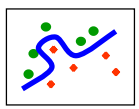
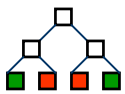
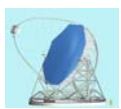
Statistical Learning in Physics Analysis

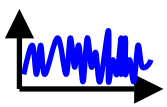
Jens Zimmermann
zimmerm@mppmu.mpg.de

Max-Planck-Institut für Physik, München
MPI für extraterrestrische Physik, München
Forschungszentrum Jülich GmbH


Statistical Learning?


Three Classes of Learning Methods


Applications in Physics Analysis


Training the Learning Methods


Examples

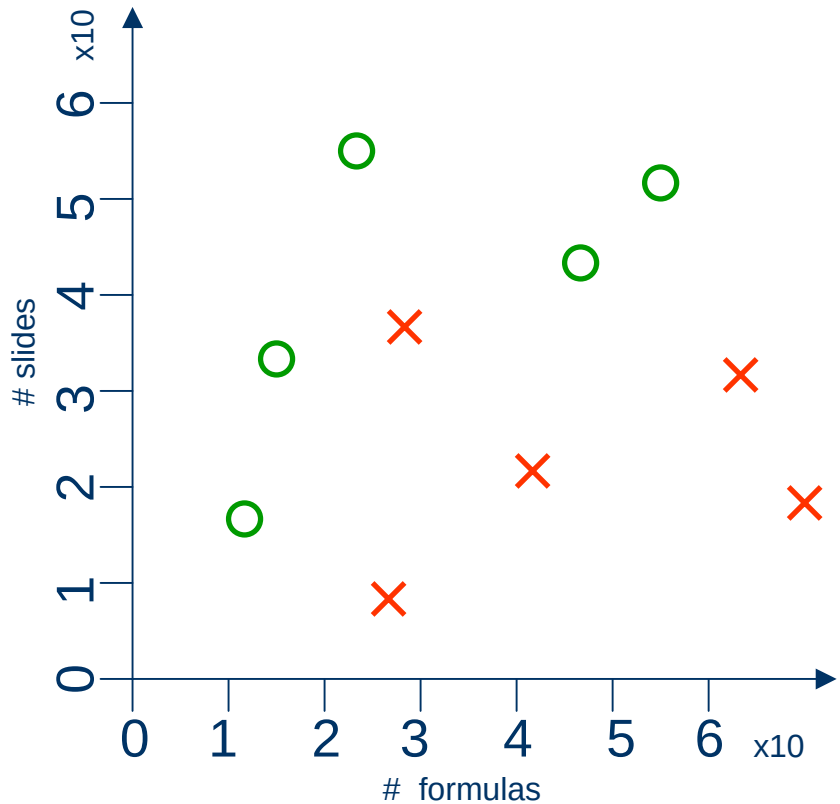

Conclusion

Some Events

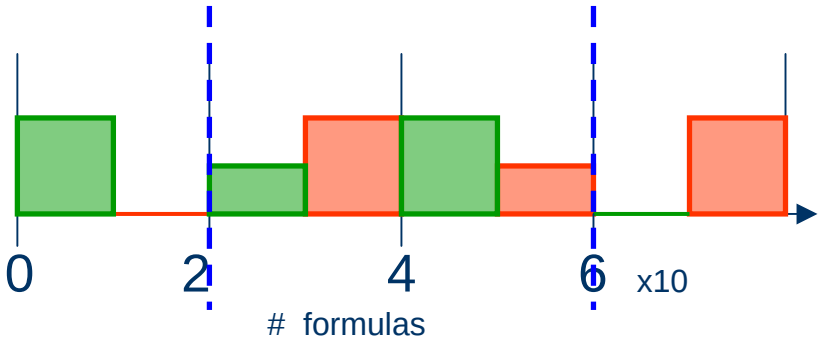
Theorists ×

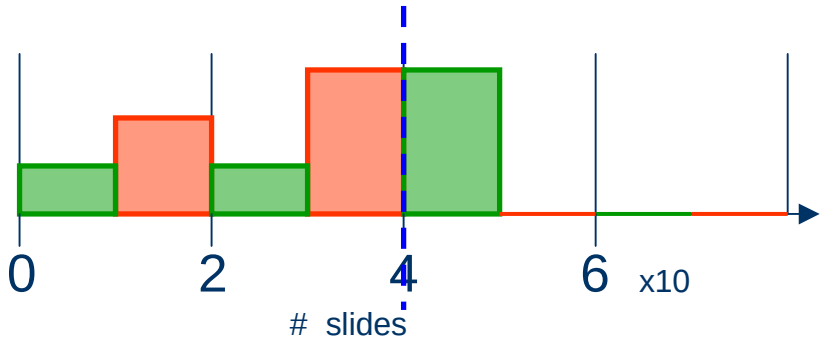
Experimentalists ○

# formulas	# slides
42	21
28	8
71	19
64	31
29	36
15	34
48	44
56	51
25	55
12	16



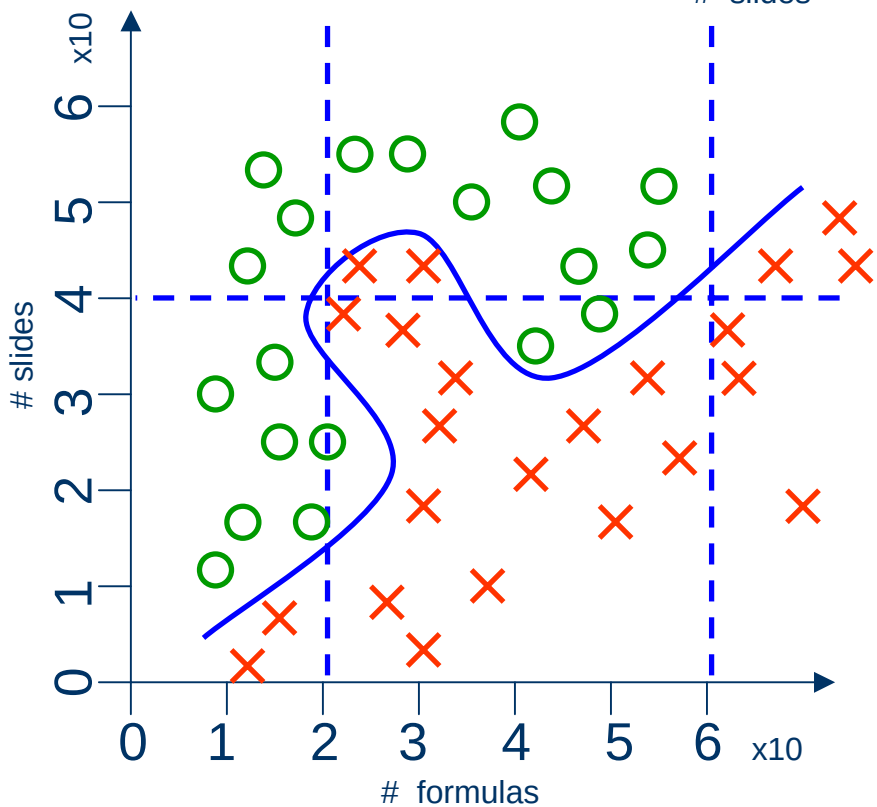
First Analysis



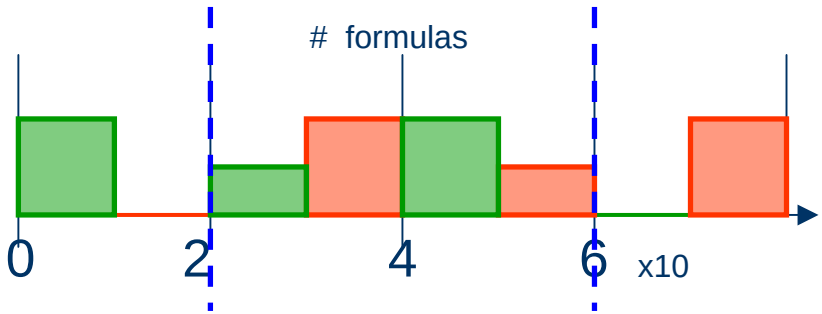


Experimentalists

Theorists



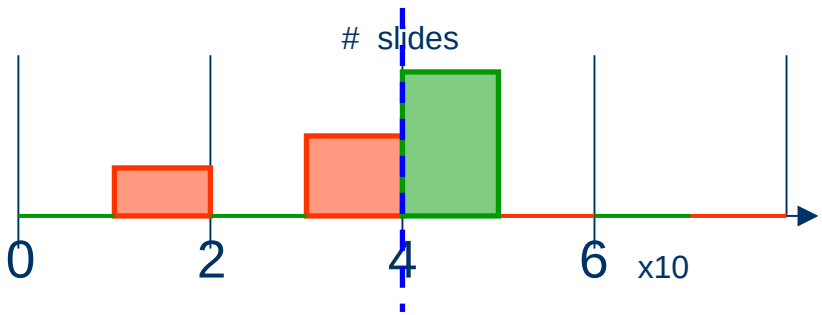
Decision Trees



#formulas < 20 → exp

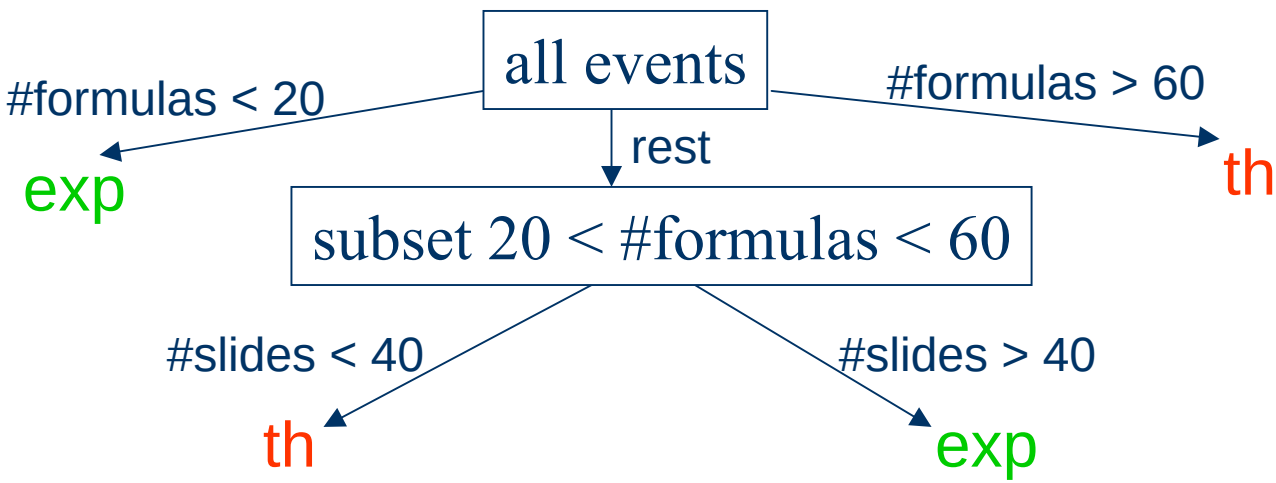
#formulas > 60 → th

20 < #formulas < 60?



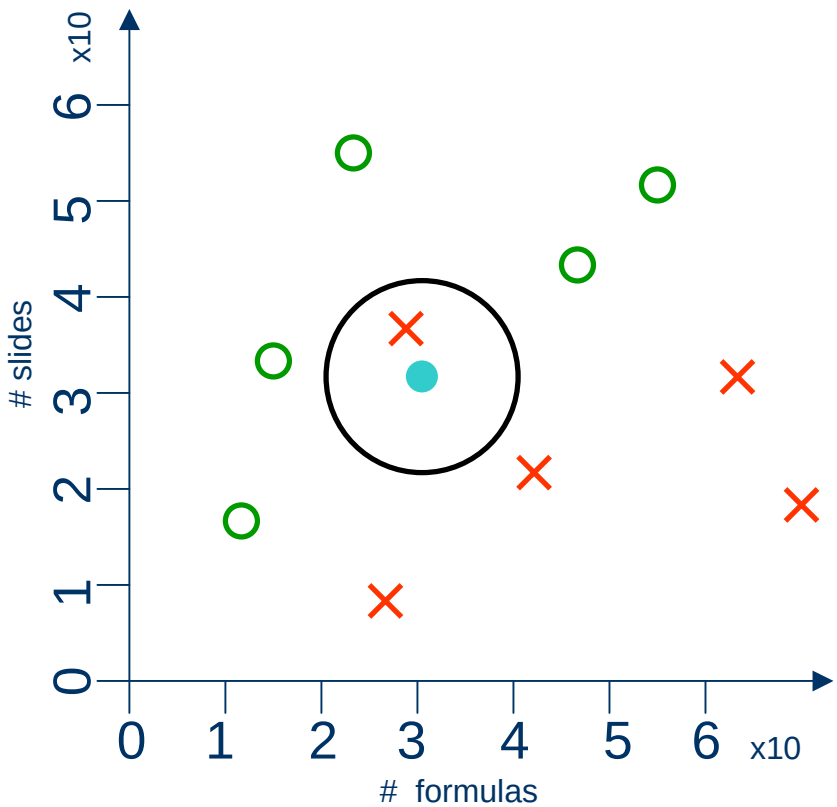
#slides > 40 → exp

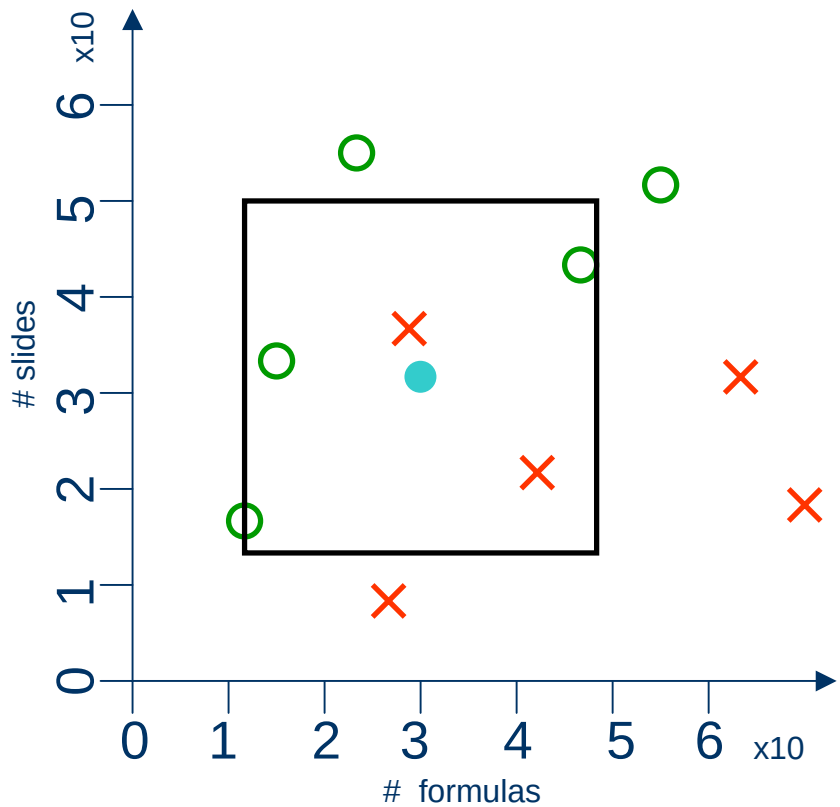
#slides < 40 → th



Local Density Estimators

Search for similar events that are already classified and count the members of the two classes.





k-Nearest-Neighbour

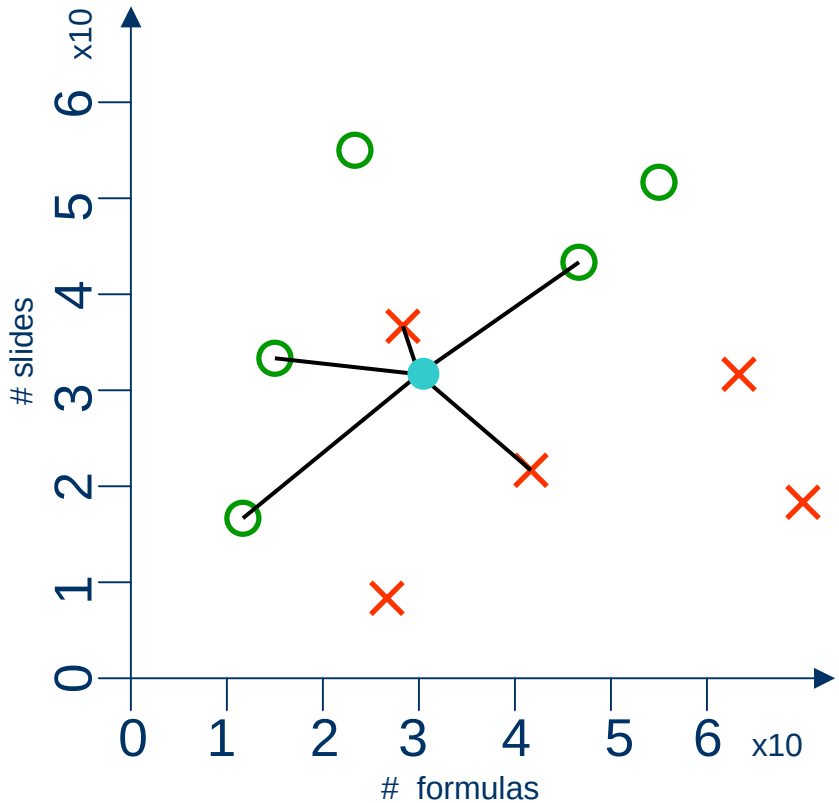
k=1
out=

k=2
out=

k=3
out=

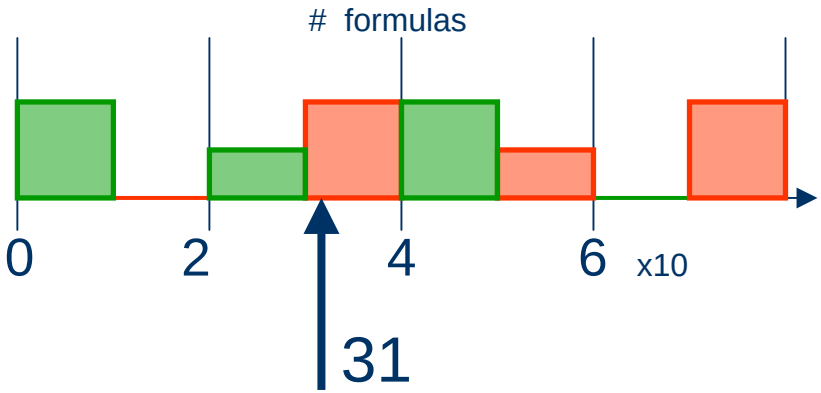
k=4
out=

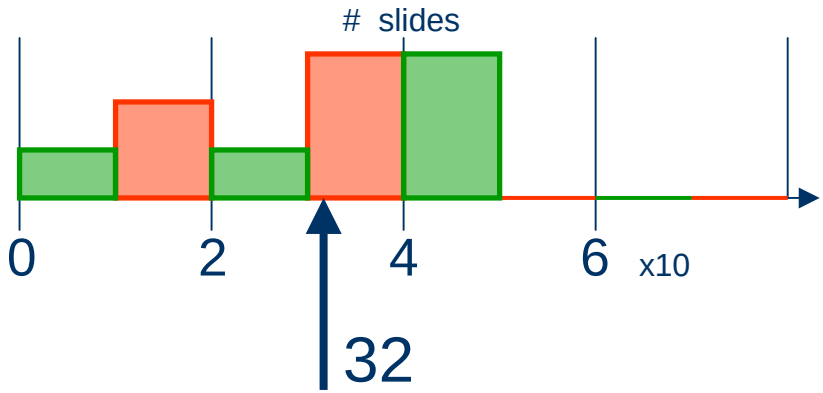
k=5
out=



For every evaluation position the distances to each training position need to be determined!

Maximum Likelihood / Naive Bayes





$$p_{Th} = \frac{2}{5} \times \frac{3}{5} = 0.24$$

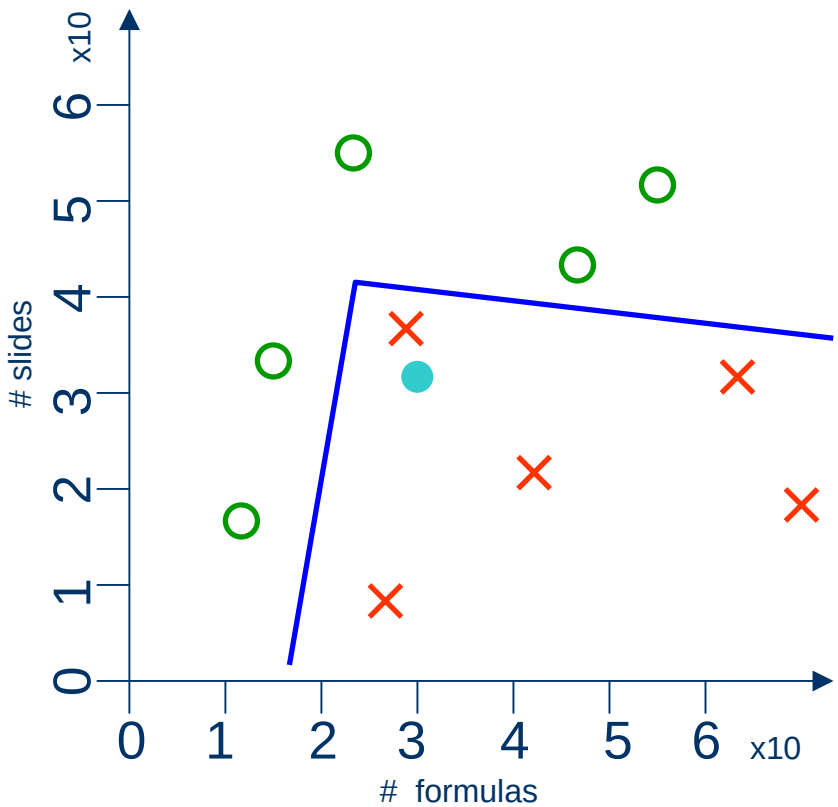
$$p_{Exp} = \frac{1}{5} \times \frac{1}{5} = 0.04$$

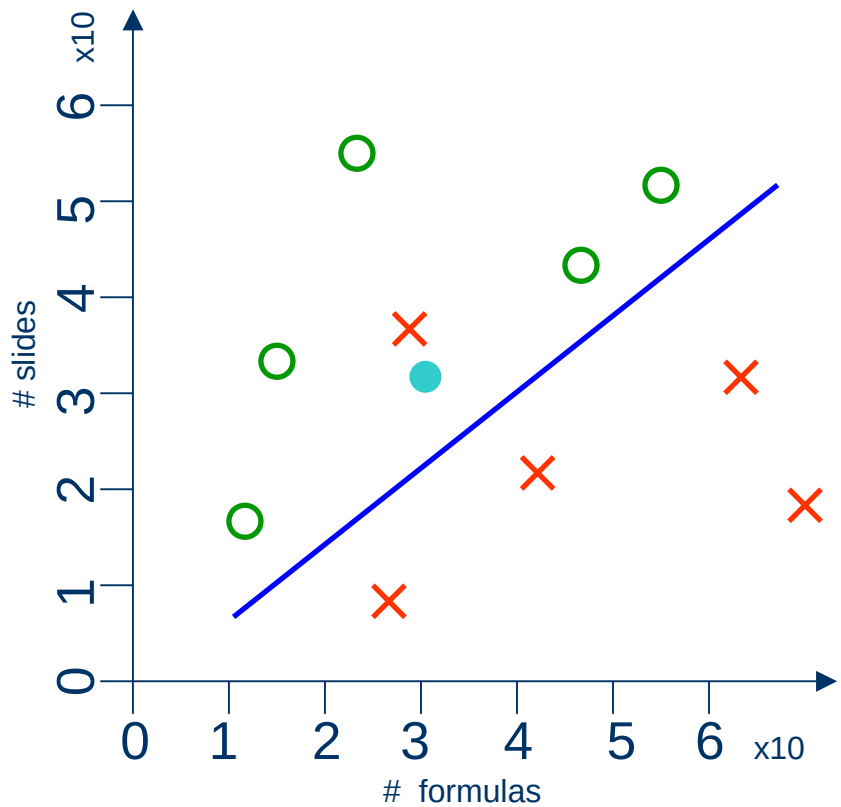
out= ×

Correlation gets lost completely by projection!

Methods Based on Linear Separation

Divide the input space into regions
separated by one or more hyperplanes.
Extrapolation is done!





Linear Discriminant Analysis

$$y_k = \gamma_0 + \gamma_1 x_{k,1} + \cdots + \gamma_d x_{k,d} + \sqrt{v}\xi_k \qquad 1 \le k \le n$$

$$A = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,d} \end{pmatrix}$$

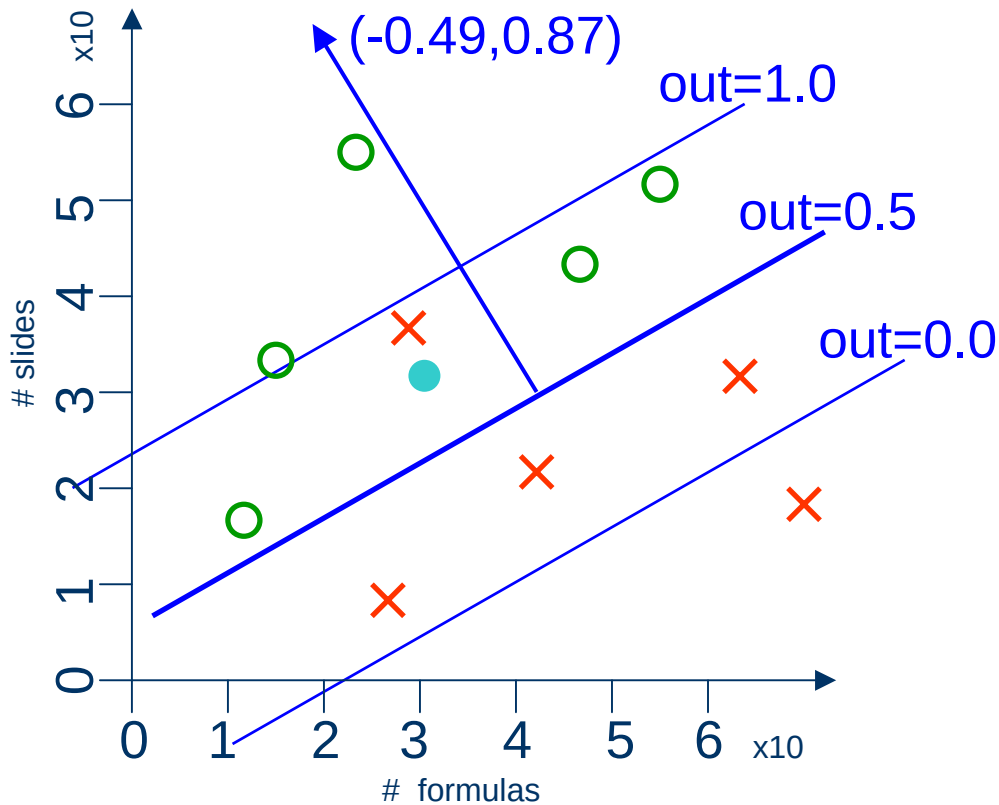
Fisher
1930

$$Y = A\gamma$$

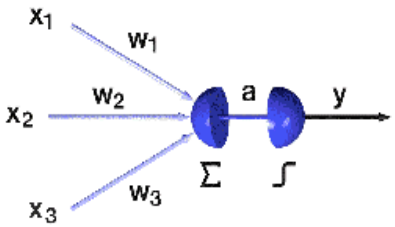
$$\hat{\gamma} = (A^T A)^{-1} A^T Y$$

$$out = 0.3 - 0.012 \times formulas + 0.021 \times slides$$

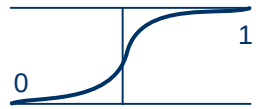
Only one separating hyperplane is usually not enough!
Can we combine two or more?



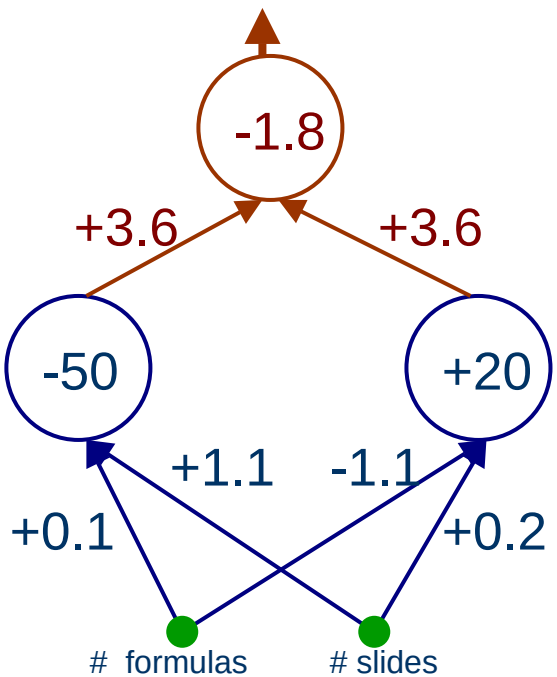
Neural Networks

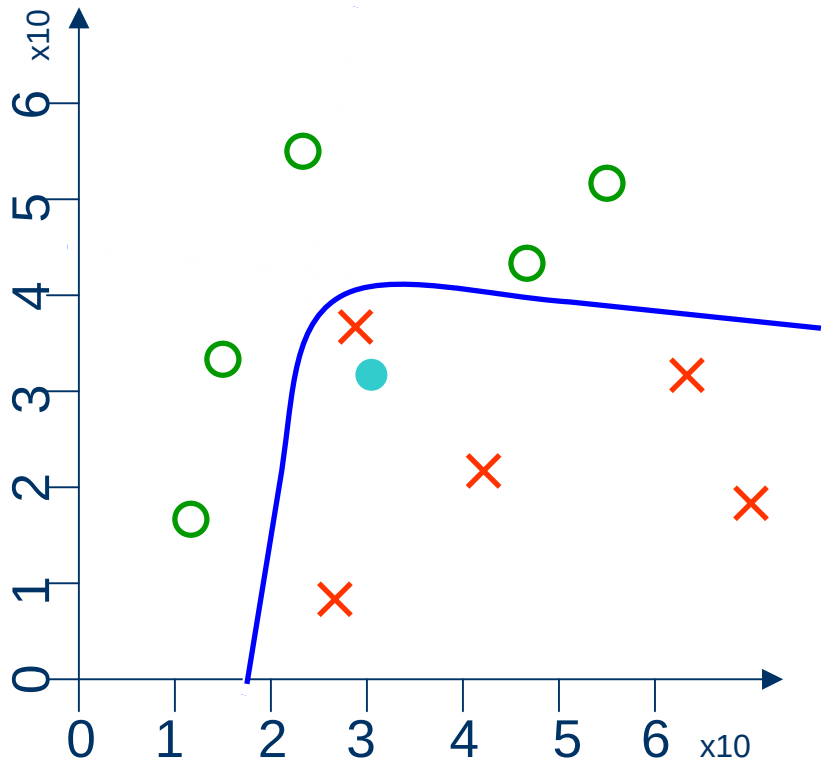


$$y = \sigma\left(\sum w_i x_i + s\right) \qquad \sigma(a) = \frac{1}{1+e^{-a}}$$



Train NN with two hidden neurons (gradient descent): $E = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - out(x_i)\right)^2$



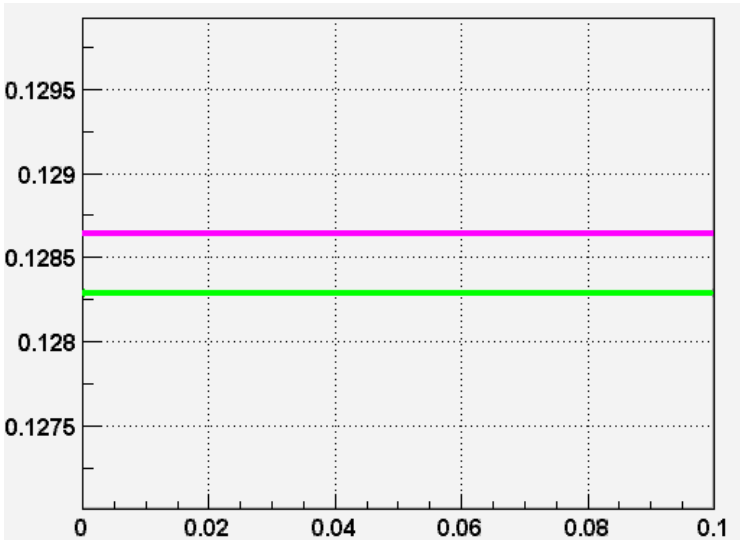


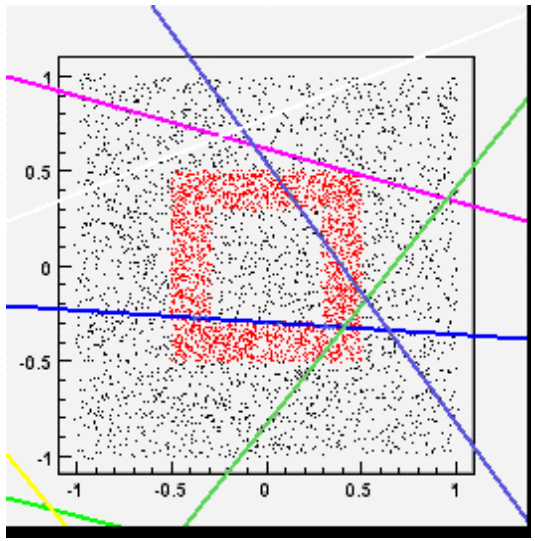
NN Training

8 hidden neurons = 8 separating lines

Test-Error

Train-Error

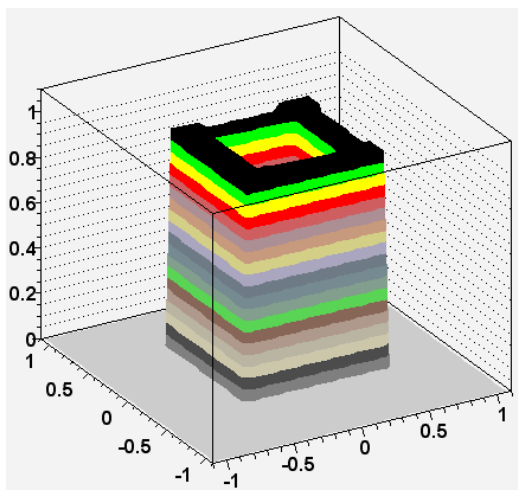




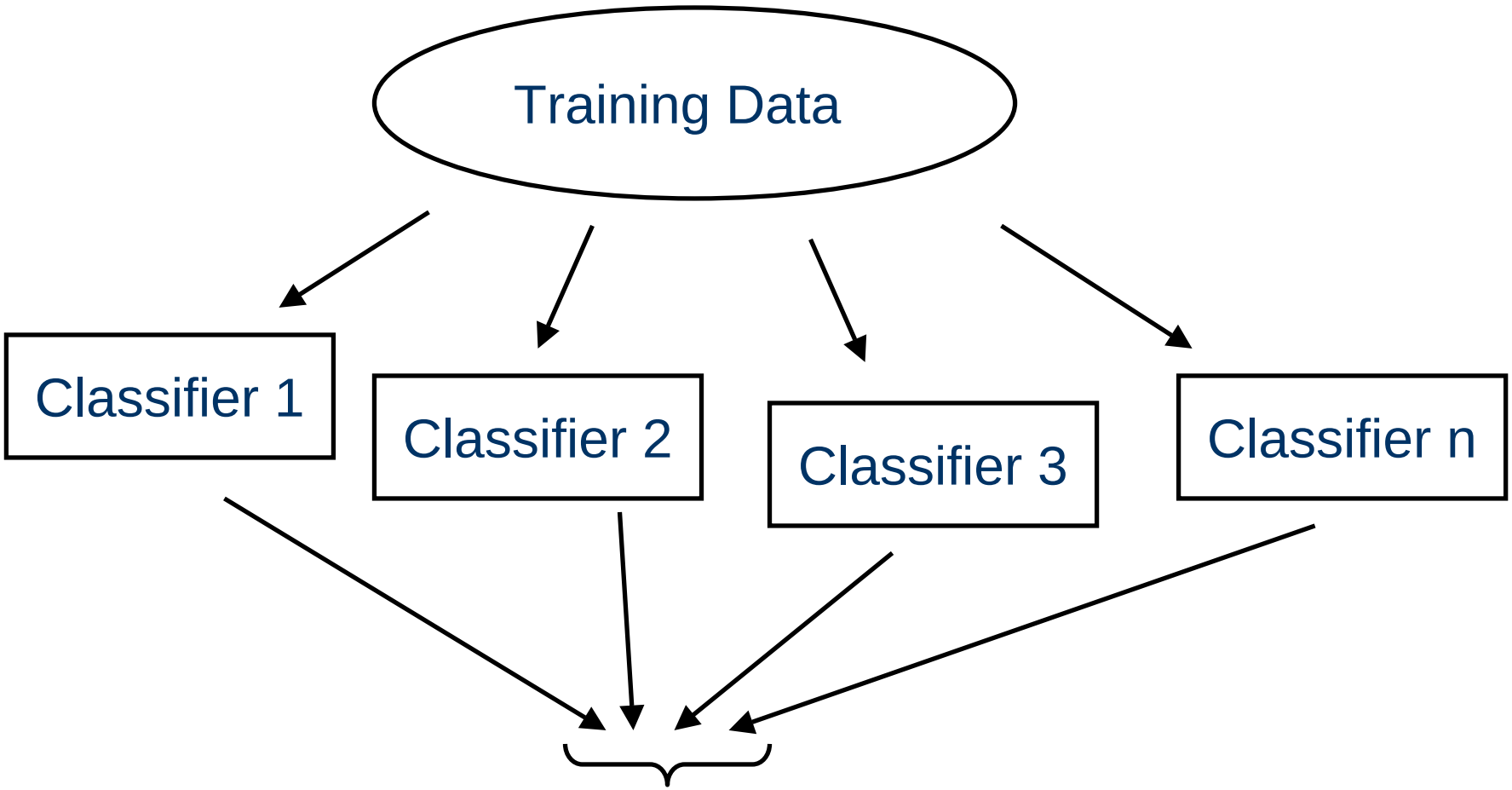
signal

background

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - out(x_i)\right)^2$$



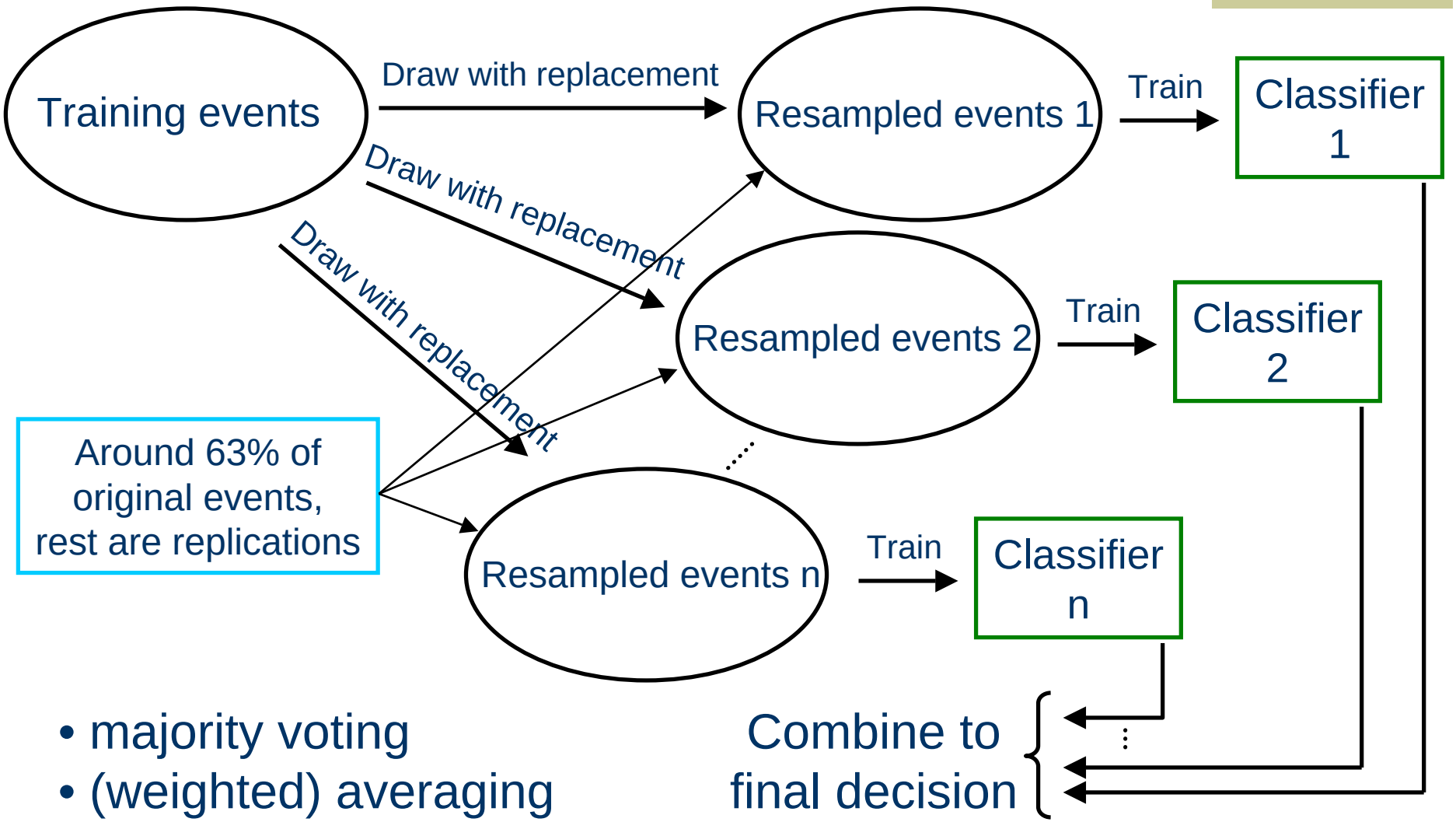
Meta-Learning Strategies



Combine different classifications
to one final decision

Bagging – Procedure

Bootstrap **agg**regat**ing**



- majority voting
- (weighted) averaging

Aim is to create strong classifiers which are as independent as possible.

Random Forests

Basis:

Decision Tree (CART) without pruning

Modification:

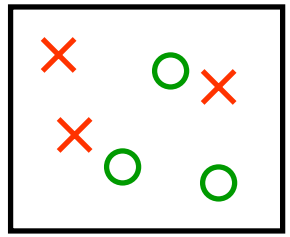
At each node of the tree: Search only through a randomly selected subset of all features

→ Use Bagging on this classifier

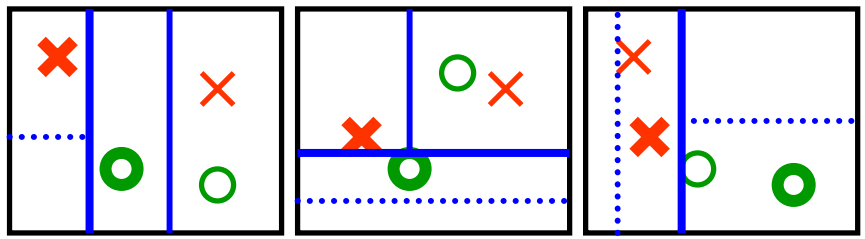
Tree, Randomness, Combination

→ RF

Training:

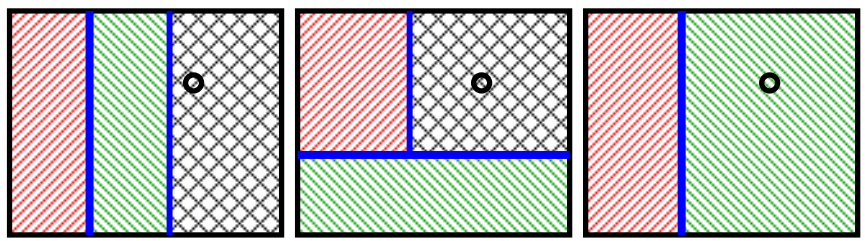


Create 3 trees

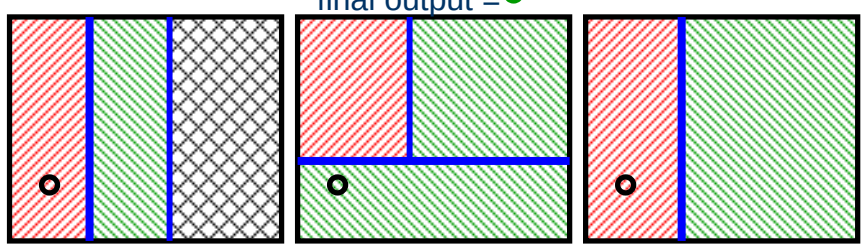


1 – 2,1 2 – 2,1 1 – 1,2

Testing/Evaluation:



final output =



final output =

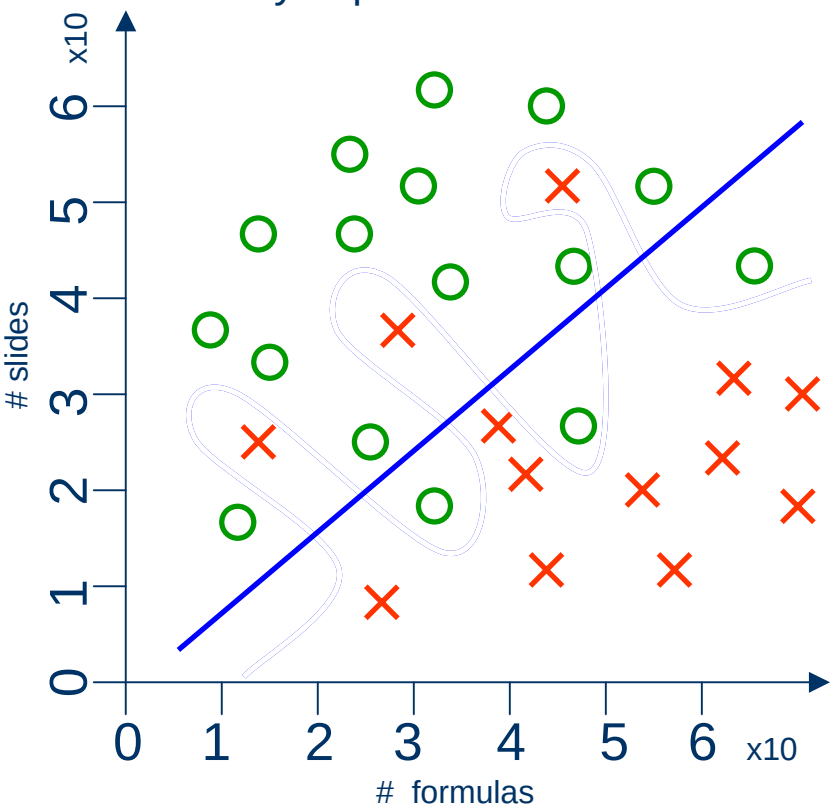
Training of Statistical Learning Methods

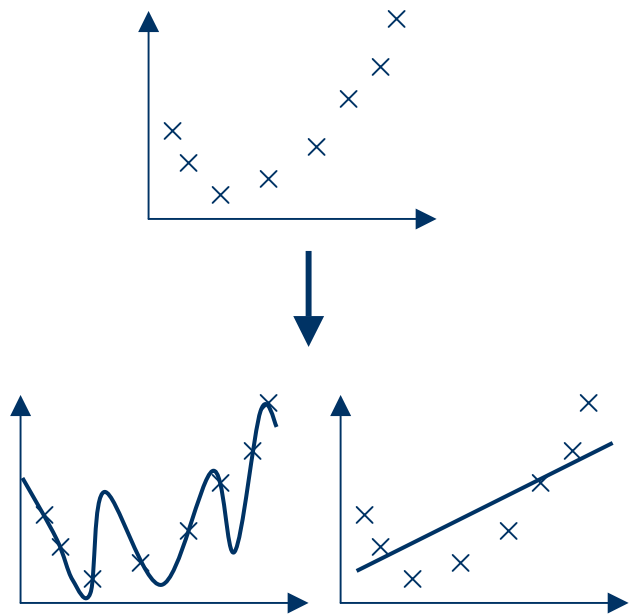
Statistical Learning Method:

From N examples $(\vec{x}_i, y_i)$ infer a rule $\vec{x} \rightarrow out(\vec{x})$

Important: Generalisation vs. Overtraining

Easily separable but with noise?



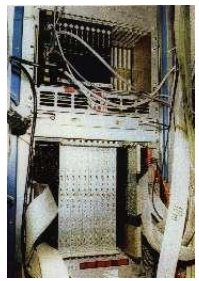


Too high degree of polynomial results in interpolation but too low degree means bad approximation

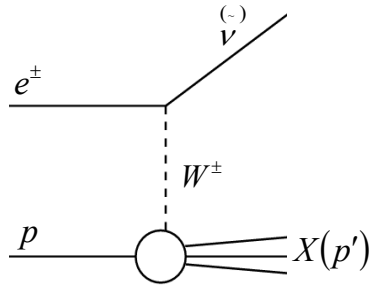
Applications in Physics Analysis

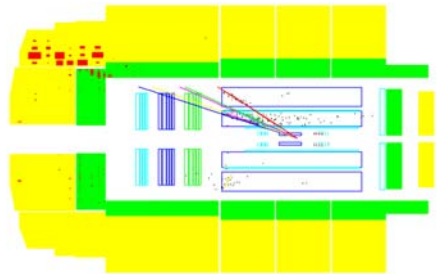
Classification Online „Trigger“

H1



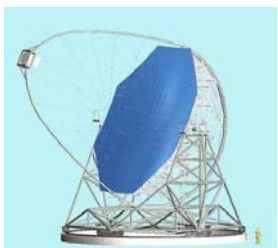
L2NN



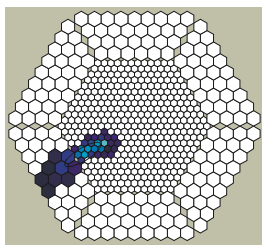


Charged Current

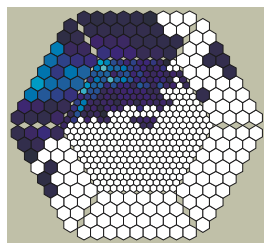
Classification Offline „Purification“



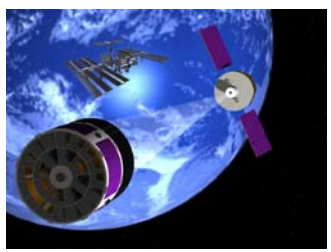
MAGIC



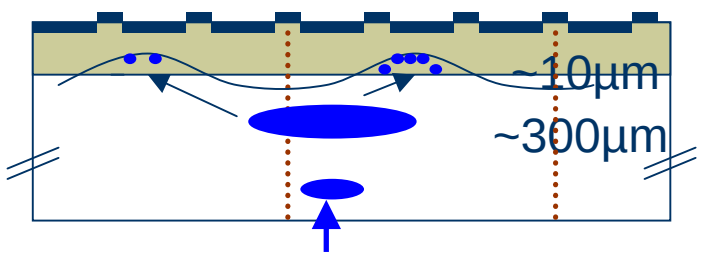
Gamma vs. Hadron



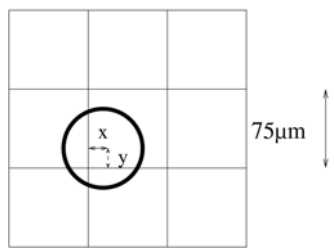
Regression



XEUS

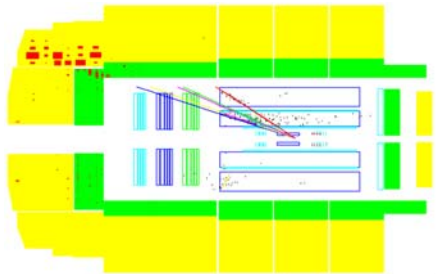


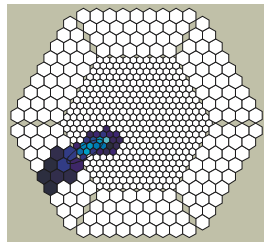
X-ray CCD

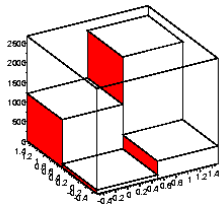


Features

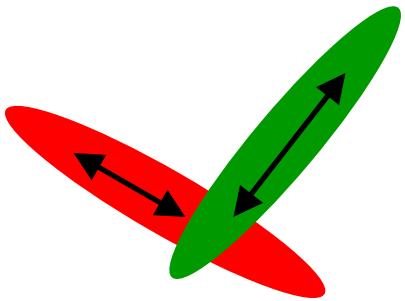
Choose raw quantities



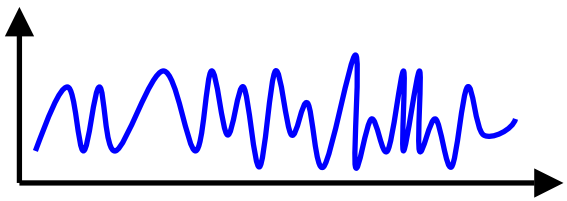


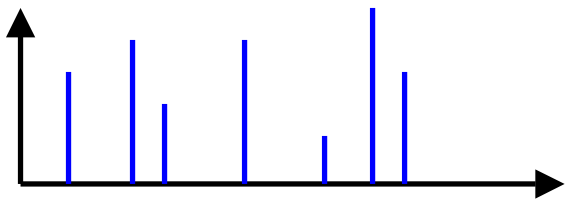


PCA

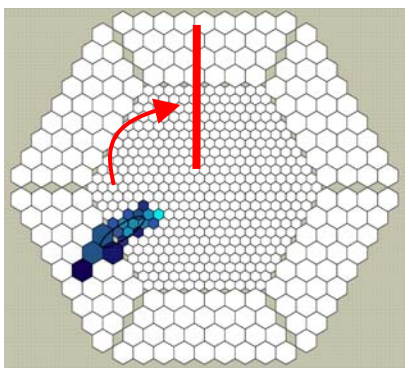


FFT

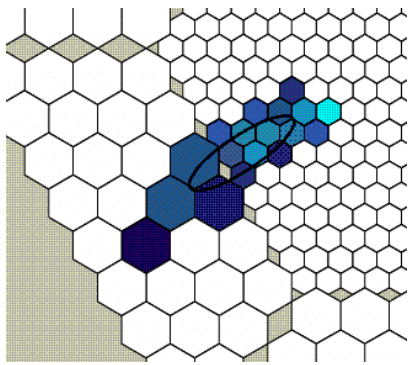




Symmetrie

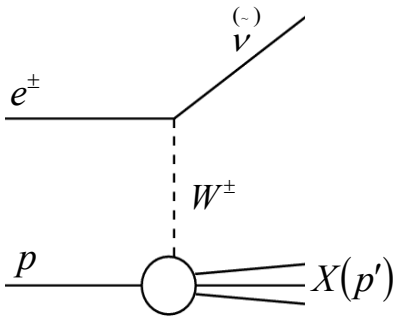


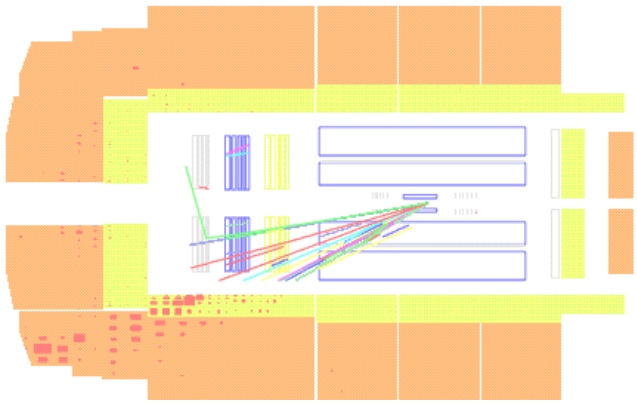
Fit



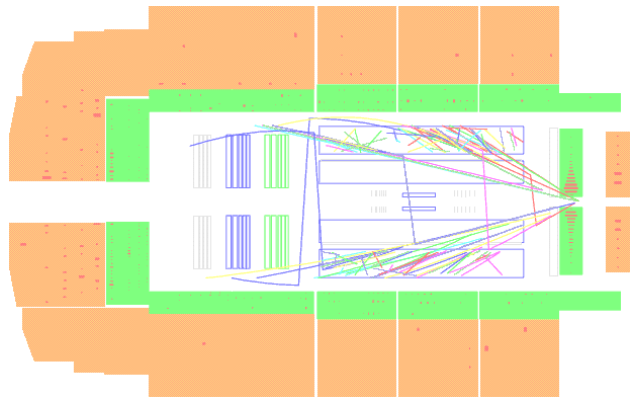
H1 - Triggering Charged Current Interactions

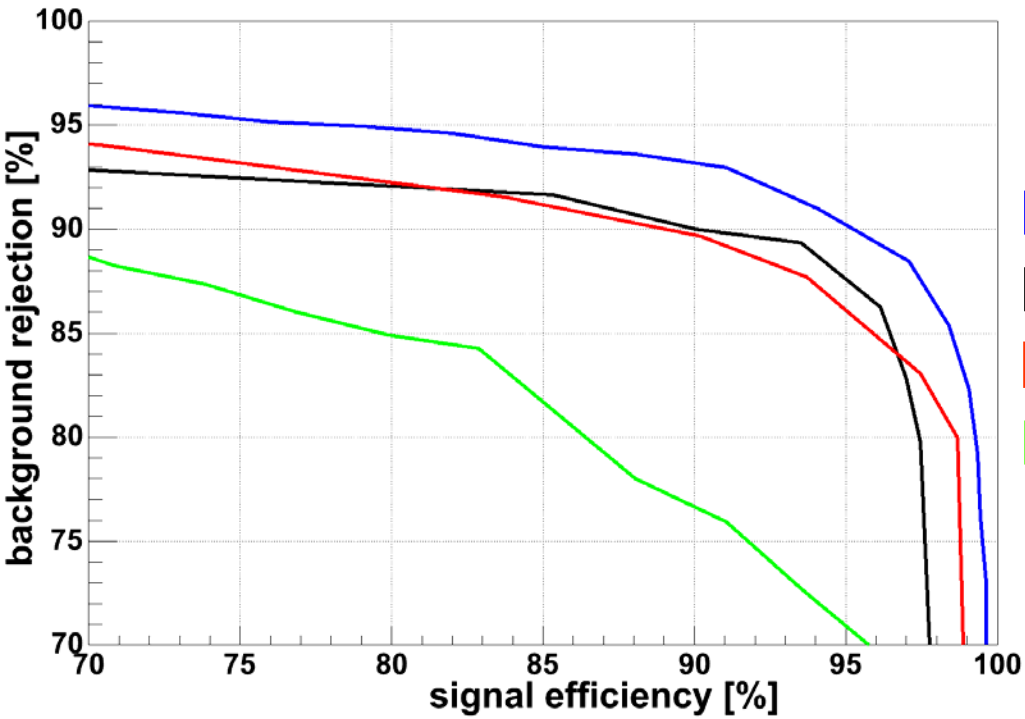
Charged Current Interaction (signal)





Beam Gas Event (background)





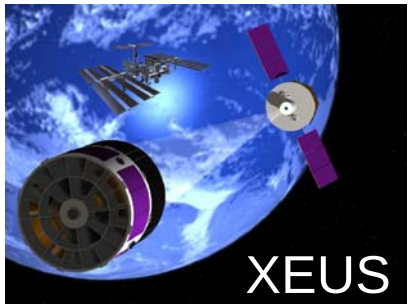
Neural Network

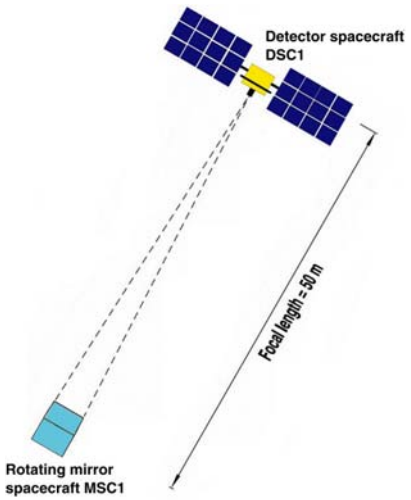
Decision Tree C4.5

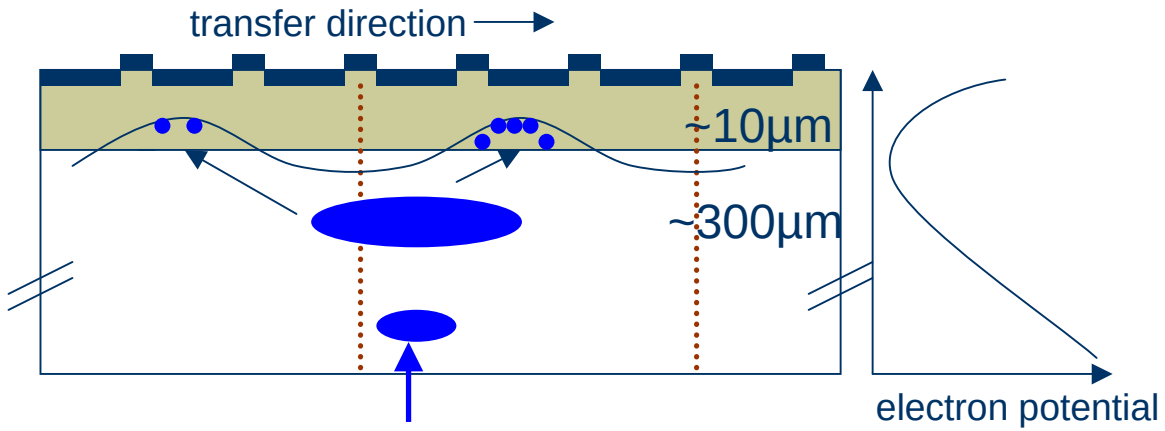
k-Nearest-Neighbour-Search

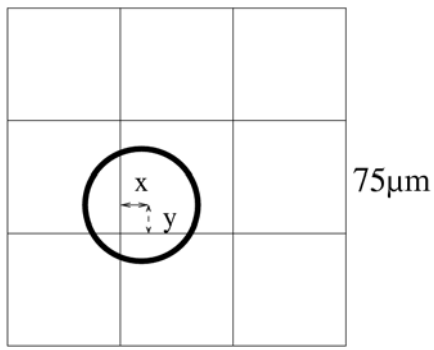
Maximum Likelihood

Regression of the Incident Position of X-ray Photons



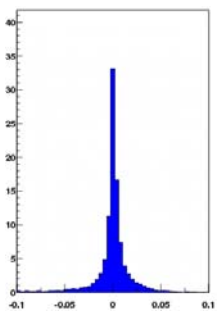






$$x_{COM} = \frac{\sum x_i c_i}{\sum c_i}$$

$$x_{CCOM} = x_{COM} + \Delta(x_{COM})$$



σ of reconstruction in µm:

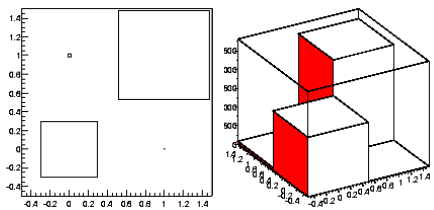
Neural Networks 3.6

k-Nearest-Neighbour 3.7

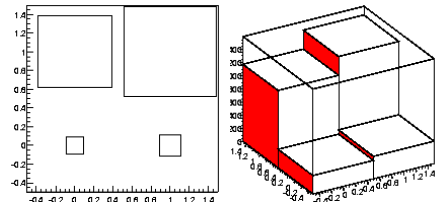
classical methods

ETA 3.9 CCOM 4.0

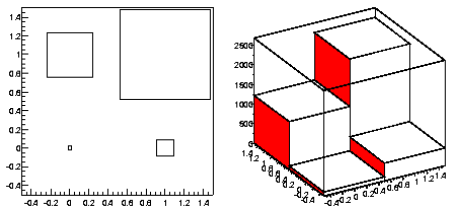
Pileup Recognition – Setup



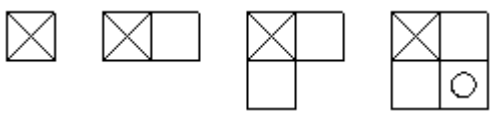
Pileup vs. Single photon



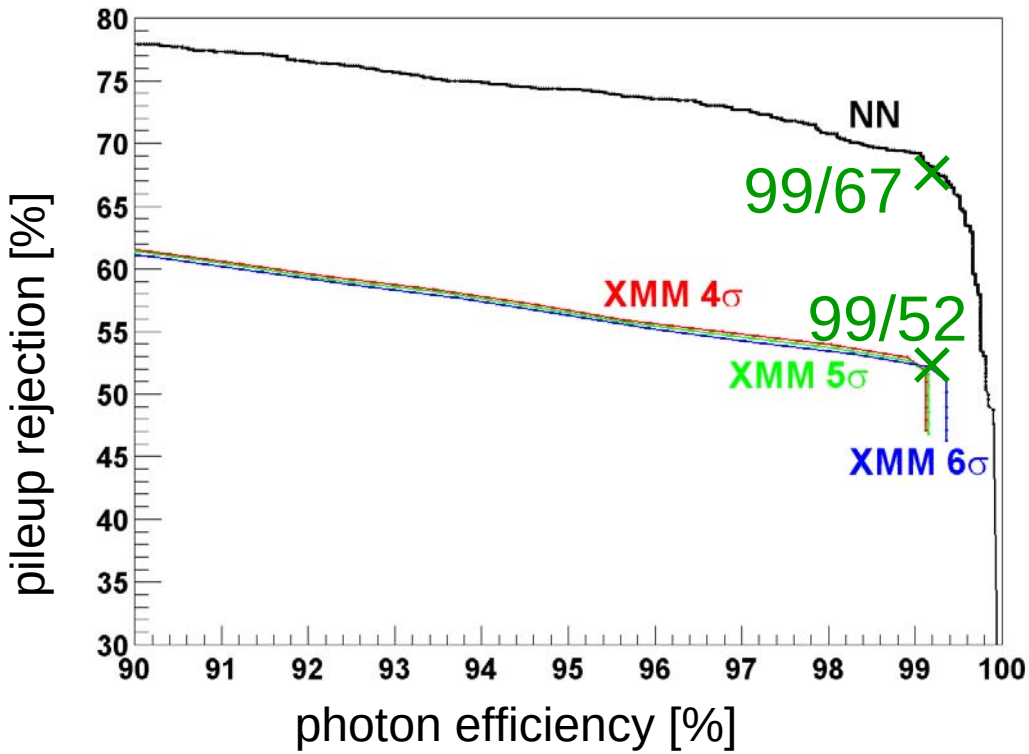
?



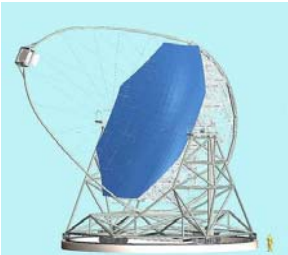
?



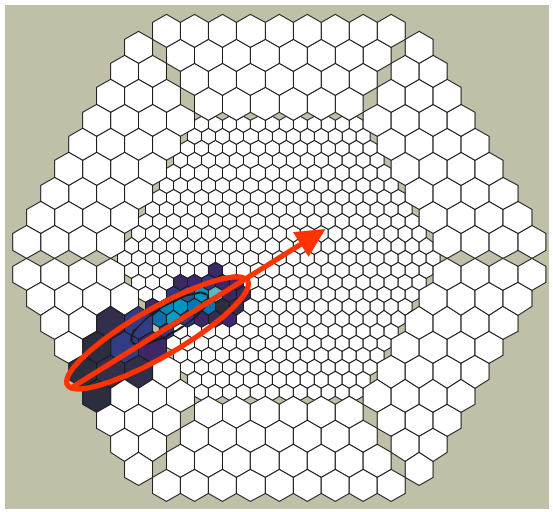
classical algorithm „XMM“

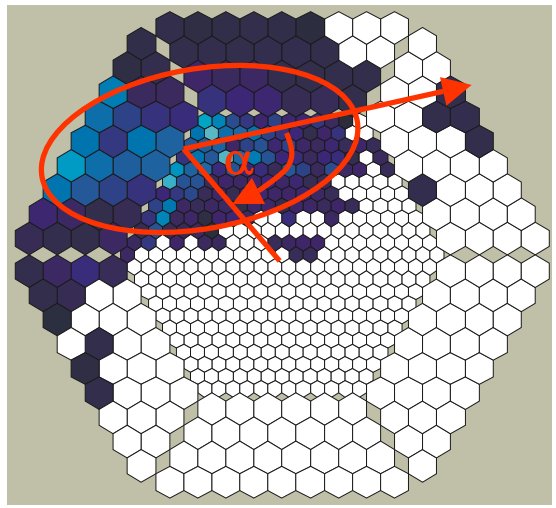


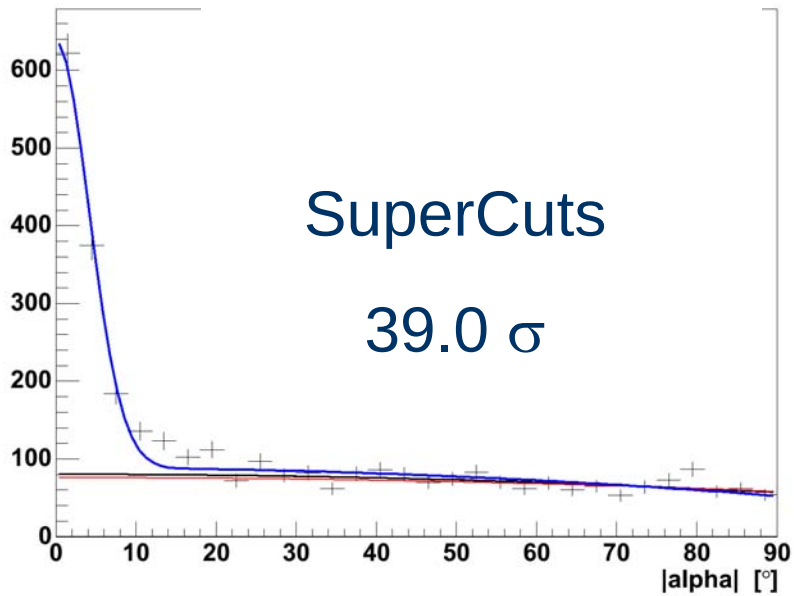
MAGIC - Gamma/Hadron Separation



Observation
Mkn421
22.04.04

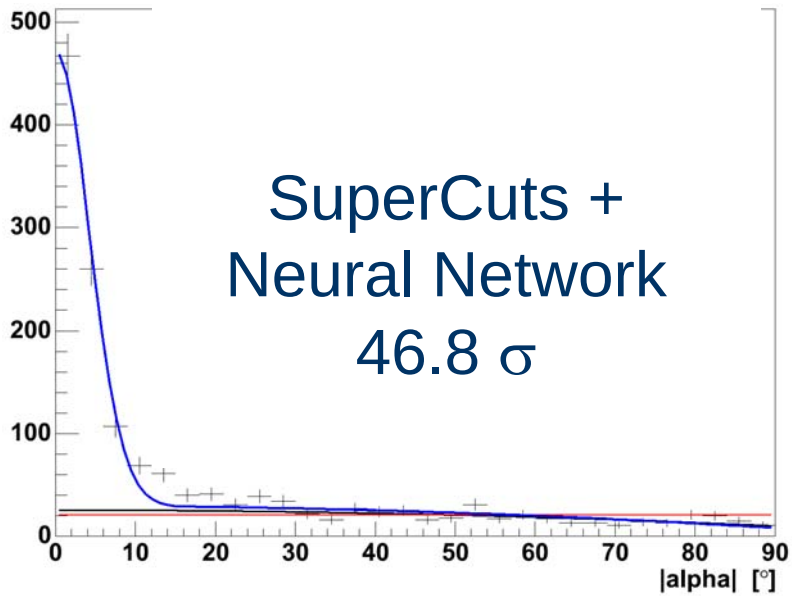






SuperCuts

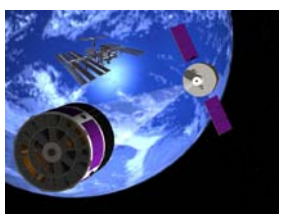
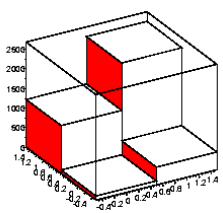
39.0 σ

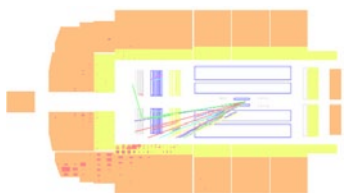
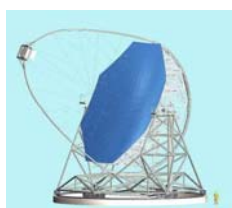
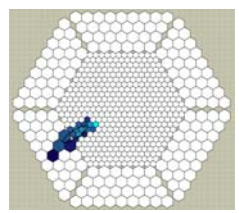


SuperCuts +
Neural Network
46.8 σ

Summary

- Three classes of statistical learning methods
  - Decision Trees (Bagging) — C4.5, CART, Random Forest
  - Local Density Estimators — Maximum Likelihood, k-Nearest-Neighbour
  - Linear Separation — Linear Discriminant Analysis, Neural Networks, Support Vector Machines
- Many applications in current experiments and analysis

- Compared to classical methods usually at least small improvements

Conclusion

Up To You

Recognize possible applications

Intelligent Trigger? Purification?

Control Training and Evaluation

Overtraining? Systematic Uncertainties?

Compare classical and different statistical learning methods

Time? Performance? Convenience?

Win efficiency / significance

Theory of Communication: Minimum Description Length Principle

Hypothesis *H* and Data *D*

$$P(H \mid D) \cdot P(D) = P(H \cap D) = P(D \mid H) \cdot P(H)$$

$$P(H \mid D) = \frac{P(H)P(D \mid H)}{P(D)}$$

Bayes

18th century

Our hypothesis should have the maximum probability given the data:

$$\text{maximize } P(H \mid D)$$

$$\text{maximize } P(H)P(D \mid H)$$

$$\text{maximize } \log P(H) + \log P(D \mid H)$$

Shannon 1948

$$I(X) \propto -\log P(X)$$

Rissanen 1990

→ MDLP $\text{minimize } I(H) + I(D \mid H)$

Statistical Learning Theory: Structural Risk Minimization

We have N training events with input x_i and correct output y_i

$$R_{emp}(\alpha) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i,\alpha)\right)^2$$

empirical risk

$$R(\alpha) = \int \left(y - f(x,\alpha)\right)^2 dP_{X,Y}$$

actual risk

Relationship is uniform convergence:

$$P\left(\sup_\alpha \left|R(\alpha) - R_{emp}(\alpha)\right| > \varepsilon\right) \xrightarrow{N\to\infty} 0$$

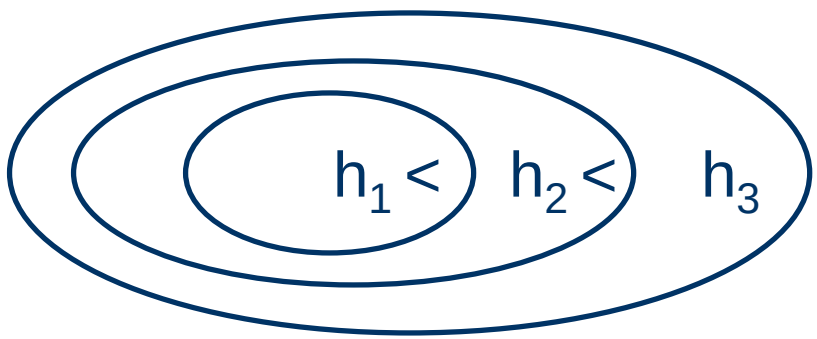
Upper bound for the actual risk (Vapnik):

1996

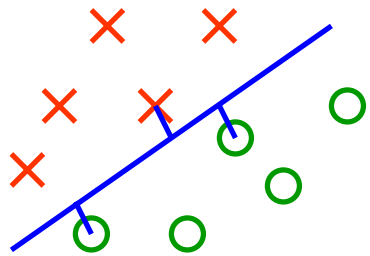
$$R(\alpha) \le R_{emp}(\alpha) + \varepsilon(N,p,h)$$

h: VC Dimension of learning method (*complexity*)

→ Create nested subsets of function spaces with rising complexity



Support Vector Machines



Separating hyperplane with maximum distance to each datapoint: Maximum margin classifier

Found by setting up condition for correct classfication $y_i(\vec{w}\vec{x}_i + b) \geq 1$ and minimizing $\|\vec{w}\|^2$ which leads to the Lagrangian

$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum \alpha_i \left(y_i(\vec{w}\vec{x}_i + b) - 1\right)$$

Necessary condition for a minimum is $\vec{w} = \sum \alpha_i y_i \vec{x}_i$

So the output becomes $out = \mathrm{sgn}\left(\sum \alpha_i y_i \vec{x}_i \vec{x} + b\right)$ KKT: only SV have $\alpha_i \neq 0$

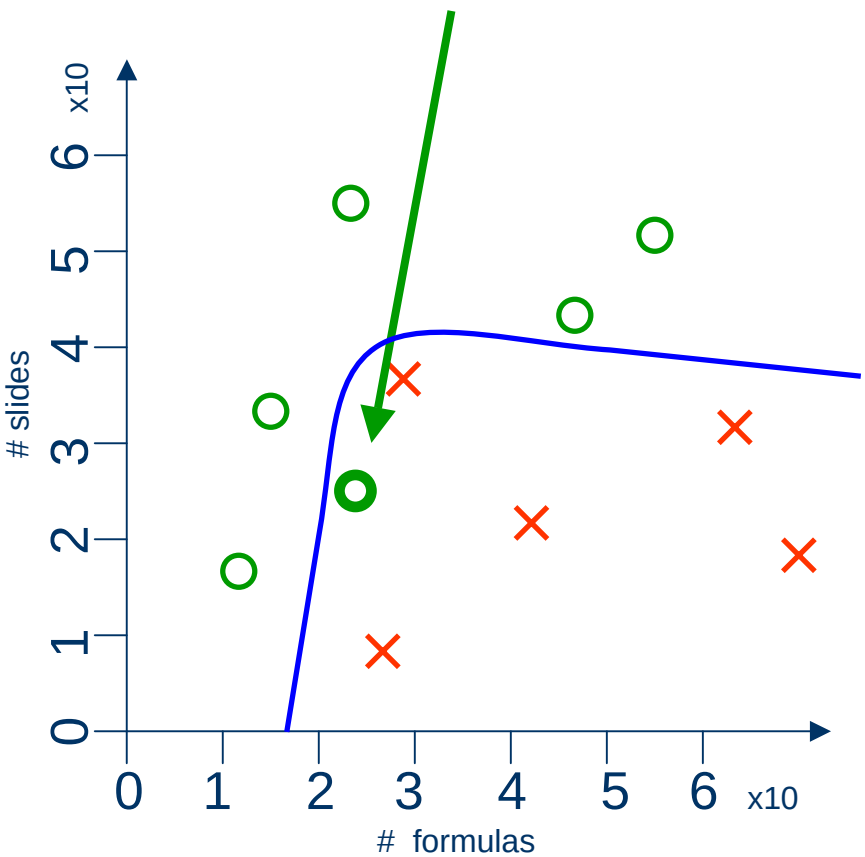
Only linear separation? No! Replace dot products: $\vec{x}\vec{y} \rightarrow \Phi(\vec{x})\Phi(\vec{y})$

The mapping to feature space $\Phi : R^d \rightarrow F$ is hidden in a kernel $K(\vec{x}, \vec{y}) = \Phi(\vec{x})\Phi(\vec{y})$

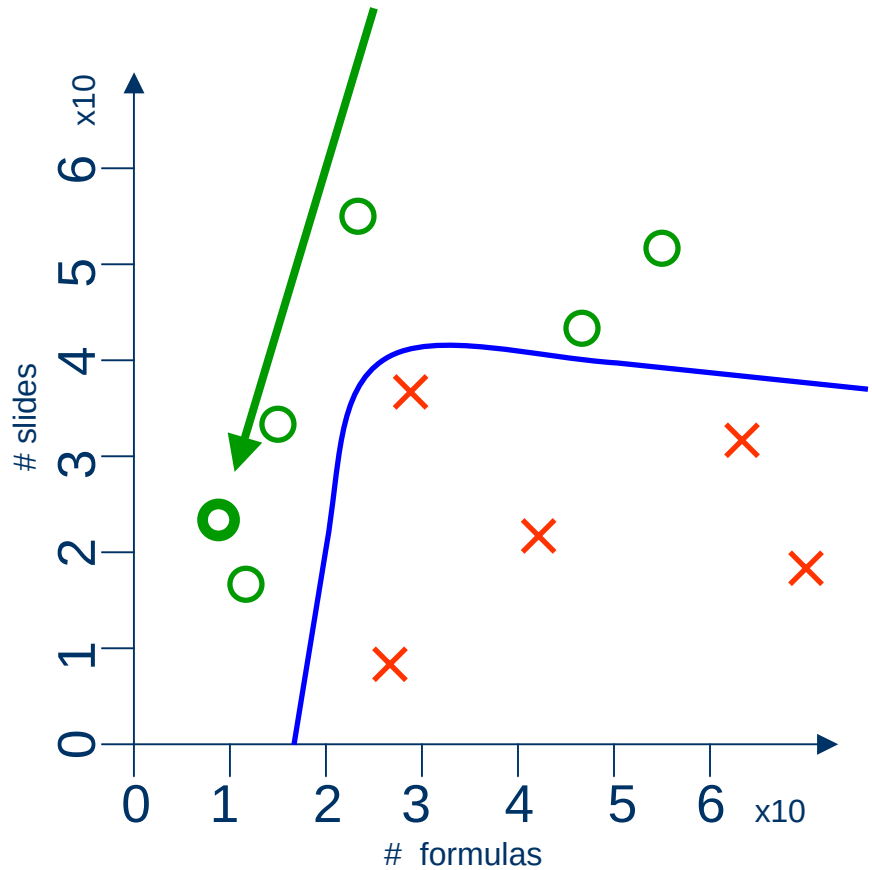
Non-separable case: $\frac{1}{2}\|\vec{w}\|^2 \rightarrow \frac{1}{2}\|\vec{w}\|^2 + C\sum \xi_i$

Finally

Include Statistical Learning Theory:
~25 formulas on 27 slides

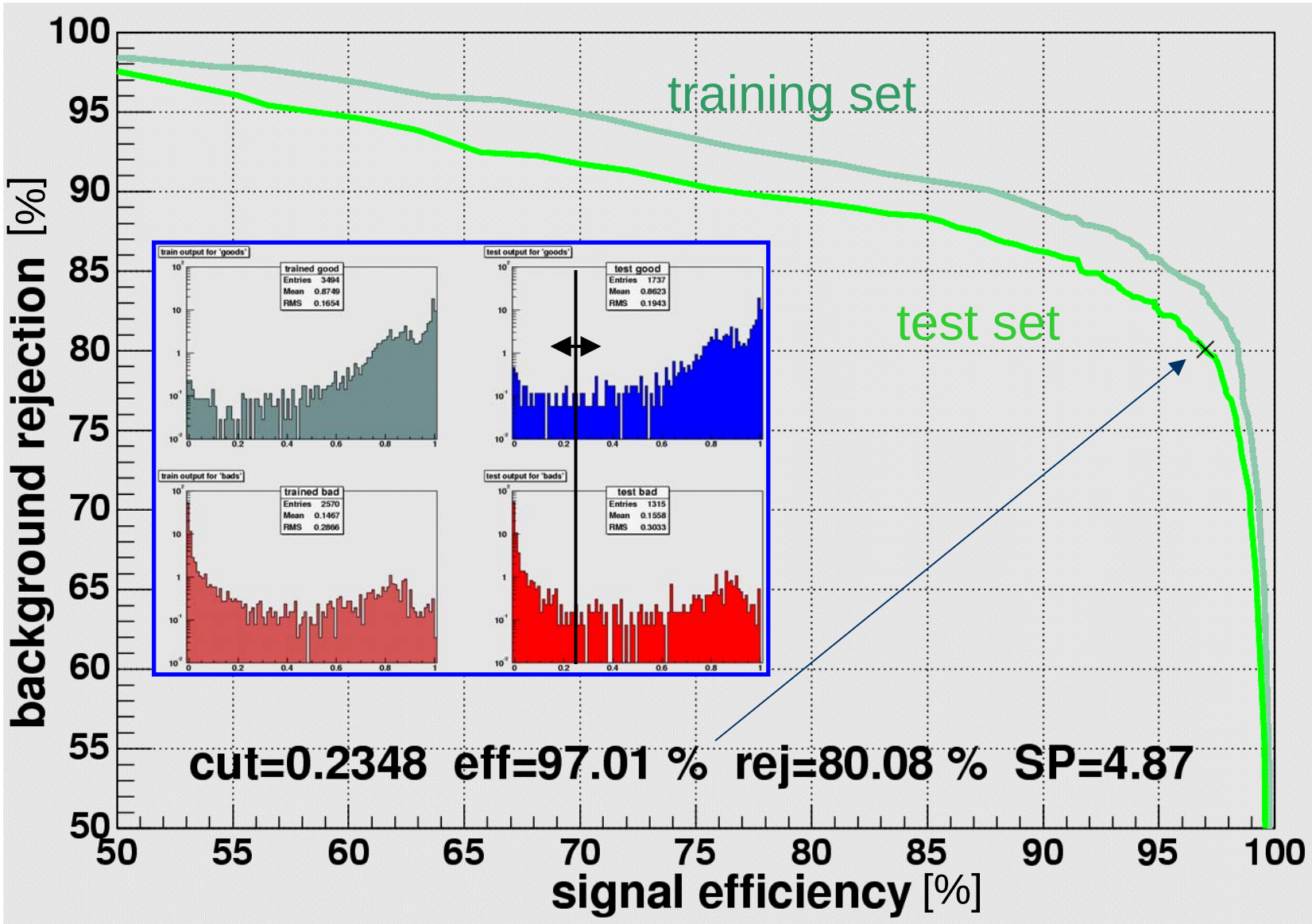


Skip Theory:
~7 formulas on 24 slides



Backups

Deeply Virtual Compton Scattering (S41) - Efficiency/Rejection



Supervised Learning needs Training Data

Consequences

- Correct output t_i is known for every learning sample because
  - Data was generated by simulation
  - Data was taken from experiment under specific conditions 

NN may train on unwanted simulation-features

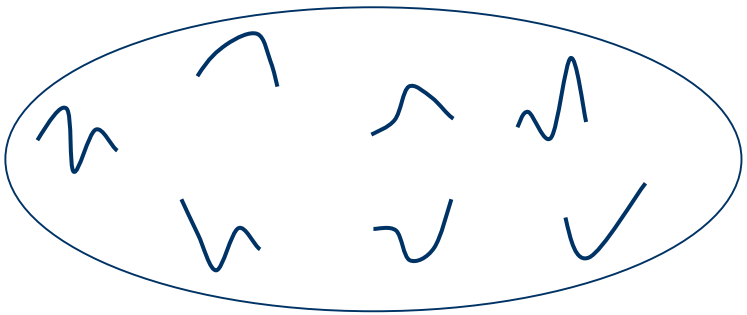
Difficult / bad precision

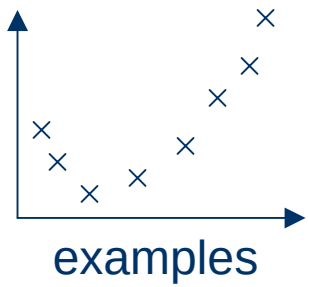
- Correct output t_i is not known but very well approximated by some complicated analysis

NN will never outperform this analysis

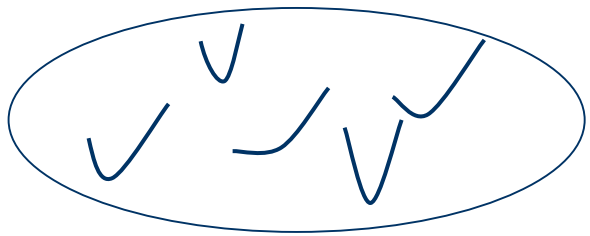
Statistical Learning vs. Parametric Models

large function space
given by learning method
„Training“

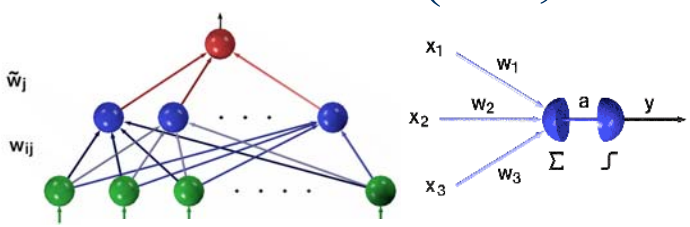




small function space
given by user (knowledge!)
„Fitting“



Gradient descent (NN)
QP-Solver (SVM)
Weighted counting (LDE)
Find Max/Min (ML,k-NN)



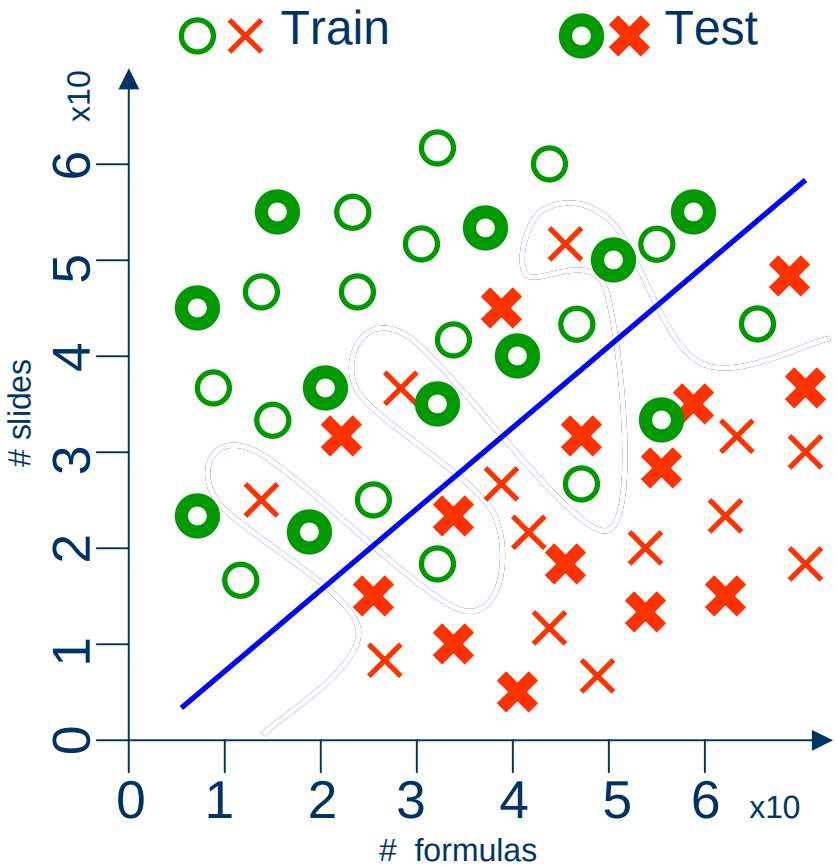
$$y = \sigma\left(\sum w_i x_i - s\right)$$

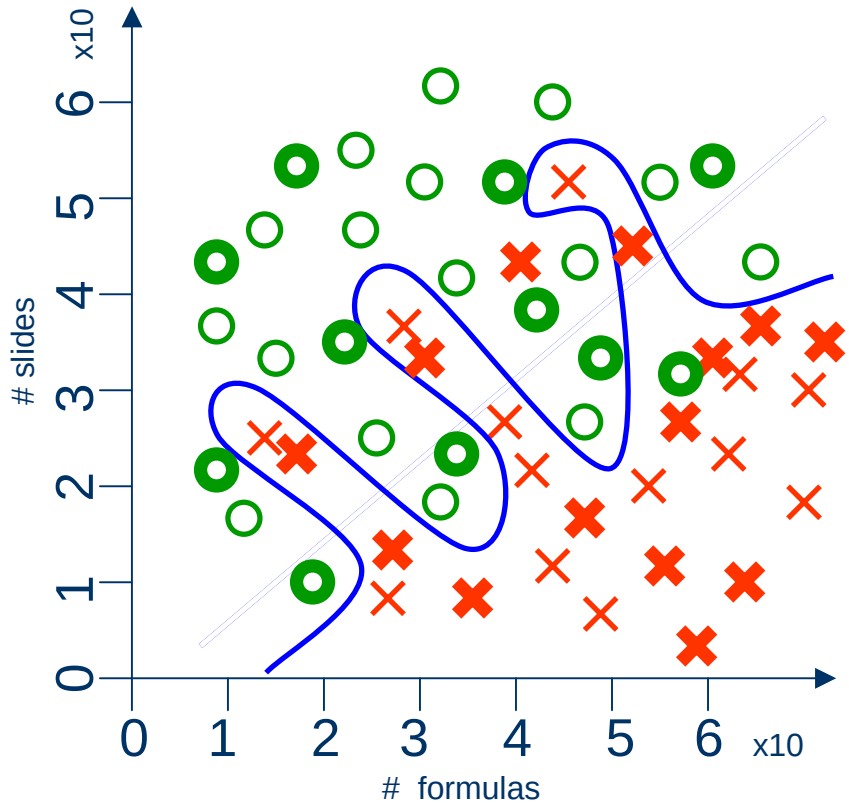
Minimize error:
e.g. least squares fit
➔ use e.g. Minuit

$$E_r = \left(a + b \cdot SIZE + c \cdot IMPACT + d \cdot LENGTH + e \cdot WIDTH + \right.$$
$$\left. + f \cdot SIZE \times LENGTH + g \cdot SIZE \times WIDTH \right) \cdot \frac{\cos^{-2}\theta + h \cdot \cos^{-4}\theta}{1+h}$$

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\left(E_i - Er_i\right)^2}{E_i}$$

See Overtraining - Want Generalization → Need Regularization

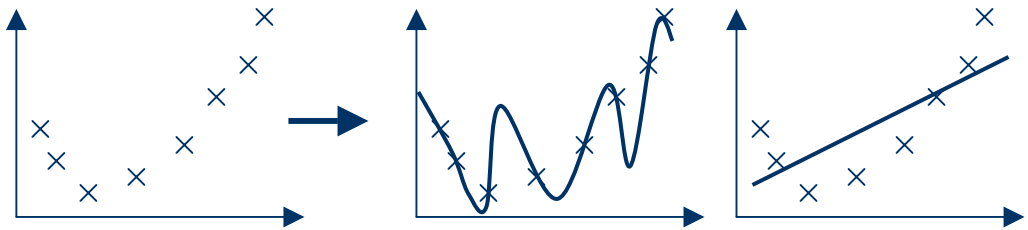




$$E = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - out(x_i)\right)^2$$

We want to tune a parameter of the learning algorithm depending on the overtraining we see!

Overtraining

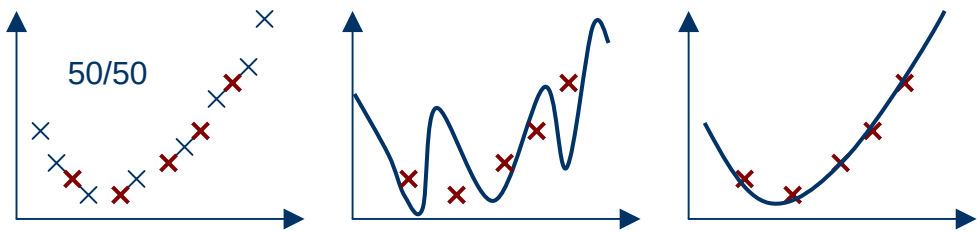


- Too many free parameters
- Too large function space
- Learning „by heart“

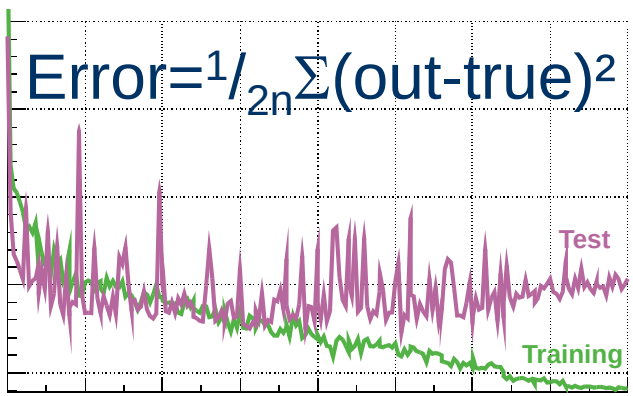
Too high degree of polynomial results in interpolation
but too low degree means bad approximation

→ MDLP

Control the behaviour:
Split learning samples randomly
into training samples ×
and test (validation) samples ×



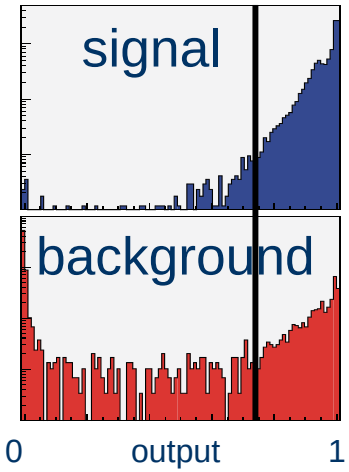
XEUS –
Classification of
photon events with
limited statistics



Overtraining in different
learning methods:
NN: # hidden neurons
LDE: width of local region
SVM: # support vectors

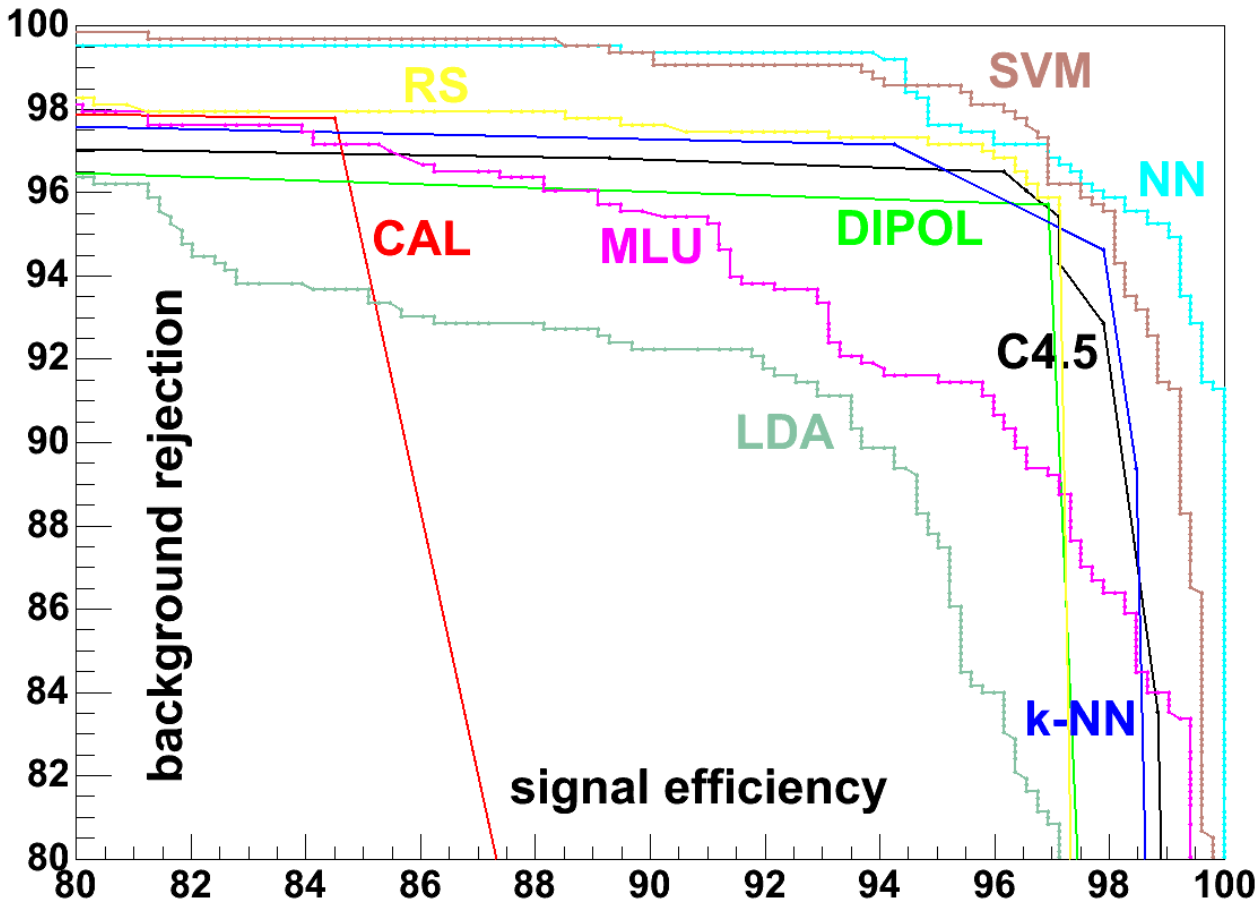
→ VC-Dim

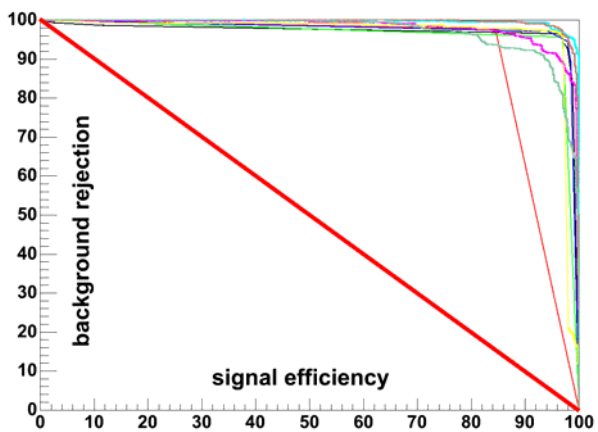
Performance Measurement - Classification



Eff@Rej = xx%
Rej@Eff = xx%

Misclassification =
200%-Eff-Rej

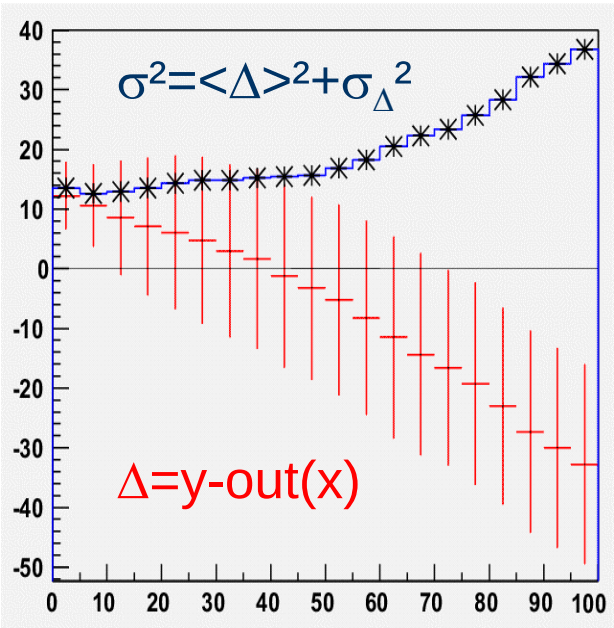




Performance Measurement - Regression

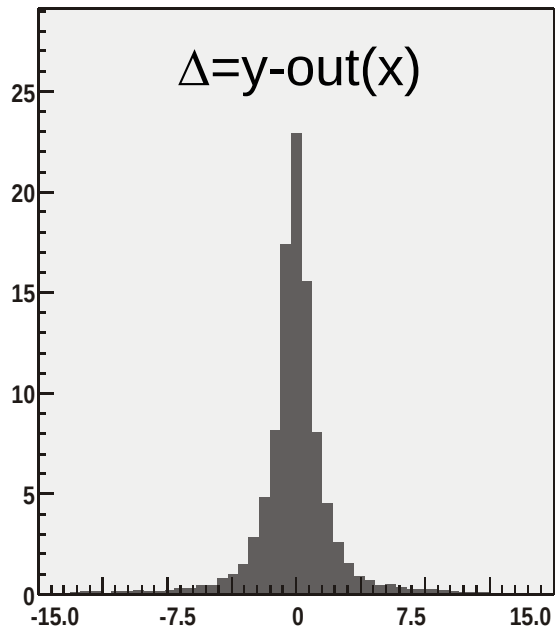
$$E = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - out(x_i)\right)^2$$

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - out(x_i)}{y_i}\right)^2$$

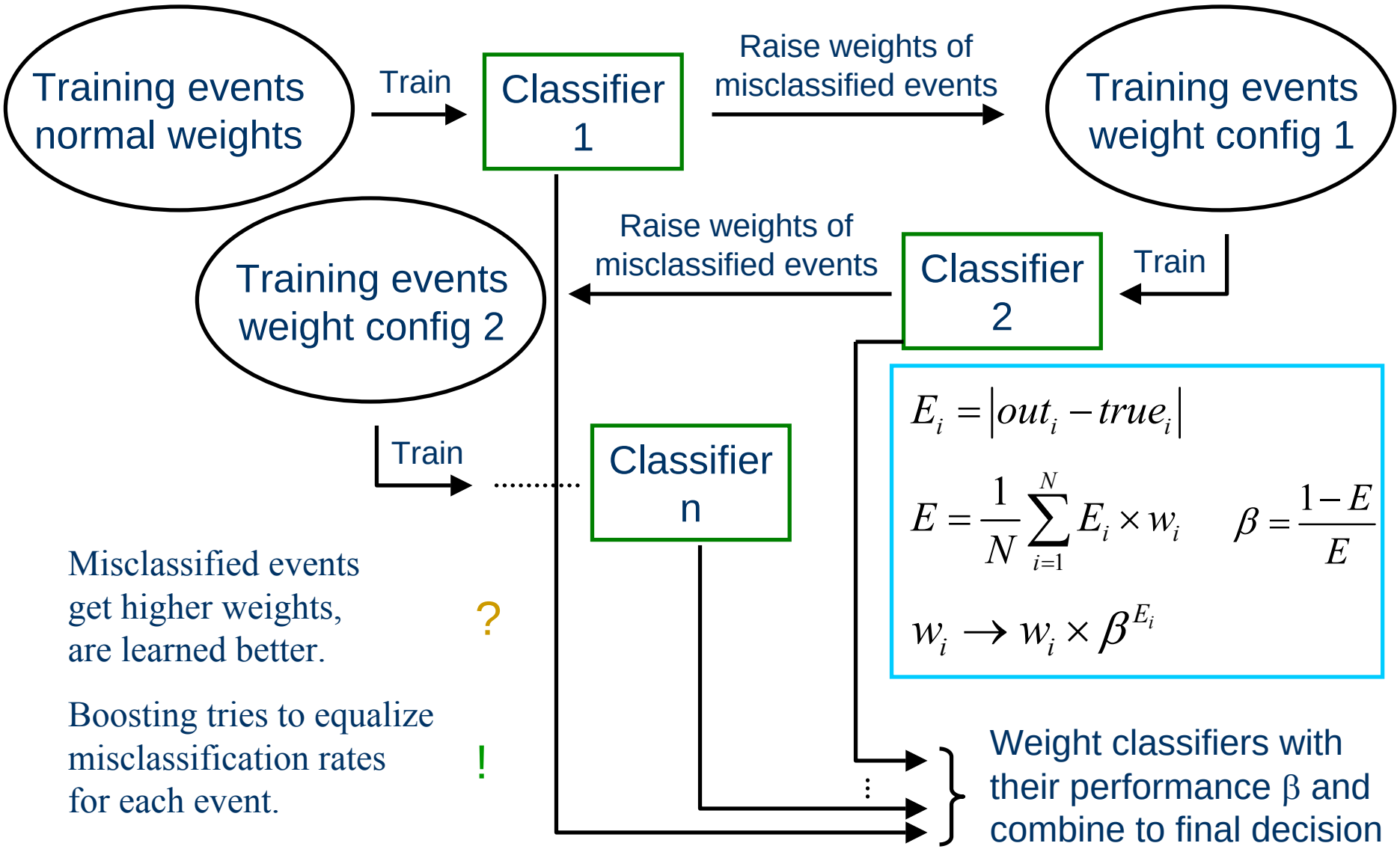


$$m = \left\langle y_i - out\left(x_i\right)\right\rangle$$

$$s^2 = \left\langle \left(y_i - out\left(x_i\right)\right)^2 \right\rangle - m^2$$

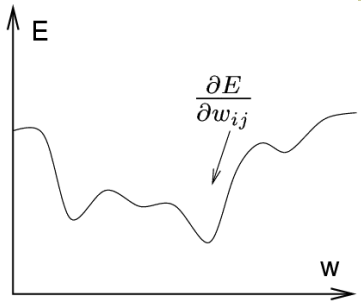


Boosting – Procedure

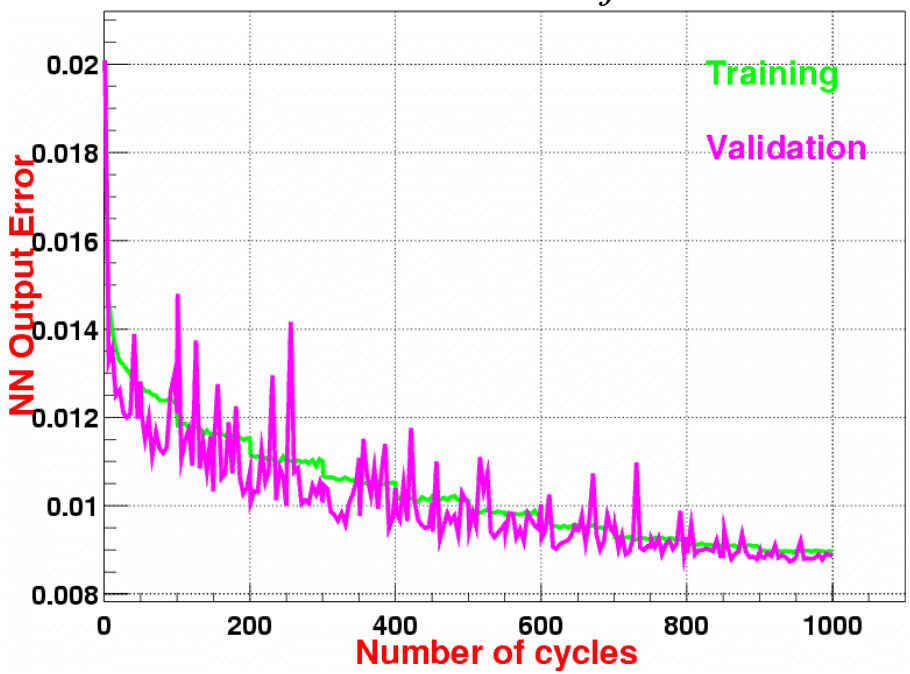


Training Neural Networks – Theory

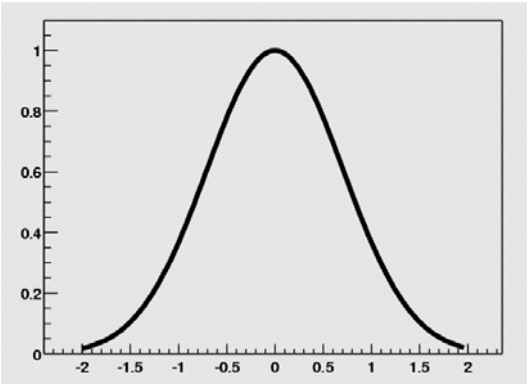
$$E = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - out(x_i)\right)^2$$



$$\Delta w_{ij}\left(t+1\right) = -\eta\frac{\partial E}{\partial w_{ij}} + \alpha\Delta w_{ij}\left(t\right)$$



Regression with Neural Networks

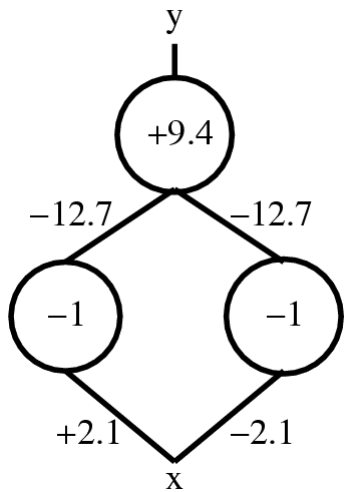


- Example: Train Gauß with two hidden neurons
- Minimization of error during training is

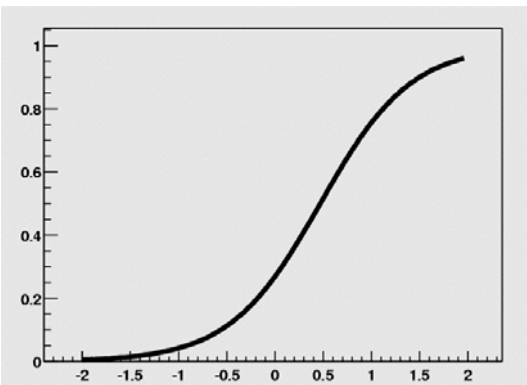
  scaling and shifting the two sigmoids

  scaling and shifting their sum

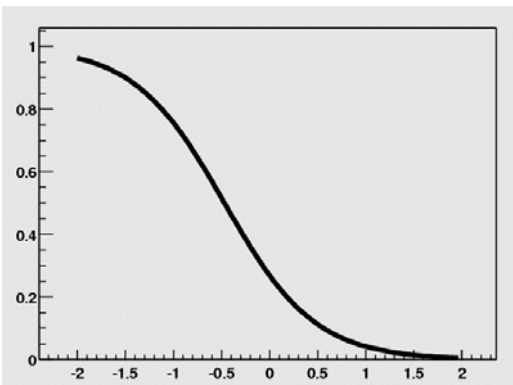
  to fit the function



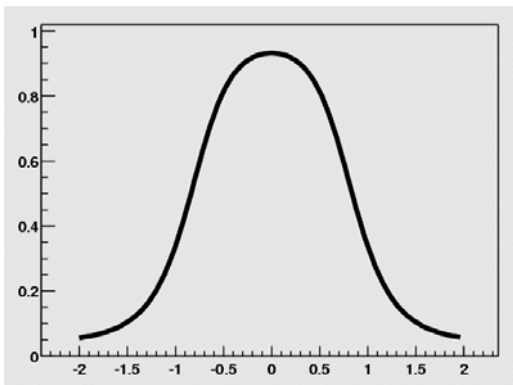
σ(+2.1x - 1)



σ(-2.1x - 1)

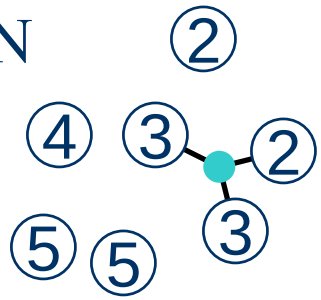


output



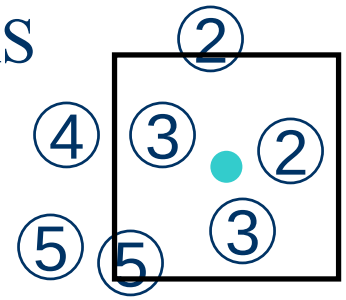
From Classification to Regression

k-NN



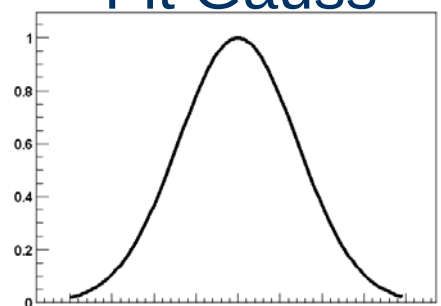
$$out = \frac{8}{3}$$

RS



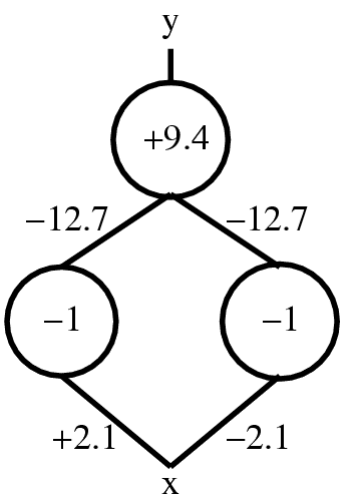
$$out = \frac{13}{4}$$

Fit Gauss

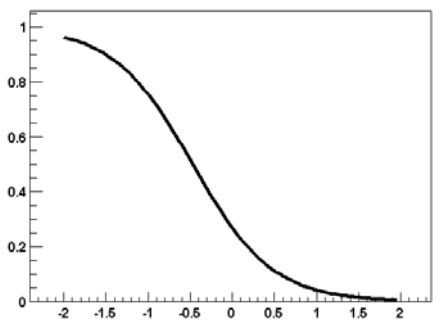


NN

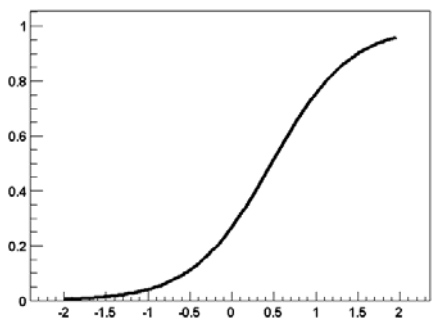
$$E = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - out(x_i)\right)^2$$



a=σ(-2.1x - 1)



b=σ(+2.1x - 1)



out=σ(-12.7a-12.7b+9.4)