L7: Kernel density estimation

Non-parametric density estimation

Histograms

Parzen windows

Smooth kernels

Product kernel density estimation

The naïve Bayes classifier

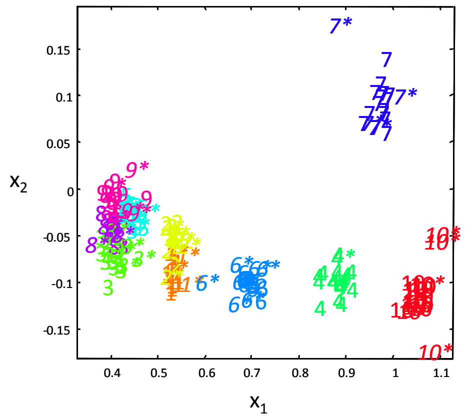
Non-parametric density estimation

In the previous two lectures we have assumed that either

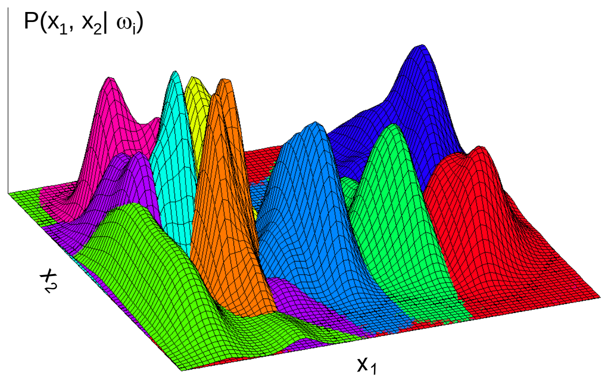
- The likelihoods $p(x|\omega_i)$ were known (LRT), or
- At least their parametric form was known (parameter estimation)

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!



$\xrightarrow{\text{non-parametric density estimation}}$



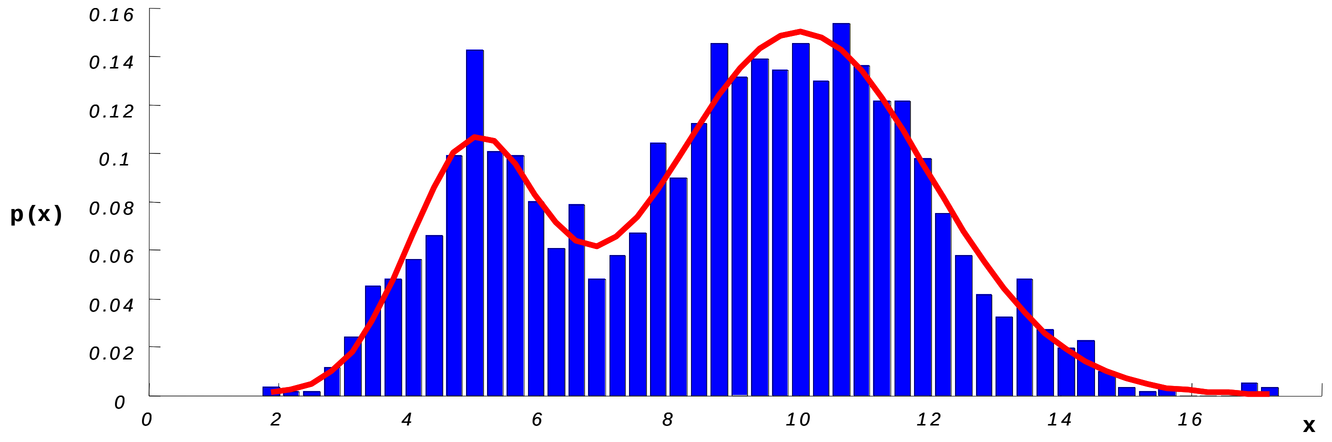
The histogram

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

$$p_H(x) = \frac{1}{N} \frac{\left[\# \: of \: x^{(k} \: in \: same \: bin \: as \: x\right]}{[width \: of \: bin]}$$

- The histogram requires two "parameters" to be defined: <u>bin width</u> and <u>starting position</u> 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 x, drawn from a distribution $p(x)$, will fall in a given region $\Re$ of the sample space is

$$P = \int_{\Re} p(x')dx'$$

- Suppose now that N vectors $\{x^{(1}, x^{(2}, \ldots x^{(N}\}$ are drawn from the distribution; the probability that k of these N vectors fall in $\Re$ is given by the binomial distribution

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

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

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

- Therefore, as $N \to \infty$ 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 $\Re$

$$P \cong \frac{k}{N}$$

[Bishop, 1995]

- On the other hand, if we assume that $\Re$ is so small that $p(x)$ does not vary appreciably within it, then

$$\int_{\Re} p(x')dx' \cong p(x)V$$

  - where V is the volume enclosed by region $\Re$
- Merging with the previous result we obtain

$$\left.\begin{array}{l} P = \int_{\Re} p(x')dx' \cong p(x)V \\ P \cong \frac{k}{N} \end{array}\right\} \Rightarrow p(x) \cong \frac{k}{NV}$$

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

In practice the total number of examples is fixed

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

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

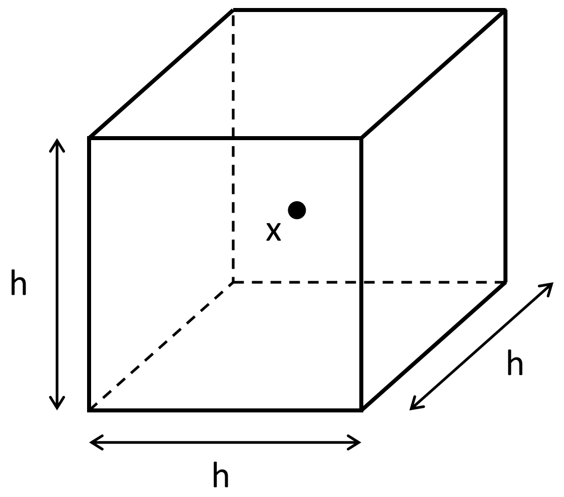
$$p(x) \cong \frac{k}{NV} \text{ where } \begin{cases} V & volume\ surrounding\ x \\ N & total\ \#examples \\ k & \#examples\ inside\ V \end{cases}$$

- When applying this result to practical density estimation problems, two basic approaches can be adopted
  - We can fix V and determine k from the data. This leads to **kernel density estimation** (KDE), the subject of this lecture
  - We can fix k and determine V 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 $N \to \infty$, provided that V shrinks with N, and that k grows with N appropriately

Parzen windows

Problem formulation

- Assume that the region $\Re$ that encloses the k 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 <u>kernel function</u> $K(u)$

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

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

[Bishop, 1995]

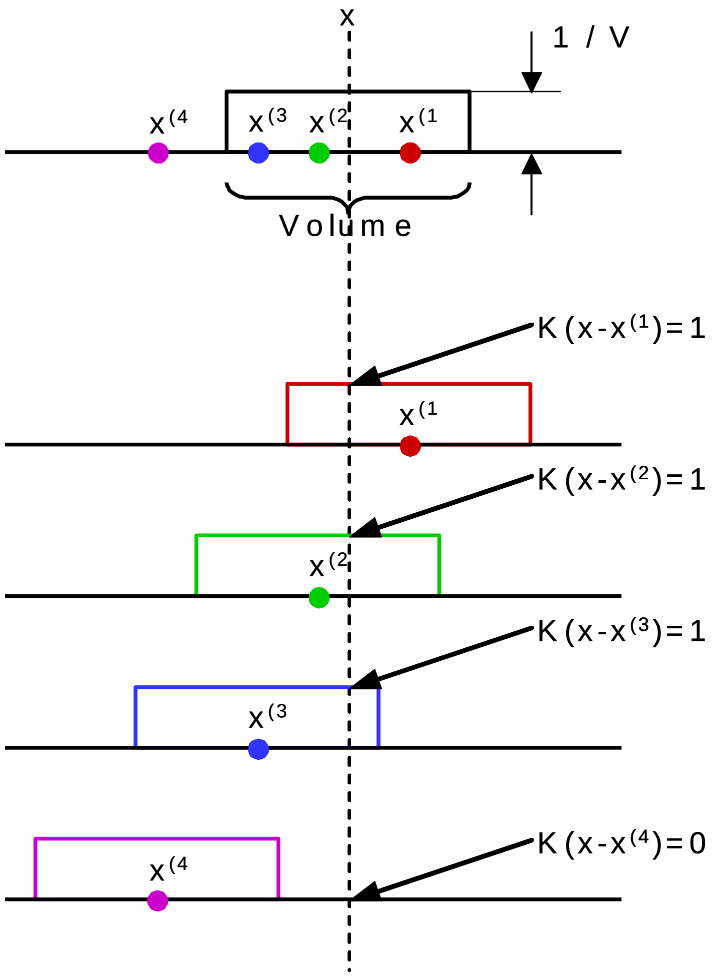
– The total number of points inside the hypercube is then

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

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 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 $p_{KDE}(x)$

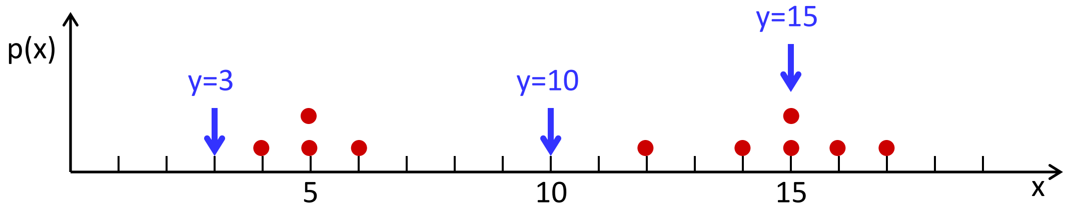
$$E[p_{KDE}(x)] = \frac{1}{Nh^D}\sum_{n=1}^{N} E\left[K\left(\frac{x-x^{(n}}{h}\right)\right]$$
$$= \frac{1}{h^D} E\left[K\left(\frac{x-x^{(n}}{h}\right)\right] = \frac{1}{h^D}\int K\left(\frac{x-x^{(n}}{h}\right)p(x')dx'$$

  - where we have assumed that vectors $x^{(n}$ are drawn independently from the true density $p(x)$

- We can see that the expectation of $p_{KDE}(x)$ is a convolution of the true density $p(x)$ with the kernel function
  - Thus, the kernel width h plays the role of a smoothing parameter: the wider h is, the smoother the estimate $p_{KDE}(x)$
- For $h \to 0$, the kernel approaches a Dirac 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 $p_{KDE}(x)$ would then degenerate to a set of impulses located at the training data points

Exercise

- Given dataset $X = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$, use Parzen windows to estimate the density $p(x)$ at $y = 3,10,15$; use $h = 4$
- Solution
  - Let's first draw the dataset to get an idea of the data



  - Let's now estimate $p(y = 3)$

$$p(y=3) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right) = \frac{1}{10 \times 4^1}\left[K\left(\frac{3-4}{4}\right) + K\left(\frac{3-5}{4}\right) + \cdots K\left(\frac{3-17}{4}\right)\right] = 0.0025$$

  - Similarly

$$p(y=10) = \frac{1}{10\times4^1}[0+0+0+0+0+0+0+0+0+0] = 0$$

$$p(y=15) = \frac{1}{10\times4^1}[0+0+0+0+0+1+1+1+1+0] = 0.1$$

Smooth kernels

The Parzen window has several drawbacks

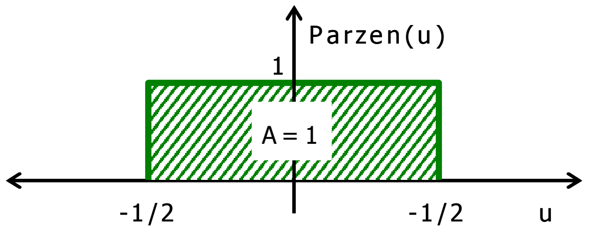
- It yields density estimates that have discontinuities
- It weights equally all points x_i, regardless of their distance to the estimation point x

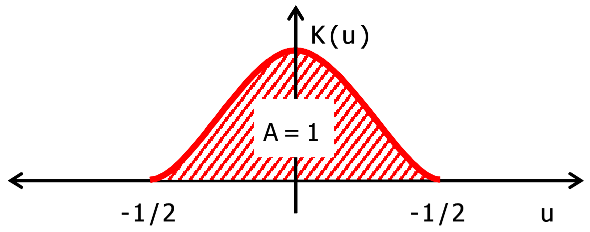
For these reasons, the Parzen window is commonly replaced with a smooth kernel function $K(u)$

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

- Usually, but not always, $K(u)$ will be a radially symmetric and unimodal pdf, such as the Gaussian $K(x) = (2\pi)^{-D/2} e^{-\frac{1}{2}x^T x}$
- Which leads to the density estimate

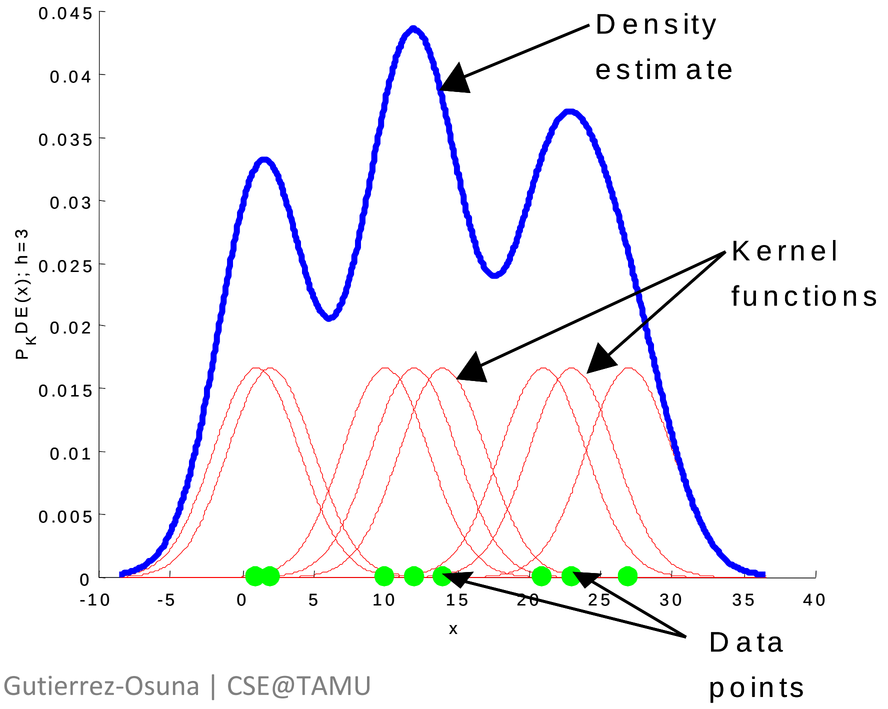
$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(k}}{h}\right)$$





Interpretation

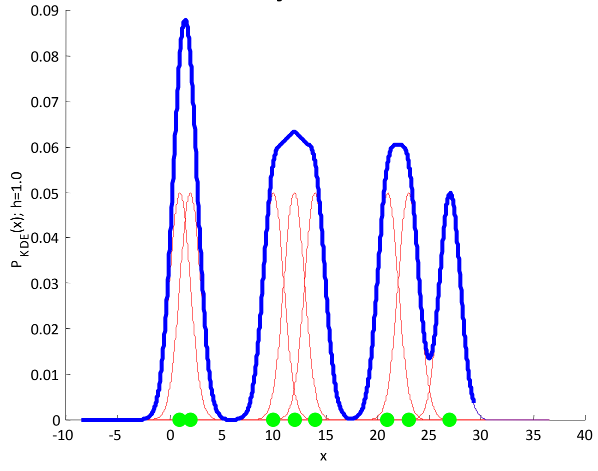
- Just as the Parzen window estimate can be seen as a sum of boxes centered at the data, the smooth kernel estimate is a sum of “bumps”
- The kernel function determines the shape of the bumps
- The parameter h, also called the <u>smoothing parameter</u> or <u>bandwidth</u>, determines their width

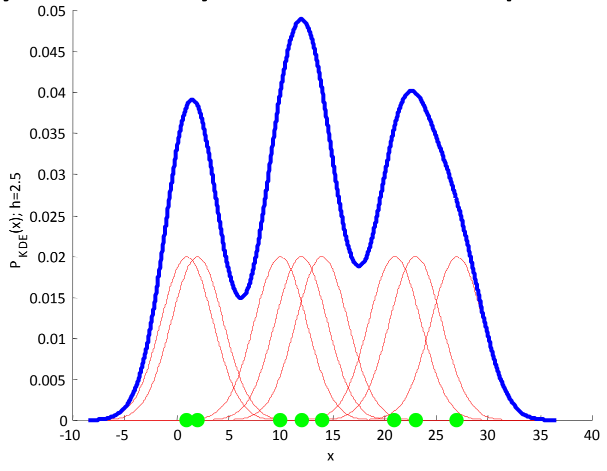


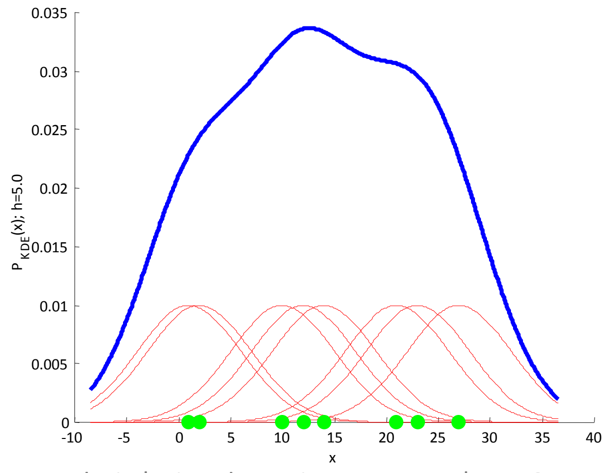
Bandwidth selection

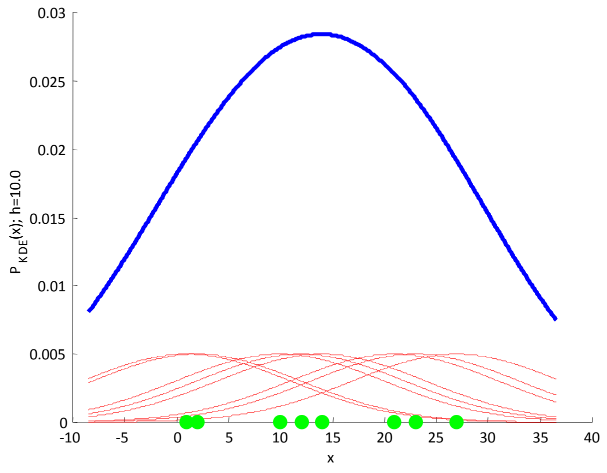
The problem of choosing h is crucial in density estimation

- 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







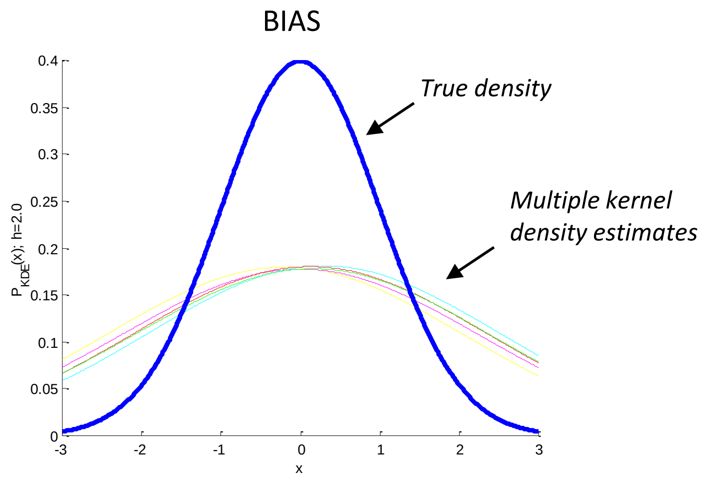


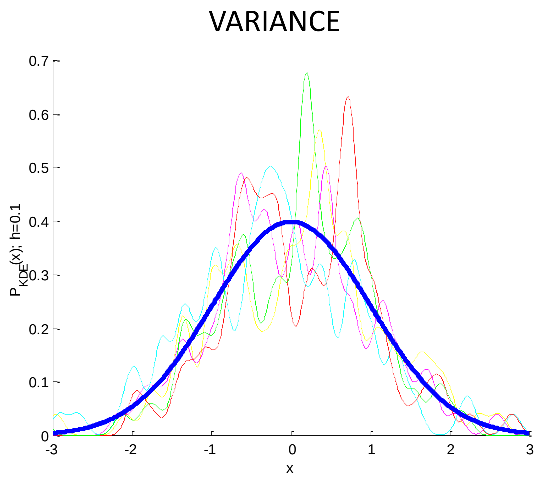
- We would like to find a value of h that minimizes the error between the estimated density and the true density
  - A natural measure is the MSE at the estimation point x, defined by

$$E[(p_{KDE}(x)-p(x))^2] = \underbrace{E[p_{KDE}(x)-p(x)]^2}_{bias} + \underbrace{var(p_{KDE}(x))}_{variance}$$

- This expression is an example of the <u>bias-variance tradeoff</u> 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 <u>systematic error</u> incurred in the estimation
  - The variance of an estimate is the <u>random error</u> 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 $p_{KDE}(x)$





Bandwidth selection methods, univariate case

Subjective choice

- The natural way for choosing 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)

$$h_{MISE} = \arg\min\left\{E\left[\int \left(p_{KDE}(x) - p(x)\right)^2 dx\right]\right\}$$

- If we assume that the true distribution is Gaussian and we use a Gaussian kernel, it can be shown that the optimal value of h is

$$h^* = 1.06\sigma N^{-1/5}$$

  - where σ 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 1.06 to better cope with multimodal densities
  - The optimal bandwidth then becomes

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

- IQR is the interquartile range, a robust estimate of the spread
  - IQR is the difference between the 75th percentile ($Q3$) and the 25th percentile ($Q1$): $IQR = Q3 - Q1$
  - A percentile rank is the proportion of examples in a distribution that a specific example is greater than or equal to

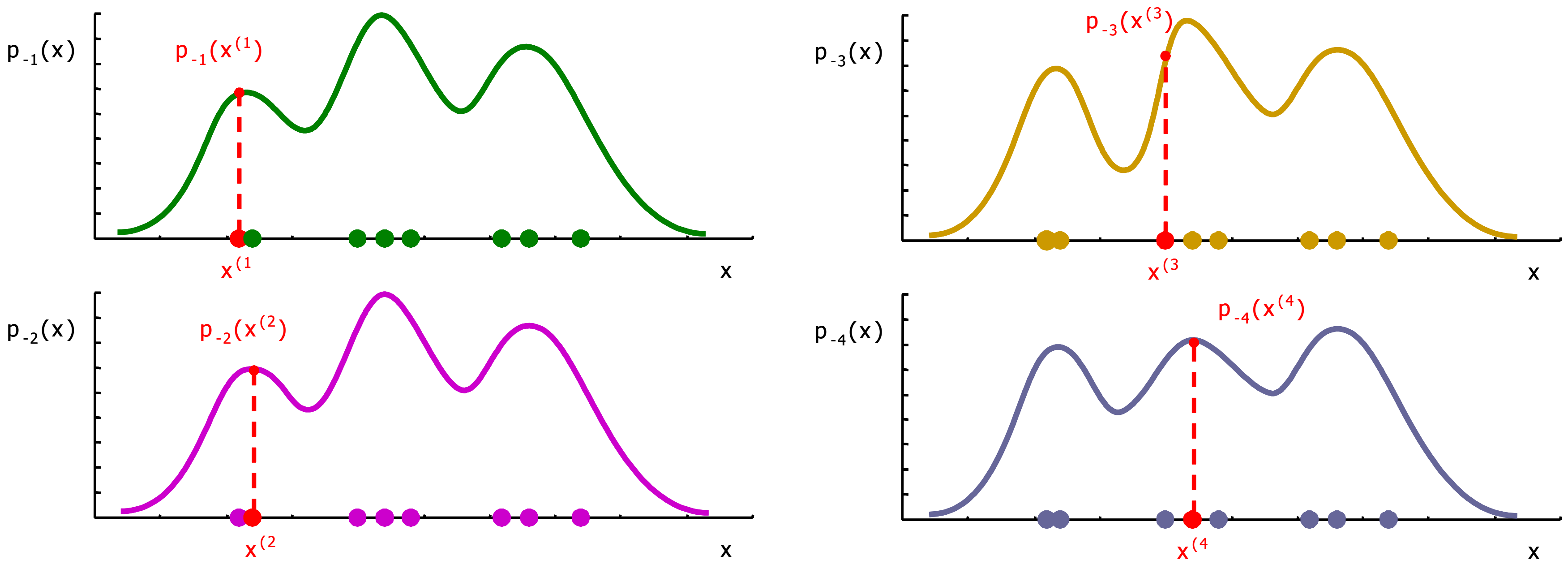
Maximum likelihood cross-validation

- The ML estimate of h is degenerate since it yields $h_{ML} = 0$, 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

$$h^* = \arg\max \left\{ \frac{1}{N} \sum_{n=1}^{N} log p_{-n}\left(x^{(n)}\right) \right\}$$

$$where\ p_{-n}\left(x^{(n)}\right) = \frac{1}{(N-1)h} \sum_{\substack{m=1 \\ m\neq \mathrm{n}}}^{N} K\left(\frac{x^{(n} - x^{(m}}{h}\right)$$

[Silverman, 1986]


$p_{-1}(x)$
$p_{-1}(x^{(1)})$
$x^{(1}$
x
$p_{-3}(x)$
$p_{-3}(x^{(3)})$
$x^{(3}$
x
$p_{-2}(x)$
$p_{-2}(x^{(2)})$
$x^{(2}$
x
$p_{-4}(x)$
$p_{-4}(x^{(4)})$
$x^{(4}$
x

Multivariate density estimation

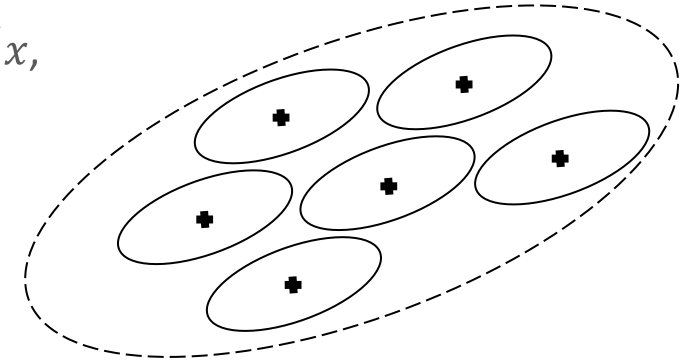
For the multivariate case, the KDE is

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

- Notice that the bandwidth h is the same for all the axes, so this density estimate will be weight all the axis equally
- 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 $\Sigma = I$), estimate the density, and then transform back [Fukunaga]
  - The whitening transform is $y = \Lambda^{-1/2} M^T x$, where Λ and M 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

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

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

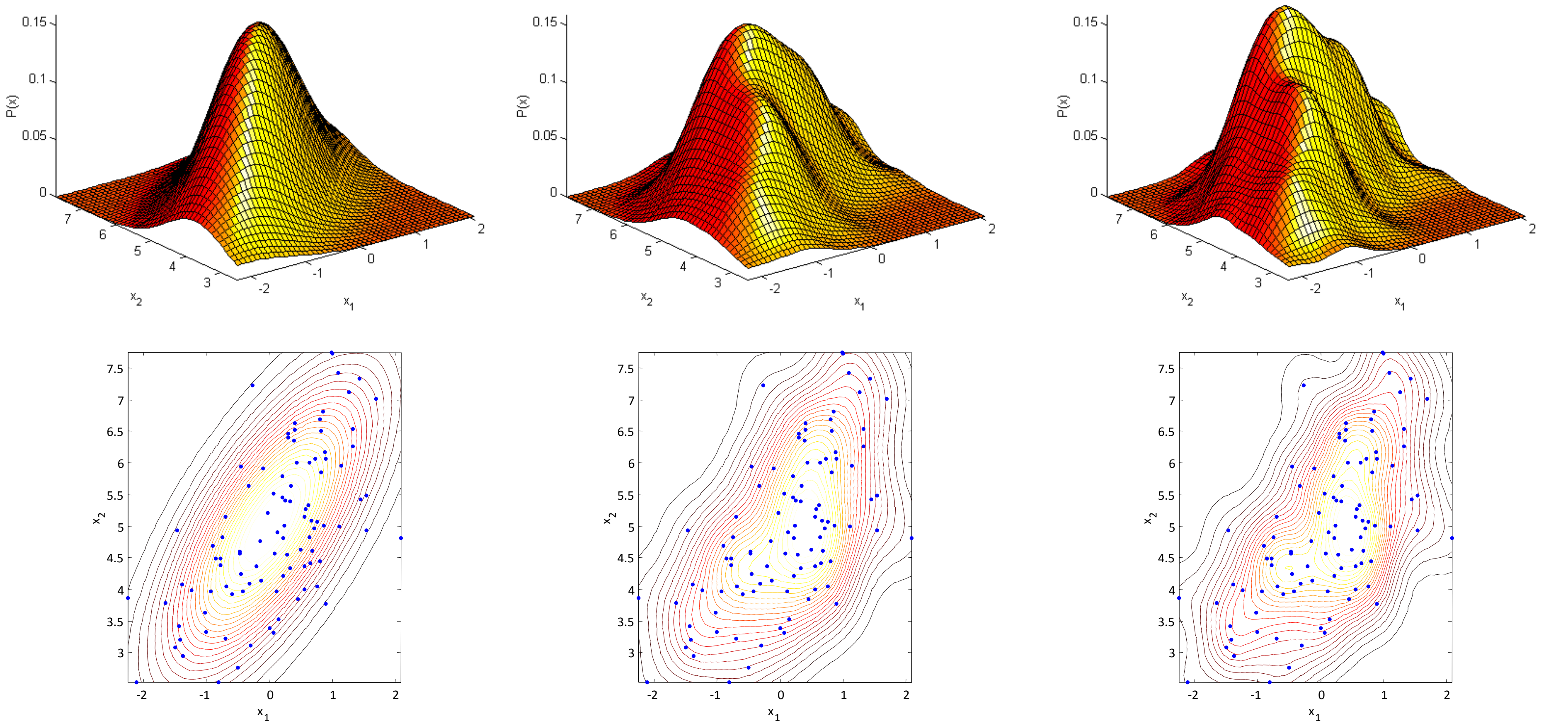
- The product kernel consists of the product of one-dimensional kernels
  - Typically the same kernel function is used in each dimension ($K_d(x) = K(x)$), and only the bandwidths are allowed to differ
  - Bandwidth selection can then be performed with any of the methods presented for univariate density estimation
- Note that although $K\left(x, x^{(n}, h_1, \dots h_D\right)$ uses kernel independence, this does not imply we assume the features are independent
  - If we assumed feature independence, the DE would have the expression

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

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

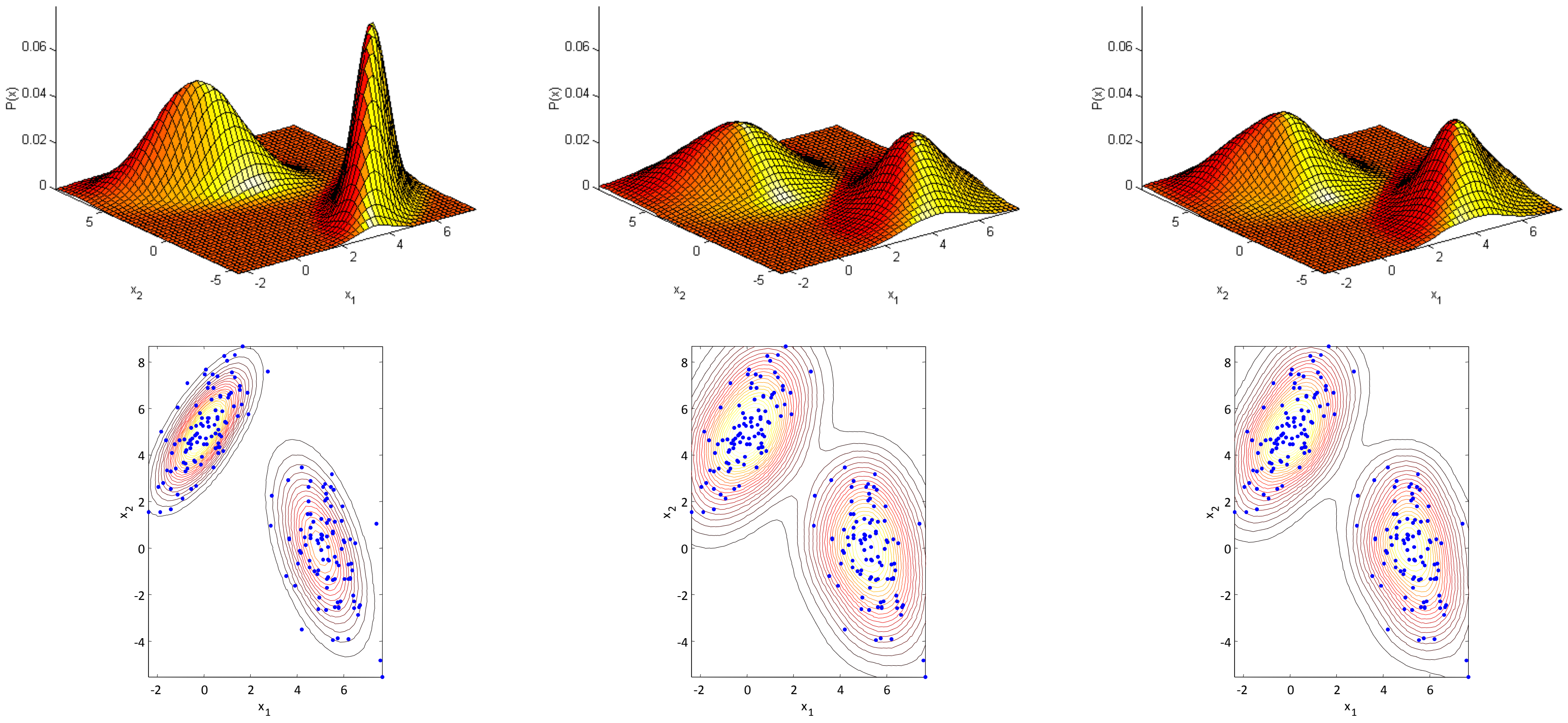
Example I

– 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 $h = 1.06\sigma N^{-1/5}$ (middle) and $h = 0.9AN^{-1/5}$ (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 $h = 1.06\sigma N^{-1/5}$ (middle) and $h = 0.9AN^{-1/5}$ (right)



Naïve Bayes classifier

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

$$chose\ \omega_i\ if\ g_i(x) > g_j(x) \forall j \neq i\ where\ g_i(x) = P(\omega_i|x)$$

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

$$g_i(x) = P(\omega_i|x) \propto p(x|\omega_i)P(\omega_i)$$

  - where $P(\omega_i)$ is our prior knowledge and $\mathrm{p}(x|\omega_i)$ is obtained through DE

- Although the DE methods presented in this lecture allow us to estimate the multivariate likelihood $p(x|\omega_i)$, 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

$$p(x|\omega_i) = \prod_{d=1}^{D} p(x_d|\omega_i)$$

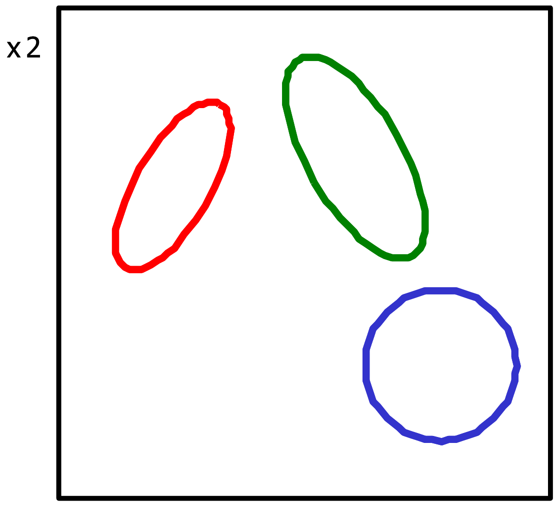
  - This assumption is not as rigid as assuming independent features $p(x) = \prod_{d=1}^{D} p(x_d)$

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

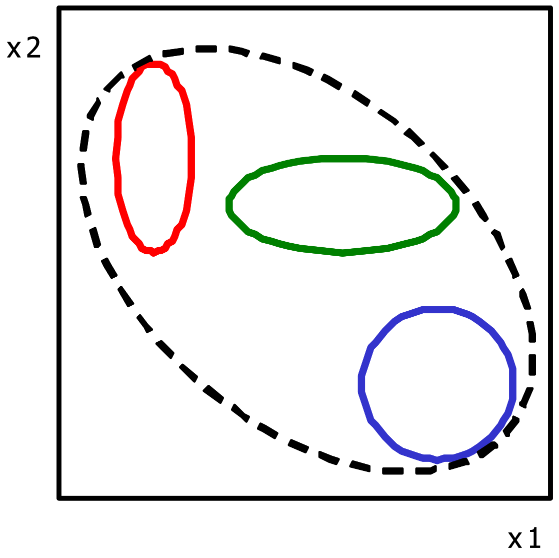
$$g_{i,NB}(x) = P(\omega_i) \prod_{d=1}^{D} p(x_d|\omega_i)$$

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

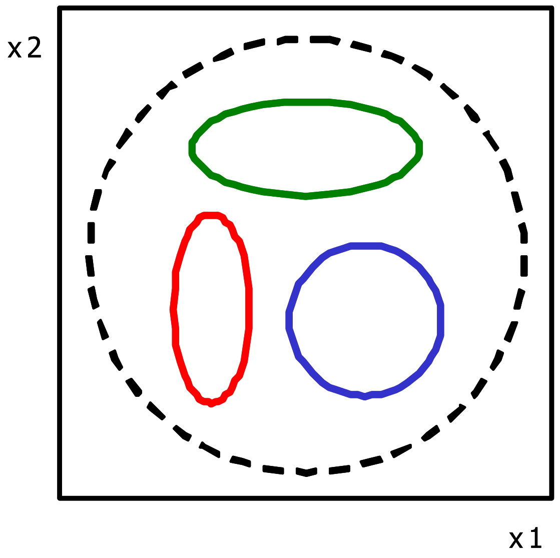
Class-conditional independence vs. independence



$$p(x|\omega_i) \neq \prod_{d=1}^{D} p(x_d|\omega_i)$$



$$p(x|\omega_i) = \prod_{d=1}^{D} p(x_d|\omega_i)$$
$$p(x) \neq \prod_{d=1}^{D} p(x_d)$$



$$p(x|\omega_i) = \prod_{d=1}^{D} p(x_d|\omega_i)$$
$$p(x) \cong \prod_{d=1}^{D} p(x_d)$$