L7: Kernel density estimation

Non-parametric density estimation

Histograms

Parzen windows

Smooth kernels

Product kernel density estimation

The naïve Bayes classifier

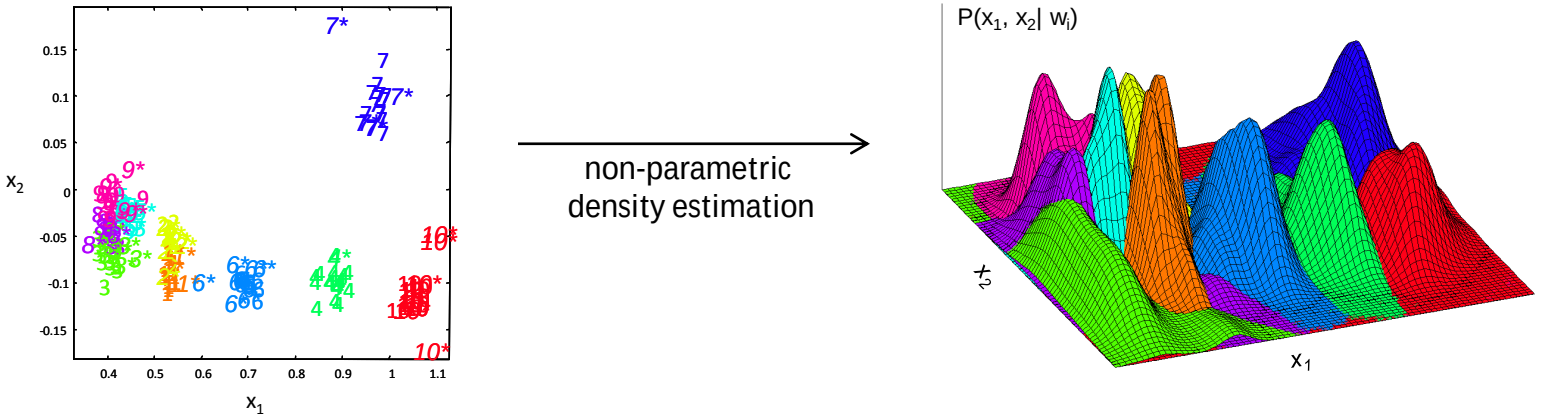
Non-parametric density estimation

In the previous two lectures we have assumed that either

- The likelihoods ή ώϑ ¡ 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!



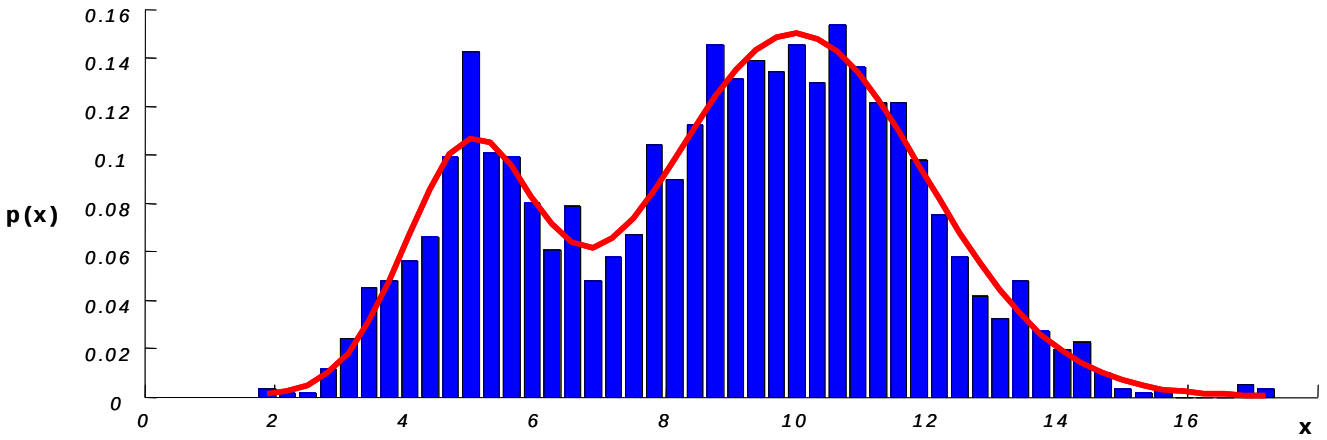
The histogram

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

$$\hat{p}(x) \approx \frac{\text{fraction of points in bin}}{\text{volume of bin}}$$

- 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 $\mathcal{R}$ of the sample space is

$$P = \int_{\mathcal{R}} p(\mathbf{x}')\,d\mathbf{x}'$$

- Suppose now that N vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ are drawn from the distribution; the probability that K of these N vectors fall in $\mathcal{R}$ 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

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

- 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 $\mathcal{R}$

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

[Bishop, 1995]

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

$$\int_P p(x)\,dx \approx p(x)\,V$$

  - where V is the volume enclosed by region P

- Merging with the previous result we obtain

$$\frac{k}{n} \approx p(x)\,V \quad \text{or} \quad p(x) \approx \frac{k}{nV}$$

- 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 P 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 P
  - Small enough to support the assumption that $p(x)$ is constant within P

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

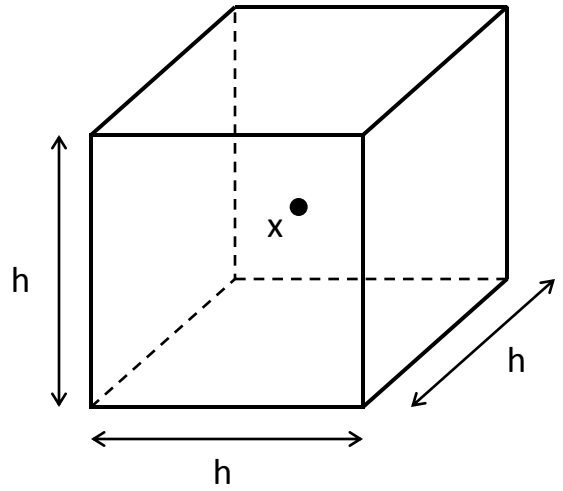
$$\hat{p}(\mathbf{x}) \approx \frac{k}{n V} \quad \text{where} \quad \begin{array}{l} V \text{ is the volume surrounding } \mathbf{x} \\ n \text{ is the total number of examples} \\ k \text{ is the number of examples inside } V \end{array}$$

- 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 P that encloses the examples is a hypercube with sides of length centered at
  - Then its volume is given by , where is the number of dimensions



- To find the number of examples that fall within this region we define a kernel function

$$\phi(\mathbf{x}) = \begin{cases} 1 & |x_j| \le \frac{h}{2} \\ 0 & \text{otherwise} \end{cases}$$

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

[Bishop, 1995]

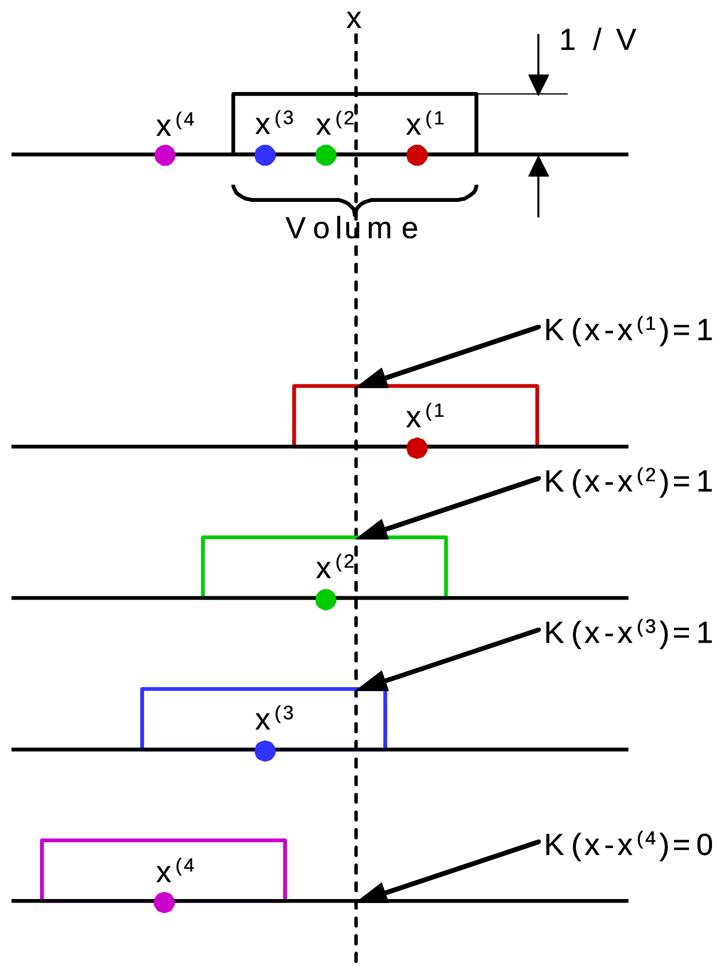
- The total number of points inside the hypercube is then

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

Substituting back into the expression for the density estimate

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

- 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 $\hat{p}_n(\mathbf{x})$

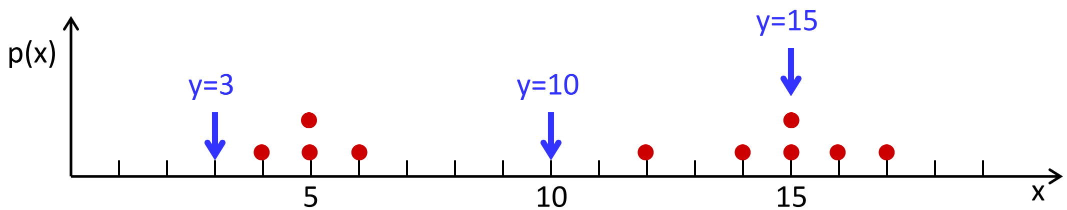
$$\mathcal{E}[\hat{p}_n(\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n} \mathcal{E}_{\mathbf{x}_i}[\varphi(\mathbf{x}-\mathbf{x}_i)] = \int \varphi(\mathbf{x}-\mathbf{v})\, p(\mathbf{v})\, d\mathbf{v}$$

$$= \frac{1}{h^d}\int \varphi\left(\frac{\mathbf{x}-\mathbf{v}}{h}\right) p(\mathbf{v})\, d\mathbf{v} = \frac{1}{h^d}\int \varphi\left(\frac{\mathbf{v}-\mathbf{x}}{h}\right) p(\mathbf{x})\, d\mathbf{x}$$

  - where we have assumed that vectors $\mathbf{x}_i$ are drawn independently from the true density $p(\mathbf{x})$
- We can see that the expectation of $\hat{p}_n(\mathbf{x})$ is a convolution of the true density $p(\mathbf{x})$ with the kernel function
  - Thus, the kernel width h plays the role of a smoothing parameter: the wider h is, the smoother the estimate $\hat{p}_n(\mathbf{x})$
- For $h \to 0$, the kernel approaches a Dirac delta function and $\hat{p}_n(\mathbf{x})$ 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 $\hat{p}_n(\mathbf{x})$ 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 $p(y = 3)$

$$p_{KDE}(y=3) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{y-x^{(n)}}{h}\right) = \frac{1}{10 \times 4^1}\left[K\left(\frac{3-4}{4}\right) + K\left(\frac{3-5}{4}\right) + \dots + K\left(\frac{3-17}{4}\right)\right] = 0.025$$

  - Similarly

$$p_{KDE}(y=10) = \frac{1}{10 \times 4^1}[0+0+0+0+0+0+0+0+0+0] = 0$$

$$p_{KDE}(y=15) = \frac{1}{10 \times 4^1}[0+0+0+0+0+1+1+1+1+0] = 0.1$$

Smooth kernels

The Parzen window has several drawbacks

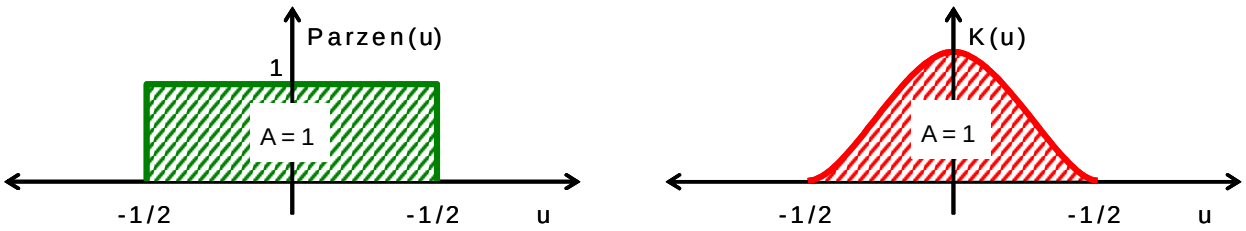
- It yields density estimates that have discontinuities
- It weights equally all points x_i regardless of their distance to the estimation point x

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

$$\sum_{i=1}^{n} K(x - x_i)$$

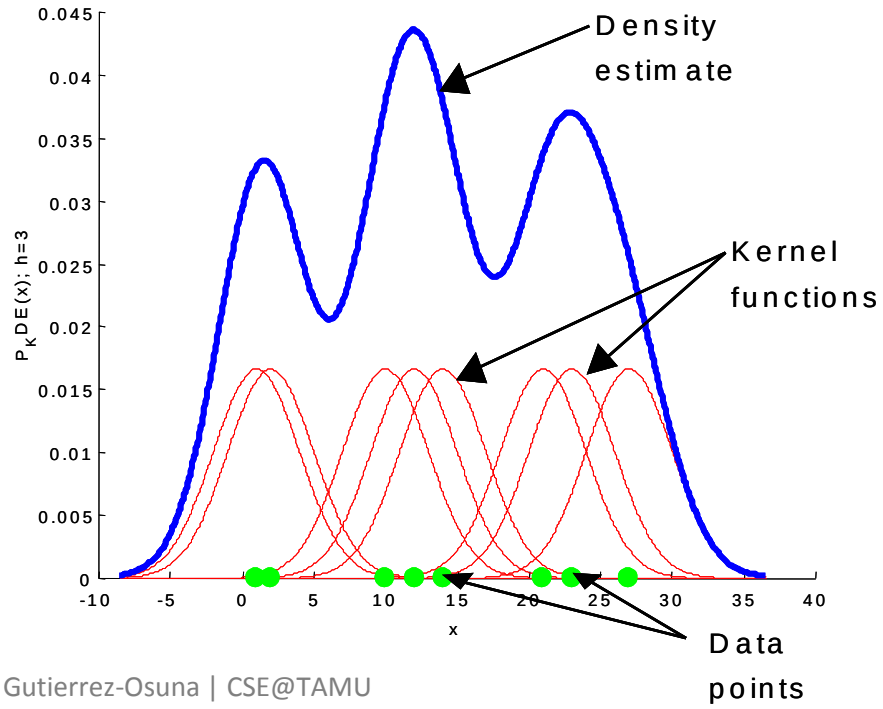
- Usually, but not always, K will be a radially symmetric and unimodal pdf, such as the Gaussian $K(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{x^2}{2\sigma^2}\right)$
- Which leads to the density estimate

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



Interpretation

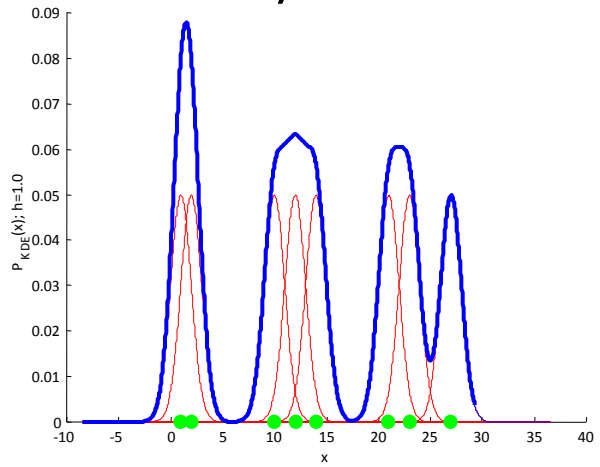
- Just as the Parzen window estimate can be seen as a sum of boxes centered at the data, the smooth kernel estimate is a sum of “bumps”
- The kernel function determines the shape of the bumps
- The parameter ”Ω also called the smoothing parameter or bandwidth, determines their width

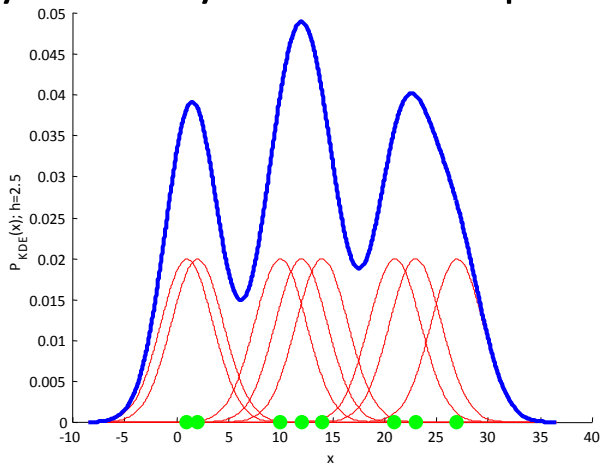


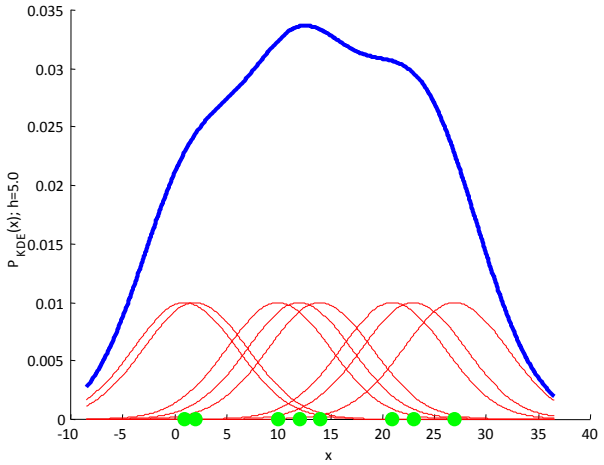
Bandwidth selection

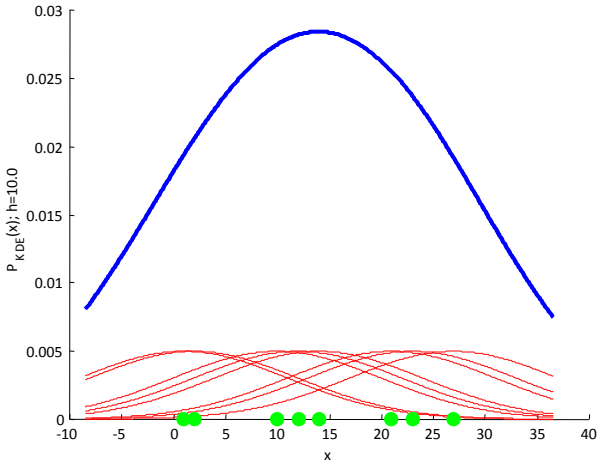
The problem of choosing ▌is crucial in density estimation

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







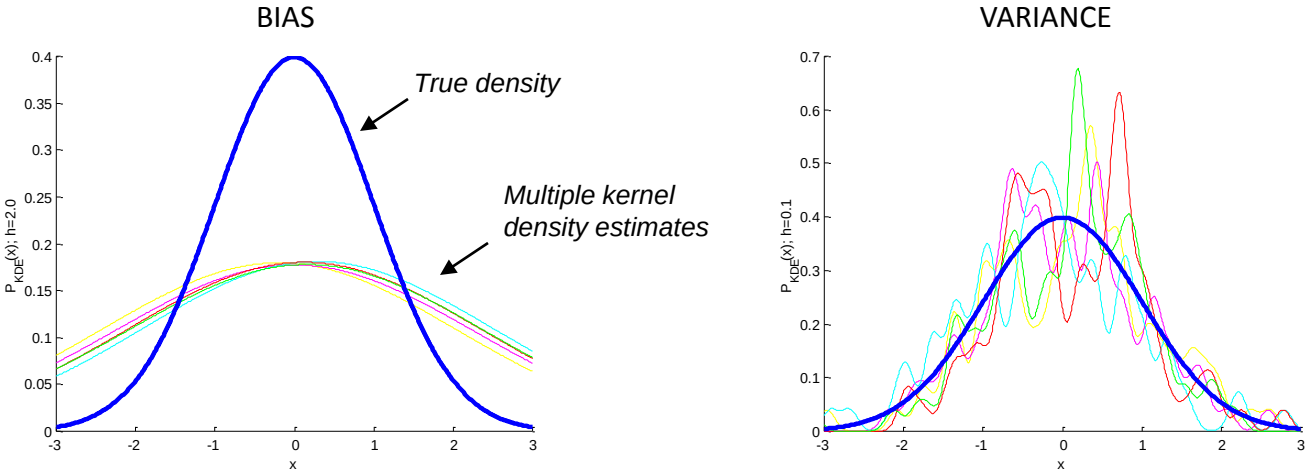


- We would like to find a value of h that minimizes the error between the estimated density and the true density
  - A natural measure is the MSE at the estimation point x, defined by

$$\mathrm{MSE}\{\hat{f}_h(x)\} = \mathrm{E}[(\hat{f}_h(x) - f(x))^2] = \underbrace{\mathrm{E}[\hat{f}_h(x) - f(x)]^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}(\hat{f}_h(x))}_{\text{variance}}$$

- 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 $\hat{p}(x)$ for different data sets (the variance), but it will increase the bias of $\hat{p}(x)$ with respect to the true density $p(x)$
  - A small bandwidth will reduce the bias of $\hat{p}(x)$, at the expense of a larger variance in the estimates $\hat{p}(x)$



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)

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

- 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 ρσ to better cope with multimodal densities
  - The optimal bandwidth then becomes

$$\text{where } \hat{I}\left(\frac{\text{IQR}}{1.34}\right)$$

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

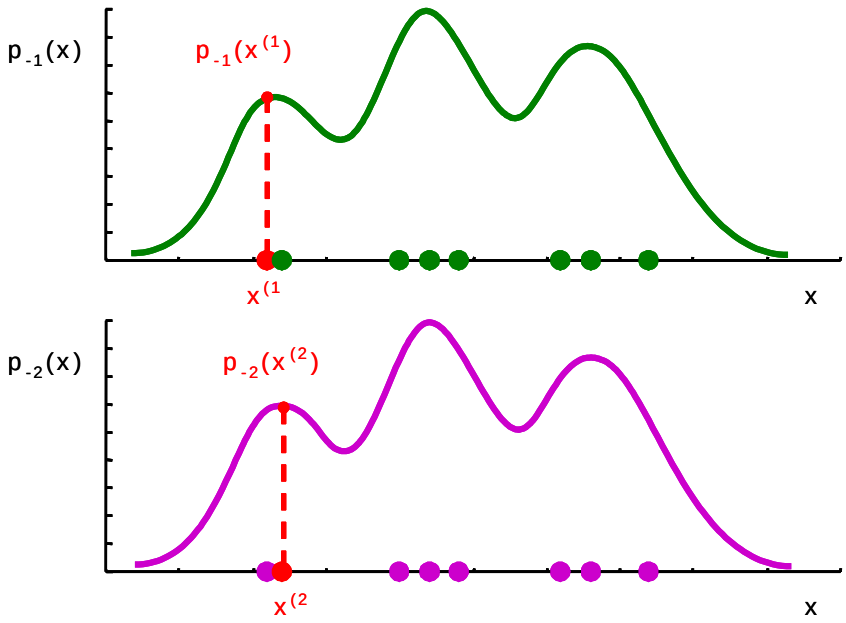
Maximum likelihood cross-validation

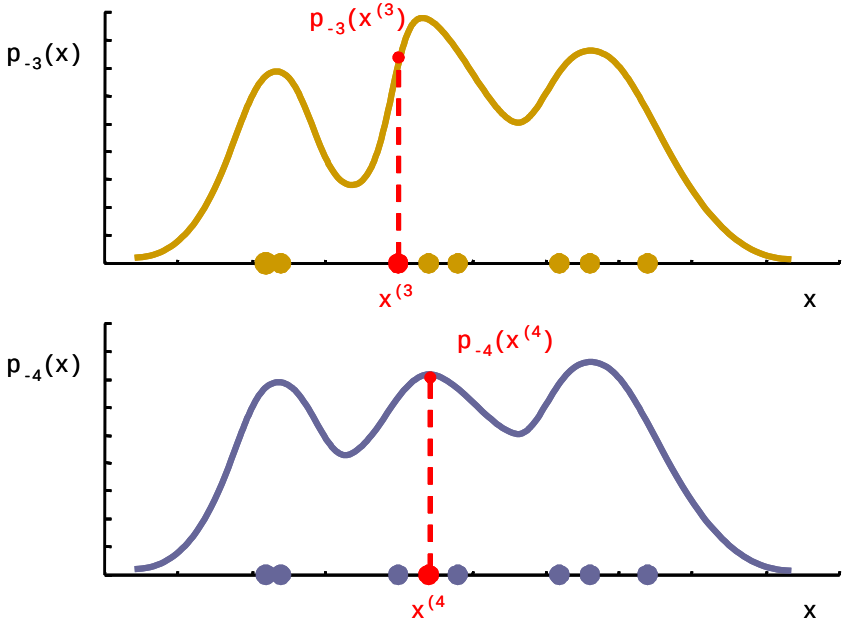
- The ML estimate of σ is degenerate since it yields $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\delta(x - x_i)$, a density estimate with Dirac delta functions at each training data point
- A practical alternative is to maximize the "pseudo-likelihood" computed using leave-one-out cross-validation

$$\sigma_{ML} = \arg\max_{\sigma}\left\{\frac{1}{n}\sum_{i=1}^{n}\log \hat{f}_{-i}(x_i)\right\}$$

$$\log \hat{f}_{-i}(x_i) = \log\frac{1}{(n-1)\sigma}\sum_{\substack{j=1 \\ j\neq i}}^{n} K\left(\frac{x_i - x_j}{\sigma}\right)$$

[Silverman, 1986]


p-1(x)
p-1(x(1))
x(1
x
p-2(x)
p-2(x(2)
x(2
x


p-3(x)
p-3(x(3))
x(3
x
p-4(x)
p-4(x(4)
x(4
x

Multivariate density estimation

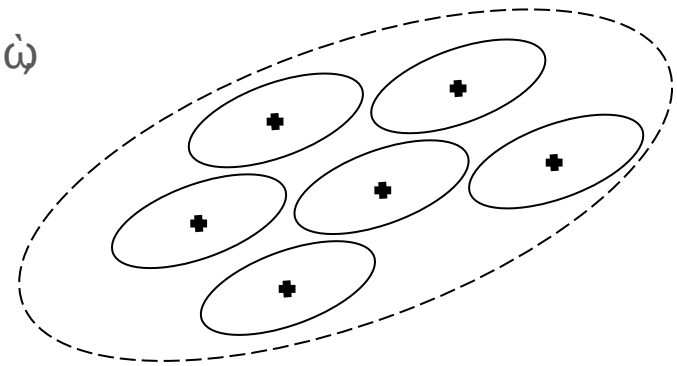
For the multivariate case, the KDE is

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

- 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 $\Sigma = I$), estimate the density, and then transform back [Fukunaga]
  - The whitening transform is $\mathbf{y} = \Lambda^{-1/2} U^T \mathbf{x}$ where Λ and U 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

$$\hat{p}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x}-\mathbf{x}_i)$$

$$K(\mathbf{x}-\mathbf{x}_i) = \frac{1}{h_1 h_2 \cdots h_d} \prod_{j=1}^{d} K_j\left(\frac{x_j - x_{ij}}{h_j}\right)$$

- The product kernel consists of the product of one-dimensional kernels
  - Typically the same kernel function is used in each dimension ($K_j = K$), 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 $K(\mathbf{x}-\mathbf{x}_i)$ uses kernel independence, this does not imply we assume the features are independent
  - If we assumed feature independence, the DE would have the expression

$$\hat{p}(\mathbf{x}) = \prod_{j=1}^{d} \frac{1}{n}\sum_{i=1}^{n} K_j\left(\frac{x_j - x_{ij}}{h_j}\right)$$

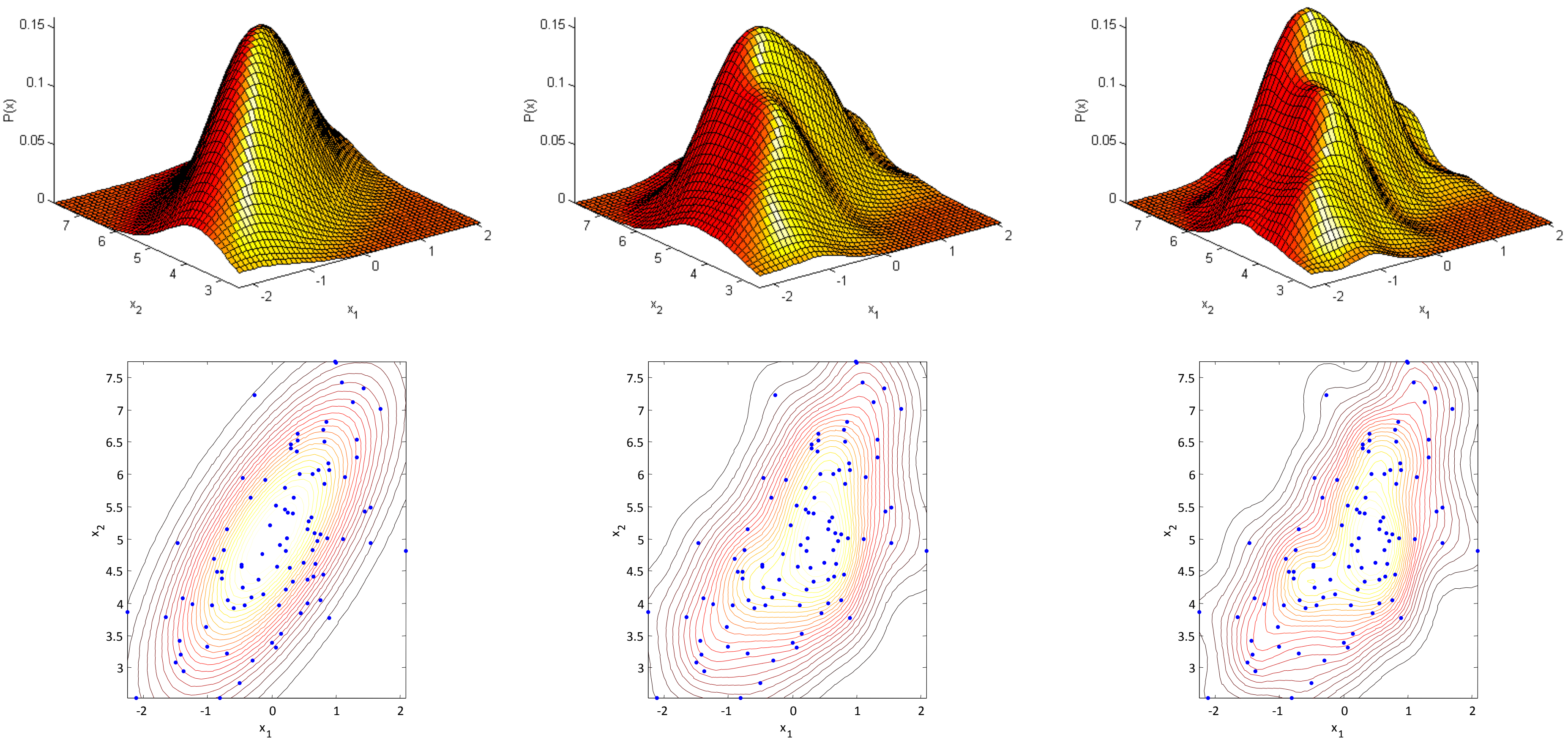
  - Notice how the order of the summation and product are reversed compared to the product kernel

Example I

ï This example shows the product KDE of a bivariate <u>unimodal</u> Gaussian

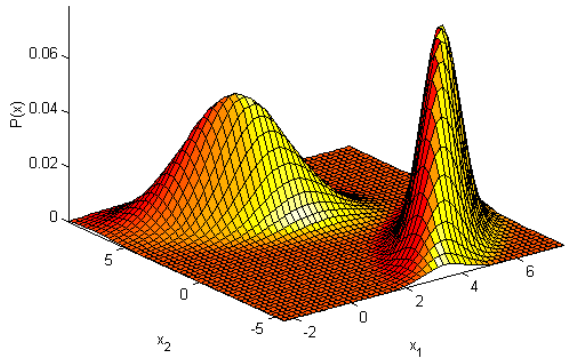
Å 100 data points were drawn from the distribution

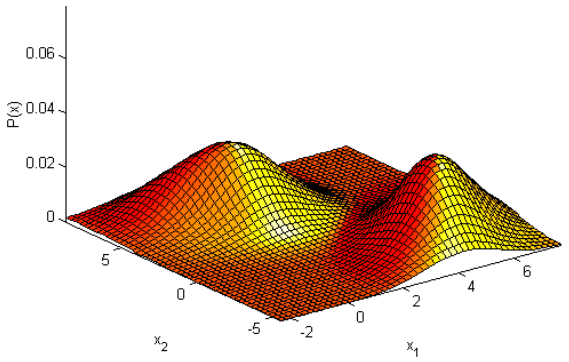
Å The figures show the true density (left) and the estimates using ΫΩسي ρ8ਛ φ ΰ ٩٥٣ (middle) and ΫΩسي π8ὢ ΰ ٩٥٣ (right)

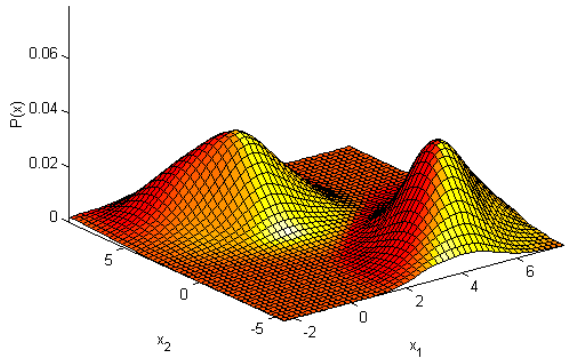


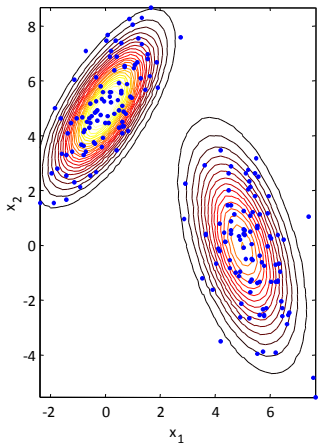
Example II

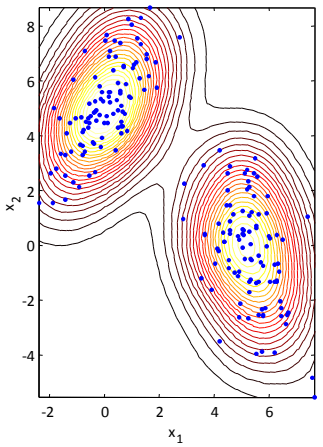
- This example shows the product KDE of a bivariate bimodal Gaussian
  - 100 data points were drawn from the distribution
  - The figures show the true density (left) and the estimates using $h = 0.25$ (middle) and $h = 0.5$ (right)

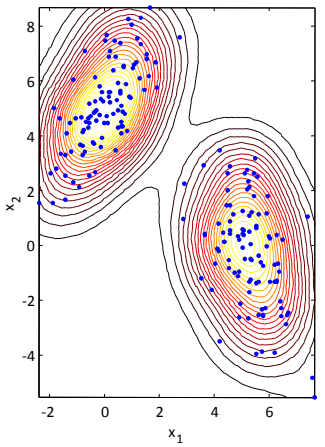












Naïve Bayes classifier

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

$$\omega_i : g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) \quad \text{or} \quad g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i)$$

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

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i) + \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i)$$

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

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

$$p(\mathbf{x}|\omega_i) = \prod_{j=1}^{d} p(x_j|\omega_i)$$

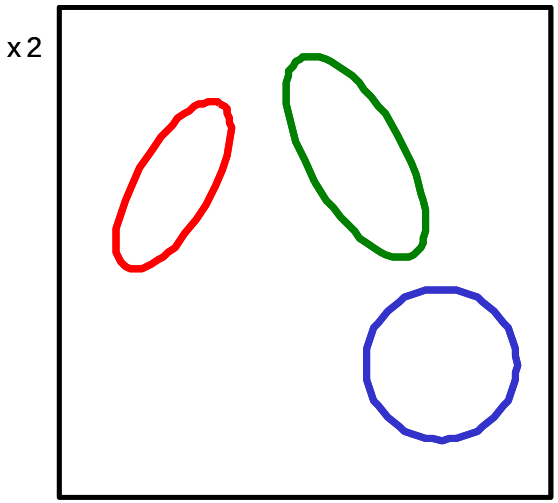
  - This assumption is not as rigid as assuming independent features $p(\mathbf{x}) = \prod_{j=1}^{d} p(x_j)$

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

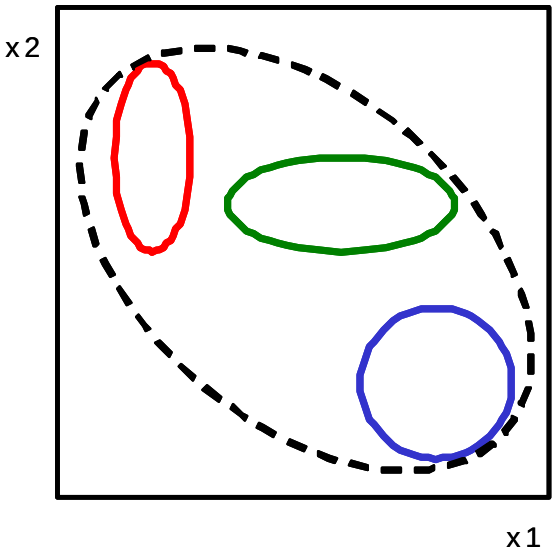
$$g_{NB_i}(\mathbf{x}) = P(\omega_i) \prod_{j=1}^{d} p(x_j|\omega_i)$$

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

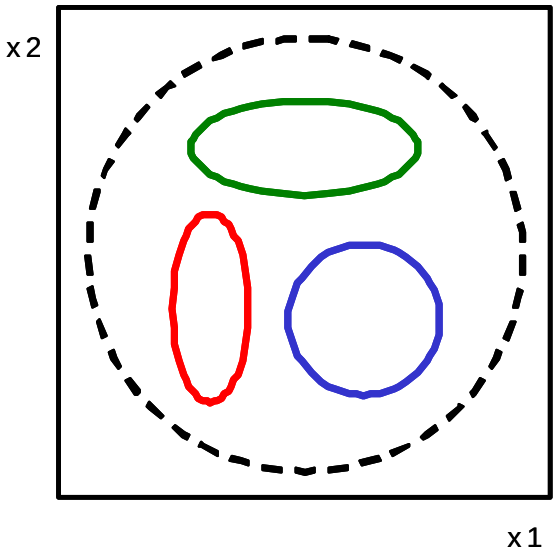
Class-conditional independence vs. independence



$p(x_1, x_2) = p(x_1)\,p(x_2)$



$p(x_1, x_2) \neq p(x_1)\,p(x_2)$
$p(x_1) = p(x_1 \mid y)$



$p(x_1, x_2) \neq p(x_1)\,p(x_2)$
$p(x_1) \neq p(x_1 \mid y)$