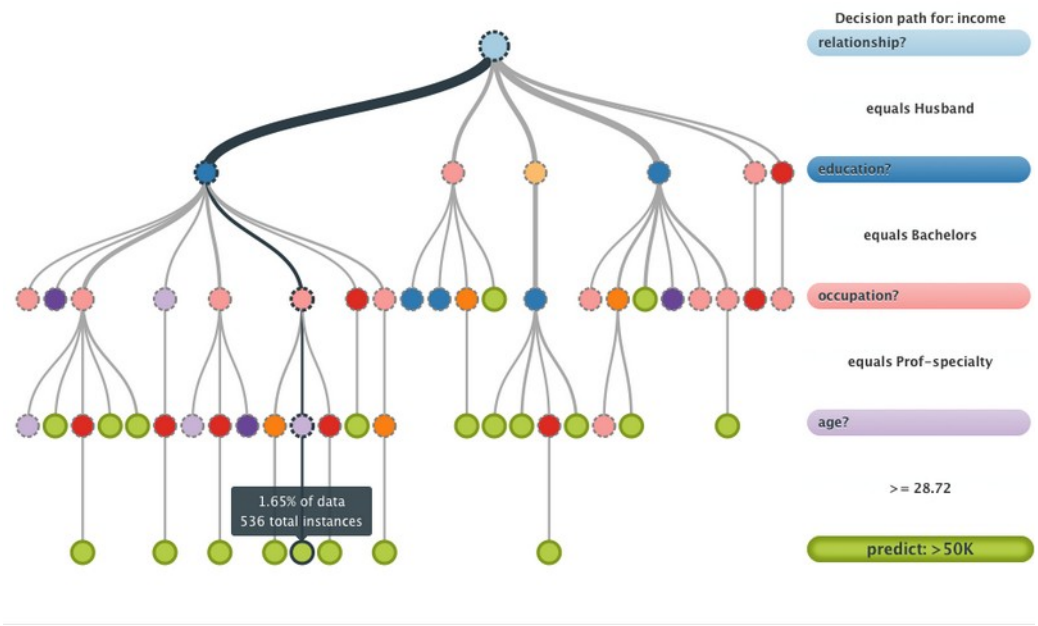
Machine learning
Lecture 2



Marcin Wolter

IFJ PAN

6 December 2019

- Simple non-linear methods like k-nearest neighbors, Parzen kernel methods, PDE_RS.
- Independent Component Analysis ICA

All slides are here: https://indico.ifj.edu.pl/event/289/

Example from the last lecture

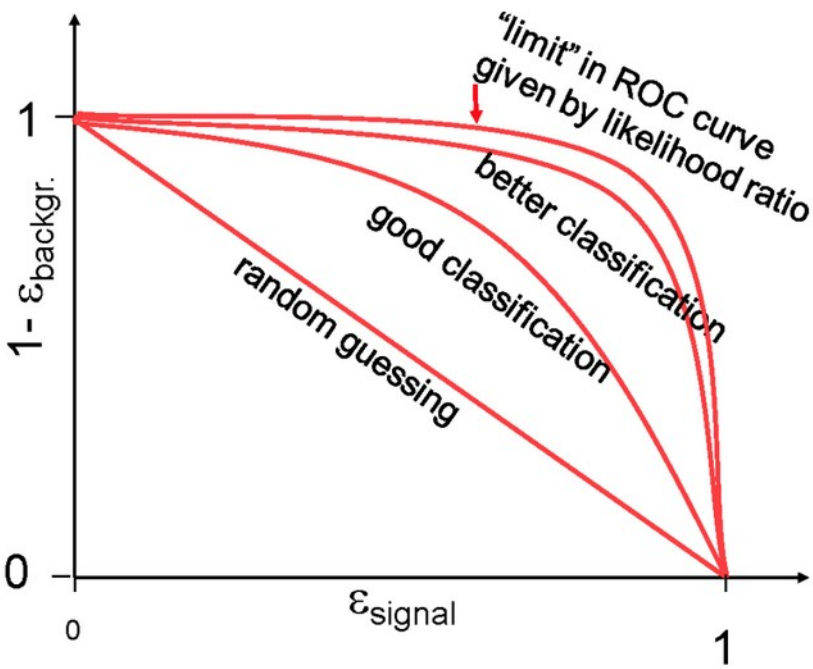
- https://github.com/marcinwolter/MachineLearnin2019/blob/master/plot_digits_classif.ipynb
- There was a bug (thanks for pointing up!)
- The bug is corrected now.

Simple classificators

- We have learned about:
  - Disciminants:
    - Cuts
    - Fisher Linear Discriminant
    - Naive Bayes
  - Unsupervised method
    - Decorrelation
    - Principal Component Analysis
- Today we will talk about:
  - Overtraining problem.
  - Other discrimimants: kernel density estimators, PDE_RS
  - Example how to use them and how to check their performance (ROC curve).
  - Example of face classification – eigenfaces.

ROC curve

- ROC (Receiver Operation Characteristic) curve was first used to calibrate radars.
- Shows the background rejection ($1-\varepsilon_B$) vs signal efficiency ε_B. Shows how good the classifier is.
- The integral of ROC could be a measure of the classifier quality:



Integral(ROC) = ½ – random

Integral(ROC) = 1 - ideal

Classification Accuracy

- Classification accuracy is the ratio of correct predictions to total predictions made.
  - **classification accuracy = correct predictions / total predictions**
- It is often presented as a percentage by multiplying the result by 100.
  - **classification accuracy = correct predictions / total predictions * 100**
- Classification accuracy can also easily be turned into a misclassification rate or error rate by inverting the value, such as:
  - **error rate = (1 - (correct predictions / total predictions)) * 100**

Confusion matrix

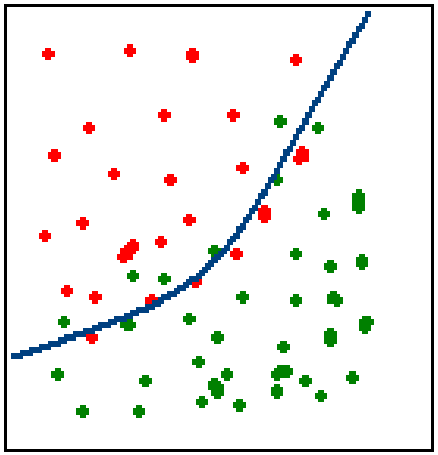
- Example confusion matrix (recognition of dogs vs. cats)

		Actual class	
		Cat	Dog
Predicted class	Cat	**5**	2
	Dog	3	**3**

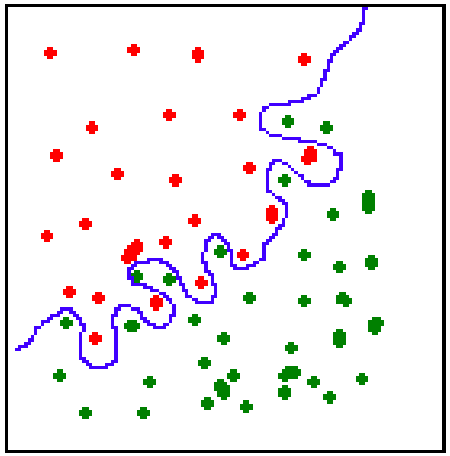
Confusion Matrix

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$

Overtraining

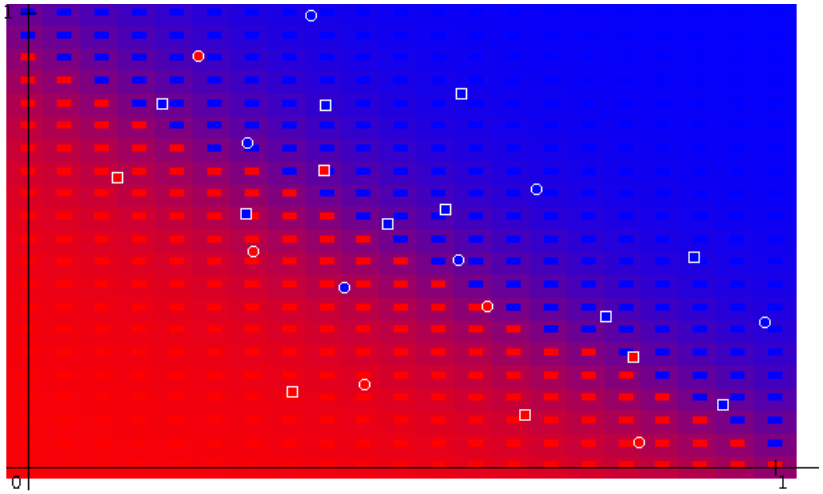


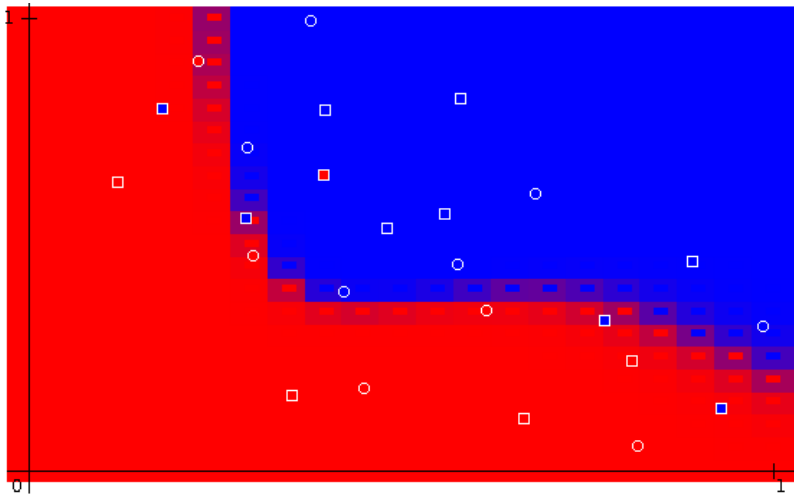
Correct

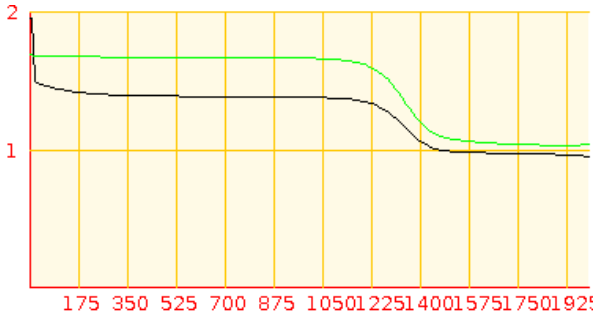


Overtraining

- **Overtraining** – algorithm "learns" the particular events, not the rules.
- <u>This effect important for all ML algorithms.</u>
- **Remedy – checking with another, independent dataset.**

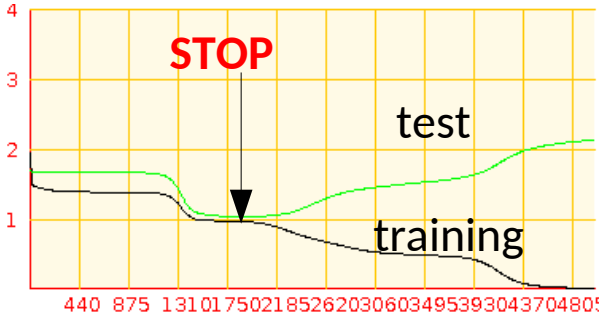






Training sample

Test sample



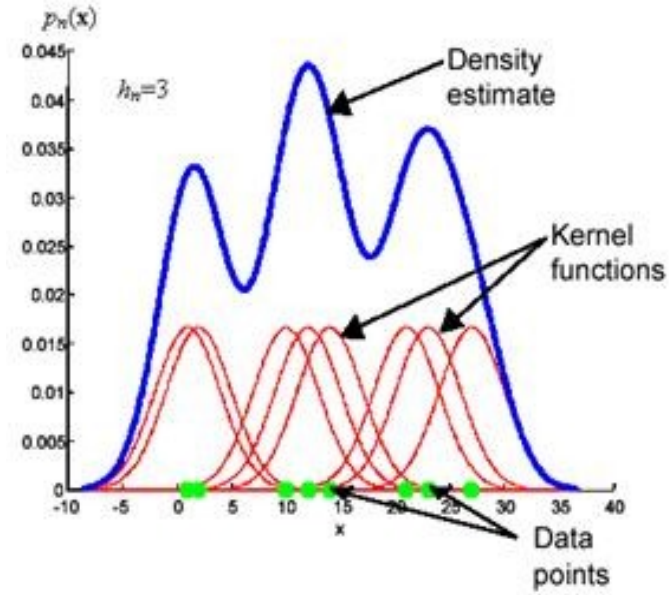
Example of using Neural Network.

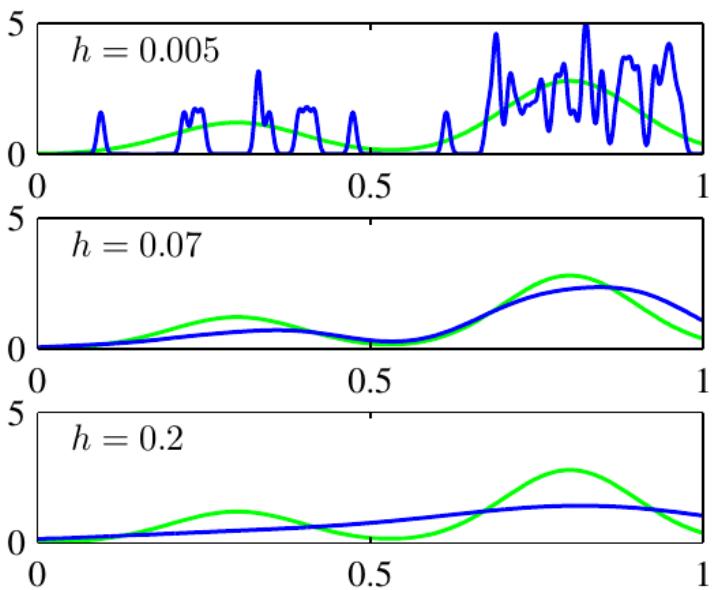
Kernel density estimators

Approximation of the unknown probability density as a sum of kernel functions centered in the points x_n of the training sample (Parzen, years 1960-ties).

$$D(x)=\frac{P(S)}{P(S)+P(B)}$$

- Typical kernel functions: Gauss, $1/x^n$ itp.
- Simple idea, but using this method requires a lot of memory and CPU.





Approximated probability density (blue) comapred to the true probability density (green) as a function of the width of the Gaussian kernel function. We can see, that the width is a smoothing parameter.

QUAERO package from the D0 experiment

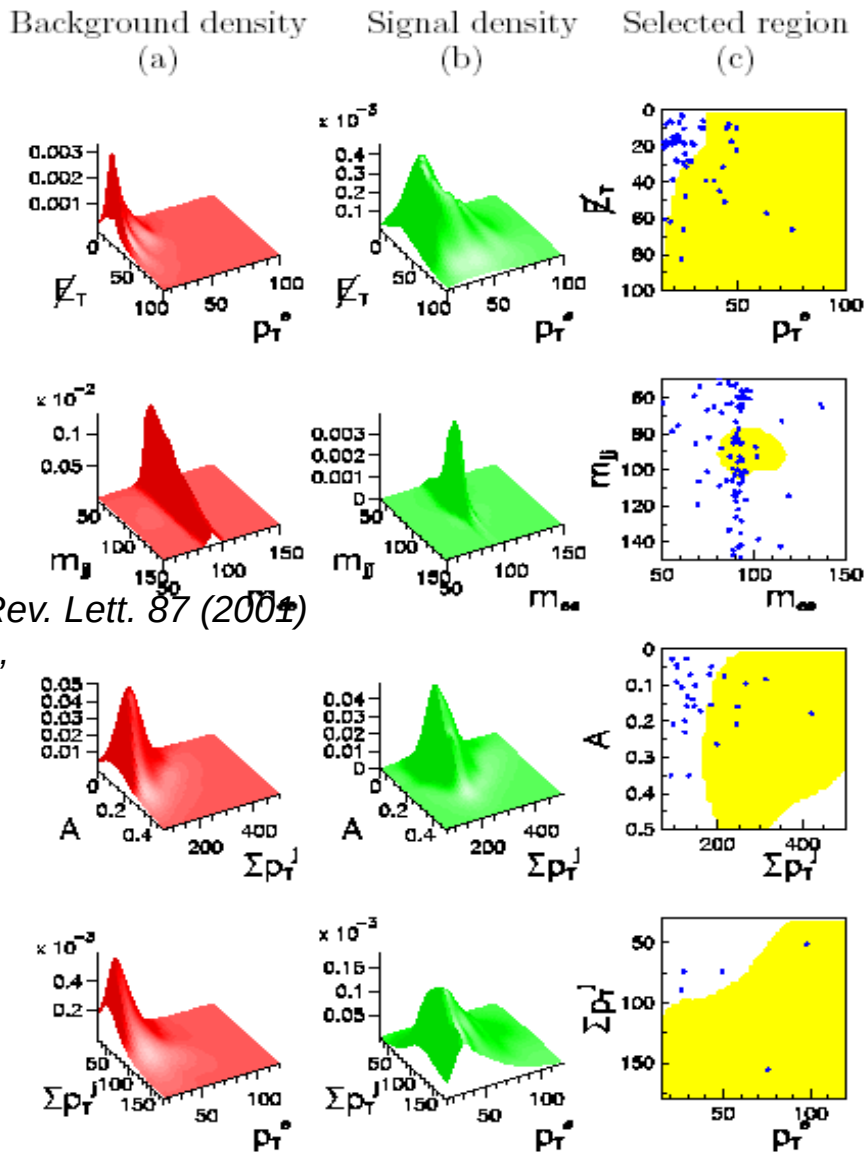


FIG. 1. The background density (a), signal density (b), and selected region (shaded) (c) determined by QUAERO for the standard model processes discussed in the text. From top to bottom the signals are: $WW \to e\mu \not{E}_T$, $ZZ \to ee\,2j$, $t\bar{t} \to e\not{E}_T\,4j$, and $t\bar{t} \to e\mu\not{E}_T\,2j$. The dots in the plots in the rightmost column represent events observed in the data.

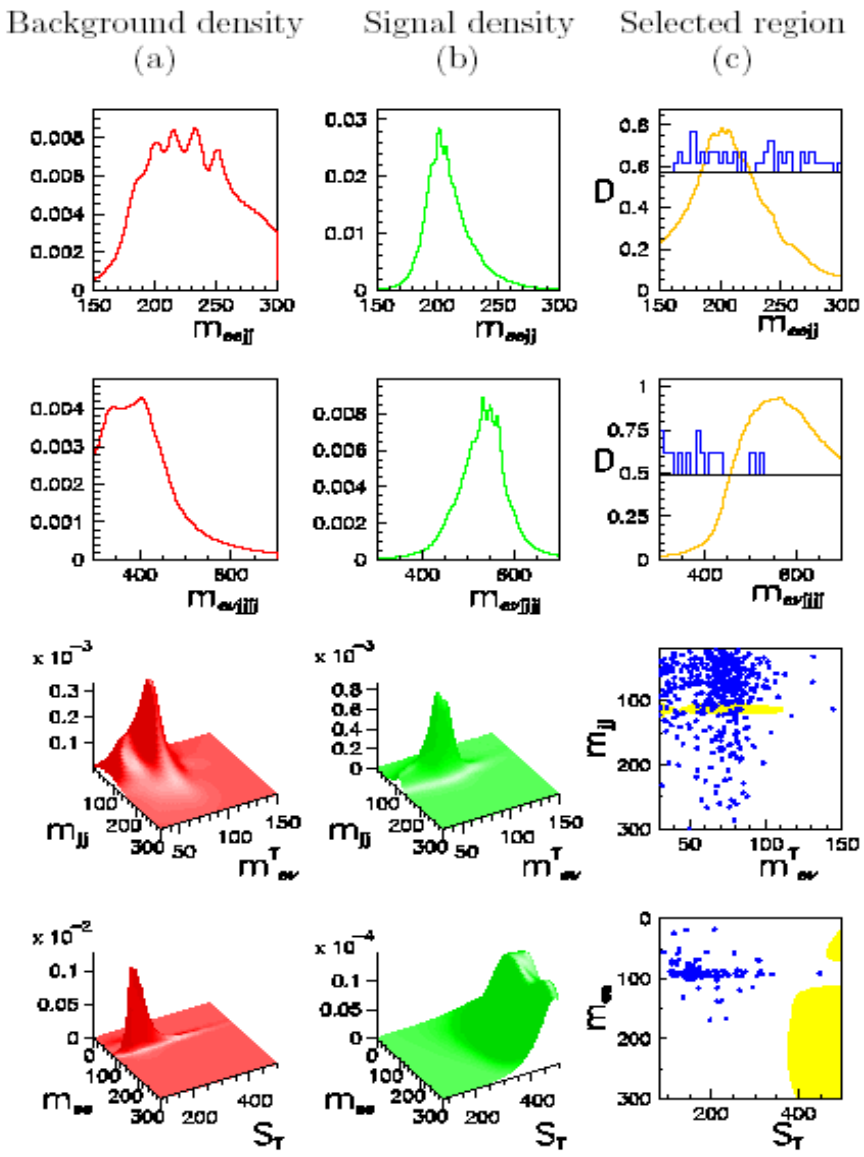


FIG. 2. QUAERO's analysis of signatures involving undiscovered particles. From top to bottom the hypothetical signals are: $h_{200} \to ZZ \to ee\,2j$, $Z'_{550} \to t\bar{t} \to e\not{E}_T\,4j$, $Wh_{115} \to e\not{E}_T\,2j$, and $LQ_{225}\overline{LQ}_{225} \to ee\,2j$. Plots (c) of the first two rows show the discriminant D (curve), the threshold $D_{\rm cut}$ (horizontal line), and the data (histogram); the region with $D > D_{\rm cut}$ is selected.

Phys. Rev. Lett. 87 (2001) 231801,

PDE_RS – extension of the Parzen methods

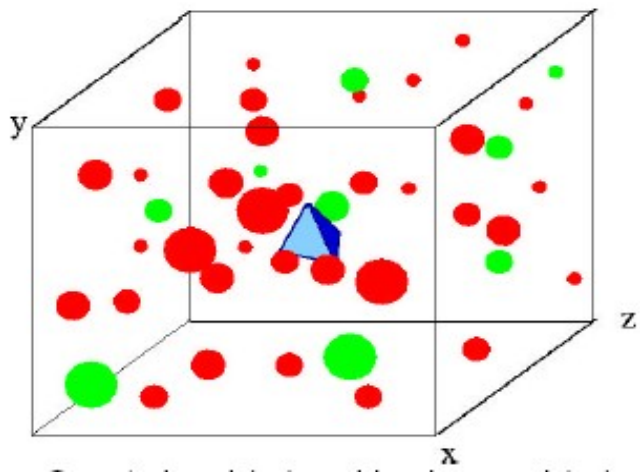
*published by T. Carli, B. Koblitz, NIM A 501 (2003) 576-588

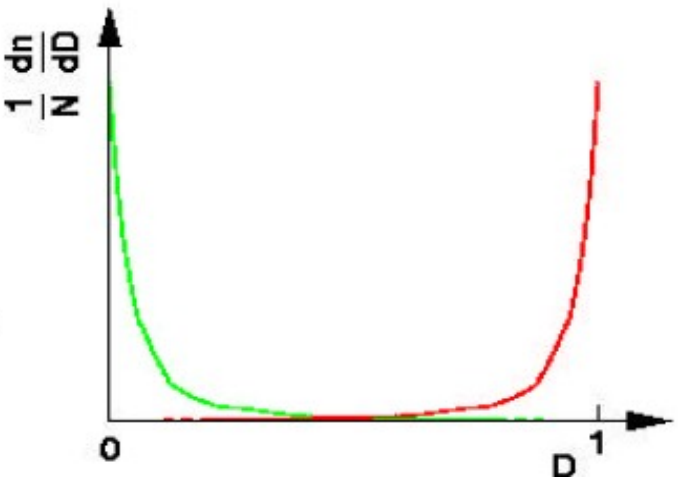
Counts signal (n_s) and background (n_b) events in N-dimensional cube around the classified event – only few events from the training sample are needed.

Size of the cube – a free parameter.

Discriminator *D(x)* :

$$D(x)=\frac{n_S}{n_S+n_b}$$



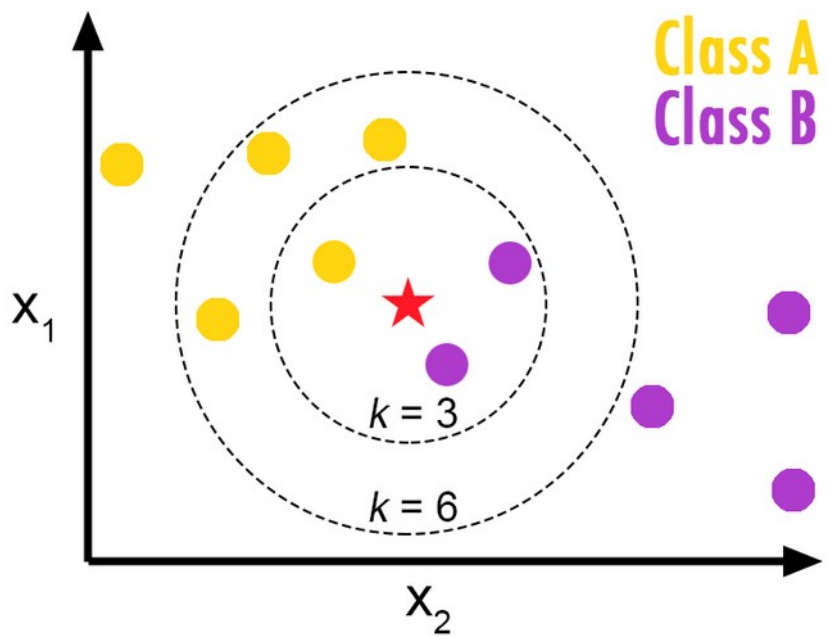


- Simple analysis.
- Events stored in the binary tree – the neighbor events are found quickly.
- It's a special case of Parzen method – the kernel function: $f(x)=\begin{Bmatrix}1\,(|x|\le d)\\0\,(|x|>d)\end{Bmatrix}$

KNN - k nearest neighbors

- Proposed already in 1951
- An event is qualified to the class, to which belongs the majority of it's k nearest neighbors,
- or we calculate the probability of belonging to the class "signal" as:

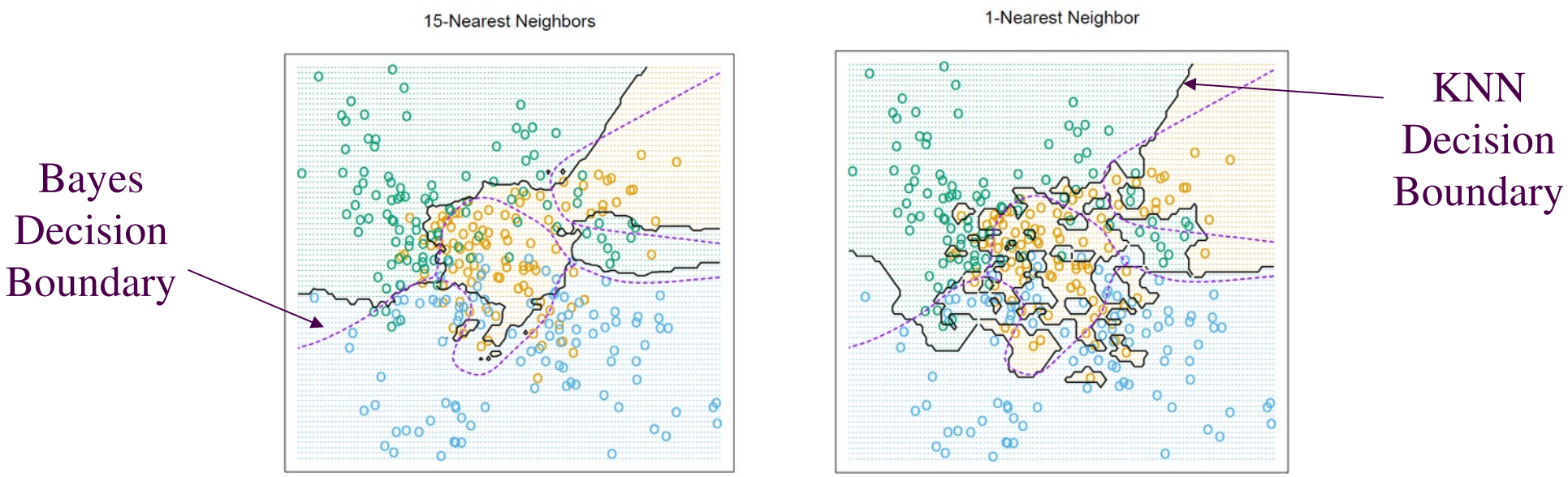
$$p(S|x) = \frac{k_S}{k}$$



K-Nearest Neighbors

- The only parameter that can adjust the complexity of KNN is the number of neighbors *k*.

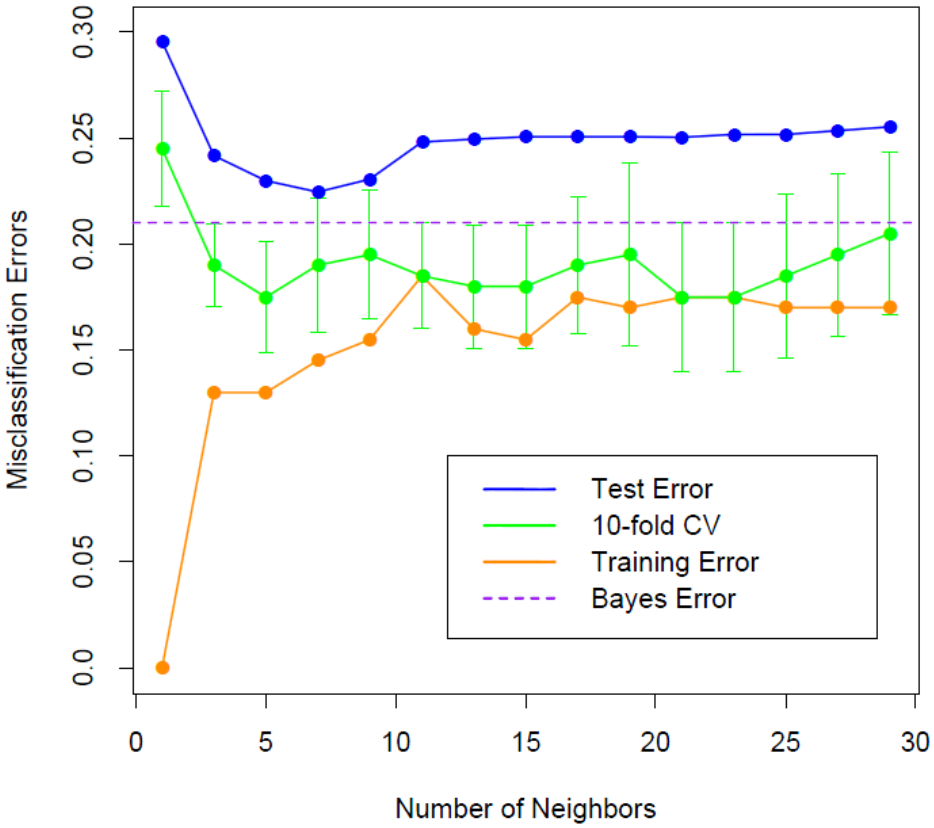
- The larger *k* is, the smoother the classification boundary. Or we can think of the complexity of KNN as lower when *k* increases.



K-Nearest Neighbors

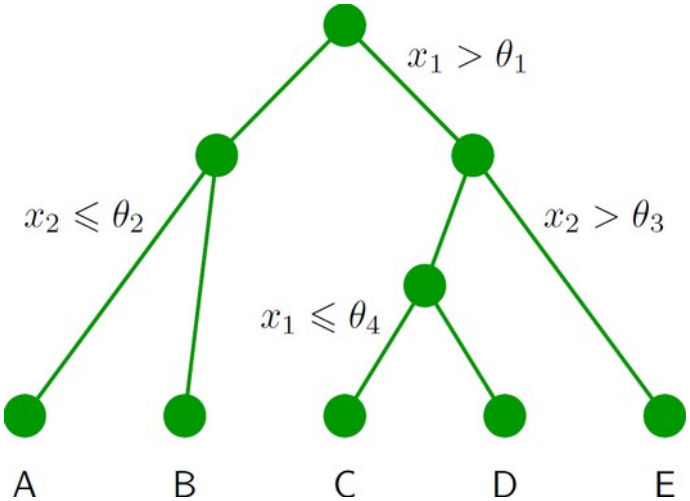
- For another simulated data set, there are two classes. The error rates based on the training data, test data, and *10*-fold cross validation are plotted against *K*, the number of neighbors.

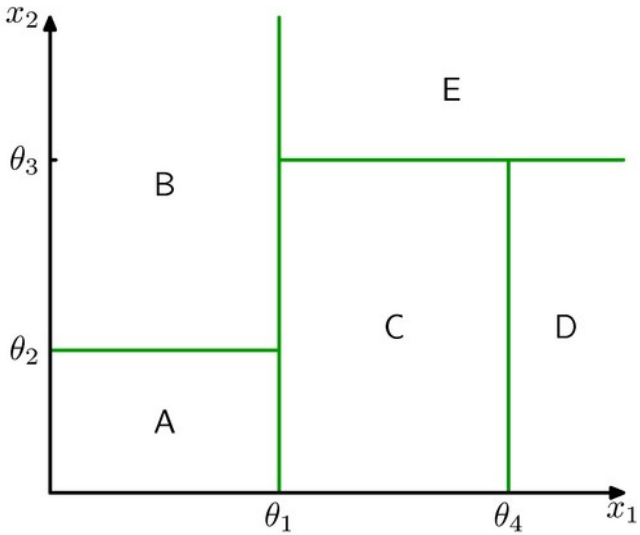
- We can see that the training error rate tends to grow when *k* grows, which is not the case for the error rate based on a separate test data set or cross-validation.



Decision trees

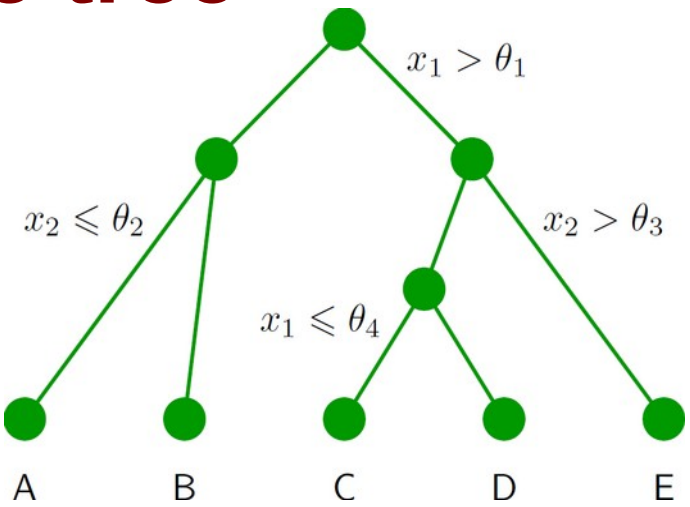
- Decision tree – a series of cuts, each „leaf” (A,B,C,D,E) has a label, for example ”signal” and “background”.





- Easy in visualization and interpretation
- Resistant for *outliers*.
- Weak variables are ignored.
- Fast training and classification.
- Unfortunately: **sensitive for fluctuations, unstable**.

Building the tree



- We start from the root.
- We divide the training sample by the best separating cut on the best variable.
- We repeat the procedure until the stopping conditions are fulfilled, for example: number of leafs, number of events in a leaf etc.
- The ratio S/B in a leaf defines the classification (binary signal or background, or a real number giving the probability, that a given event is a signal).

Definitions of separation:

- Gini inpurity: (*Corrado Gini 1912, invented the Gini index used to measure the inequality of incomes*)

  p (1-p) : p= P(signal), *purity*

- Cross-entropy:

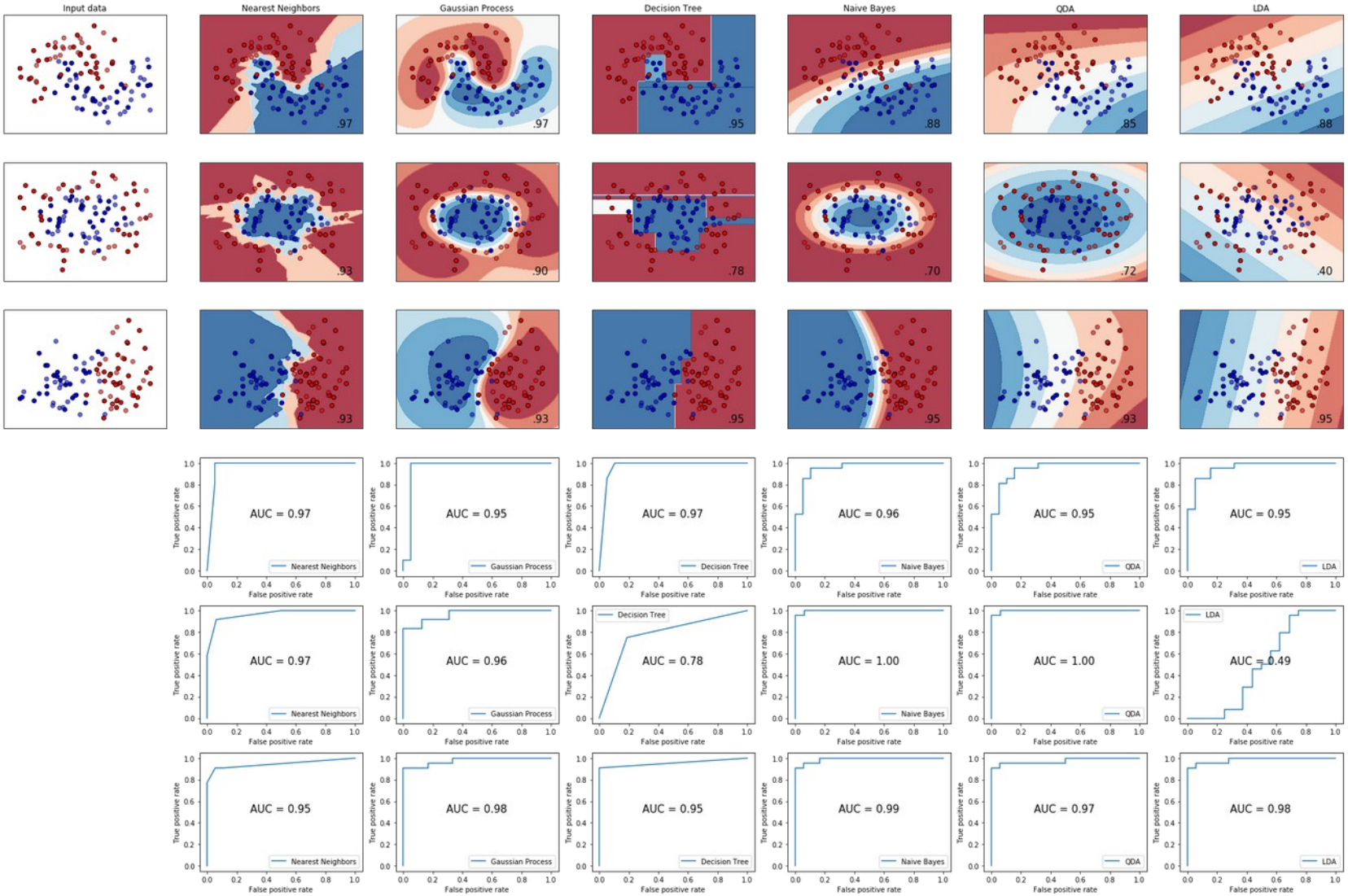
  -(p ln p + (1-p)ln(1-p))

- Missidentification:

  1-max(p,1-p)

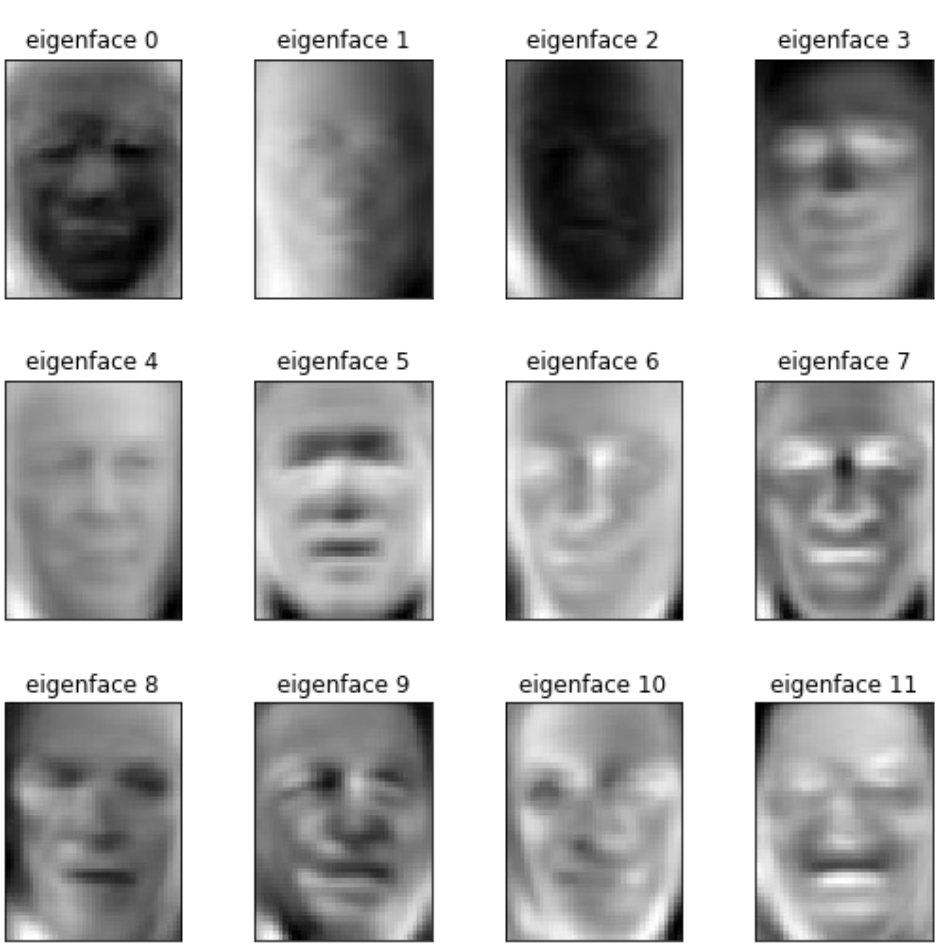
Example of simple classifiers

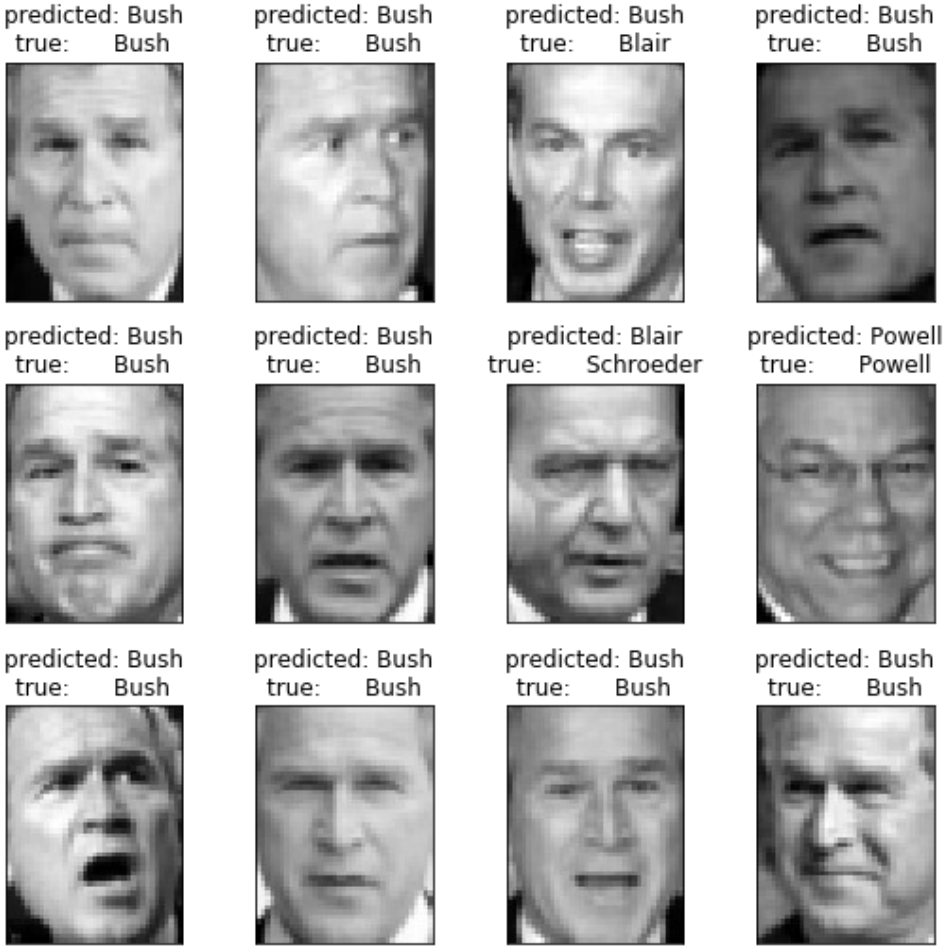
- https://github.com/marcinwolter/MachineLearnin2019/blob/master/simple_classifier_comparison.ipynb



Faces classification

- PCA – each face can be represented as a combination of a limited number of “eigenfaces”
- https://github.com/marcinwolter/MachineLearnin2019/blob/master/plot_face_recognition.ipynb





Summary

- We have learned about simple classifiers….
- ...how to train them, how to optimize them, how to measure their performance...
- ...and about unsupervised methods like Principal Component Analysis.

- Next time we talk about:
  - Independent Component Analysis ICA
  - Boosted Decision Trees BDT