

Non-parametric density estimation

Gosia Migut

Slides credit: David Tax

Admin stuff

- Registrations on Brightspace: 800+
- Lab week 2 downloads: 309
- Questions asked on Thursday: 39
- Lab questions on the exam!

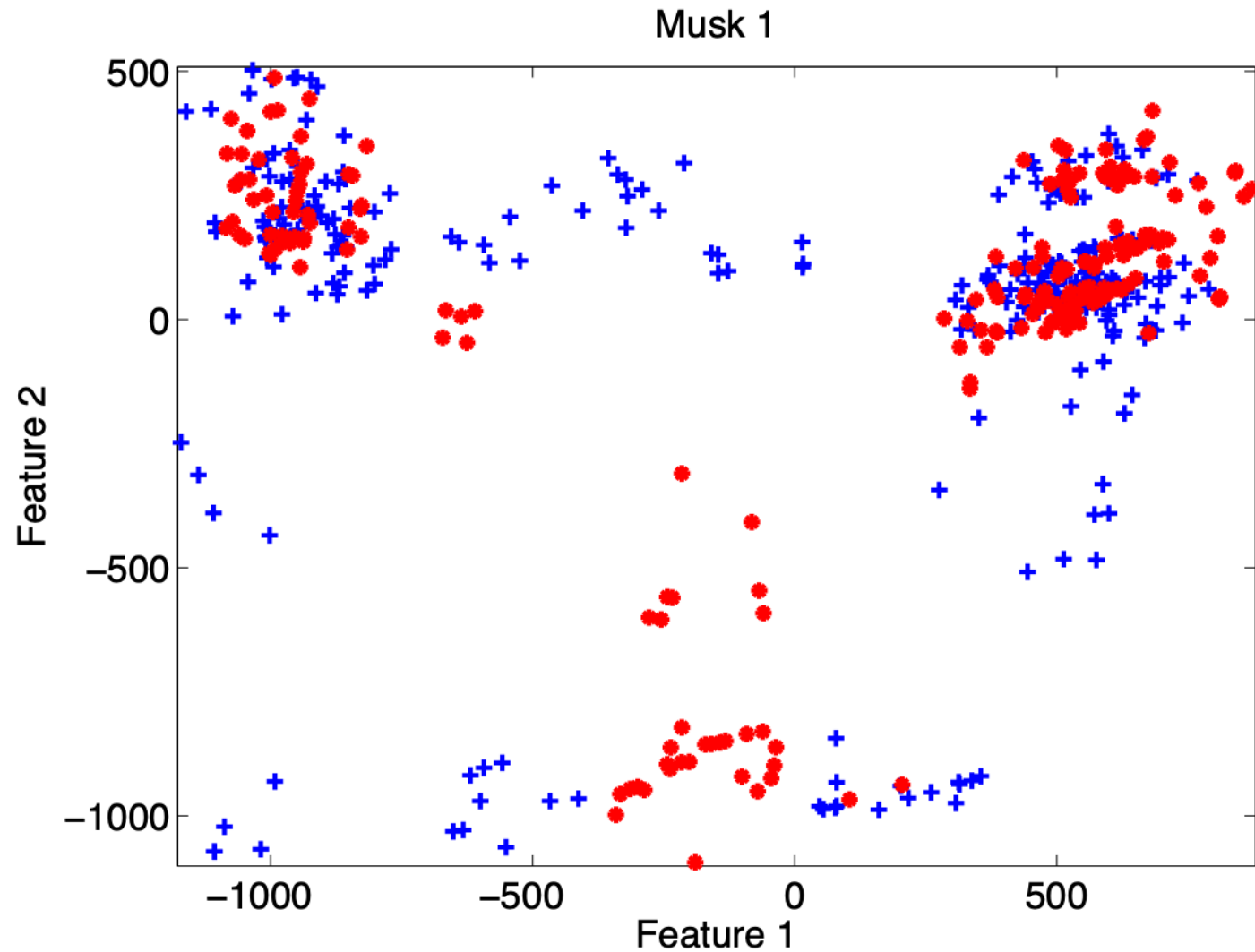
Recap last week

- Parametric density estimation
 - known distribution
 - estimate the parameters on training set
- Assume a single Gaussian distribution for each of the classes:

$$\hat{p}(x|y_i) = N(x|\mu_i, \Sigma_i)$$

$$N(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{p/2} |\Sigma_i|^{1/2}} \exp\left(\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)\right)$$

But in real life...

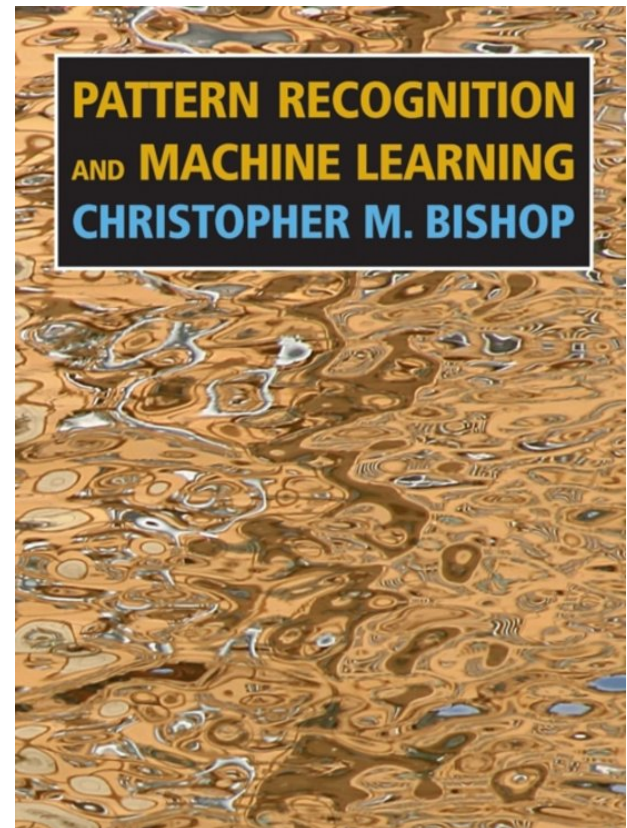


Non-parametric density estimation

- Non-parametric
 - no knowledge on the distribution
 - manage the smoothness of the distribution
- Popular non-parametric algorithms are:
 - Parzen, k-Nearest Neighbors (today)
 - Naïve Bayes (Friday)

Literature

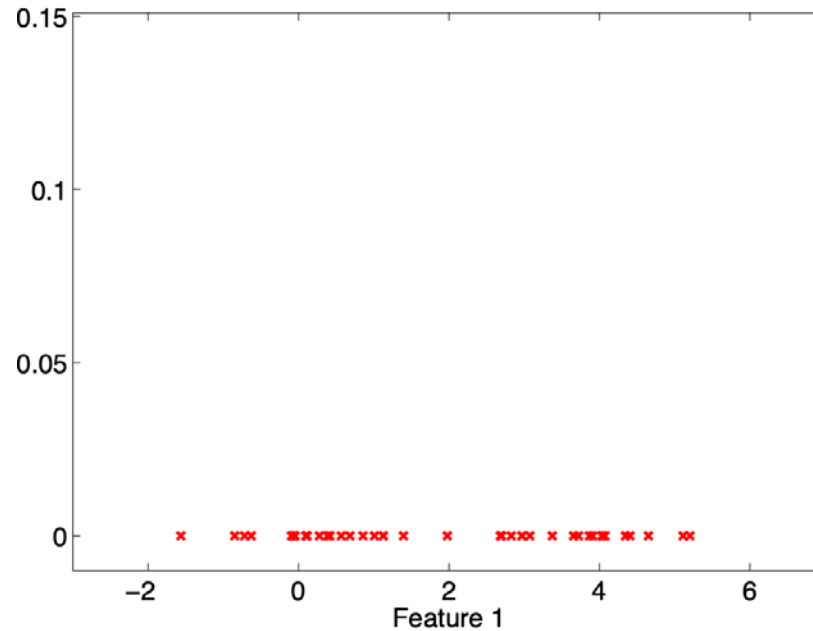
- Chapter 2 section 2.5 from:
Bishop (2006). Pattern Recognition and Machine Learning. Springer, UK.
- Available online



After practicing with the concepts of this lecture you should be able to:

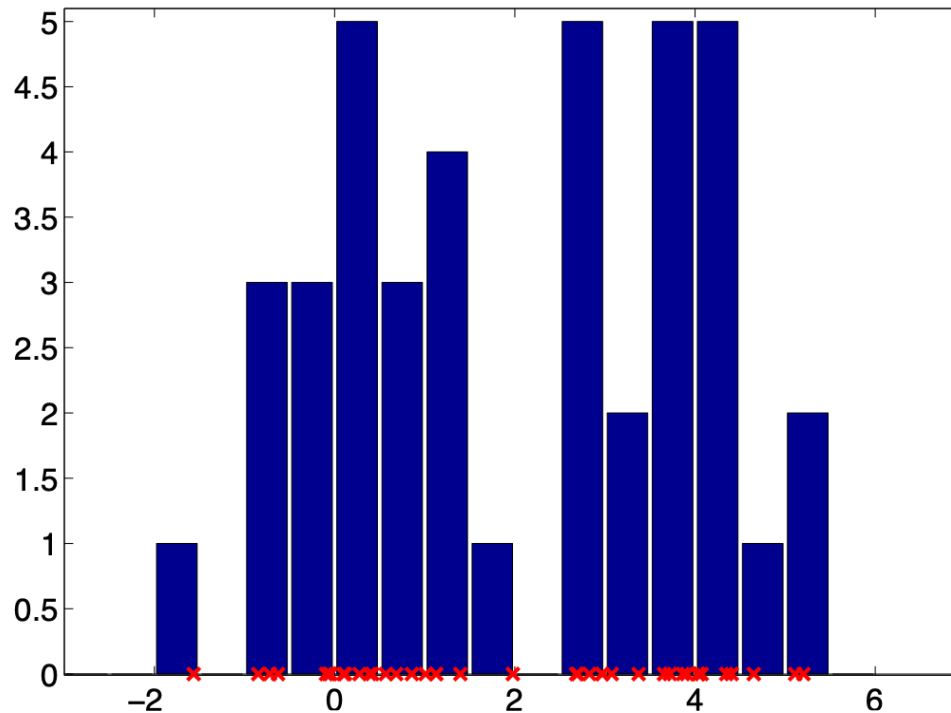
- Explain the difference between parametric and non-parametric density estimation
- Explain Parzen density estimation and classification
- Explain k-nn density estimation and classification
- Explain the advantages and disadvantages of Parzen and k-nn
- Implement k-nn classifier in Python

Histogram



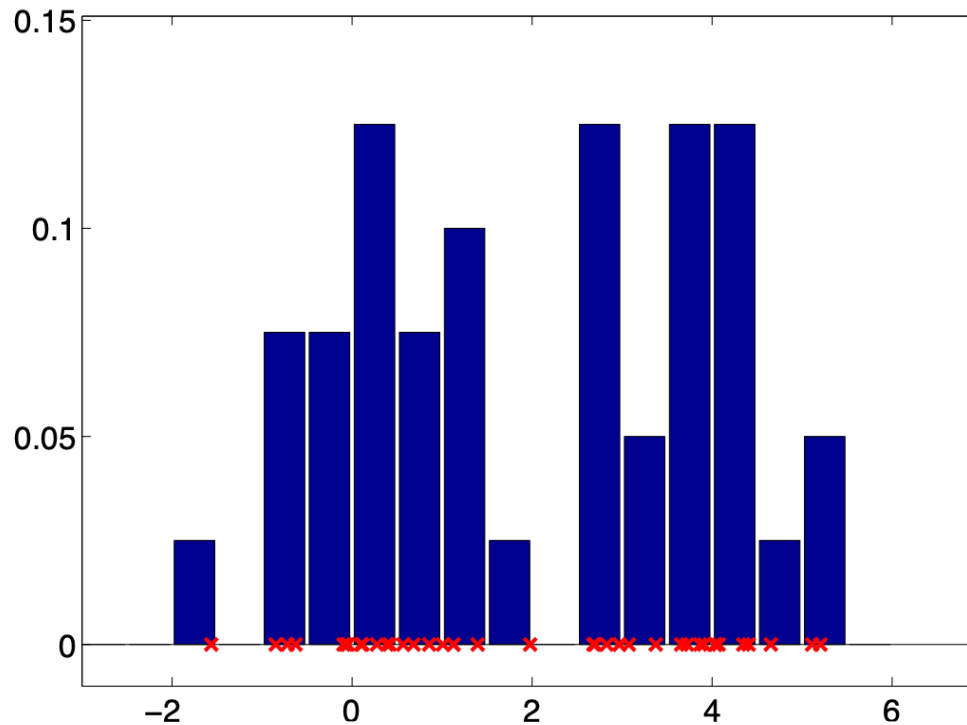
- Example: we have one feature
- We have 40 examples: how to estimate the probability density?

Histogram



- Split the feature in subregions: width h
- Count the number of objects in each region: k_N

Histogram method



- Probability density estimate: fraction of points/volume

$$\hat{p}(x) = \frac{1}{h} \frac{k_N}{N}$$

Histogram

- How many bins?
- For a reasonable precise approximation, h can't be too large
- For a stable estimate, h cannot be too small
- Depends on the number of training examples
- In practice, two very related methods are used:
 - Parzen density estimate
 - k-Nearest-neighbor density estimate

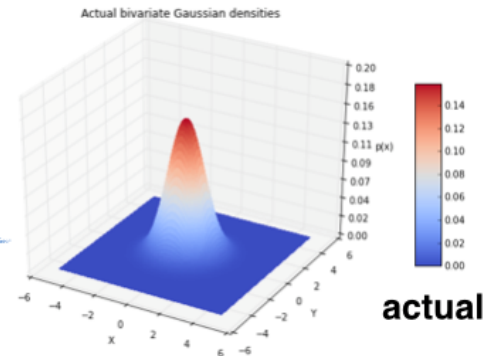
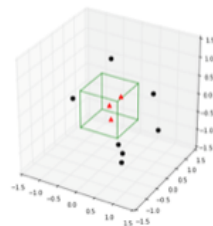
Parzen density estimation

Non-parametric density estimation

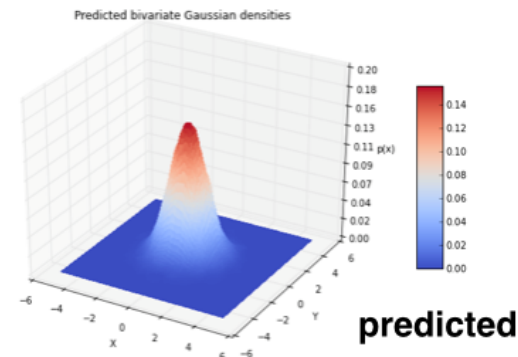
Our goal:



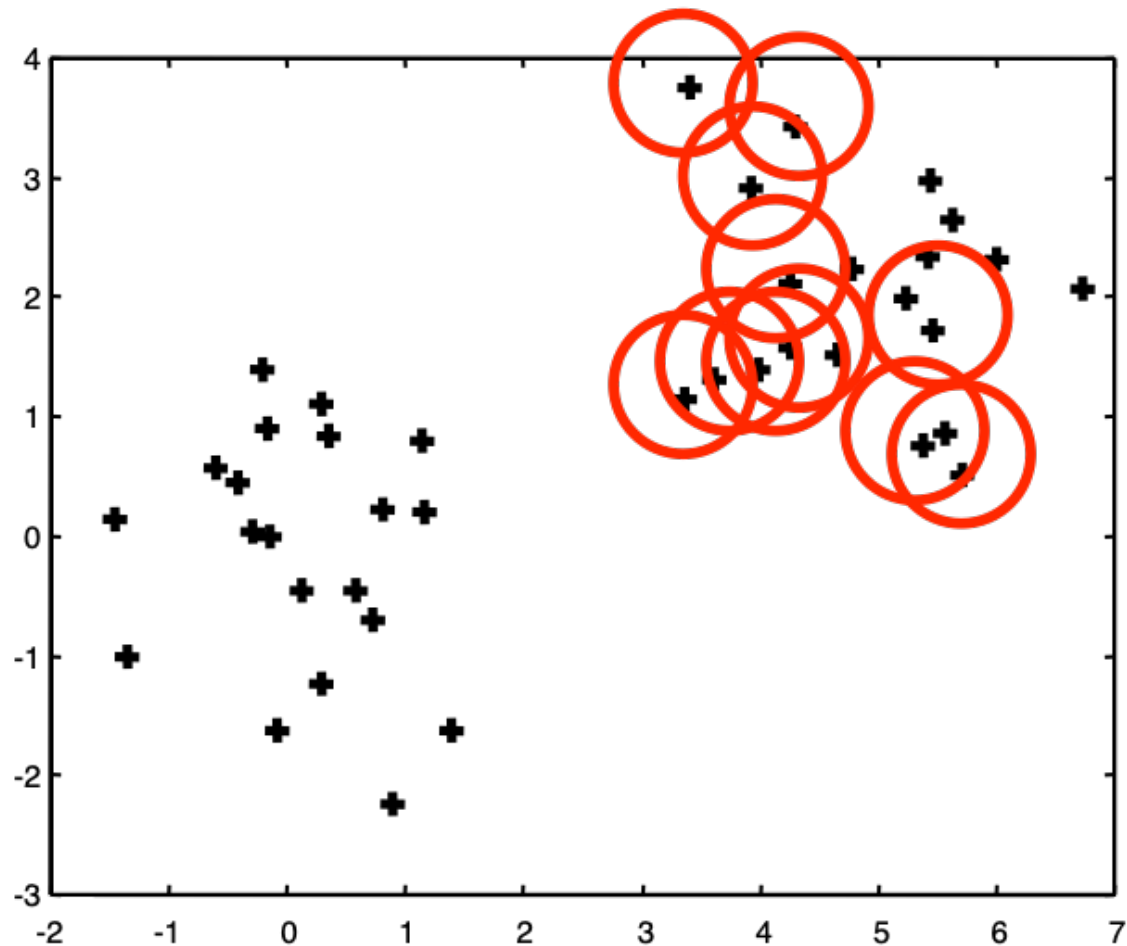
We want to **estimate** probability densities at certain points



Assuming we have samples drawn from a **unknown** distribution (here: bivariate Gaussian)

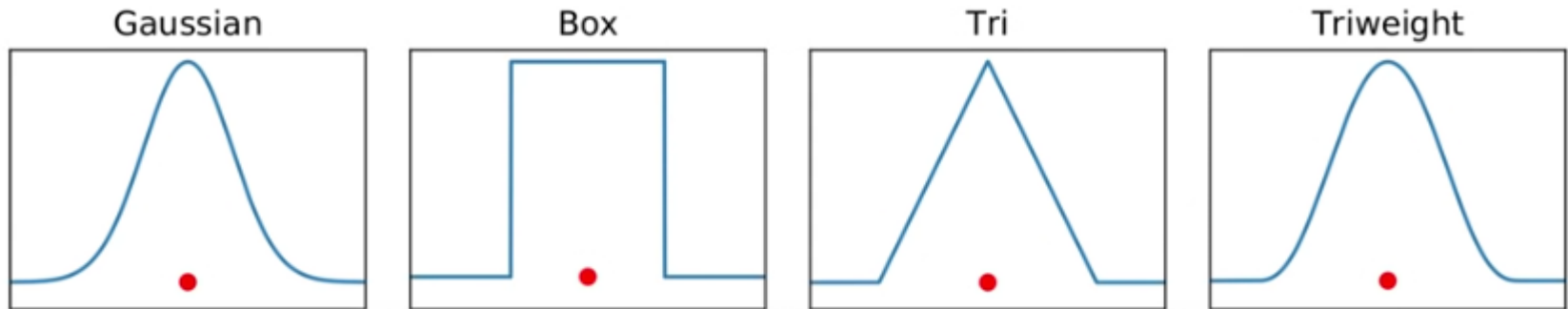
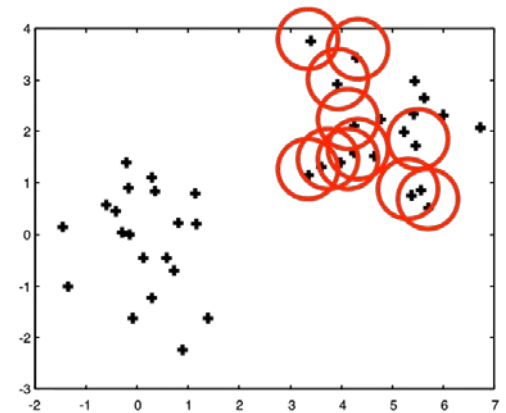


Parzen density estimation



Parzen density estimation

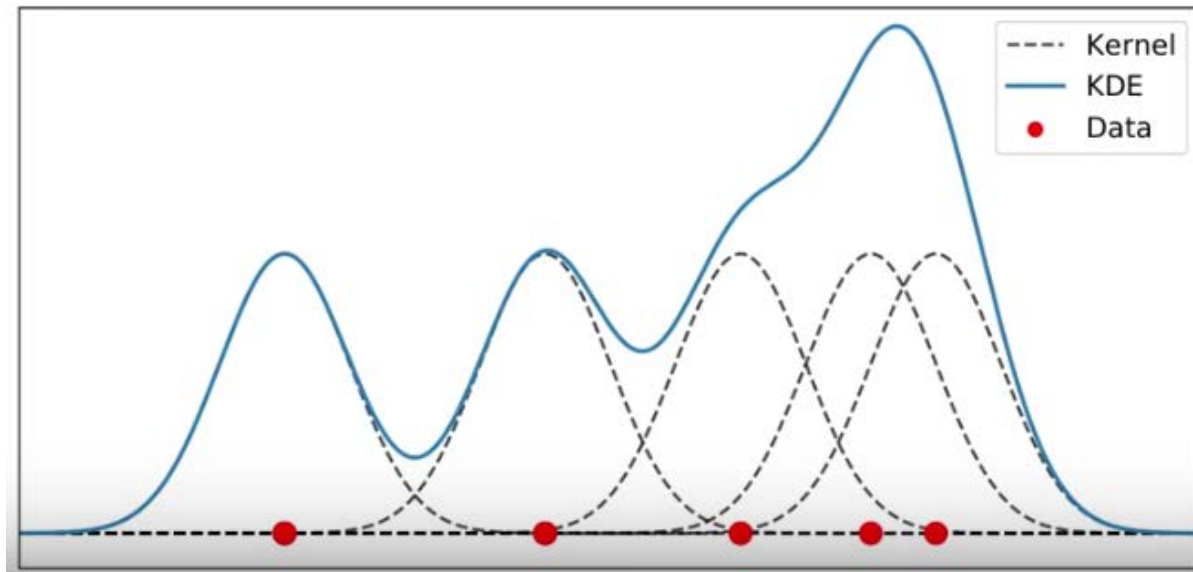
- Procedure:
 - Define cell shape (kernel) eg. Gaussian



- Fix size of cell (h)
- Add contributions of cells

Parzen density estimation

- On every datapoint x_i place a kernel function K .



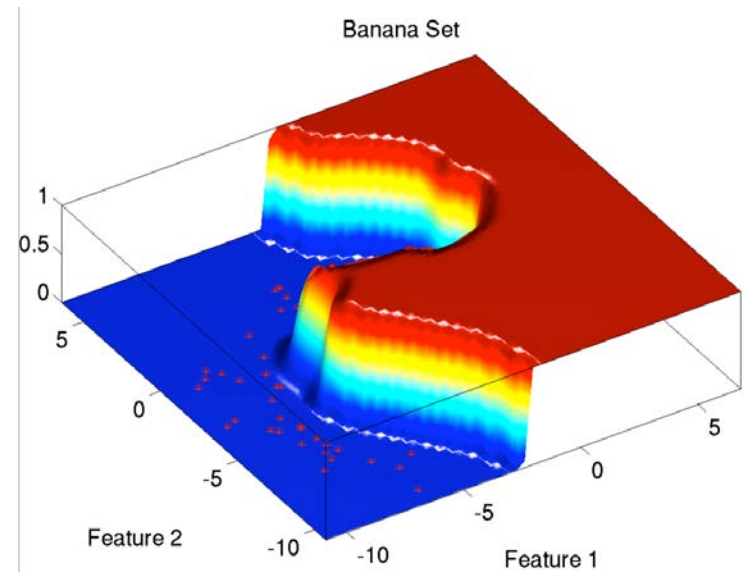
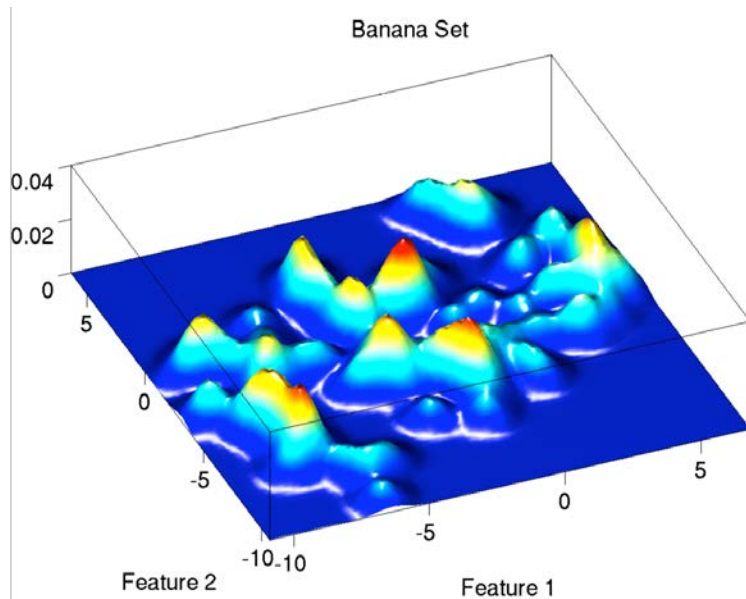
- Parzen density estimate is:

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

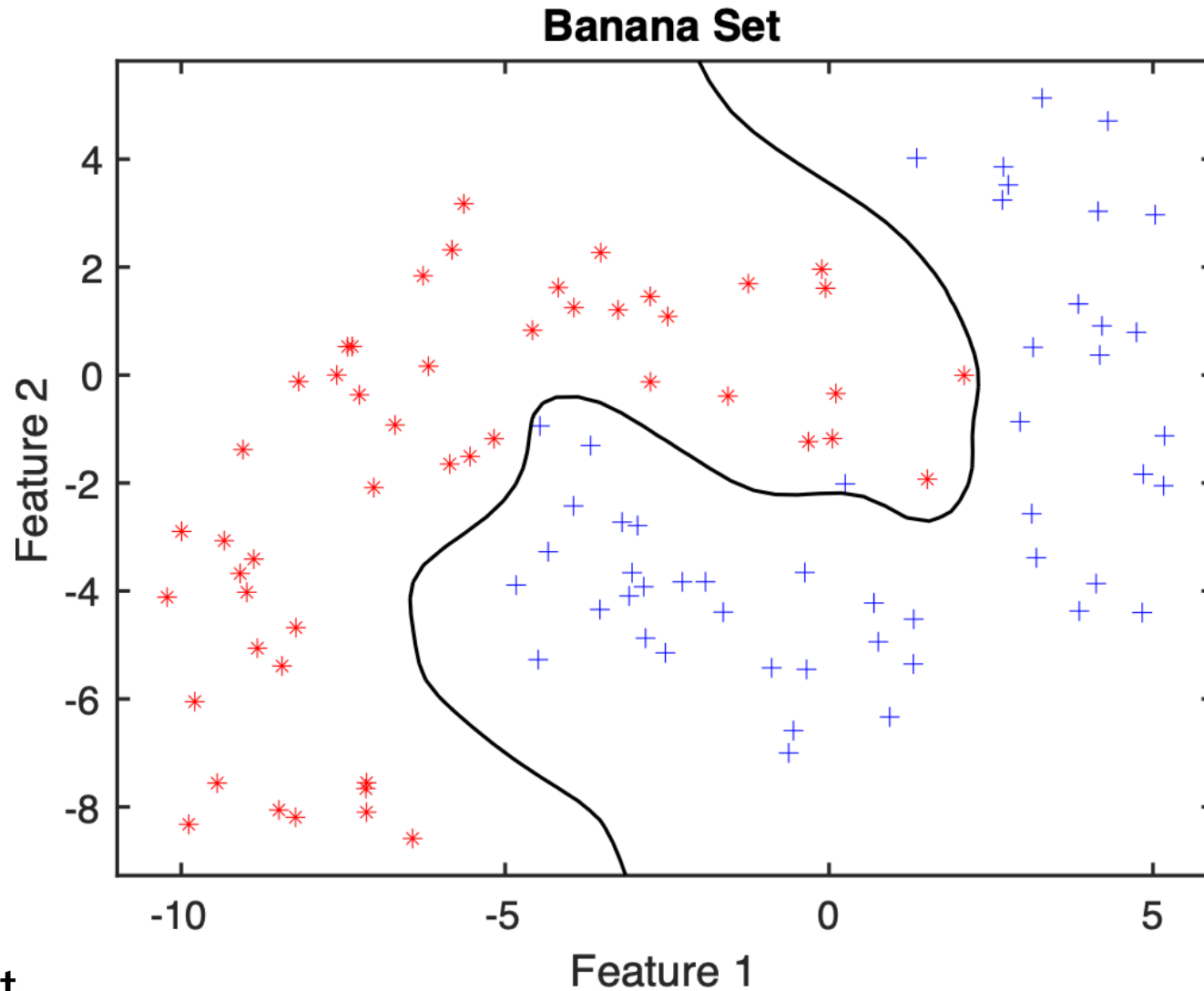
Parzen classifier

- Using Normal density

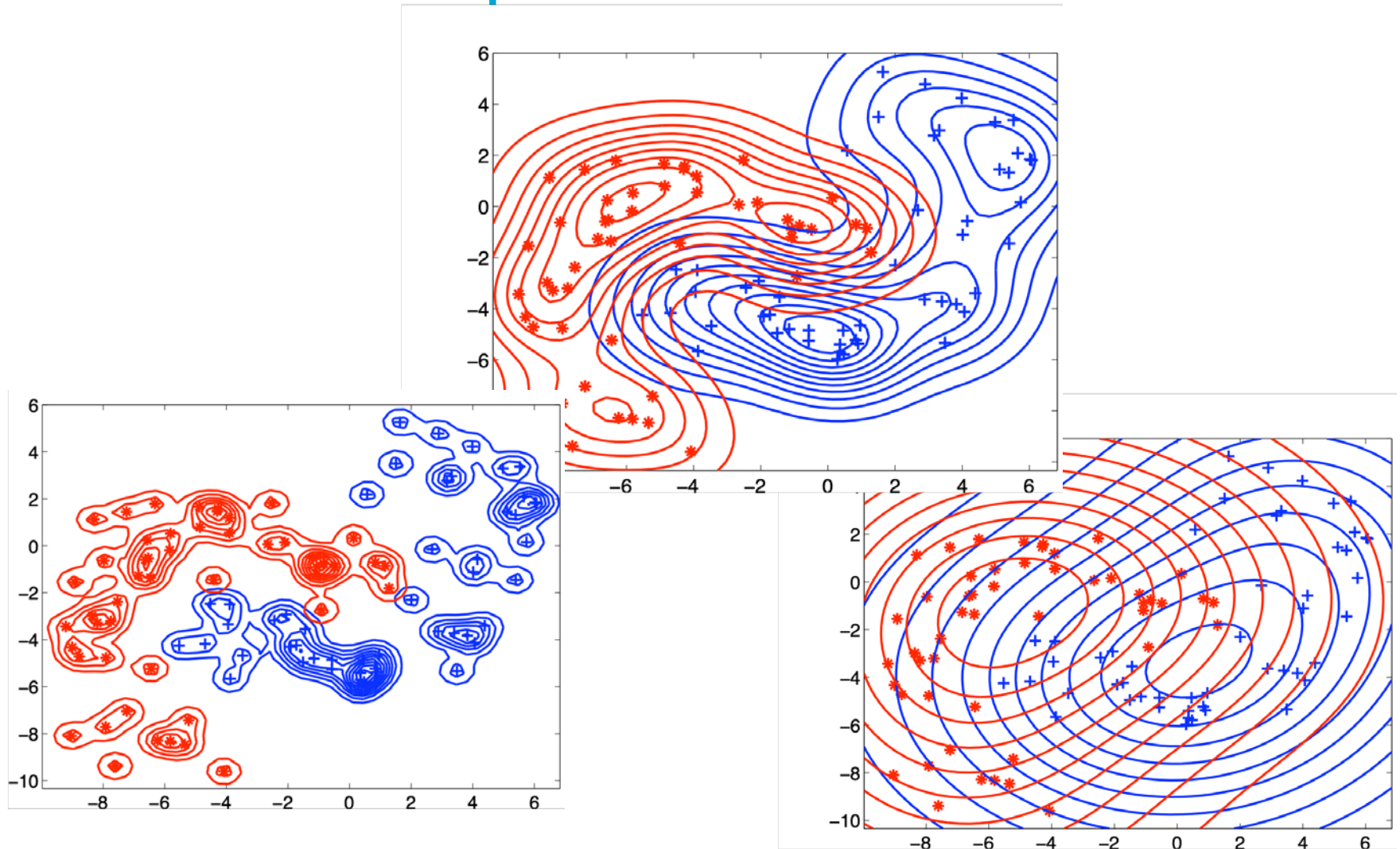
$$\hat{p}(x|y_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} N(x|x_j^{(i)}, hI)$$



Parzen classifier

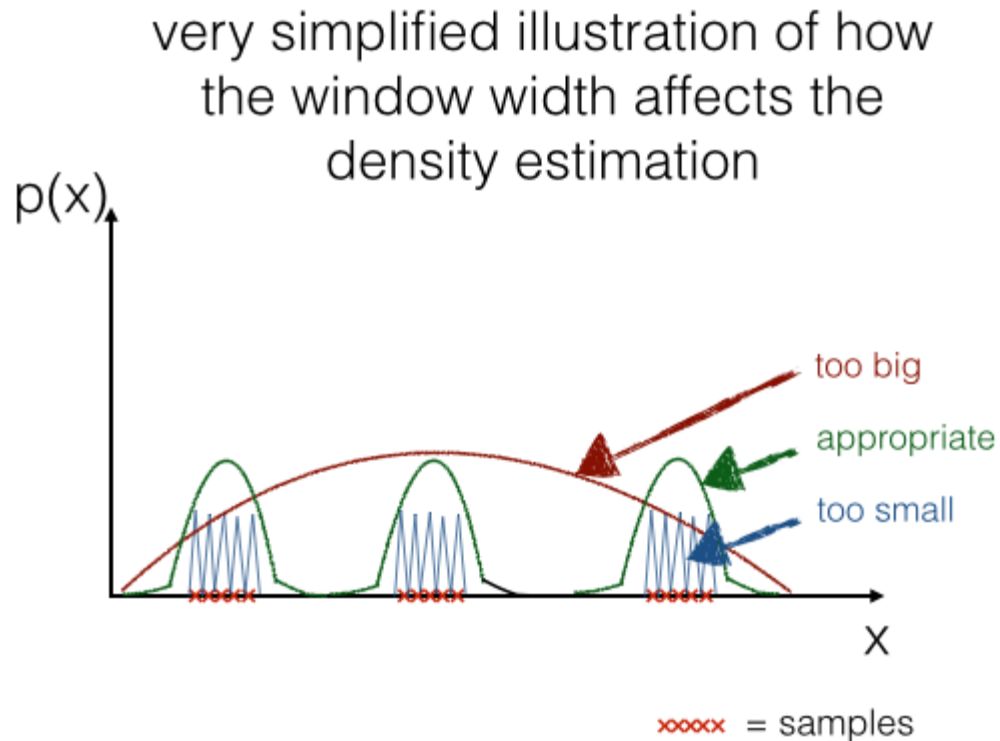


Parzen width parameter



Parzen width parameter

- The choice of h is important



Optimization of h

- Use a heuristic

$$h = \sigma \left(\frac{4}{p+2} \frac{1}{p+4} \right) n^{\frac{-1}{p+4}}$$

$$\sigma^2 = \frac{1}{p} \sum_{i=1}^p s_{ii}$$

- Optimize the likelihood

$$\prod_{i=1}^n \hat{p}(x_i)$$

- Use the average k-nearest neighbor distance
(k=10 is suggested)

Question

- Given a set of five data points:
 $x_1 = 2, x_2 = 2.5, x_3 = 3, x_4 = 1, x_5 = 6,$
find Parzen probability density function (pdf)
estimates at $x = 3,$
using the Gaussian function with $\sigma = 1$ as
window function.

Solution

$$\frac{1}{\sqrt{2\pi}} \exp \left(-\frac{(x_1 - x)^2}{2} \right)$$

$$= \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{(2 - 3)^2}{2} \right) = 0.2420$$

$$\frac{1}{\sqrt{2\pi}} \exp \left(-\frac{(x_2 - x)^2}{2} \right)$$

$$= \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{(2.5 - 3)^2}{2} \right) = 0.3521$$

Solution

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_3 - x)^2}{2}\right) = 0.3989$$

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_4 - x)^2}{2}\right) = 0.0540$$

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_5 - x)^2}{2}\right) = 0.0044$$

so

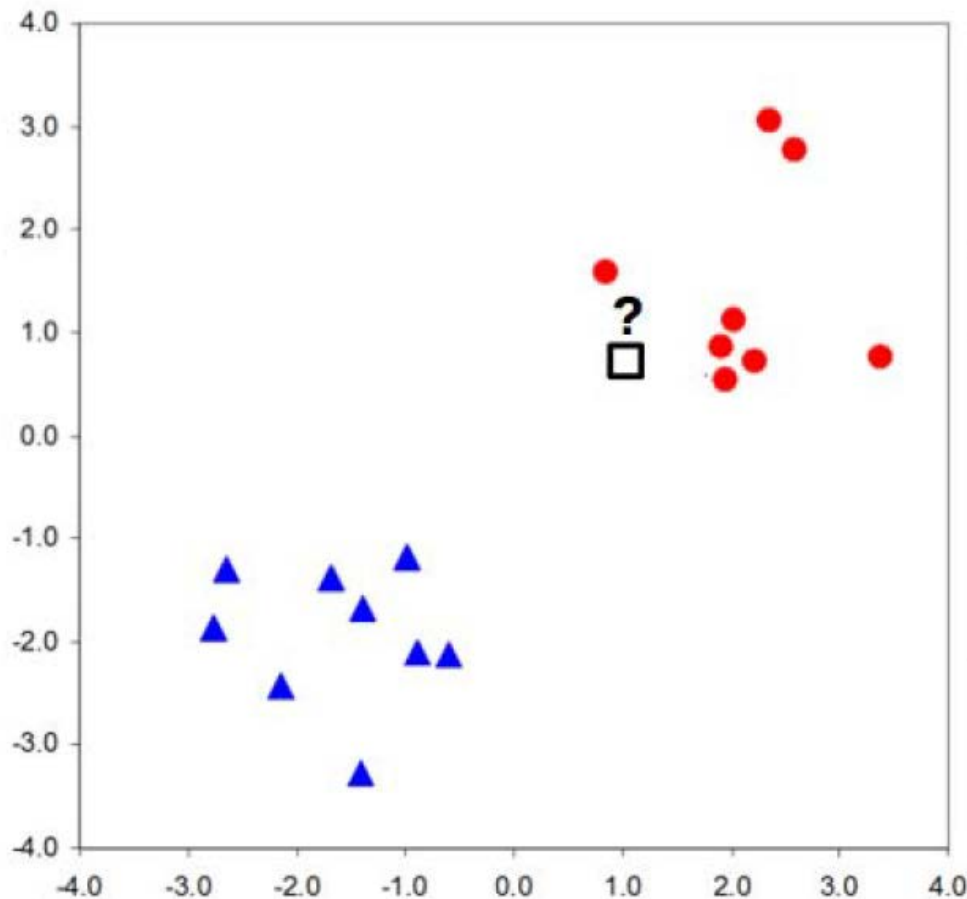
$$\begin{aligned} p(x = 3) &= (0.2420 + 0.3521 + 0.3989 \\ &\quad + 0.0540 + 0.0044)/5 = 0.2103 \end{aligned}$$

Parzen summary

- Estimates probability densities using kernel function
- Kernel function of fixed shape and size
- The choice of window shape and size is important

K-nearest Neighbours

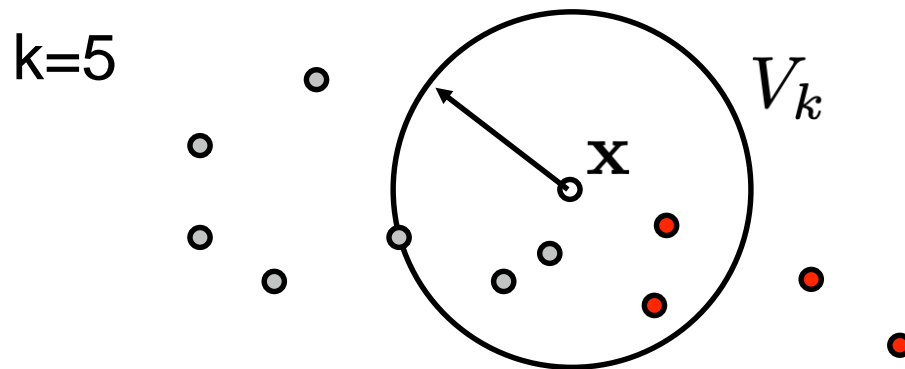
K-nearest neighbour intuition



- Set of points (x_1, x_2)
 - Two classes
- Is the box red or blue?
- How did you do it?
- Nearby points are red
 - Use as a bias for a learning algorithm

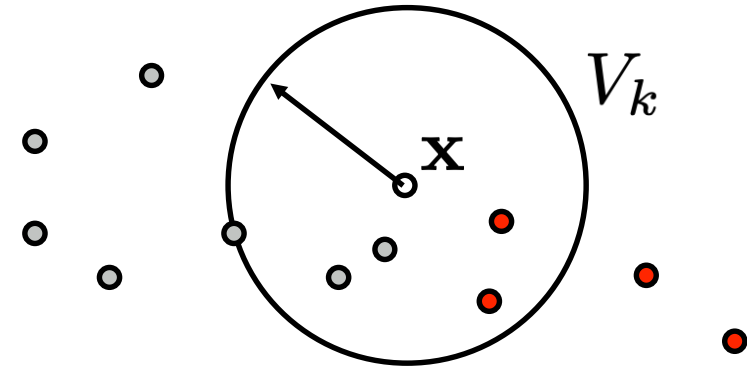
K nearest neighbor

- Use the intuition to classify a new point x :
 - Locate the cell on the new point x
 - Do **not** fix the volume of the cell:
grow the cell until it covers k objects:
find the k -th neighbors
 - predict the class y of new point x



K-nn density estimation

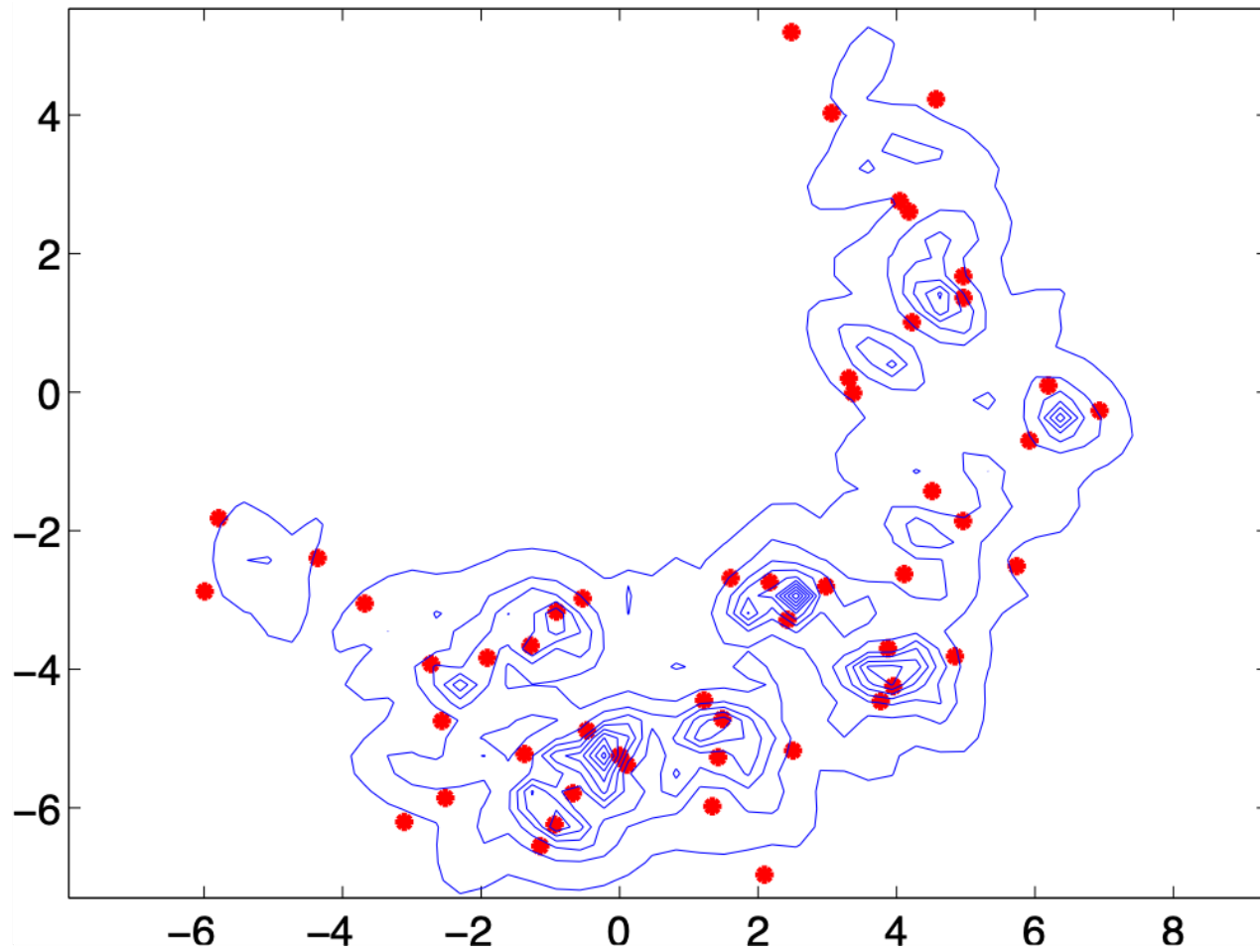
- $k = 5$
- $k_1 = 3$
- $k_2 = 2$



$$\hat{p}(x|y_i) = \frac{k_i}{n_i V_k}$$

- Where V_k is the volume of the sphere centered at x with radius r , being the distance to the k -th nearest neighbor
- Class priors: $\hat{p}(y_i) = \frac{n_i}{n}$
- Bayes: $\hat{p}(x|y_i)\hat{p}(y_i) > \hat{p}(x|y_j)\hat{p}(y_j) \rightarrow k_i > k_j$

K-nn density estimate

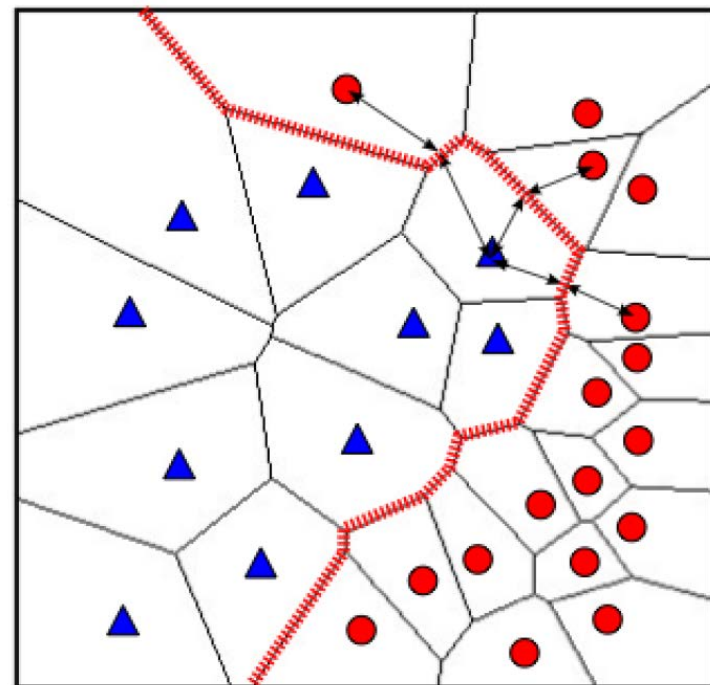


K-nn classification algorithm

- Given:
 - training examples $\{x_i, y_i\}$
 - x_i attribute-value representation of examples
 - y_i class label: {male, female}, digit {0,1, ... 9} etc.
 - testing point x that we want to classify
- Algorithm:
 - compute distance $D(x, x_i)$ to every training example x_i
 - select k closest instances $x_{i1} \dots x_{ik}$ and their labels $y_{i1} \dots y_{ik}$
 - output the class y^* which is most frequent in $y_{i1} \dots y_{ik}$ (**majority vote**)

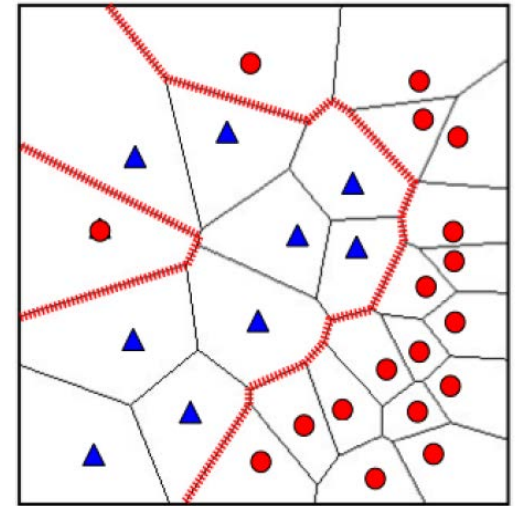
Decision boundary 1-nn

- Voronoi tessellation
 - partitions space into regions
 - boundary: points at same distance from two different training examples
- Classification boundary
 - non-linear, reflects classes well
 - impressive for simple method

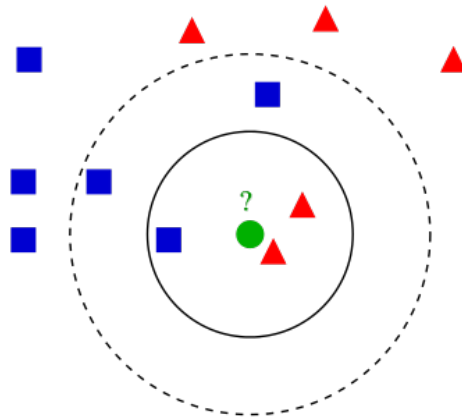


Nearest neighbor outliers

- Algorithm is sensitive to outliers
 - single mislabeled example dramatically changes boundary
- Idea:
 - use more than one nearest neighbor to make decision
 - count class labels in k most similar training examples
 - many "triangles" will outweigh single "circle" outlier

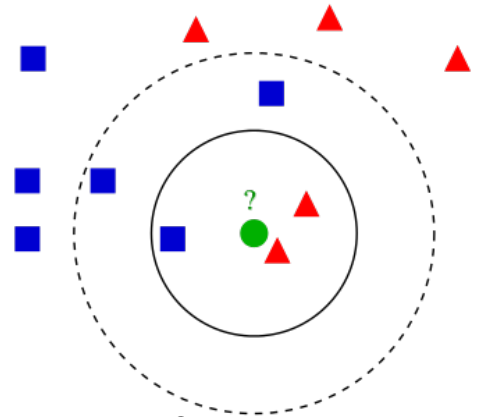


What is the influence of k ?



- What is the largest/smallest value of k that you can choose?
 - What will be the classification error then?

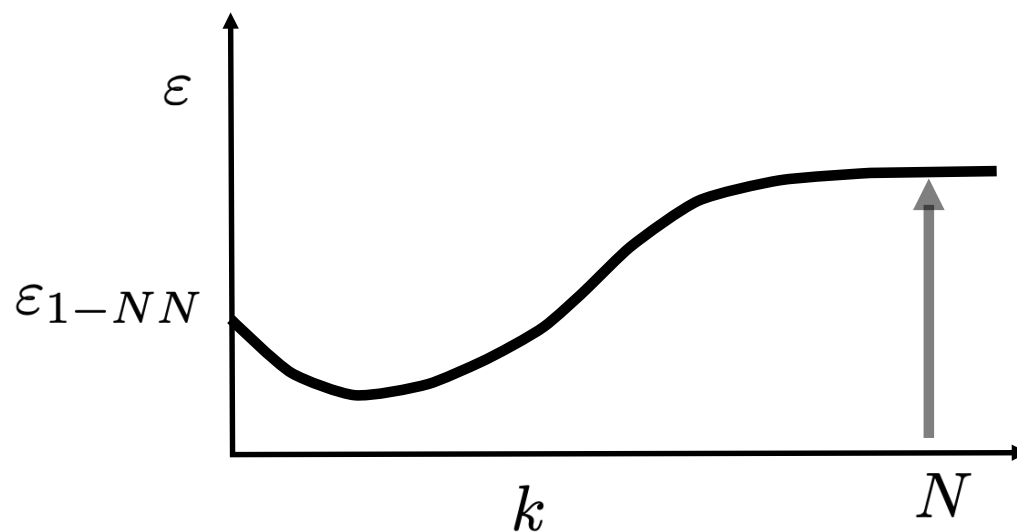
What is the influence of k ?



- Value of k has strong effect on k-nn performance
 - Large value \rightarrow everything classified as the most probable class
 - Small value \rightarrow highly variable, unstable decision boundaries
 - Small changes to training set \rightarrow large changes to classification
 - affects “smoothness” of the boundary

Choosing the value of k

- Selecting the value of k
 - set aside a portion of the training data (validation set)
 - vary k
 - Pick k that gives best generalization performance

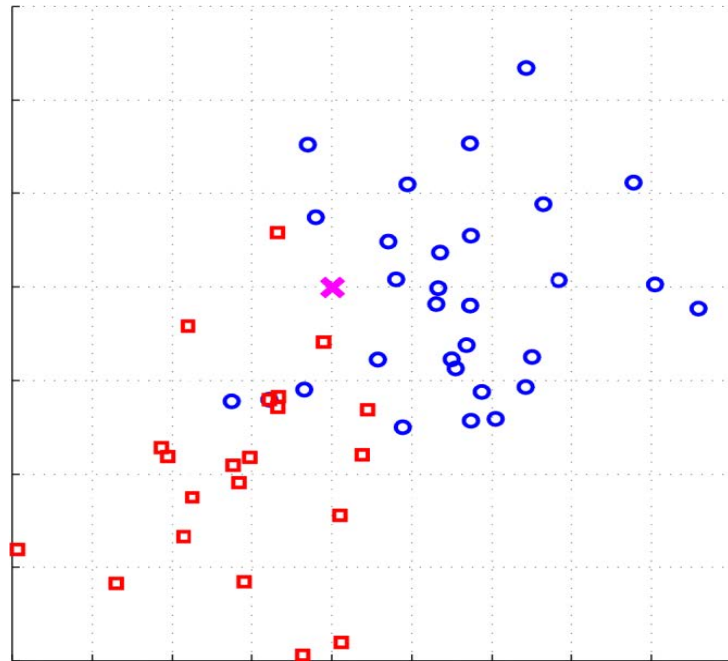


Question

- According to a k-NN classifier, to which class (red box or blue circle) does the new point **X** belong?

1. If $k=5$?

2. If $k=2$?



K-nn resolving ties

- Resolving ties:
 - equal number of positive/negative neighbours
 - use odd k (doesn't solve multi-class)
 - breaking ties:
 - random: flip the coin to decide positive/negative
 - prior: pick class with greater prior
 - nearest: use 1-nn classifier to decide

Distance measures

- The key component of the kNN algorithm
 - defines which examples are similar and which aren't
 - can have strong effect on performance
- Euclidean (numeric attributes):

$$D(x, x') = \sqrt{\sum_d |x_d - x'_d|^2}$$

- symmetric, spherical, treats all dimensions equal
- sensitive to extreme differences in single attribute

Distance measures

- Hamming (categorical attributes):
 - number of attributes where x , x' differ

$$D(x, x') = \sum_d 1_{x_d \neq x'_d}$$

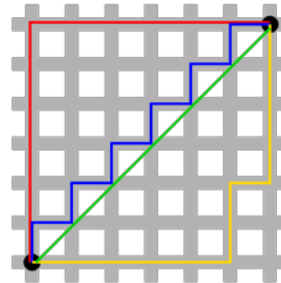
- Other
 - Kullback-Leibler (KL) divergence (for histograms)
 - Custom distance measures (BM25 for text)

Distance measures

- Minkowski distance (p-norm):

$$D(x, x') = \sqrt[p]{\sum_d |x_d - x'_d|^p}$$

- $p = 2$: Euclidean
- $p = 1$: Manhattan
- $p = 0$: Hamming
- $p = \infty$: $\max_d |x_d - x'_d|$



Question

- Given a labeled two-dimensional data set:
 - Blue label: (1,4); (2,2); (3,3); (3,4);
 - Red label: (3,7); (5,7); (5,6); (6,5);
- Predict the label of a new point (4, 5) using 3-nn classifier with Manhattan distance.

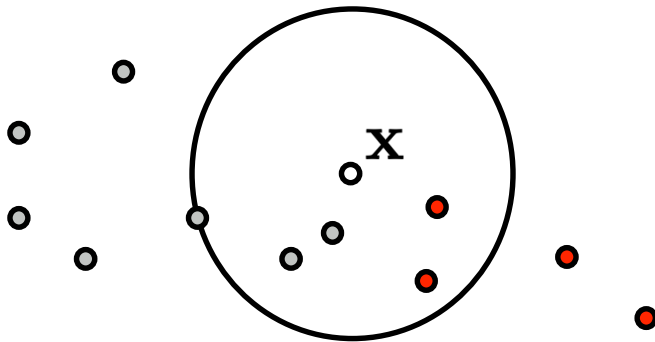
K-nn missing values

- Missing values
 - have to “fill in”, otherwise can't compute distance
 - keyconcern: should affect distances as little as possible
 - reasonable choice: average value across entire dataset

Why is k-nn slow?

- Find nearest neighbors of the new point X

- What you see:



- What algorithm sees:

- Training set

$\{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$

- Testing instance:

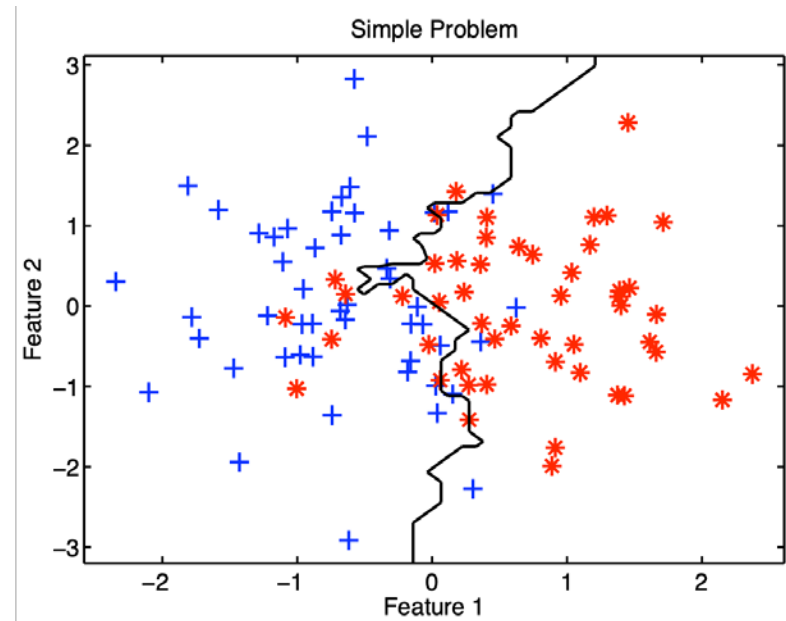
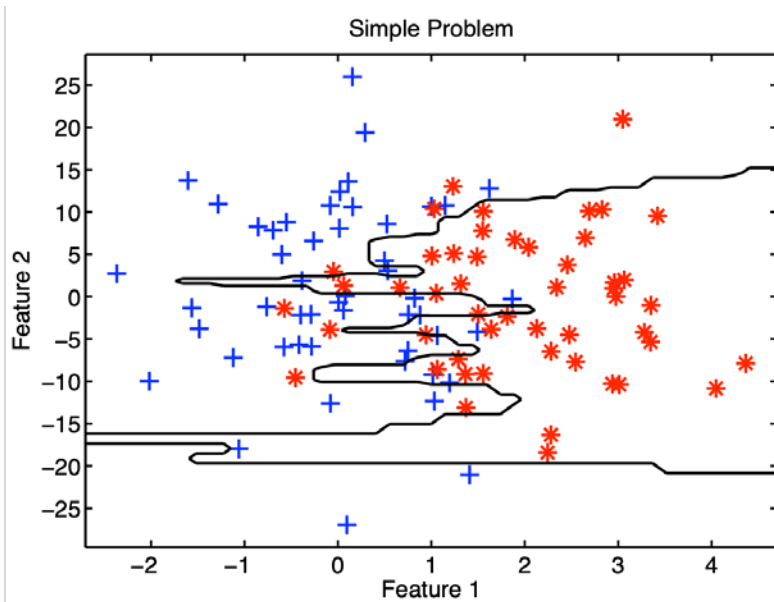
$(7, 4)$

- Nearest neighbors?

compare one-by-one to each training instance

Sometimes strange results

- How is this possible?



Scale your features!

K-nn pros and cons

- Simple and flexible classifiers
- often a very good classification performance
- it is simple to adapt the complexity of the classifier
- relatively large training sets are needed
- the complete training set has to be stored
- distances to all training objects have to be computed
- the features have to be scaled sensibly
- the value for k has to be optimized

After practicing with the concepts of this lecture you should be able to:

- Explain the difference between parametric and non-parametric density estimation
- Explain Parzen density estimation and classification
- Explain k-nn density estimation and classification
- Explain the advantages and disadvantages of Parzen en k-nn
- Implement k-nn classifier in Python

Lab

- Starts at 13:45
- I will be at the lab today from 15:00