} \end{cases}$$

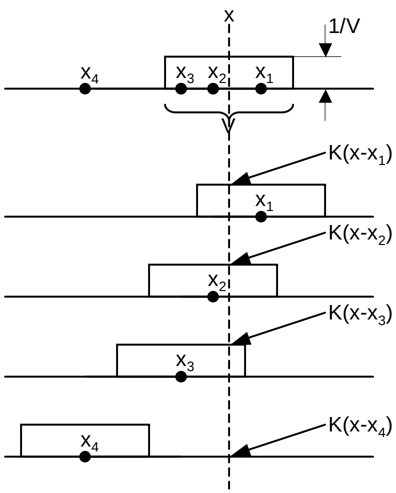
  - This kernel, which corresponds to a unit hypercube centered at the origin , is known as a **<u>Parzen window</u>** or the **<u>naïve estimator</u>**
- **The total number of points inside the hypercube is then**

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

  - $K((x-x^{(n})/h)$ is equal to unity if and only if the point $x^{(n}$ falls inside a hypercube of side h centered at x
- **Substituting back into the expression for the density estimate**

$$P_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right)$$

- **Notice that the Parzen window density estimate resembles the histogram, except that the cell locations are determined by the data points**



KDE: Parzen windows (2)

- **To understand the role of the kernel function we compute the expectation of the probability estimate P(x)**

$$E[P_{KDE}(x)] = \frac{1}{Nh^D}\sum_{n=1}^{N} E\left[K\left(\frac{x - x^{(n}}{h}\right)\right] =$$
$$= \frac{1}{h^D} E\left[K\left(\frac{x - x^{(n}}{h}\right)\right] =$$
$$= \frac{1}{h^D}\int K\left(\frac{x - x'}{h}\right)P(x')dx'$$

  - where we have assumed that the vectors $x^{(n}$ are drawn independently from the true density P(x)
- **We can see that the expectation of the estimated density $P_{KDE}(x)$ is a convolution of the true density P(x) with the kernel function**
  - The width w of the kernel plays the role of a smoothing parameter: the wider the kernel function, the smoother the estimate $P_{KDE}(x)$
- **For $h \rightarrow 0$, the kernel approaches a delta function and $P_{KDE}(x)$ approaches the true density**
  - However, in practice we have a finite number of points, so h cannot be made arbitrarily small, since the density estimate $P_{KDE}(x)$ approaches a set of delta functions centered at the data points

KDE: smooth kernels

- **The Parzen window has several drawbacks**
  - Yields density estimates that have discontinuities
  - Weights equally all the points x_i, regardless of their distance to the estimation point x
- **It is easy to to overcome some of these difficulties by generalizing the Parzen window with a smooth kernel function K(u) which satisfies the condition**

$$\int_{R^D} K(x)dx = 1$$

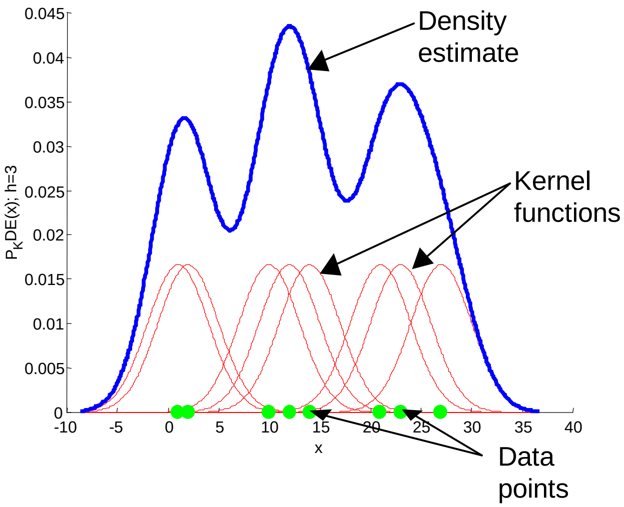
  - Usually, but not not always, K(u) will be a radially symmetric, unimodal probability density function, such as the multivariate Gaussian density function

$$K(x) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}x^T x\right)$$

  - where the expression of the density estimate remains the same as with Parzen windows

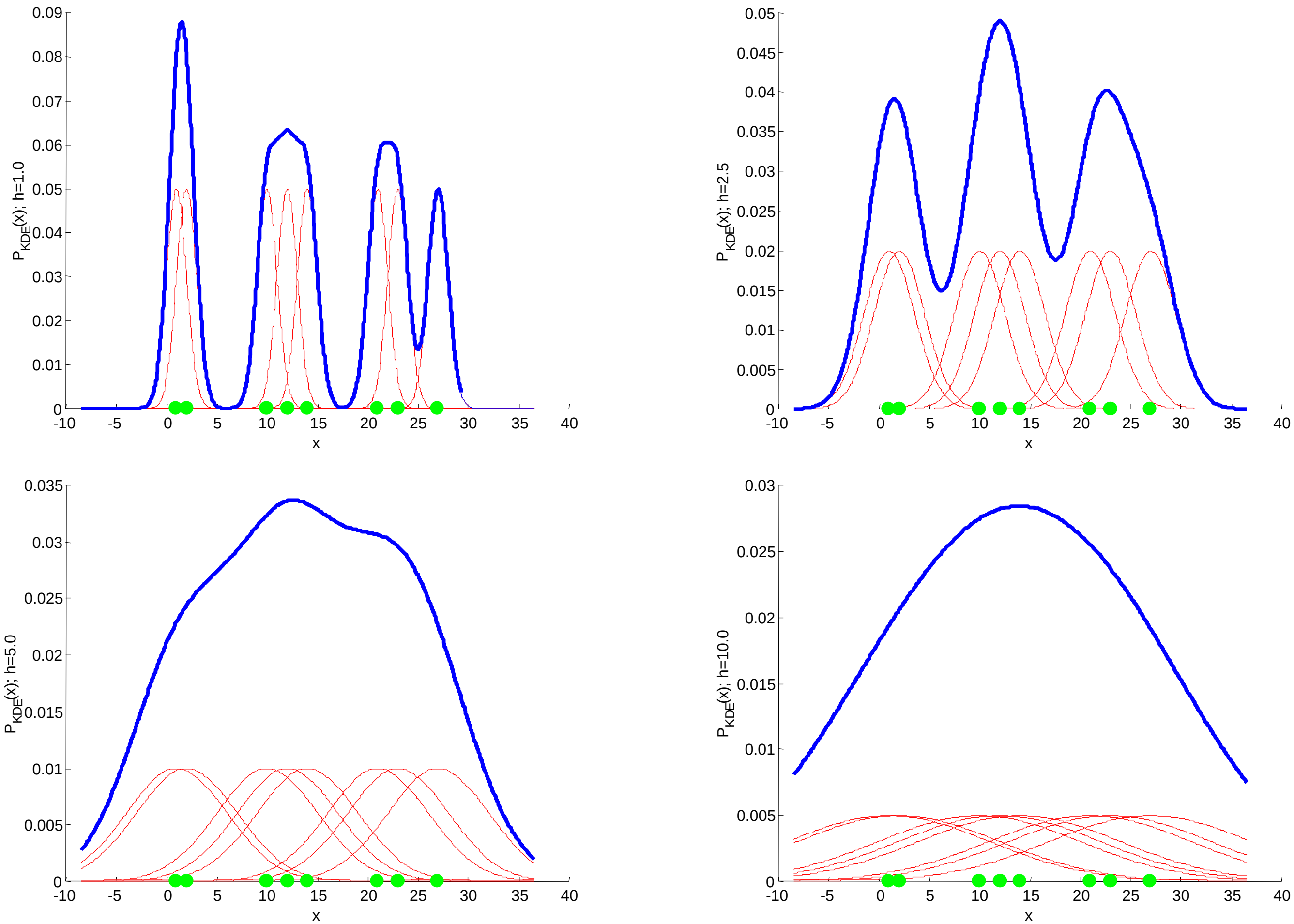
$$P_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right)$$

- **Just as the Parzen window estimate can be considered a sum of boxes centered at the observations, the smooth kernel estimate is a sum of "bumps" placed at the observations**
  - The kernel function determines the shape of the bumps
  - The parameter h, also called the **smoothing parameter** or **bandwidth**, determines their width



Choosing the bandwidth: univariate case (1)

- **The problem of choosing the bandwidth is crucial in density estimation**
  - A large bandwidth will over-smooth the density and mask the structure in the data
  - A small bandwidth will yield a density estimate that is spiky and very hard to interpret



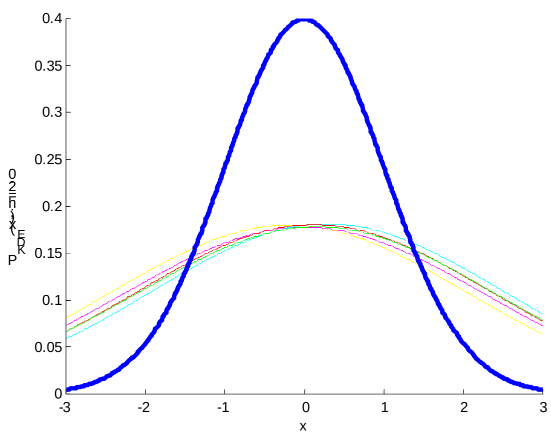
Choosing the bandwidth: univariate case (2)

- **We would like to find a value of the smoothing parameter that minimizes the error between the estimated density and the true density**
  - A natural measure is the mean square error at the estimation point x, defined by

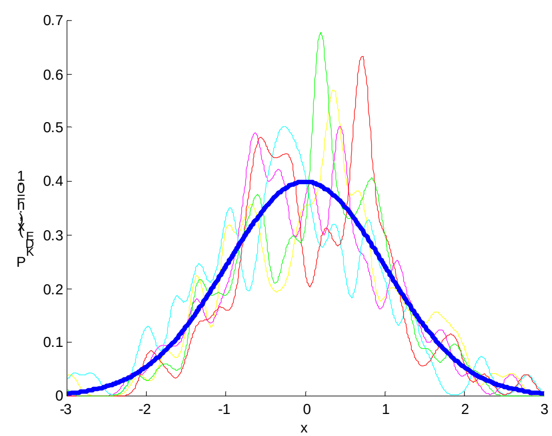
$$MSE_x(P_{KDE}) = E\left[(P_{KDE}(x) - P(x))^2\right] = \underbrace{\{E[P_{KDE}(x) - P(x)]\}^2}_{\text{bias}} + \underbrace{\text{var}(P_{KDE}(x))}_{\text{variance}}$$

- **This expression is an example of the <u>bias-variance dilemma</u> of statistics: 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 $P_{KDE}(x)$ for different data sets (the variance) but it will increase the bias of $P_{KDE}(x)$ with respect to the true density P(x)
  - A small bandwidth will reduce the bias of $P_{KDE}(x)$, at the expense of a larger variance in the estimates $P_{KDE}(x)$

BIAS



VARIANCE



Bandwidth selection methods, univariate case (1)

- **Subjective choice**
  - The natural way for choosing the smoothing parameter is to plot out several curves and choose the estimate that is most in accordance with 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)

$$h_{opt} = \underset{h}{\arg\min}\{MISE(P_{KDE}(x))\} = \underset{h}{\arg\min}\left\{E\left[\int (P_{KDE}(x) - P(x))^2 dx\right]\right\}$$

  - If we assume that the true distribution is a Gaussian density and we use a Gaussian kernel, it can be shown that the optimal value of the bandwidth becomes [Silverman]

$$h_{opt} = 1.06\sigma N^{-1/5}$$

    - where σ is the sample variance and N is the number of training examples

Bandwidth selection methods, univariate case

- Better results can be obtained if we use a robust measure of the spread instead of the sample variance and we reduce the coefficient 1.06 to better cope with multimodal densities [Silverman]. With this in mind, the optimal bandwidth becomes

$$h_{opt} = 0.9AN^{-1/5} \qquad \text{where } A = \min\left(\sigma, \frac{IQR}{1.34}\right)$$

  - IQR is the interquartile range, a robust estimate of the spread. It is computed as one half the difference between the 75th percentile (Q3) and the 25th percentile (Q1). The formula for semi-interquartile range is therefore: (Q3-Q1)/2
    - A percentile rank is the proportion of examples in a distribution that a specific example is greater than or equal to

Likelihood cross-validation

- The ML estimate of h is degenerate since it yields $h_{ML}=0$, a density estimate with delta functions at each training data point
- An practical alternative is to maximize the “pseudo-likelihood” computed using cross-validation

$$h_{MLCV} = \arg\max_h \left\{ \frac{1}{N} \sum_{n=1}^{N} \log f_i\left(x^{(n)}\right) \right\}$$

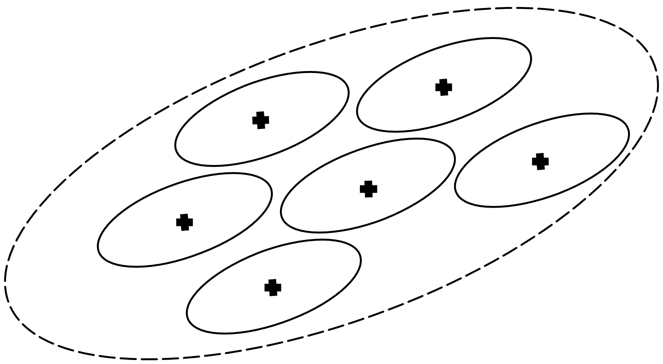
$$\text{where } f_i\left(x^{(m)}\right) = \frac{1}{(N-1)h} \sum_{n=1, n \neq m}^{N} K\left(\frac{x^{(m)} - x^{(n)}}{h} \right)$$

Multivariate density estimation

- **The derived expression of the estimate $P_{KDE}(x)$ for multiple dimensions was**

$$P_{KDE}(x) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right)$$

  - Notice that the bandwidth h is the same for all the axes, so this density estimate will be weight all the axis equally
- **However, if the spread of the data is much greater in one of the coordinate directions 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 kernel density estimate**
  - **Pre-scale each axis** (normalize to unit variance, for instance)
  - **Pre-whiten the data** (linearly transform to have unit covariance matrix), estimate the density, and then transform back [Fukunaga]
    - The whitening transform is simply $y=\Lambda^{-1/2}M^T x$, where Λ and M are the eigenvalue and eigenvector matrices of the sample covariance of x
    - Fukunaga's method is equivalent to using a hyper-ellipsoidal kernel



Product kernels

- **A very common method of performing multivariate density estimation is the <u>product kernel</u>, defined as**

$$P_{PKDE}(x) = \frac{1}{N}\sum_{i=1}^{N} K\left(x, x^{(n}, h_1, \ldots, h_D\right)$$

$$\text{where } K\left(x, x^{(n}, h_1, \ldots, h_D\right) = \frac{1}{h_1 \cdots h_D}\prod_{d=1}^{D} K_d\left(\frac{x(d) - x^{(n}(d)}{h_d}\right)$$

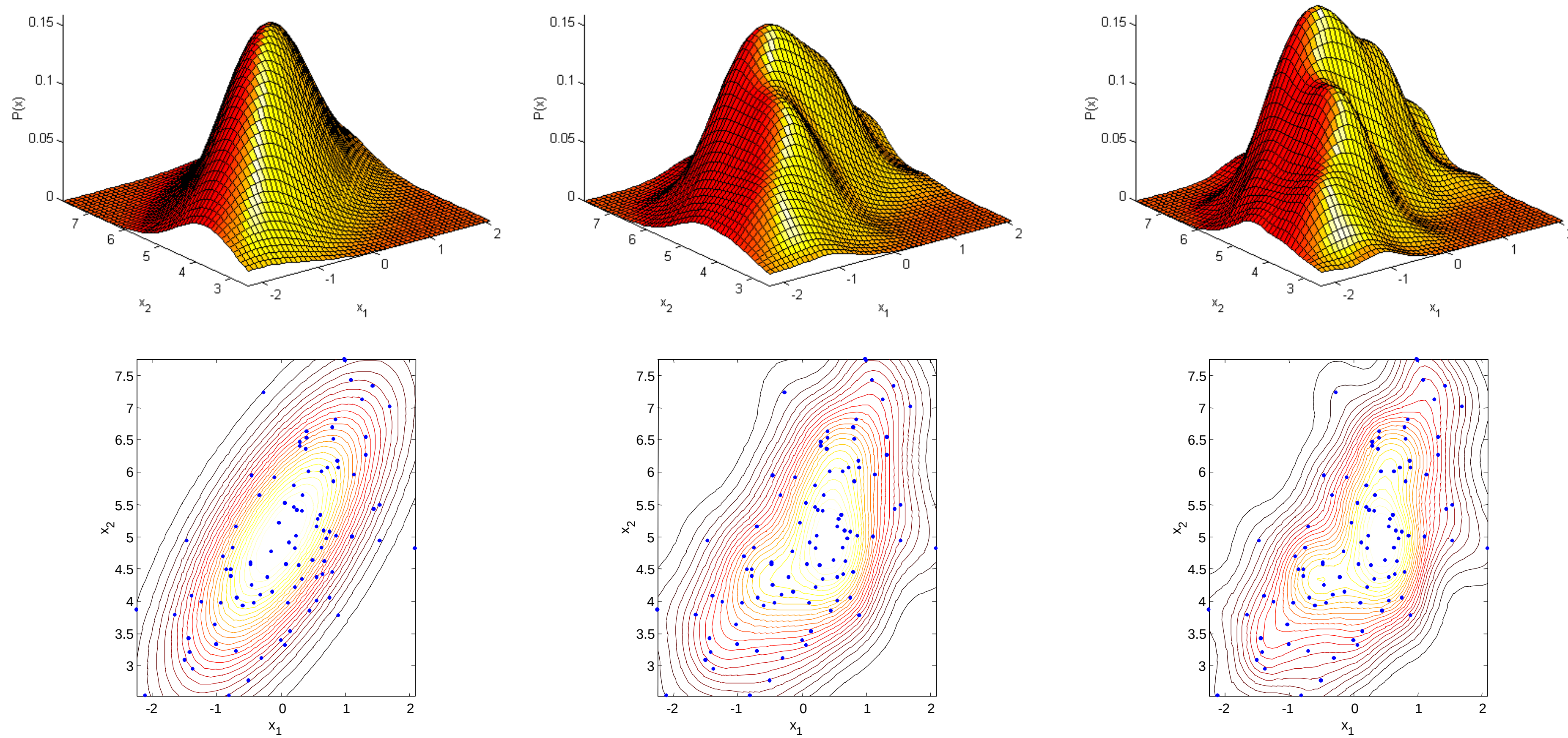
  - The product kernel consists of the product of one-dimensional kernels
- **Typically the same kernel function is used in each dimension ($K_d(x)=K(x)$), and only the bandwidths are allowed to differ**
  - Bandwidth selection can then be performed with any of the methods presented for univariate density estimation
- **It is important to notice that although the expression of $K(x,x^{(n},h_1,\ldots h_D)$ uses <u>kernel independence</u>, this does not imply that any type of feature independence is being assumed**
  - A density estimation method that assumed feature independence would have the following expression

$$P_{FEAT-IND}(x) = \prod_{d=1}^{D}\left(\frac{1}{Nh_d}\sum_{i=1}^{N} K_d\left(\frac{x(d) - x^{(n}(d)}{h_d}\right)\right)$$

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

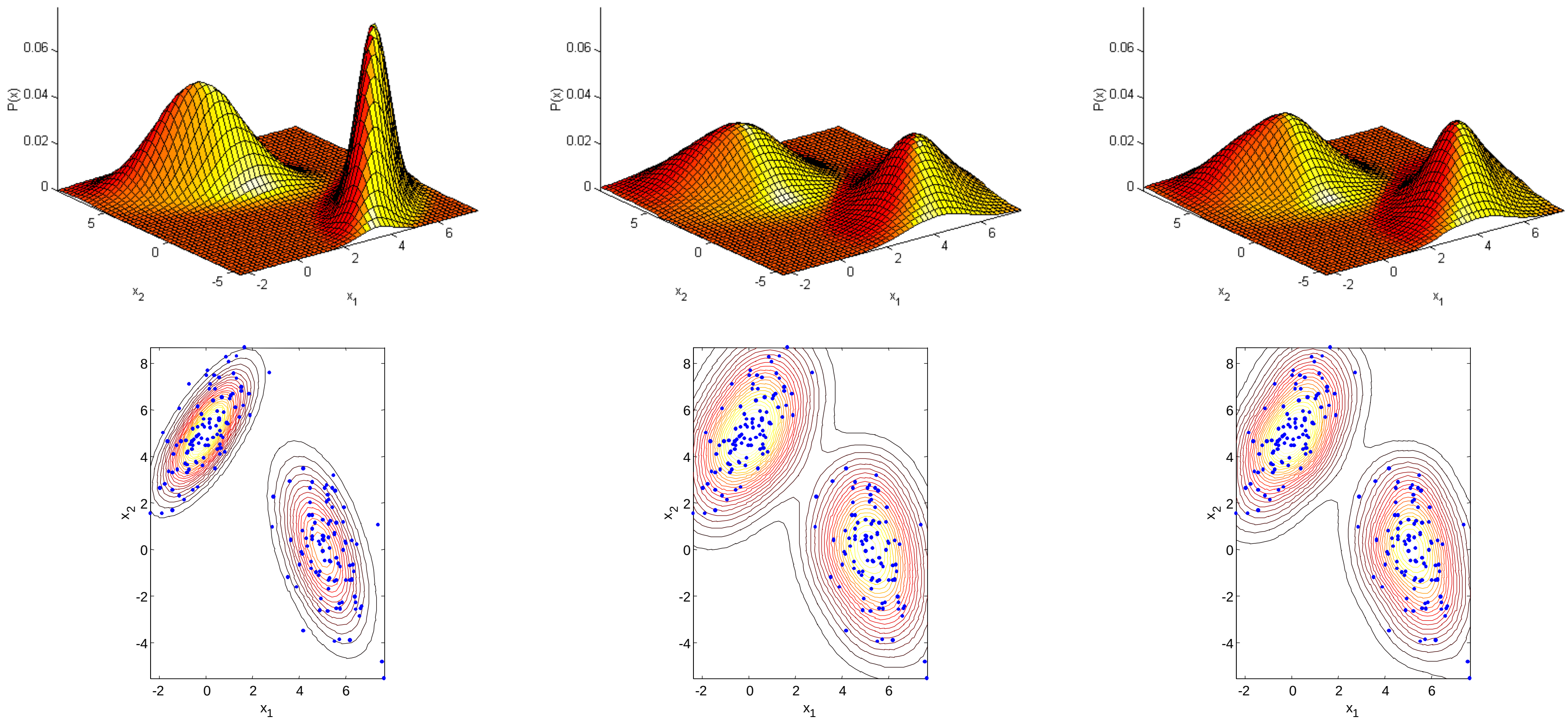
Product kernel, example 1

- **This example shows the product kernel density estimate of a bivariate unimodal Gaussian distribution**
    - 100 data points were drawn from the distribution
    - The figures show the true density (left) and the estimates using $h=1.06\sigma N^{-1/5}$ (middle) and $h=0.9AN^{-1/5}$ (right)



Product kernel, example 2

- **This example shows the product kernel density estimate of a bivariate bimodal Gaussian distribution**
  - 100 data points were drawn from the distribution
  - The figures show the true density (left) and the estimates using $h=1.06\sigma N^{-1/5}$ (middle) and $h=0.9AN^{-1/5}$ (right)



Naïve Bayes classifier

- **Recall that the Bayes classifier is given by the following family of discriminant functions**

$$\text{choose } \omega_i \text{ if } g_i(x) > g_j(x) \ \forall j \neq i$$
$$\text{where } g_i(x) = P(\omega_i \mid x)$$

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

$$g_i(x) = P(\omega_i \mid x) \propto P(x \mid \omega_i)P(\omega_i)$$

  - where $P(\omega_i)$ is our prior knowledge and $P(x|\omega_i)$ is obtained through density estimation
- **Although we have presented density estimation methods that allow us to estimate the multivariate likelihood $P(x|\omega_i)$, the curse of dimensionality still poses problems**
- **One highly practical simplification of the Bayes classifier is the so-called <u>Naïve Bayes</u> classifier**
  - The Naïve Bayes classifier makes the assumption that the <u>features are class-conditionally independent</u>

$$P(x \mid \omega_i) = \prod_{d=1}^{D} P(x(d) \mid \omega_i)$$

    - It is important to notice that this assumption is not as rigid as assuming independent features $\mathbf{P(x)} = \prod_{d=1}^{D} \mathbf{P(x(d))}$
  - Merging this expression into the discriminant function yields the decision rule for the Naïve Bayes classifier

$$\boxed{g_{i,NB}(x) = P(\omega_i)\prod_{d=1}^{D} P\left(x(d) \mid \omega_i\right) \qquad \textbf{Naïve Bayes Classifier}}$$

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