Machine Learning

A) Bayes Classifier
b) Nonparametric techniques

Cigdem Beyan

A. Y. 2024/2025

Introduction

- A fundamental **statistical approach** for classification.
- Hypothesis:
  - The decision problem is cast in probabilistic terms,
  - All relevant **probabilities are known**.
- Goal:
  - Discriminate the different decision rules using the probabilities and the associated costs
    - Estimating the **joint probability *p(x, y)*** from the training data set

An Easy Example

- Let ω (*called state of nature*) is what we want to classify and it is to be probabilistically described
- There are **two classes** $\boldsymbol{\omega_1}$ and $\boldsymbol{\omega_2}$ which correspond to two possible states:

  a) $P(\omega = \omega_1) = 0.7$
  b) $P(\omega = \omega_2) = 0.3$

  → **a-priori** or **prior probability**

  - There are no measurements or observations available to help make a decision about which class ω belongs to.
- Decision rule:
  - Decide ω_1 if $P(\omega_1) > P(\omega_2)$; otherwise decide ω_2
  - Given the probabilities (70% for ω_1 and 30% for ω_2), the decision rule would lead to classifying ω as ω_1 since 0.7 is greater than 0.3.

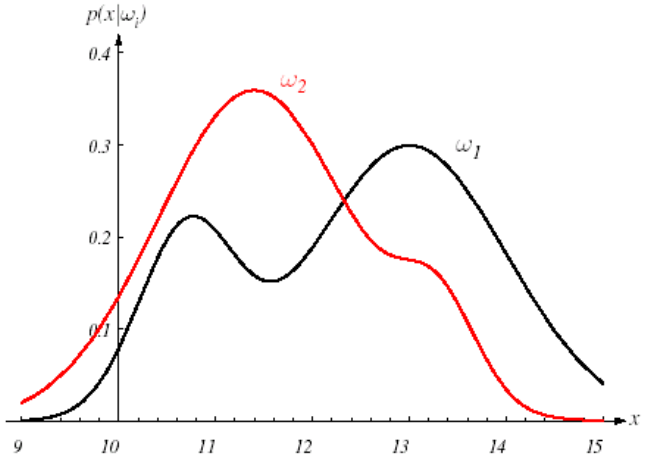
Another Example

- Having the previous hypothesis and additionally **a single measurement** ***x***, which is a random variable that depends on the class ω_j, we can get:

$$p(x \mid \omega_j)_{j=1,2}$$ = **Likelihood** *or* **class-conditional probability density function**

i.e. *the probability of having the measurement x knowing that the state of nature is* ω_j

- When we fix the measurement ***x***, the higher $p(x \mid \omega_j)$ is, the more likely ω_j is the correct state.



Bayes Formula

- Assuming known $\boldsymbol{P(\omega_j)}$ and $\boldsymbol{p(x|\omega_j)}$, the decision of ω (the state of nature) becomes, for Bayes

$$p(\omega_j, x) = P(\omega_j \mid x)\, p(x) = p(x \mid \omega_j) P(\omega_j)$$

that is

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)} \propto p(x \mid \omega_j) P(\omega_j)$$

where:

- $P(\omega_j)$ = **Prior**
- $p(x \mid \omega_j)$ = **Likelihood**
- $P(\omega_j \mid x)$ = ***Posterior***
- $p(x) = \sum_{j=1}^{J} p(x \mid \omega_j) P(\omega_j)$ = ***Evidence***

Bayes Decision Rule

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)} \quad \Longleftrightarrow \quad posterior = \frac{likelihood \times prior}{evidence}$$

- *The posterior or* ***a-posteriori probability*** *is the probability that the state of nature is ω_j given the observation x.*
- The most important factor is the product *likelihood* × *prior*; the evidence $p(x)$ is simply a scale factor, which ensures that

$$\sum_j P(\omega_j \mid x) = 1$$

- From the formula of Bayes derives the **Bayes' decision rule**:
  Decide w_1 if $P(w_1 \mid x) > P(w_2 \mid x)$, and w_2 otherwise

Naïve Bayes Classifier

- A probabilistic classification algorithm based on based on Bayes' theorem.
- Assumes independence among features.
- Calculates the posterior probability using

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)}$$

- Assumes features $x_1, x_2, \ldots, x_n$ are independent given the class label ω:

$$P(x|\omega) = P(x_1|\omega) \cdot P(x_2|\omega) \cdots P(x_n|\omega)$$

- Probability of each class is determined from training data.

Naïve Bayes Classifier

- Classify an observation x to class ω_j if:

$$P(w_j|x) = \frac{P(x|w_j) \cdot P(w_j)}{P(x)} \text{ is maximized}$$

- **Advantages:**
  - Fast to train and predict, suitable for large datasets.
  - Works well with high-dimensional data.
- **Limitations:**
  - Independence assumption may not hold in real-world scenarios, leading to suboptimal performance.
  - Zero probability problem: requires techniques for unseen feature-class combinations.

Problems

- To create an optimal classifier that uses the Bayesian decision rule you need to know:
  - **Prior probabilities** $P(\omega_i)$
  - **Class-conditional densities** $p(\mathbf{x} \mid \omega_i)$

- The performance of a classifier <u>strongly</u> depends on the **goodness** of these components.

BUT PRACTICALLY ALL THIS INFORMATION IS NEVER AVAILABLE!!

Problems

- More often we only have:
  - A vague knowledge of the problem, from which to extract vague **a-priori probabilities**.
  - Some particularly representative patterns, training data, used to train the classifier (**often too few!**)

- Estimating a-priori probabilities is usually not particularly difficult.

- Estimating conditional densities is **more complex**.

Problems

- Conditional densities can be estimated by estimating the unknown parameters of known function $\boldsymbol{p(\mathbf{x} \mid \omega_j)}$
  - E.g., estimate the vector

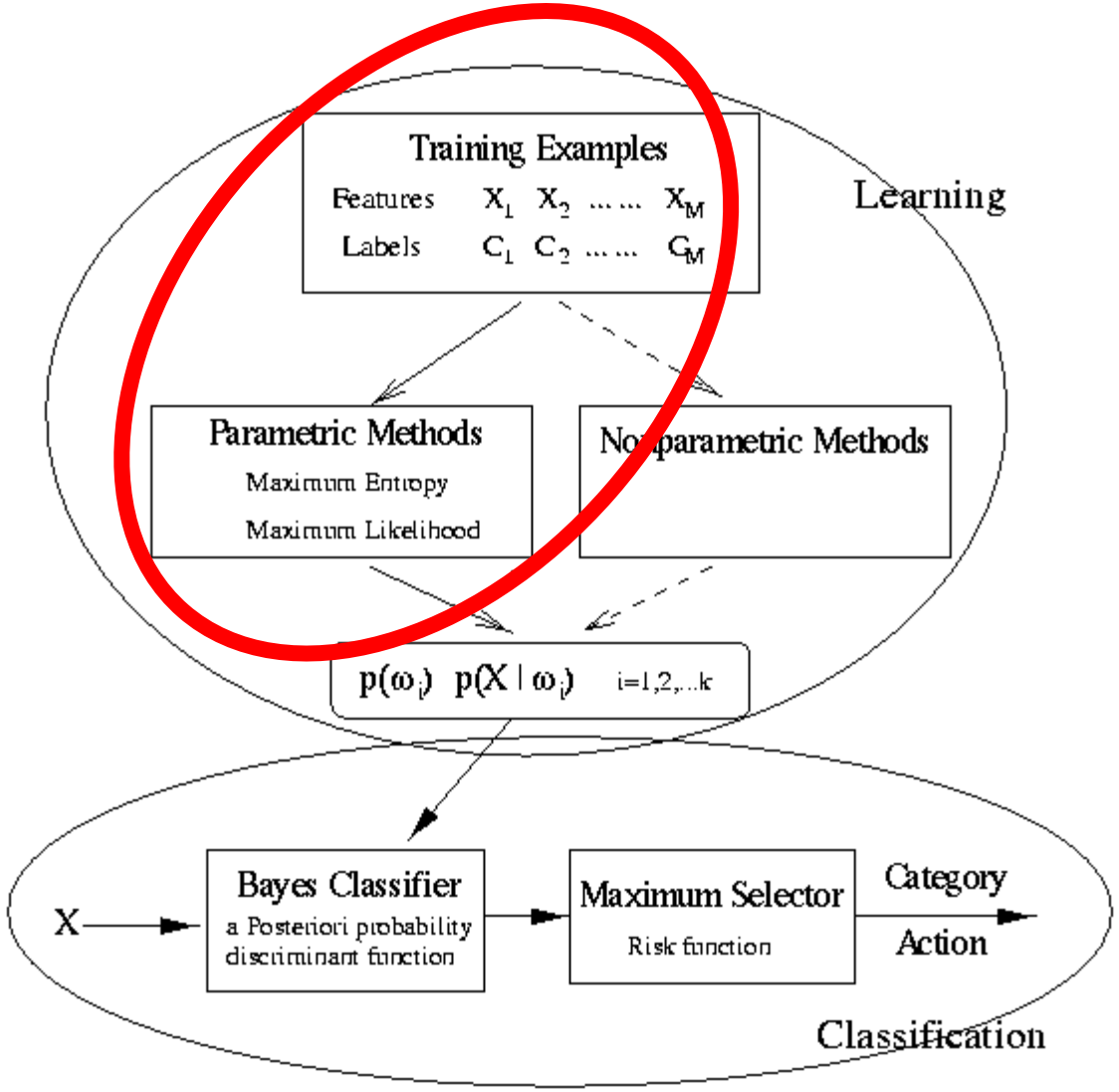
when $\boldsymbol{\theta}_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$

$$p(\mathbf{x} \mid \omega_j) \approx N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

Parameter Estimation

- Roadmap:
  - Estimate the parameters from the training data.
  - Use the resulting estimated as true values.
  - Use the Bayesian decision theory to build a classifier.

$$p(\mathbf{x} \mid \omega_j) \approx N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$



Parameter Estimation

- Suppose we have a set of *n training samples* and each has a class id (w_i).
- Then $P(\omega_i) = \frac{n_i}{n}$ where n_i is the number of samples with label w_i .
- Keep in mind that a-priori probabilities are not so useful as much as conditional densitites.
  - Parameter estimation to estimate the conditional densities ➔

Parameter Estimation

- Given a training set D={x_1, x_2, ..., x_n}
- $p(\mathbf{x} \mid \omega)$ is determined by $\boldsymbol{\theta}$, which is a vector representing the necessary parameters
  - such as $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if $p(\mathbf{x} \mid \omega) \approx N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
  - We need to find the best parameter $\boldsymbol{\theta}$ using the training set.

- To do so, we have two MAIN approaches
  - Maximum likelihood estimation (ML)
  - Bayesian estimation

2 APPROACHES

- **Maximum Likelihood approach**
  - Parameters are seen as quantities whose values are fixed but unknown.
  - The best estimation of their value is defined to be the one that **maximizes the probability** of obtaining the samples actually observed (training data).
  - Good to use it when we have sufficient data and want a frequent estimation approach.
  - Sensitive to overfitting, especially when the data is sparse.

2 APPROACHES

- **Bayesian approach**
  - Uses the full posterior distribution of the parameters rather than a single point like ML.
  - It involves integrating over the posterior distribution to make predictions.
  - Used when you want a full probabilistic model, including uncertainty about parameter values.
  - Provides a complete distribution over possible parameter values, offering more information than point estimates.
  - Computationally expensive!

Maximum Likelihood Approach

- Given the starting hypothesis and the samples of training set **D** are i.i.d. – independent and identically distributed, we have:

$$p(\mathbf{D} \mid \boldsymbol{\theta}) = \prod_{k=1}^{n} p(x_k \mid \boldsymbol{\theta})$$

- is a function of $\boldsymbol{\theta}$ when $p(\mathbf{D} \mid \boldsymbol{\theta})$ is called the likelihood of $\boldsymbol{\theta}$ w.r.t. the set of samples **D**.
- The maximum likelihood estimate of $\boldsymbol{\theta}$ is, the value $\hat{\boldsymbol{\theta}}$ that maximizes $p(\mathbf{D} \mid \boldsymbol{\theta})$;
- Keep in mind that $\boldsymbol{\theta}$ is fixed but unknown.

Maximum Likelihood Algorithm

- Define the likelihood function $p(\mathbf{D}\,|\,\boldsymbol{\theta}) = \prod_{k=1}^{n} p(x_k\,|\,\boldsymbol{\theta})$
- Calculate the log-likelihood $l(\theta) \equiv \ln\, p(D\,|\,\theta) = \sum_{k=1}^{n} \ln\, p(x_k\,|\,\theta)$
  - Notice that the product become a sum, which is easier to handle mathematically and computationally.
- Differentiate the log-likelihood: This step is needed to obtain the equations that we can solve to find the optimal $\boldsymbol{\theta}$
  - Take the derivative and set it to zero.

$$\nabla_\theta l(\boldsymbol{\theta}) = \sum_{k=1}^{n} \nabla_\theta \ln\, p(x_k\,|\,\boldsymbol{\theta}) \qquad \nabla_\theta l(\boldsymbol{\theta}) = 0$$

Maximum Likelihood Algorithm

- Solve the equation set to zero to find the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$. The solution gives the parameter values that make the observed data most likely.

- Let's apply maximum likelihood algorithm for Gaussian Distribution

Maximum Likelihood: Gaussian Distribution

- Suppose the samples are from a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
- For simplicity, let's first consider the case where only the mean $\boldsymbol{\mu}$ is unknown.
- Under this condition, we consider a sample point $\mathbf{x}_k$ and find:

$$\ln p(\mathbf{x}_k \mid \boldsymbol{\mu}) = -\frac{1}{2}\ln\left[(2\pi)^d |\boldsymbol{\Sigma}|\right] - \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}_k - \boldsymbol{\mu})$$

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

$$\nabla_{\boldsymbol{\mu}} \ln p(\mathbf{x}_k \mid \boldsymbol{\mu}) = \boldsymbol{\Sigma}^{-1}(\mathbf{x}_k - \boldsymbol{\mu})$$

Maximum Likelihood: gaussian Distribution

- Identifying $\boldsymbol{\theta}$ with $\boldsymbol{\mu}$, we see that the Maximum Likelihood estimate $\boldsymbol{\mu}$ must satisfy:

$$\sum_{k=1}^{n} \boldsymbol{\Sigma}^{-1}\left(\mathbf{x}_k - \hat{\boldsymbol{\mu}}\right) = 0$$

- Multiplying for $\boldsymbol{\Sigma}$ and rearranging the sum, we obtain

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k$$

that is the arithmetic **mean** of the training samples, sometimes written as $\hat{\boldsymbol{\mu}}_n$ to clarify its dependence on the number of samples.

Maximum Likelihood: Gaussian Distribution

- In the more general typical multivariate normal case, **neither the mean μ nor the covariance matrix Σ is known**.
- Let's consider first the univariate case with $\boldsymbol{\theta} = (\theta_1, \theta_2) = (\mu, \sigma^2)$
- The log-likelihood of a single point is

$$\ln p(x_k \mid \boldsymbol{\theta}) = -\frac{1}{2}\ln\left[2\pi\theta_2\right] - \frac{1}{2\theta_2}(x_k - \theta_1)^2$$

- and its derivative is:

$$\nabla_{\boldsymbol{\theta}} l = \nabla_{\boldsymbol{\theta}} \ln\ p(x_k \mid \boldsymbol{\theta}) = \begin{bmatrix} \frac{1}{\theta_2}(x_k - \theta_1) \\ -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} \end{bmatrix}$$

Maximum Likelihood: Gaussian Distribution

- Set both equations to be zero:

$$\sum_{k=1}^{n} \frac{1}{\theta_2}(x_k - \hat{\theta}_1) = 0 \qquad -\sum_{k=1}^{n} \frac{1}{\hat{\theta}_2} + \sum_{k=1}^{n} \frac{(x_k - \hat{\theta}_1)^2}{\hat{\theta}_2^{\,2}} = 0$$

where $\hat{\theta}_1$ and $\hat{\theta}_2$ are the ML estimates for θ_1 and θ_2.

- By substituting $\hat{\mu} = \hat{\theta}_1$ and $\sigma^2 = \hat{\theta}_2$ we obtain the ML estimates for mean and variance

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})^2$$

Bayesian Parameter Estimation

- In this case, we do not assume that $\boldsymbol{\theta}$ is fixed but instead it is a **random variable**.
- In this case the training dataset **D** allows us to convert a prior distribution $p(\boldsymbol{\theta})$ into a posterior probability density $p(\boldsymbol{\theta}|\mathbf{D})$.

$$p(\boldsymbol{\theta}) \Rightarrow p(\boldsymbol{\theta}|\mathbf{D})$$

- The computation of the posterior probabilities $P(\mathrm{w}_i|\boldsymbol{x})$ requires Bayesian classification, meaning that we need to know
  - Prior probabilities $P(\omega_i)$
  - Condiitonal densities $p(\boldsymbol{x}|\omega_i)$
- When these quantities are unknown, the best we can do is to compute $p(\boldsymbol{x}|\mathrm{w}_i)$ using ***all of the information at our disposal***.

Bayesian Parameter Estimation

- Given the training set D, the Bayes' formula then becomes:

$$P(\omega_i \mid \mathbf{x}, D) = \frac{p(\mathbf{x} \mid \omega_i, D) P(\omega_i \mid D)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D) P(\omega_j \mid D)}$$

- Assumptions:
  - Reasonably, $P(\omega_i \mid D) \Rightarrow P(\omega_i)$
  - Since we are treating the supervised learning case, the training set D can be partitioned into c subsets $D_1, D_2, \ldots, D_c$, with the samples in D_i belonging to ω_i
  - The samples belonging to D_i have no influence on the parameters of $p(\mathbf{x} \mid \omega_j, D)$ if $i \neq j$.

Bayesian Parameter Estimation

- Considering these assumptions, we obtain:

$$P(\omega_i \mid \mathbf{x}, D) = \frac{p(\mathbf{x} \mid \omega_i, D) P(\omega_i \mid D)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D) P(\omega_j \mid D)}$$

$$P(\omega_i \mid \mathbf{x}, D) = \frac{p(\mathbf{x} \mid \omega_i, D_i) P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D_j) P(\omega_j)}$$

Bayesian Parameter Estimation

- Because each class can be treated independently, we do not need distinctions among classes, so we can simplify our notation by **reducing to *c* different instances of the same problem**, i.e.:

$$P(\omega_i \mid \mathbf{x}, D) = \frac{p(\mathbf{x} \mid \omega_i, D_i) P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D_j) P(\omega_j)}$$

$p(\mathbf{x} \mid D)$

$p(\mathbf{x})$

- *Use a set of samples D, drawn independently according to the fixed but unknown probability distribution* $p(\mathbf{x})$, *to determine* $p(\mathbf{x}|D)$.

Bayesian Parameter Estimation

- We are realizing the calculation of $p(\boldsymbol{x}|D)$ to estimate $p(\boldsymbol{x})$, converting the problem of estimating a probability density to a problem of estimating a parameter vector.
  - Observe how $p(\boldsymbol{x}|D)$ is obtained via an implicit parameter model $\boldsymbol{\theta}$.
  - Consequently, we have

$$p(\mathbf{x}\,|\,D) = \int p(\mathbf{x}, \boldsymbol{\theta}\,|D)d\boldsymbol{\theta}$$

where the integration extends over the entire parameter space.

Bayesian Parameter Estimation

- Then $p(\mathbf{x} \mid D) = \int p(\mathbf{x}, \boldsymbol{\theta} \mid D) d\boldsymbol{\theta}$

$$= \int p(\mathbf{x} \mid \boldsymbol{\theta}, D) p(\boldsymbol{\theta} \mid D) d\boldsymbol{\theta}$$

- Since, by hypothesis, the selection of $\mathbf{x}$ is done independently from the training samples in D, given $\boldsymbol{\theta}$,

$$p(\mathbf{x} \mid D) = \int p(\mathbf{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid D) d\boldsymbol{\theta}$$

- That is, the distribution of ***p(x)*** is known completely once we know the value of the parameter vector $\boldsymbol{\theta}$.

Bayesian Parameter Estimation

$$p(\mathbf{x}\,|\,D) = \int p(\mathbf{x}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta}\,|\,D)d\boldsymbol{\theta}$$

- Links the desired class-conditional density ***p(x|D)*** to the posterior density $p(\boldsymbol{\theta}|\mathrm{D})$ through the unknown parameter vector $\boldsymbol{\theta}$.
- If $p(\boldsymbol{\theta}|\mathrm{D})$ peaks very sharply about some value $\hat{\boldsymbol{\theta}}$, then we obtain an estimation of the most likely vector, thus

$$p(\mathbf{x}|\mathrm{D}) \approx p(\mathbf{x}\,|\,\hat{\boldsymbol{\theta}})$$

- But this approach allows to **take into account the effects of all other models**, described by the value of the **integral function**, for ***all possible models***.

$$p(\mathbf{x}\,|\,D) = \int p(\mathbf{x}\,|\,\boldsymbol{\theta})p(\boldsymbol{\theta}\,|\,D)\,d\boldsymbol{\theta}$$

Comparisons: ML vs. Bayesian estimation

- ML gives us a point estimate $\hat{\boldsymbol{\theta}}$, instead, Bayes approach gives us a distribution on $\boldsymbol{\theta}$.
- ML and Bayes solutions are equivalent in the asymptotic limit of infinite training data.
- Practically, the approaches are different for various reasons:
  - Computational complexity: ML is less
  - Interpretability: ML is easy to interpret
  - Confidence in the prior information: Bayesian heavily relies on prior info.
  - Compromise between estimation accuracy and variance: ML prone to overfitting, leading to estimates with high variance.

Nonparametric Techniques

- The problem with the Bayesian classifier is that it is based on **prior** and **conditional probabilities**:
  - Typically, <u>unknown</u> and must be <u>estimated</u> from the data.

- There are three types of methods for estimating these densities:
  - **Parametric methods:** the shape of the density is assumed, its parameters are estimated from the data (e.g., gaussian) ➔ Done
  - **Non-parametric methods:** no assumption on density form, fully estimated by data (e.g. K-NN) ➔ NOW

Nonparametric Techniques

- Given a set of examples, model the density function of the data without making any assumptions about the form of the distribution (e.g., Gaussian distribution)
  - Estimate the probability density function directly from the samples.

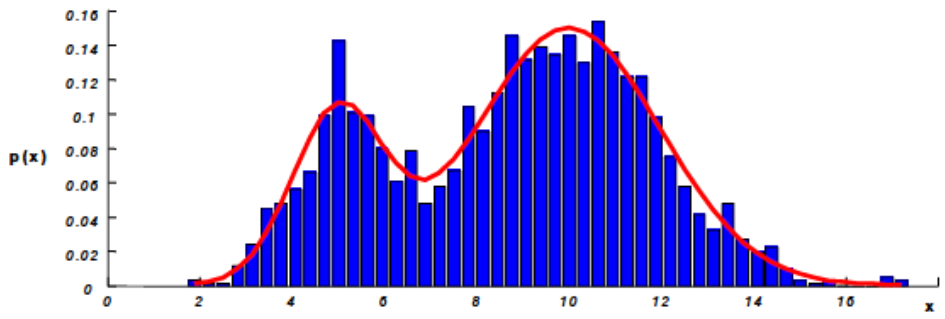
- The simplest form of non-parametric density estimation is the histogram.

Histogram

- Dividing 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,

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

- The histogram requires two "parameters" to be defined: **bin width** and **starting position** of the first bin.



Histogram

- 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 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**.

- **Curse of dimensionality** refers to the challenges and issues that arise when working with high-dimensional data in machine learning.
  - As the number of features or dimensions in a dataset increases, the amount of data required to effectively cover the feature space also increases exponentially.
  - To mitigate the curse of dimensionality, various techniques such as **feature selection**, **dimensionality reduction** (e.g., principal component analysis), and **regularization methods** are employed to reduce the dimensionality of the data while preserving its important characteristics.

Nonparametric Techniques

Before we proceed any further 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 region $\Re$ of the sample space is $P = \int_{\Re} P(x')dx'$
- Suppose now that *N* vectors {*x(1), x(2), …, x(N)*} are drawn ***independently and identically distributed (i.i.d.)*** from the distribution. The probability that ***k*** of these ***N*** vectors fall in $\Re$ is given by the binomial distribution

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

where $\binom{N}{k} = \frac{N!}{k! \cdot (N-k)!}$ and the expected value for k is E[k]=NP

Nonparametric Techniques

- The mean and variance of the ratio k/N are

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

- Therefore, as $N \to \infty$, the distribution becomes sharper, the variance gets smaller, thus we can obtain a better estimate of the probability P from the mean fraction of the points that fall within $\Re$

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

- 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 $\Re$.

Nonparametric Techniques

- Merging with the previous result we obtain

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

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

Nonparametric Techniques

- <u>In practice the value of ***N*** (the total number of examples) is fixed</u>
  - In order to improve the accuracy of the estimate ***P(x)*** we could let ***V*** to approach **zero** but then the region $\Re$ would become so **small** that it would enclose no examples.
  - This means that in practice we will have to find a compromise value of the volume ***V***,

    - Large enough to include enough examples within $\Re$
    - Small enough to support the assumption that P(x) is constant within $\Re$.

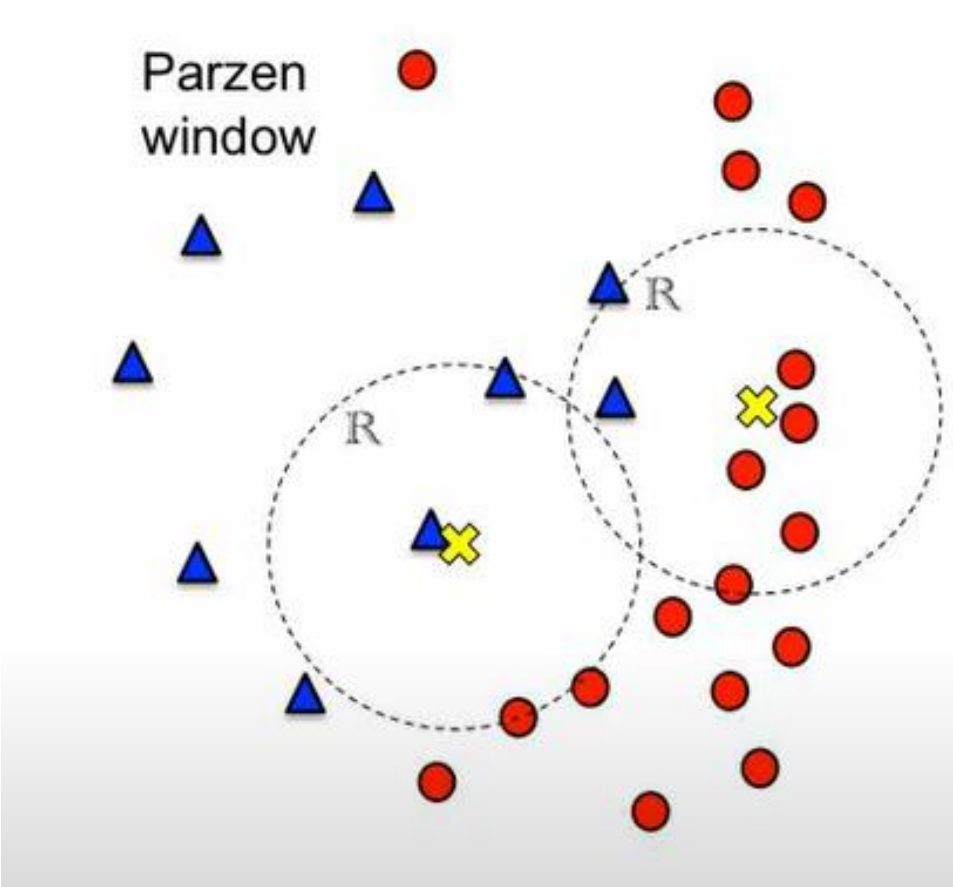
Nonparametric Techniques

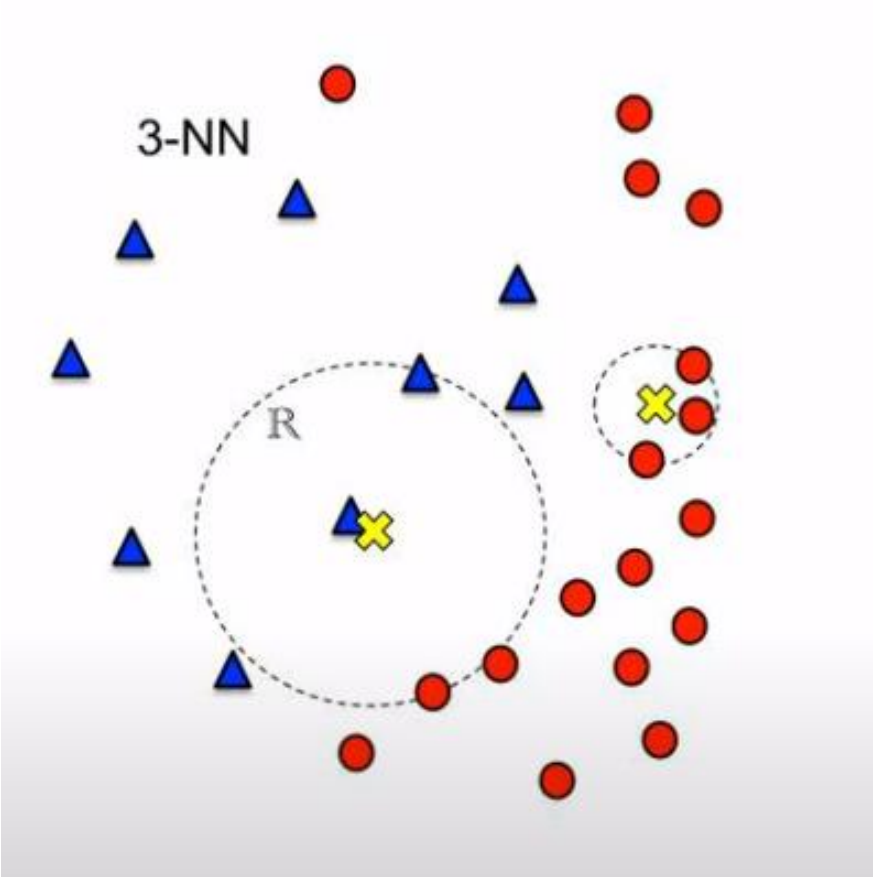
- The general expression for nonparametric density estimation is

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

- When applying this result to practical density estimation problems, two basic approaches can be adopted
  - Fix V and determine k from the data. This leads to *kernel density estimation (KDE)*, e.g. Parzen windows.
  - Fix k and determine V from the data. This gives rise to the *k-nearest-neighbor (k-NN) approach*.
- Both k-NN and KDE converge to the true probability density as $N \rightarrow \infty$, provided that V shrinks with N, and that k grows with N appropriately.

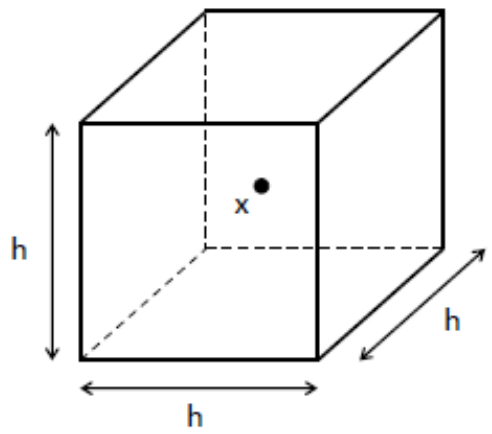
KNN VS. PARZEN WINDOWS





Parzen Windows

- Assuming 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 samples falling into 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 ***K(u)***, which corresponds to a unit hypercube centered at the origin, is known as a **Parzen window**
- The quantity $K((x-x^{(n)})/h)$ is then equal to 1 if $x^{(n}$ is inside a hypercube of side h centered on x, and zero otherwise.

Parzen Windows

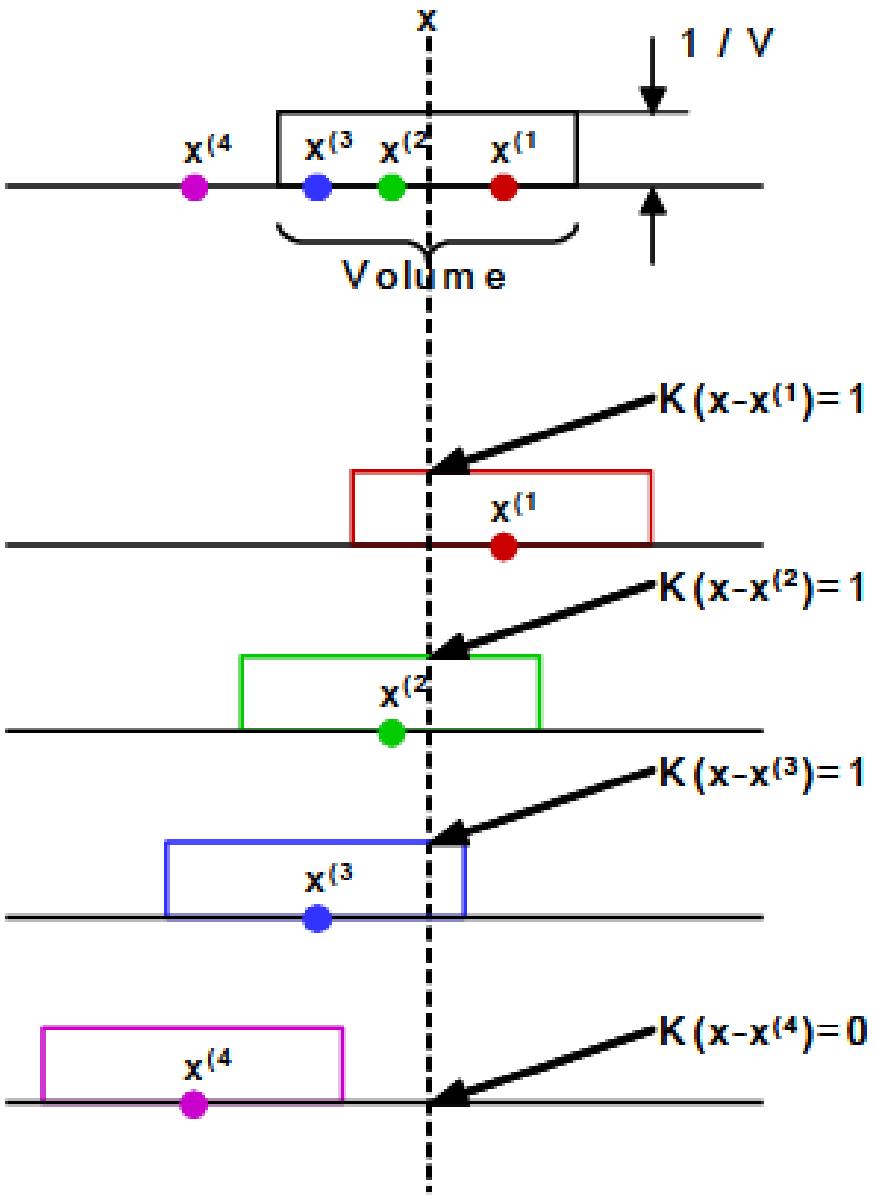
- The total # of samples inside the hypercube is then

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

- The density estimates become

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

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



Parzen Windows

- **What is the role of the kernel function?**
  - Let's compute the expectation of the estimate p_{KDE} (x)

  Recall expectation of *f(x)* is $\mathbb{E}[f(x)] = \sum_x f(x) \cdot p(x)$

  Then, 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 $p(x)$

- Notice that the expectation of p_{KDE} (x) is a convolution of the true density p(x) with the kernel function.
  - Kernel width $\boldsymbol{h}$ plays the role of a smoothing parameter such that the wider $\boldsymbol{h}$ is, the smoother the estimate p_{KDE} (x).

Parzen Windows

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

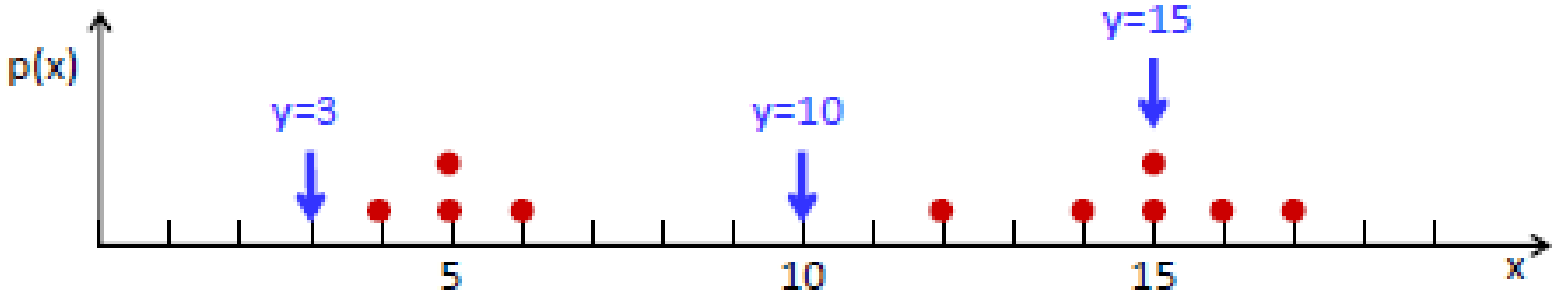
- Kernel width $\boldsymbol{h}$ plays the role of a smoothing parameter such that the wider $\boldsymbol{h}$ is, the smoother the estimate p_{KDE} (x).

- For $h \rightarrow 0$, 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.

Parzen Windows - 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



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

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

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

Parzen Windows - drawbacks

- It yields density estimates that have discontinuities
  - Sudden changes or jumps in value at certain points.
- Equal importance to all points x_i, regardless of their distance to the estimation point x.
- Therefore, the Parzen window is usually replaced with a **smooth kernel** function ***K(u),*** which is typically radially symmetric and unimodal probability density function
  - Such as Gaussian

Gaussian Parzen

Gaussian pdf:

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

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

In practice, given $\boldsymbol{n}$ is the number of data points, $\boldsymbol{x_i}$ represents each data point, $\boldsymbol{d}$ is the dimensionality of the data, $\boldsymbol{h}$ is the **bandwidth** parameter (also known as the window width or **smoothing parameter**). The Gaussian Parzen window estimate for a given point $\boldsymbol{x}$:

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(2\pi h^2)^{d/2}} \exp\left(-\frac{\|x - x_i\|^2}{2h^2}\right)$$

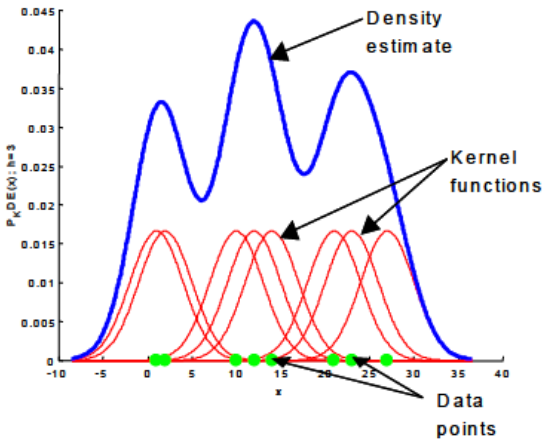
$\|x - x_i\|$:Euclidean distance

Gaussian Parzen

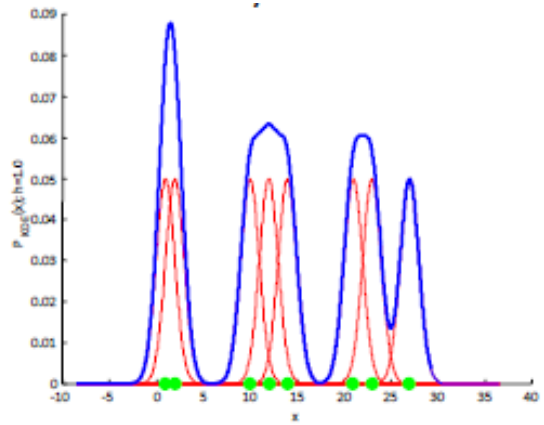
$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(2\pi h^2)^{d/2}} \exp\left(-\frac{\|x - x_i\|^2}{2h^2}\right)$$

- The bandwidth parameter h controls the smoothness of the estimate.
- A smaller h results in a more localized estimate with higher variance, while a larger h leads to a smoother estimate with lower variance but potentially more bias.
- Selecting an appropriate bandwidth is crucial for obtaining accurate density estimates.

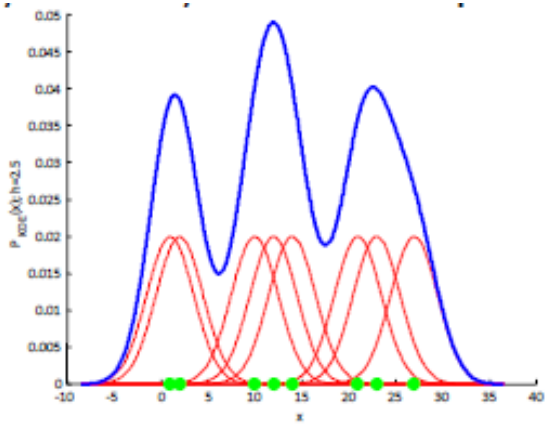
Parzen Windows – Bandwidth selection

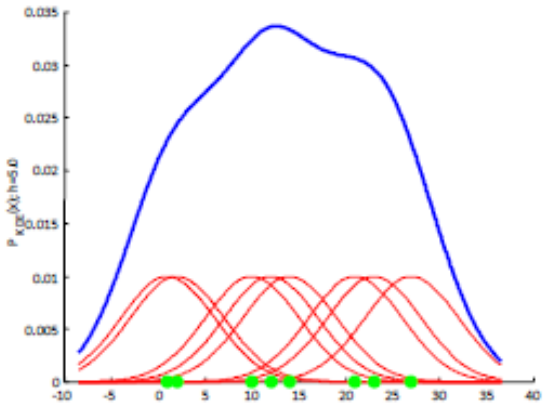


h=1

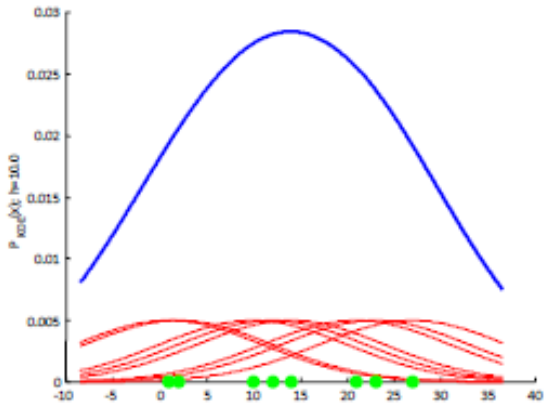


h=2.5





h=5



h=10

- A smaller h results in a more localized estimate with higher variance and very hard to interpret.
- A larger h leads to a smoother estimate with lower variance but mask the structure of the data.

Parzen Windows – Bandwidth selection

- So how to decide $\boldsymbol{h}$? (univariate case)
  - We would like to find a value of $\boldsymbol{h}$ that minimizes the error between the estimated density and the true density
  - The natural way for choosing $\boldsymbol{h}$ is to plot out several curves and choose the estimate that best matches one's prior (subjective) ideas
  - However, this method is not practical since we typically have high-dimensional data
  - Refer to standard distribution
    - Assume a standard density function and find the value of the bandwidth that minimizes the **integral of the square error (MISE)**

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

Parzen Windows – Bandwidth selection

- So how to decide h? (multivariate case)
  - The bandwidth h is applied uniformly across all axes. This means that the density estimate will treat each dimension equally, without considering differences in the spread or scale of individual features.
  - If some features have a larger spread or variability compared to others, using a single bandwidth parameter may not be sufficient.
  - Instead, it might be necessary to use a **vector of smoothing parameters** or even a **full covariance matrix**.
    - This approach takes into account the varying degrees of spread across different dimensions and allows for more flexible density estimation.

Parzen Windows – Bandwidth selection

- So how to decide h? (multivariate case)
  - Introducing a vector of **smoothing parameters** or a **full covariance** matrix complicates the bandwidth selection procedure. However, this can result in more accurate density estimates, especially when dealing with datasets with heterogeneous feature distributions.
  - Normalization is another technique that can help address the issue of unequal feature spread in multivariate kernel density estimation.
    - Normalization involves transforming the data such that each feature has a **similar scale** or **range of values**. This ensures that all features contribute equally to the density estimate, regardless of their original scale.

Nonparametric Techniques

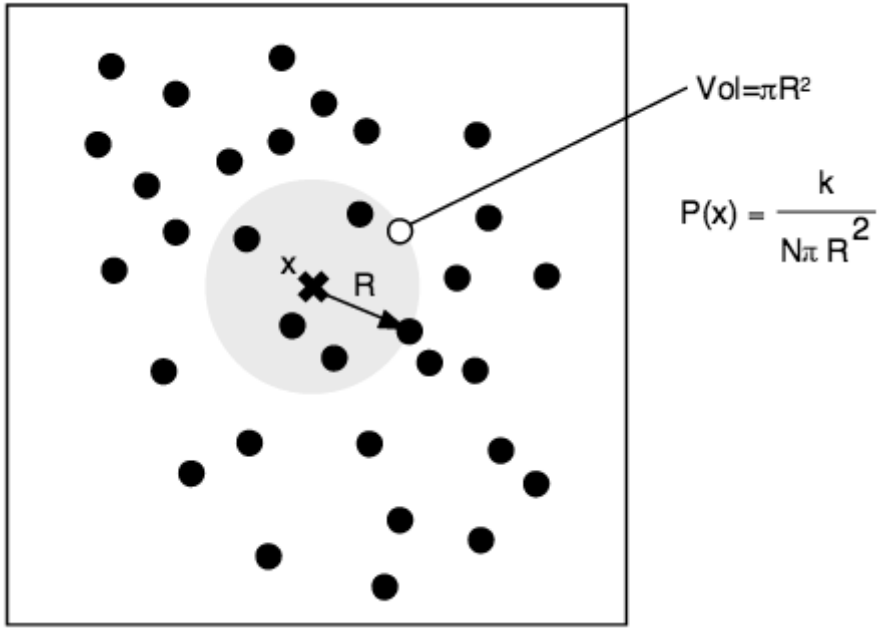
- The general expression for nonparametric density estimation is

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

- When applying this result to practical density estimation problems, two basic approaches can be adopted
  - Fix V and determine k from the data ➔ Parzen windows ✓
  - Fix k and determine V from the data. ➔ *k-nearest-neighbor (k-NN) approach*

K-Nearest Neighbors (k-NN)

- The **volume** is grown surrounding the estimation point *x*, until it encloses a total of ***k*** data points.



K-Nearest Neighbors (K-NN)

- Given an unknown point x^*, we consider $s_i \in \{s_1,\ldots,s_N\} = S$ the nearest point (NN) of the point x^* if

$$d(x^*, s_i) = \min_l d(x^*, s_l), \quad l = 1 \ldots N$$

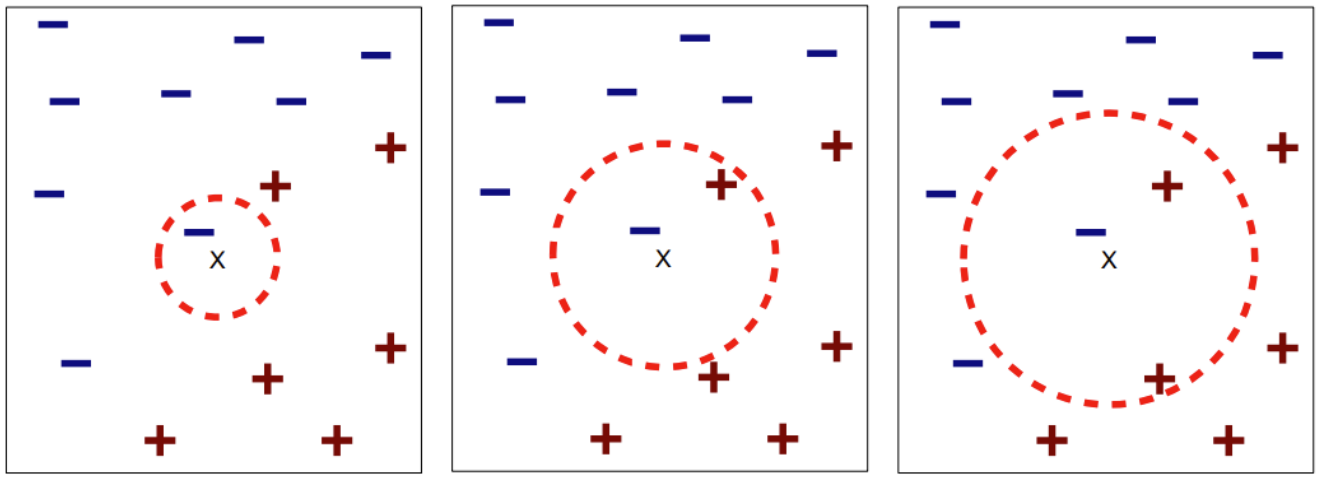
where $d(.)$ is a **distance measure** defined on the feature space.

- If you interpret each point of set S as a prototype of a class, you can then see the equation above as a rule of classification 1-NN associating the sample x to the class j to which the point s_i belongs.

K-Nearest Neighbors (K-NN)

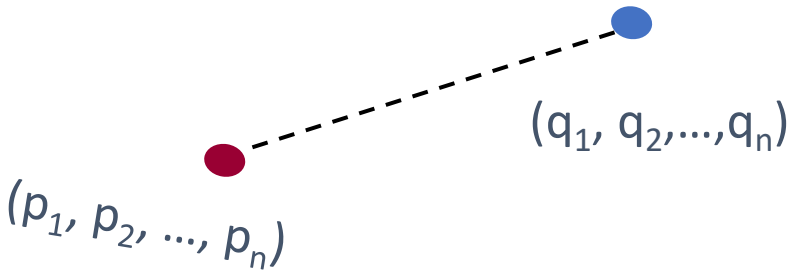
- Rule 1-NN can be generalized to define a k-NN rule, which consists of determining the k points belonging to the set S closest to the observation x.
- The classification rule k-NN associates observation x with class i having the largest number of elements among the nearest k.
- We call $U(x)$ the set of k points closer to x.
  - For example, with odd k and two classes, ω_1 and ω_2, the decision rule can choose the class with the most samples in $U(x)$. ➔ **majority voting**

K-Nearest Neighbors (K-NN)



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

- To calculate the distance, we usually use **Euclidean distance**.



$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + .. + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

K-Nearest Neighbors (K-NN)

- All the closest neighbors has the same weights, meaning that they are equally important to make a decision.
  - Therefore, the algorithm is very sensitive to the number of ***k***.

- This can be reduced by weighing the contribution of neighbors based on distance $w = 1/d(x^*, s_i)$, such that the closest neighbor would have more importance.

K-Nearest Neighbors (K-NN)

- The choice of k is important because:
  - If k is too small, the approach is sensitive to noise
  - If k is too large, the neighborhood can include examples belonging to other classes

- To operate correctly the attributes must have the same value scale and must therefore be normalized in the preprocessing phase

K-Nearest Neighbors (K-NN)

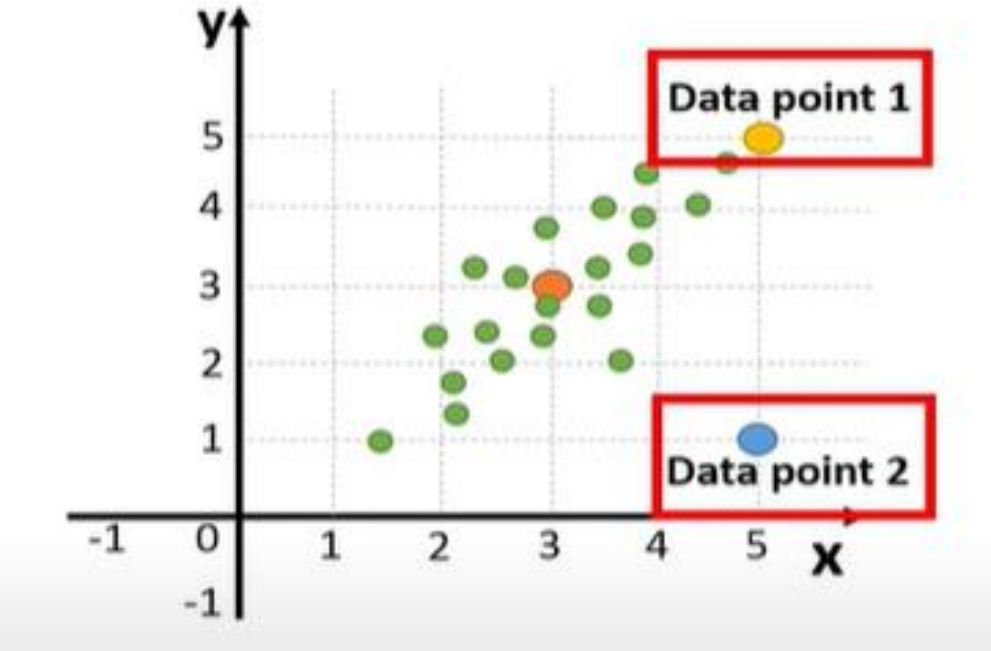
- Normalizing the scale may not be sufficient when there are different data distributions.
  - In such cases instead of using Euclidean distance, one can also try using Mahalanobis distance.
- The Mahalanobis distance is a measure of the distance between a point and a distribution. It's a generalized form of the Euclidean distance that accounts for correlations between variables and different variances along different axes.

$$D = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

- $\mathbf{x}$ is the n-dimensional vector representing the point's coordinates.
- $\boldsymbol{\mu}$ is the n-dimensional vector representing the mean of the distribution.
- $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix of the distribution.
- $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix.

K-Nearest Neighbors (K-NN)

- When to use Mahalanobis distance instead if Euclidean distance?
  - **Correlated Features:** changes in one feature are associated with changes in another.
  - **Unequal Variances:** MD normalized the distance based on the variances of the features, preventing features with larger variances from dominating the distance calculation.
  - **Outlier Detection:** Outliers often have a large MD from the rest of the data points, especially if they deviate from the normal covariance pattern.
  - In **high dimensional spaces**, ED can be problematic (***curse of dimensionality***). Incorporating information about the covariance structure, can mitigate this issue to some extent.



$$\text{Centroid} = \begin{pmatrix} \overline{X} \\ \overline{Y} \end{pmatrix} = \begin{pmatrix} 3.1 \\ 3.0 \end{pmatrix}$$

Euclidean distance between the centroid and data point 1:

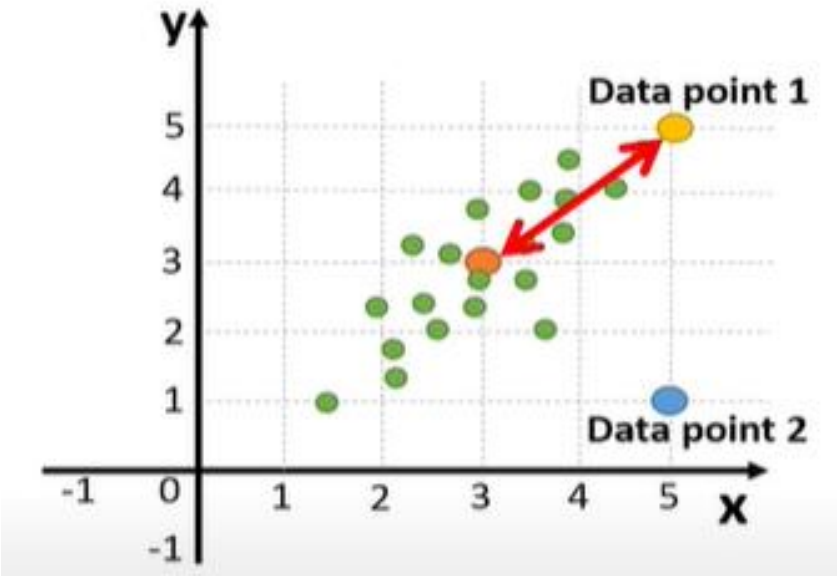
$$\boxed{d = \sqrt{(5-3.1)^2 + (5-3.0)^2} = 2.76}$$

Euclidean distance between the centroid and data point 2:

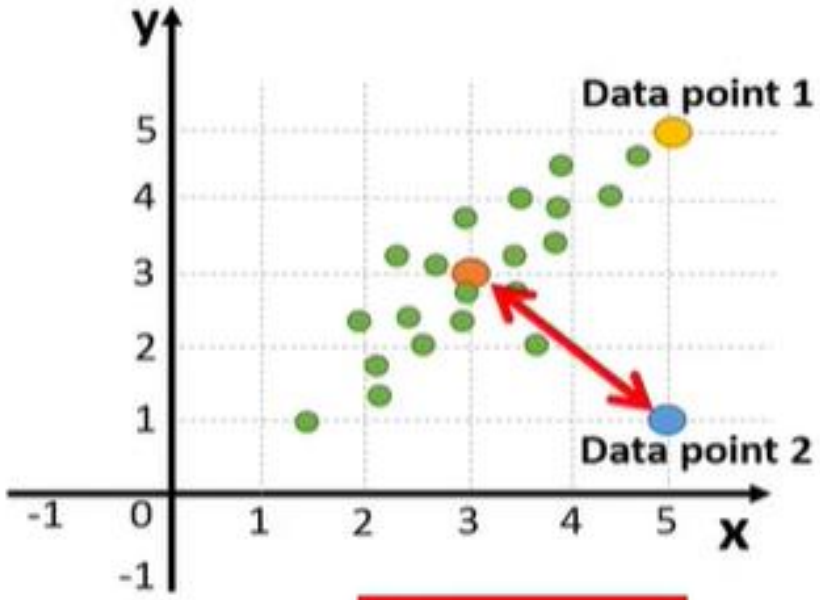
$$\boxed{d = \sqrt{(5-3.1)^2 + (1-3.0)^2} = 2.76}$$

- Based on Euclidean dist. both points have the same distance to the centroids

Mahalanobis distance



$$MD_1 = 2.26$$



$$MD_2 = 6.45$$

- Why MD_1 is so different from MD_2?

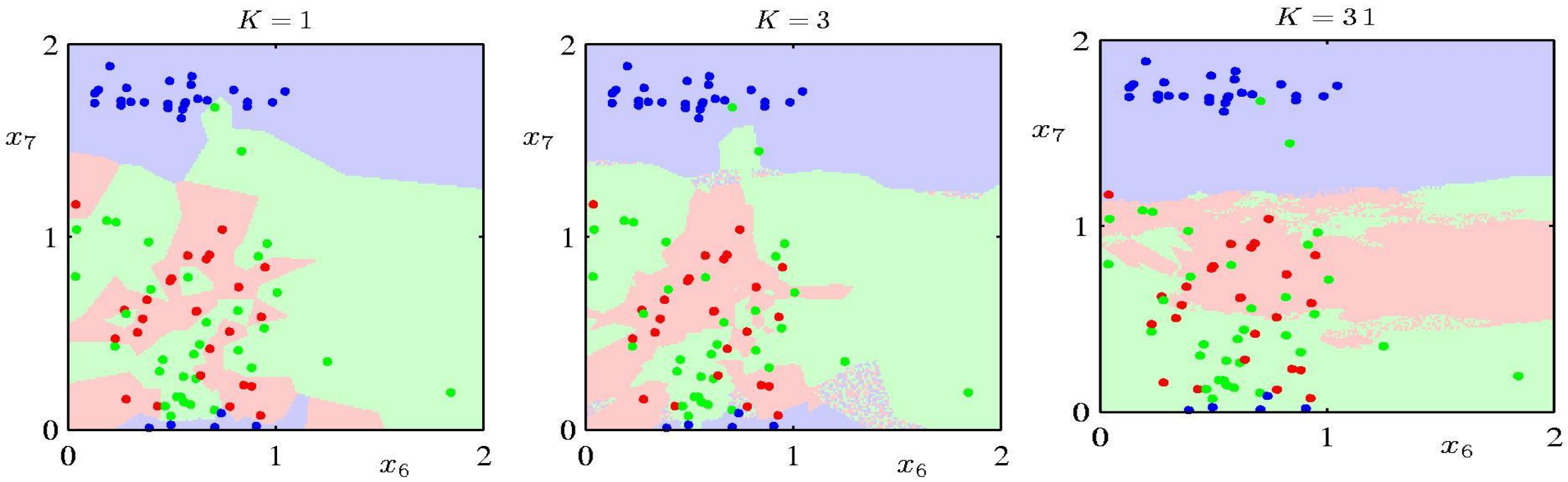
Bcs, MD takes into account the correlation in the data.

K-Nearest Neighbors (k-NN)

How to select ***k*** in k-NN?

- Common heuristics:
  - often 3, 5, 7
  - choose an odd number to avoid ties (*undetermined cases*)
- Use your training and validation data to decide ***k***.
  - Change ***k*** from ***k=1*** until the validation performance decreases.
  - Pay attention not to underfit and overfit the data.
  - Finalize the decision of ***k*** and use the same ***k*** to classify the test data.
- A typical choice is $k \cong \sqrt{N}$

How to decide K of K-NN?



What is the role of *k*?
How does it relate to overfitting and underfitting?

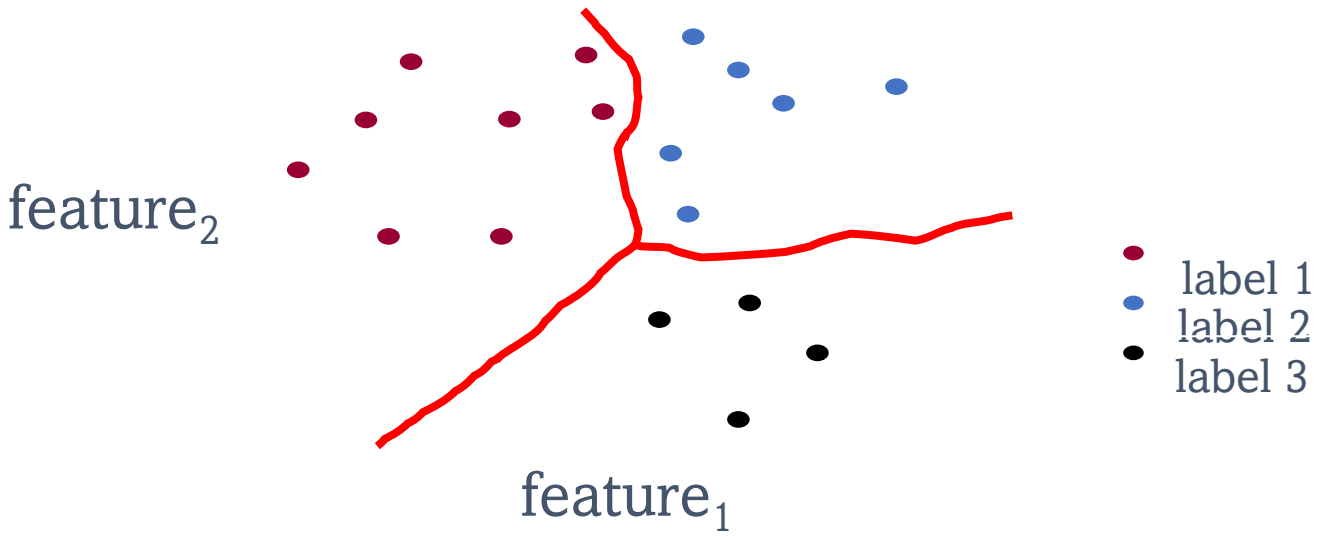
K-Nearest Neighbors (K-NN)

- If there are undetermined cases (ties) there are some alternatives:
  - Arbitrary choice
  - Assign x to the class (among those "tie") that has the nearest average sample (calculated between the k_i samples)
  - Assign x to the class that has the k_i samples with least distance from it

K-Nearest Neighbors (k-NN): pros

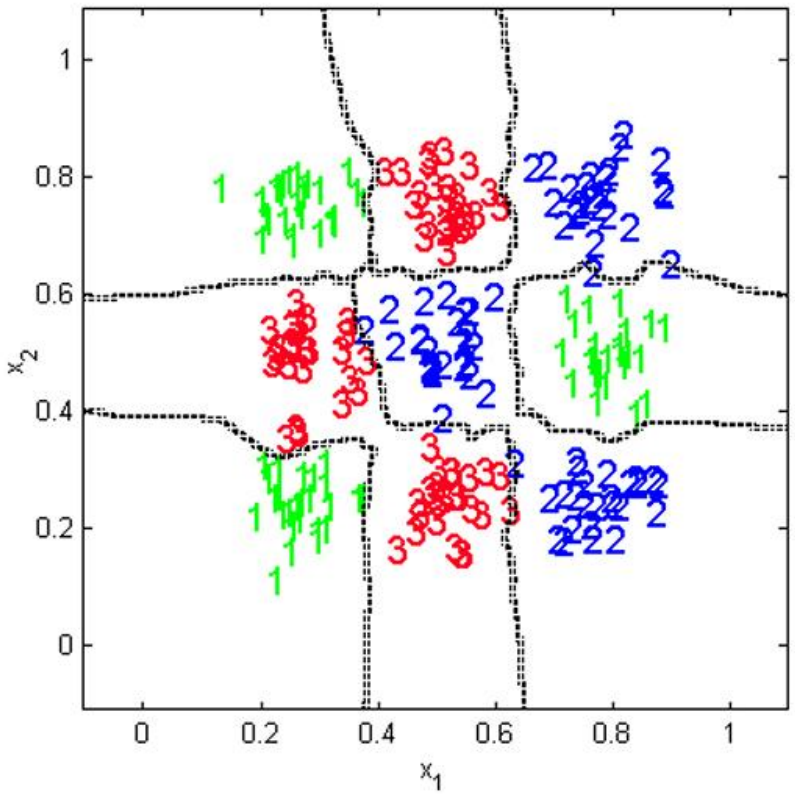
- Almost no assumptions about the data
- Non-parametric
- There is no training phase, cost is 0
- Allow us to construct **non-linear** class boundaries and are therefore more flexible.

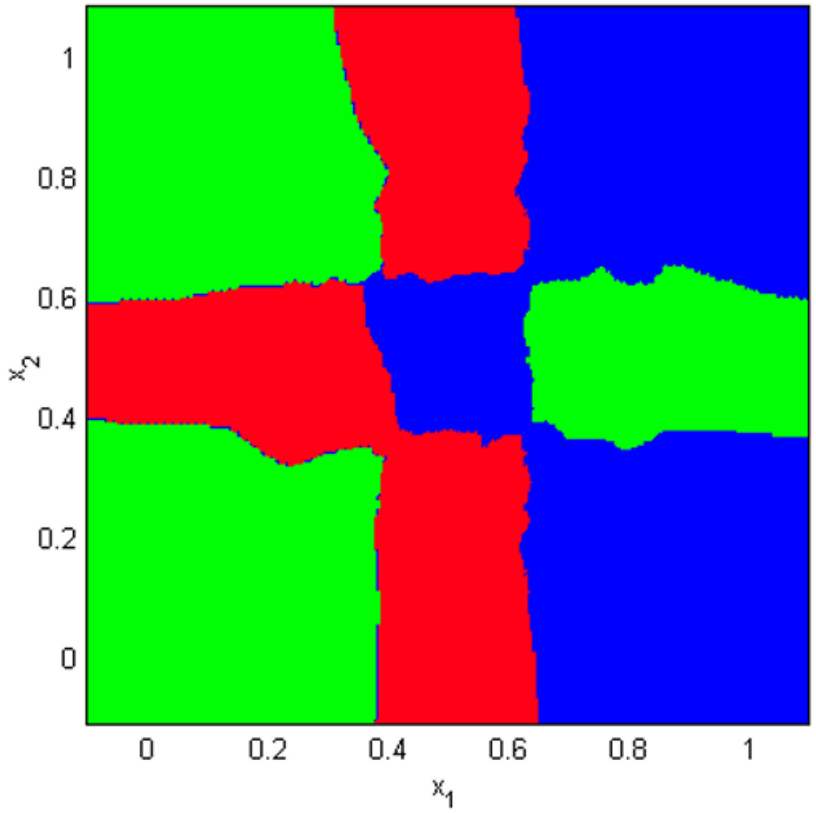
k-NN gives **locally** defined decision boundaries between classes



K-Nearest Neighbors (K-NN): pros

- Allow us to construct **non-linear** class boundaries and are therefore more flexible.





K-Nearest Neighbors (k-nn): cons

- Need to handle missing data.
- The distance metric should be chosen carefully.
- Sensitive to class outliers (e.g., mislabeled training instances)
- Sensitive to many irrelevant attributes (which influence the distance)
- Computationally expensive:
  - Poor scalability. For each new example, we have to perform as many comparisons as there are labeled data.
  - It is necessary to store all data points. Therefore, the method is more appropriate when the number of samples N is low.

How k-NN was born: pdf estimation

Going back to the pdf estimation…..

- Instead of fixing V as in the case of kernels, we fix the value of k and enlarge the radius of the sphere to include k samples
- Let's see the **a-priori probability**:
  - Given a class ω_j, the frequency of occurrence of samples N_j of class j is generally simply measured in relation to the total number of samples N, i.e.,

$$P(\omega_j) \equiv \hat{p}(\omega_j) = \frac{N_j}{N}$$

Density estimation and Nearest-Neighbor rule

- 2 classes, w_i and w_j, containing N_i and N_j samples, $N = N_i + N_j$
- The local density estimation for w_i is calculated as (and similarly for w_j)

$$\hat{p}(x|\omega_i) = \frac{1}{V}\frac{k_i}{N_i}$$

such that the ratio of k_i (k_j) points to the total N_i (N_j) belonging to the class w_i (w_j) contained in the volume V.

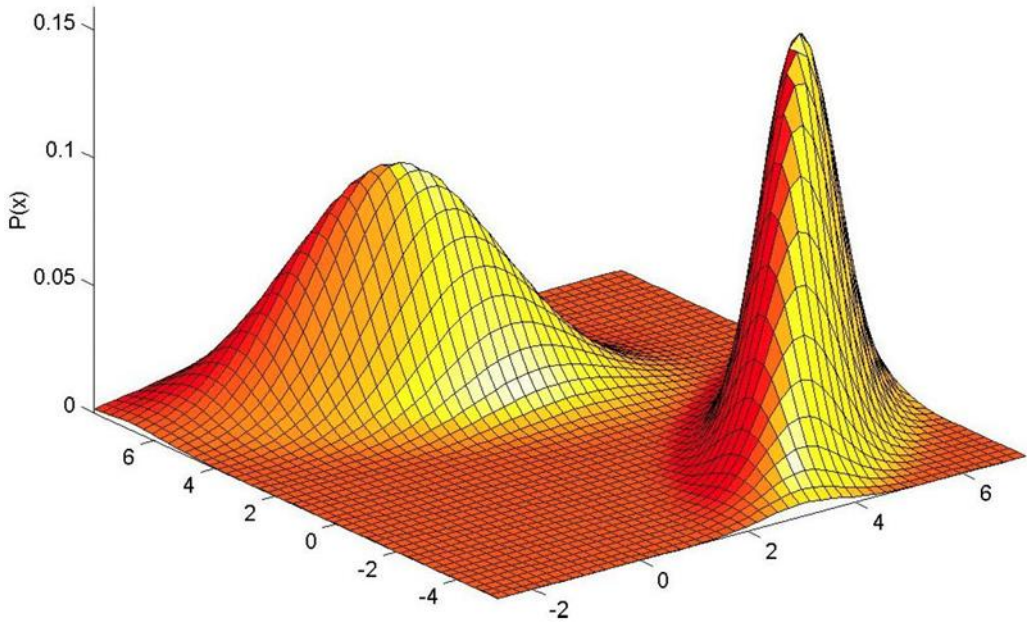
The Bayes rule says $p(x \mid \omega_i)P(\omega_i) > p(x \mid \omega_j)P(\omega_j)$, then

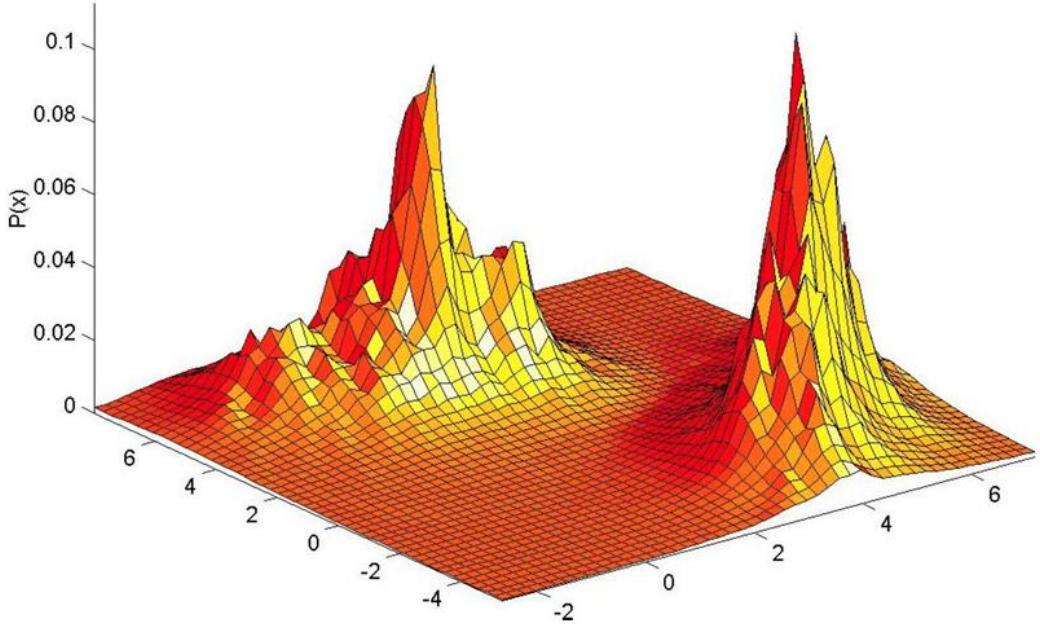
$$\hat{p}(x|\omega_i)\hat{p}(\omega_i) > \hat{p}(x|\omega_j)\hat{p}(\omega_j)$$

$$\Rightarrow \frac{1}{V}\frac{k_i}{N_i}\frac{N_i}{N} > \frac{1}{V}\frac{k_j}{N_j}\frac{N_j}{N} \quad \Rightarrow \quad k_i > k_j$$

Density estimation and Nearest-Neighbor rule

The true density, a mixture of two bivariate Gaussians





Density estimate for *k=10* neighbors and *N=200* examples.

- Notice that the estimate can be "jagged" and discontinuities in the slopes generally occur along lines away from the positions of the points themselves.

Cigdem Beyan
cigdem.beyan@univr.it
https://cbeyan.github.io/