Multivariate analysis methods in high energy physics

Marcin Wolter

IFJ PAN, Kraków

Warszawa, 5 May 2006

What is the machine learning?

- "*The goal of machine learning is to build computer systems that can adapt and learn from their experience.*" - Tom Dietterich
- These algorithms identify the dependencies between data elements using examples.
- **When we DO NOT need them?** When we know and understand all the functional dependencies in the system under investigation. It is not a common case.
- **What do we need for analysis?** Collection of examples, could be Monte Carlo data.

Tasks:

- Classification (discrete output states, signal/background)
- Regression (continuous output spectrum)

Learning methods:

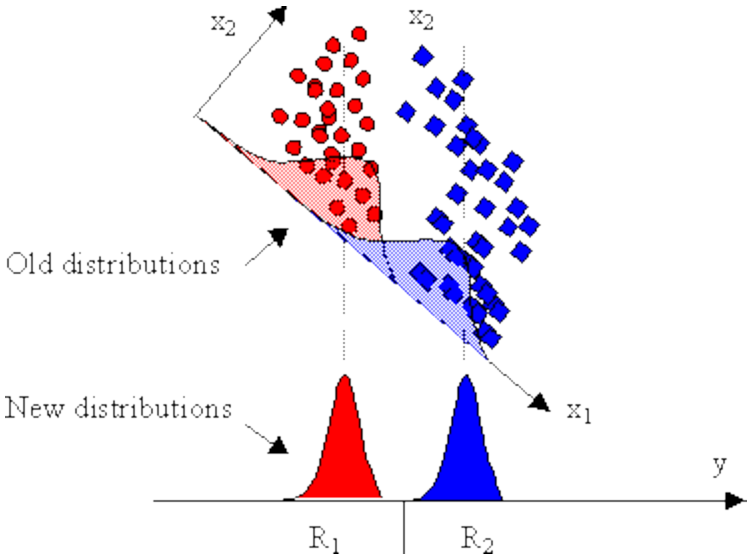
- Supervised learning (proper answers are known for the training sample).
- Unsupervised learning – system reflects the statistical structure of data (Kohonen, Hopfield networks) – *topic for a separate seminar, but will be mentioned here.*

Multivariate analysis methods

- **Multivariate analysis** – makes use of the dependencies between the variables. Multidimensional probability distribution contain, in general, more information then single variable spectra.
- **Linear and non-linear analysis** – in most cases we deal with non-linear problems, however in some applications linear methods are very useful (fast and robust).
- **Popular algorithms:**
  - **Linear**
    - Fisher linear discriminants
    - Principal Component Analysis
    - Independent Component Analysis
  - **Non-linear**
    - Probability Density Estimator
    - Neural Network (feed-forward)
    - Support Vector Machine

Fisher Linear Discriminant

Projection on one dimension and signal/background separation



Corresponds to a linear decision boundary.

$$f(\vec{x}) = \vec{w} \cdot \vec{x}$$

$$S = \frac{\sigma^2_{between}}{\sigma^2_{within}} \qquad maximize\ separation$$

$$\sigma^2_{between} = (\vec{w} \cdot \vec{\mu}_{signal} - \vec{w} \cdot \vec{\mu}_{background})^2$$

$$\sigma^2_{within} = \sigma^2_{signal} + \sigma^2_{background}$$

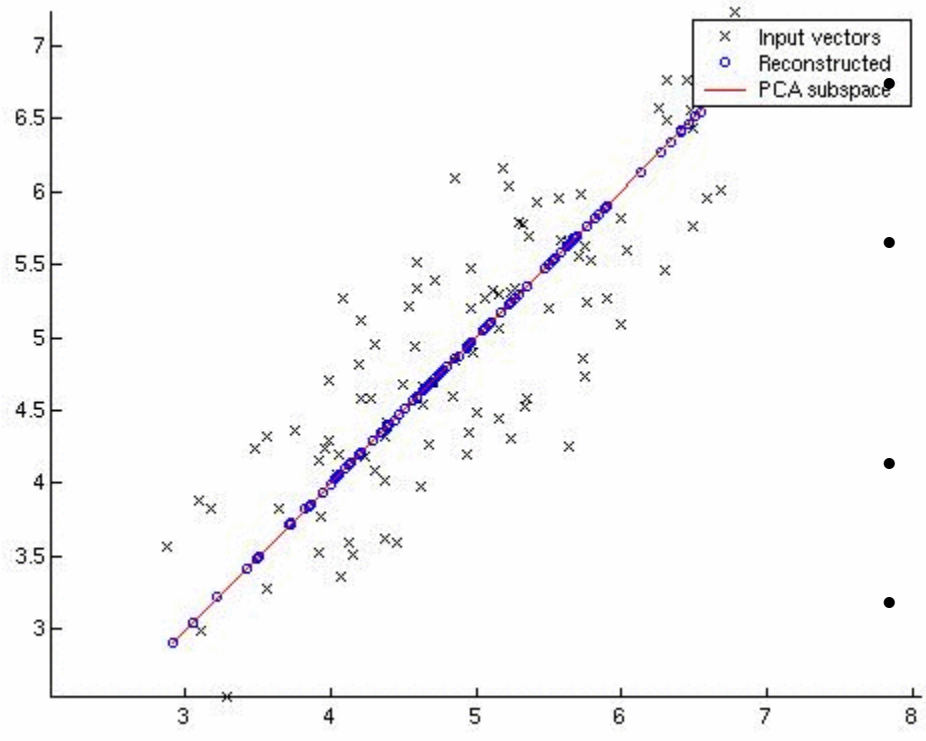
$$\sigma^2 = \vec{w}^T \cdot S \cdot \vec{w} \qquad S - variance$$

$$f(\vec{x}) = (S_{signal} + S_{background})^{-1} \cdot (\vec{\mu}_{signal} - \vec{\mu}_{background}) \cdot \vec{x}$$

Successfully used since 1936.
Non-linear separation needed: can be converted to the non-linear kernel Fisher discriminant using a *kernel trick (to be described later).*

Principal Component Analysis

- Reduces dimensionality of data, loosing as little information as possible.
- Finds orthogonal basis of the covariance matrix, eigenvectors with smallest eigenvalues are removed.



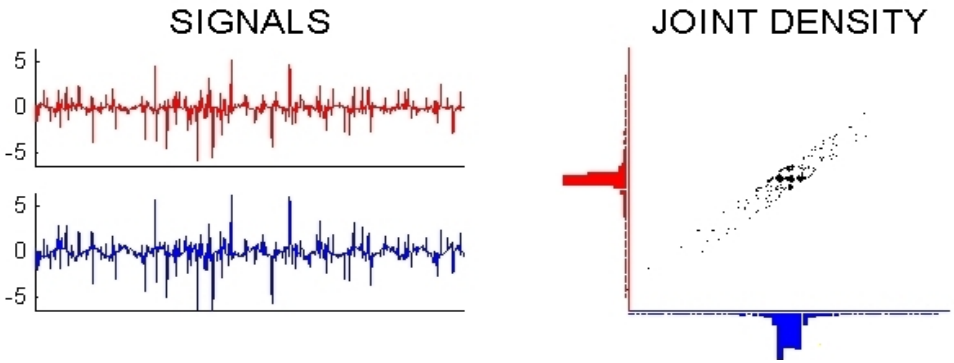
Procedure:

- Compute Cov(X)
- Compute its eigenvalues λ_i and eigenvectors v_i
- Eliminate u_i with smallest amount of variation

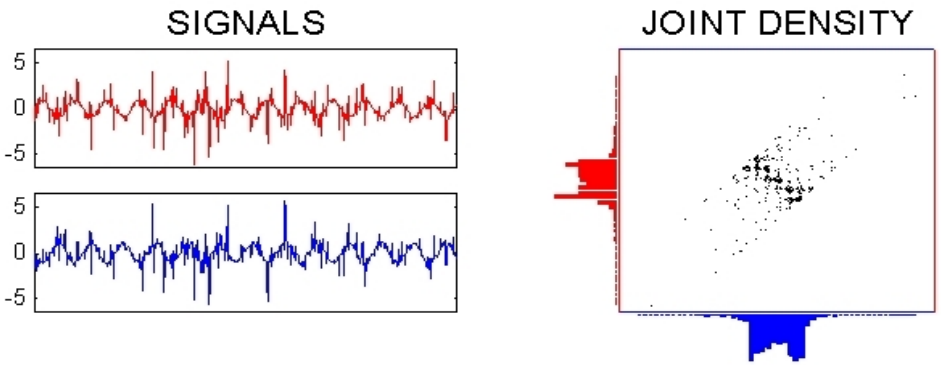
- Unsupervised technique, relies entirely on the input data.
- Can be converted into non-linear technique by a kernel trick or using neural networks (described later).

Independent Component Analysis

- Novel technique – Helsinki University of Technology
- Basic Idea
  - Assume **X** = $(x_1,..,x_N)^T$ is a linear sum **X = AS** of independent sources **S** = $(s_1,..,s_N)^T$. Both **A**, the *mixing* matrix, and **S** are *unknown.*
  - Find a ***de-mixing*** matrix **T** such that the components of **U = TX** are *statistically independent.*
- Applications:
  - filtering a single source of sound from other sources,
  - signal separation in telecommunication,
  - separation of brain activity from artifacts in magnetoencephalography (MEG) recordings,
  - in astrophysics to separate various signal sources,
  - decomposition of the Fermilab booster beam monitor data into independent source signals

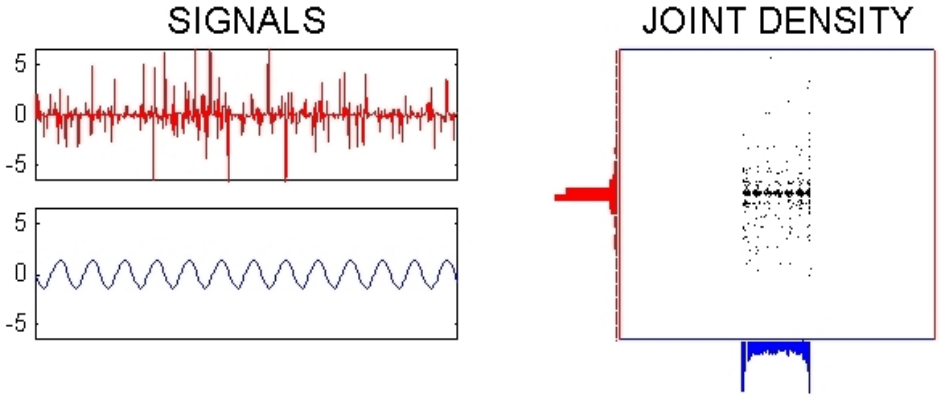


- Two measured signals to be separated into two independent sources.



- After whitening, i.e. linear transformation forcing signals to be uncorrelated.

Whitened signals and density



- ICA transformation– rotation, signals should be as non-Gaussian as possible (kurtosis is a measure of non-Gaussianity).

- *Kurtosis - $\mu_4/\sigma_4 - 3$, where μ_4 is the fourth moment about the mean and σ is the standard deviation.*

Separated signals after 5 steps of FastICA

Probability Density Estimation – a non-linear method

- **Purpose**
  - Signal/background discrimination (classification).
  - Parameter estimation (regression).
- **Basic Idea**
  - Parzen estimation (1960s) – approximation of the unknown probability. as **a sum of kernel functions** placed at the points x_n of the training sample:

$$p(x) = \frac{1}{N}\sum_n \frac{1}{h^d}\varphi\left(\frac{x - x_n}{h}\right) \qquad 1 \le n \le N$$

$$\varphi(x - x_n) \to \delta(x - x_n)$$ when the points density grows to infinity.

  - Discriminant:

$$D(x) = \frac{p(sig|x)}{p(sig|x) + p(bckg|x)}$$

  - Typical kernels: Gaussian, $1/x^n$, square etc.
  - Conceptually simple, no problems with local minima (in contrast to neural networks).
  - Each vector to be classified requires looping over all points from the training data set (CPU and memory demanding).

QUAERO package from D0 experiment

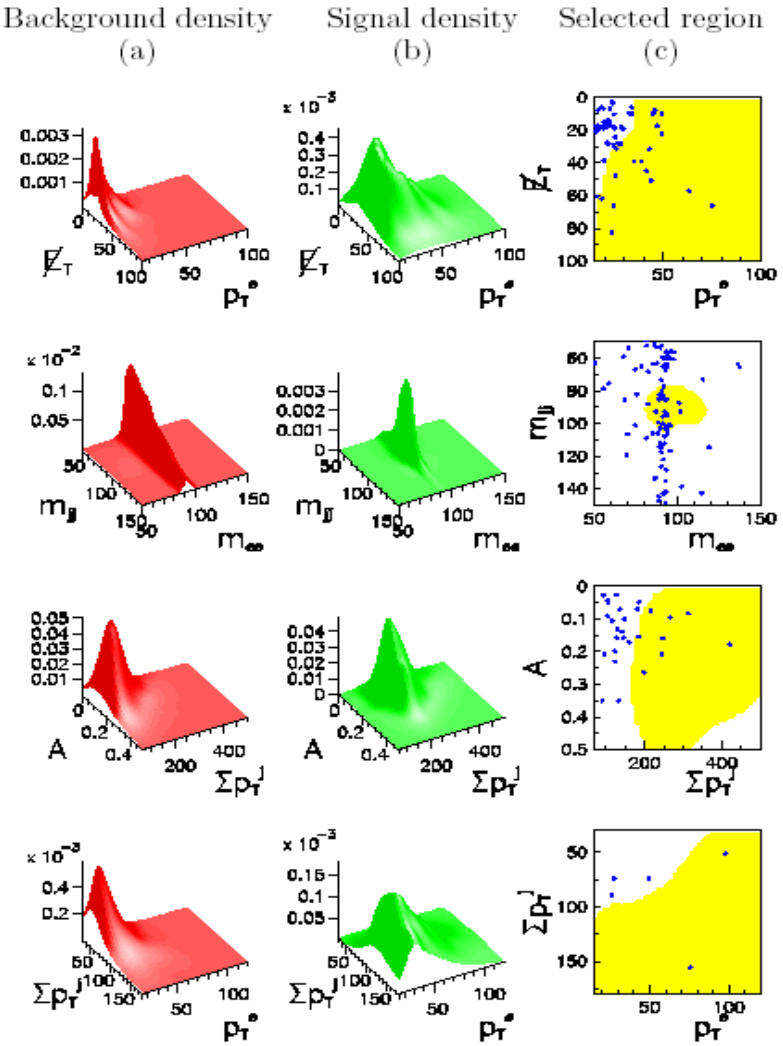


FIG. 1. The background density (a), signal density (b), and selected region (shaded) (c) determined by QUAERO for the standard model processes discussed in the text. From top to bottom the signals are: $WW \to e\mu\not{E}_T$, $ZZ \to ee\,2j$, $t\bar{t} \to e\not{E}_T\,4j$, and $t\bar{t} \to e\mu\not{E}_T\,2j$.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

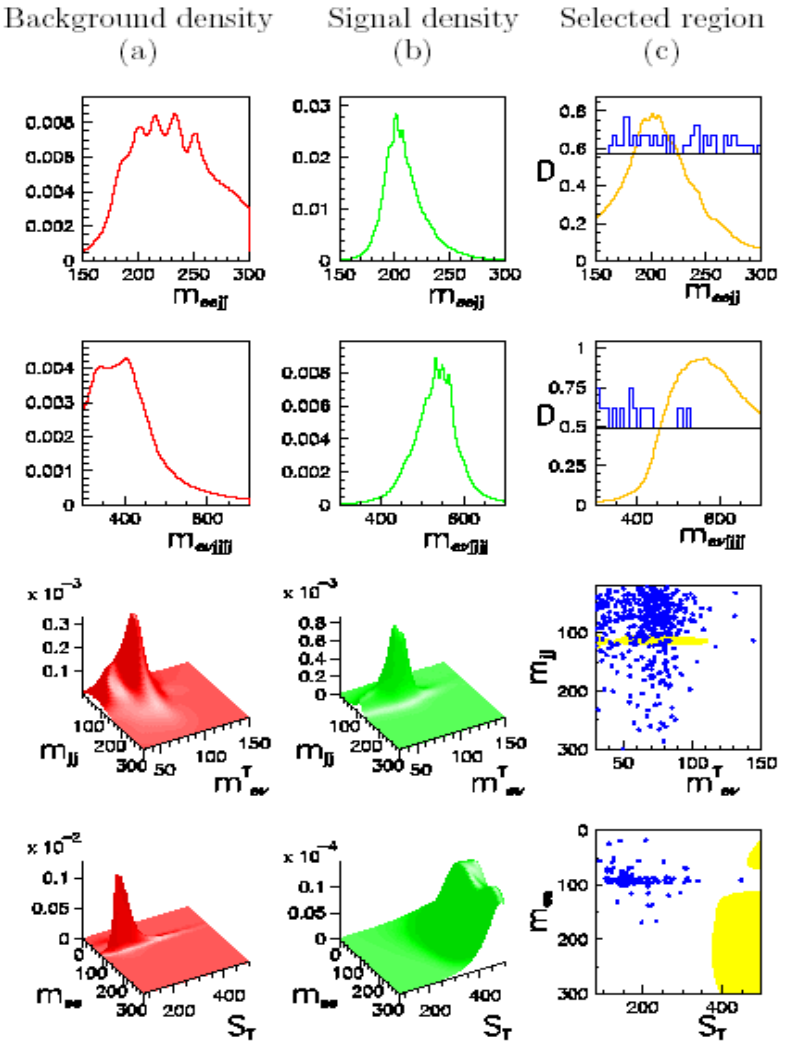


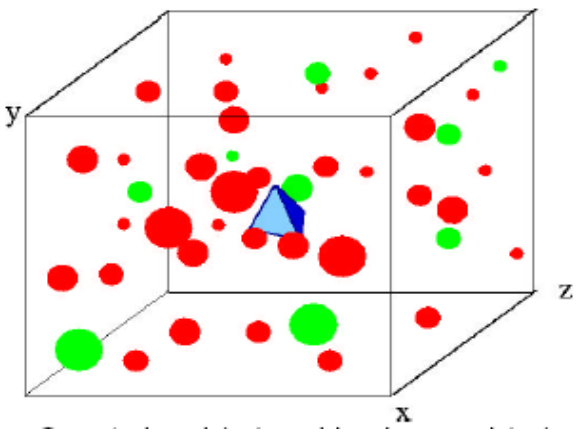
FIG. 2. QUAERO's analysis of signatures involving undiscovered particles. From top to bottom the hypothetical signals are: $h_{200} \to ZZ \to ee\,2j$, $Z'_{550} \to t\bar{t} \to e\not{E}_T\,4j$, $Wh_{115} \to e\not{E}_T\,2j$, and $LQ_{225}\overline{LQ}_{225} \to ee\,2j$. Plots (c) of the first two rows 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.

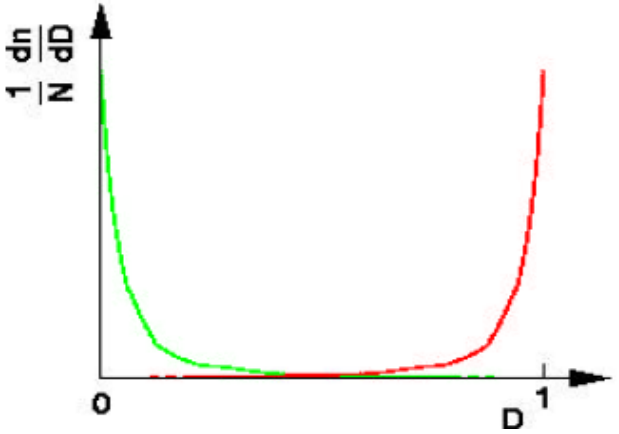
How to speed it up? PDE_RS method

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

- Count signal (n_s) and background (n_b) events in N-dim hypercube around the event classified – only few events from the training sample needed.
- Hypercube dimensions are free parameters to be tuned.
- Discriminator *D(x)* given by signal and background event densities:

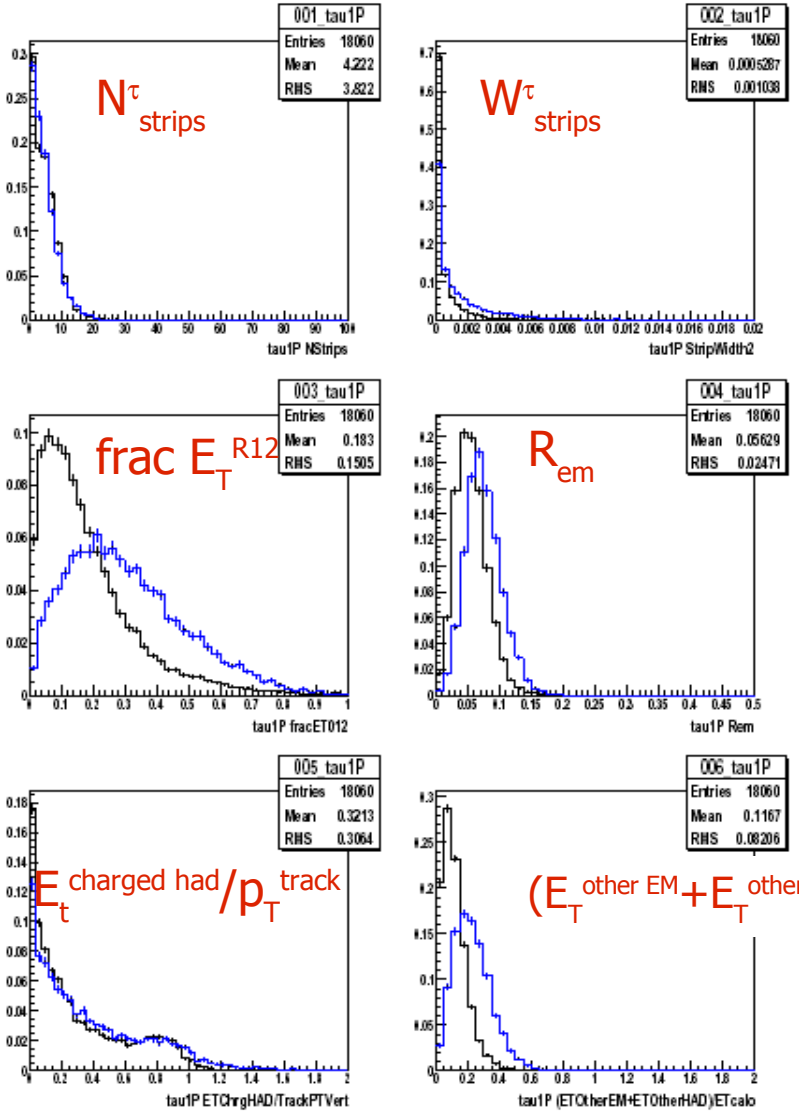
$$D(x) = \frac{n_S}{n_S + n_b}$$





- Straight forward error calculation possible.
- Event stored in the binary tree – easy to find neighbor events.

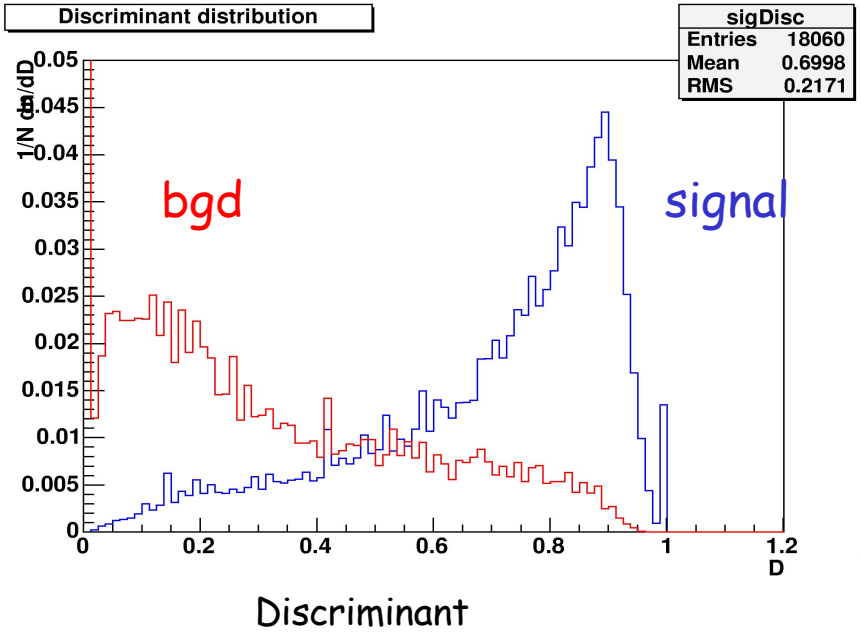
Application to τ identification in ATLAS



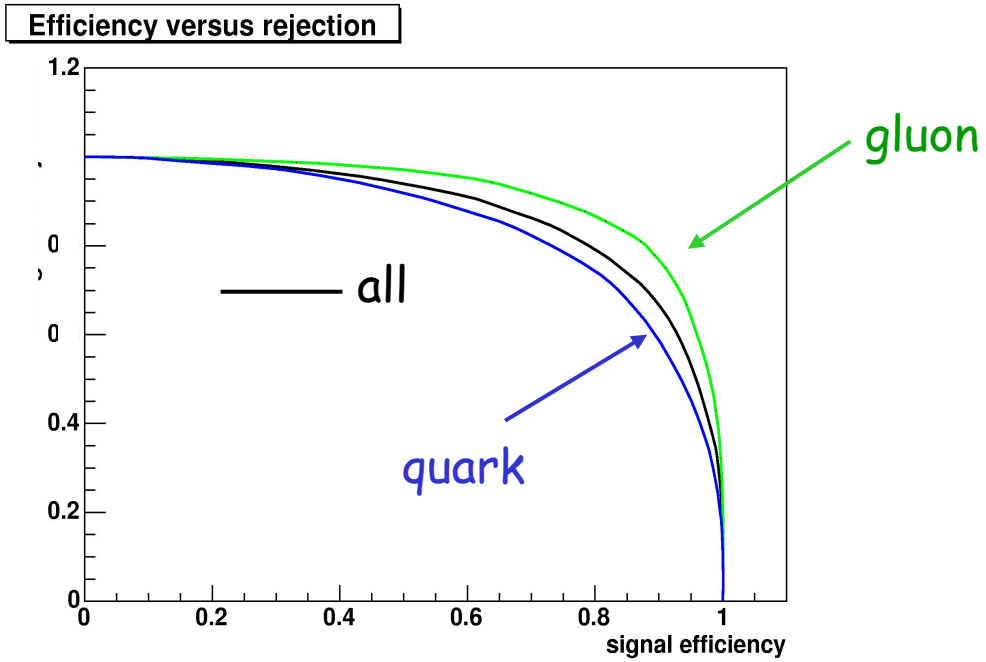
*analysis performed by E. Richter-Was, Ł. Janyst and T. Szymocha

- Six discriminating variables, none of them give separately really good discrimination.
- Variables not necessarily independent
- Hard to separate signal from background.

Application of PDE_RS



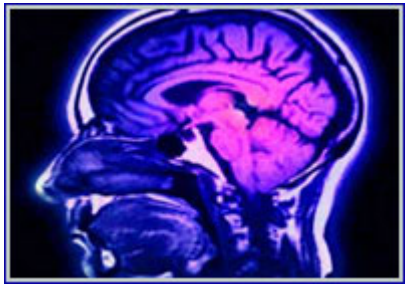
	Cut analysis sig.	Cut analysis bg	PDE-RS sig.	PDE-RS bg.
Full sample	58.9%	14.3%	58.9%	9.3%
Only gluon	58.9%	10.0%	58.9%	4.8%
Only quark	58.9%	21.5%	58.9%	11.8%



Approximately two times less background than for standard cut analysis.

Feed-Forward Neural Networks

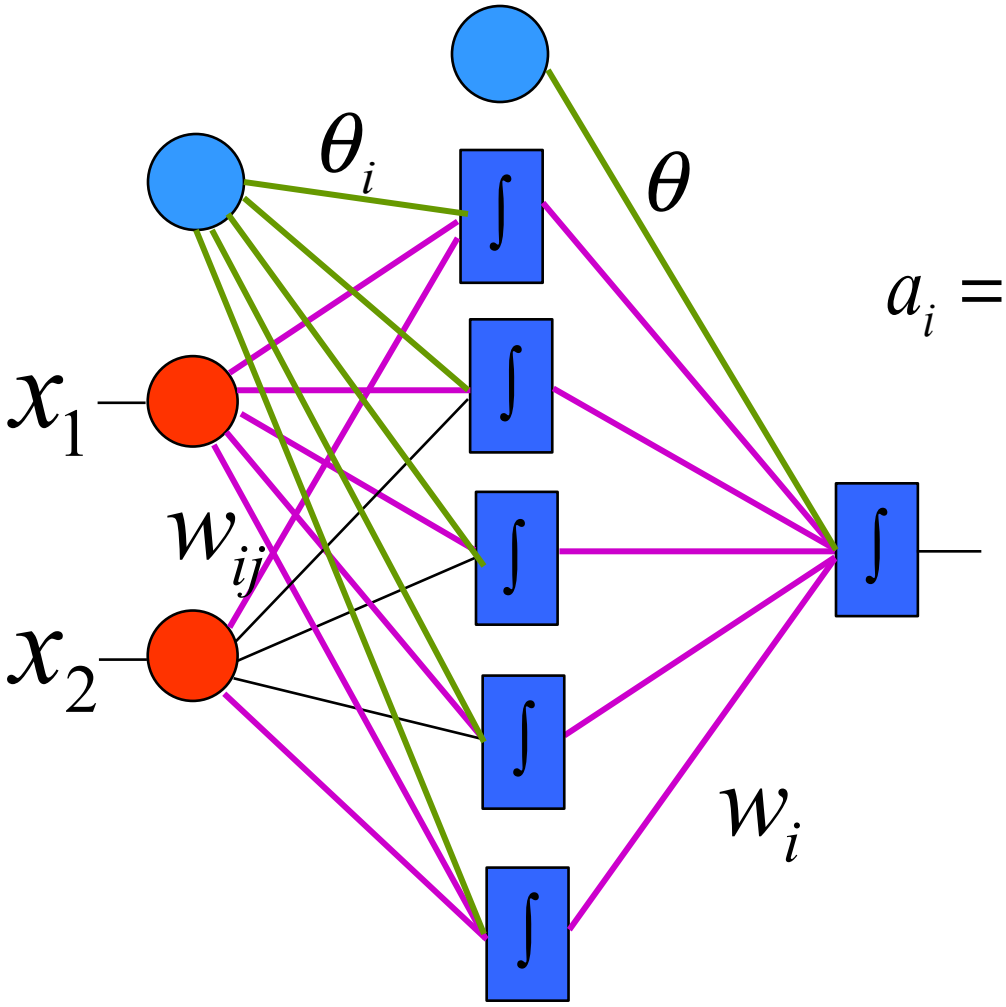
- **What can they do:**
  - Signal/background discrimination
  - Parameter estimation



  - Hardware implementations in trigger systems

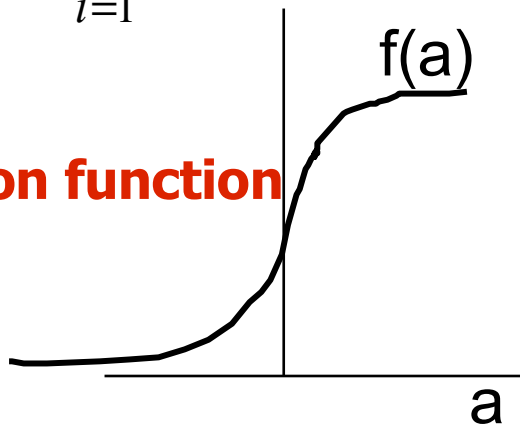
- **Characteristics**
  - Can model any function as a composition of many basic functions assigned to single neurons.
  - Can be trained on examples – network training is similar to fitting a function to data.

Feedforward Network



$$a_i = \sum_{j=1}^{2} w_{ij} x_j + \theta_i \quad \rightarrow f(a_i)$$

$$n(x,w) = f\left(\sum_{i=1}^{5} w_i f(a_i) + \theta\right)$$



Input nodes Hidden nodes Output node

Training a neural network

Minimize the *empirical risk function* with respect to the set of weights ω:

$$R(\omega) = \frac{1}{N}\sum_i [t_i - n(x_i, \omega)]^2$$

A network trained for classification (B-background, S- signal) returns a probability of a given event to be a signal *p(S|x)*:

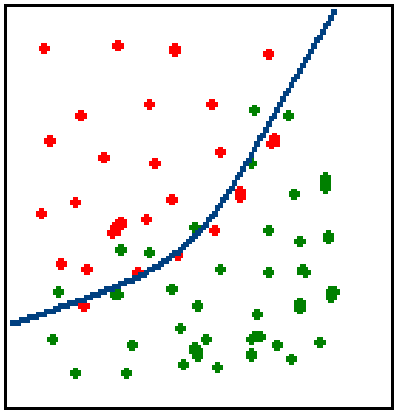
$$n(x, \omega) \to p(S|x) = \frac{p(x|S)\,p(S)}{\sum_{k=S,B} p(x|k)\,p(k)}$$

How to train a multilayer network? How to tune the weights of hidden layers? This problem stopped the NN development for 30 years.

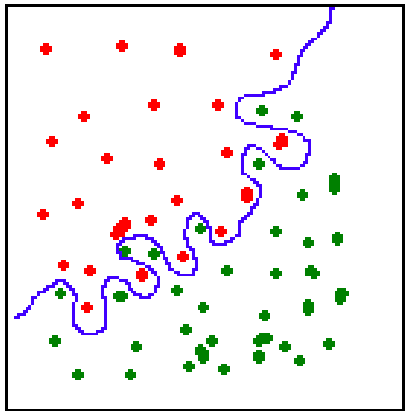
Solution - a **backpropagation** method. An error $\delta = t - n(x, \omega)$ is propagated back through the net using the present weights ("revolution" in neural networks in middle 80').

Overtraining

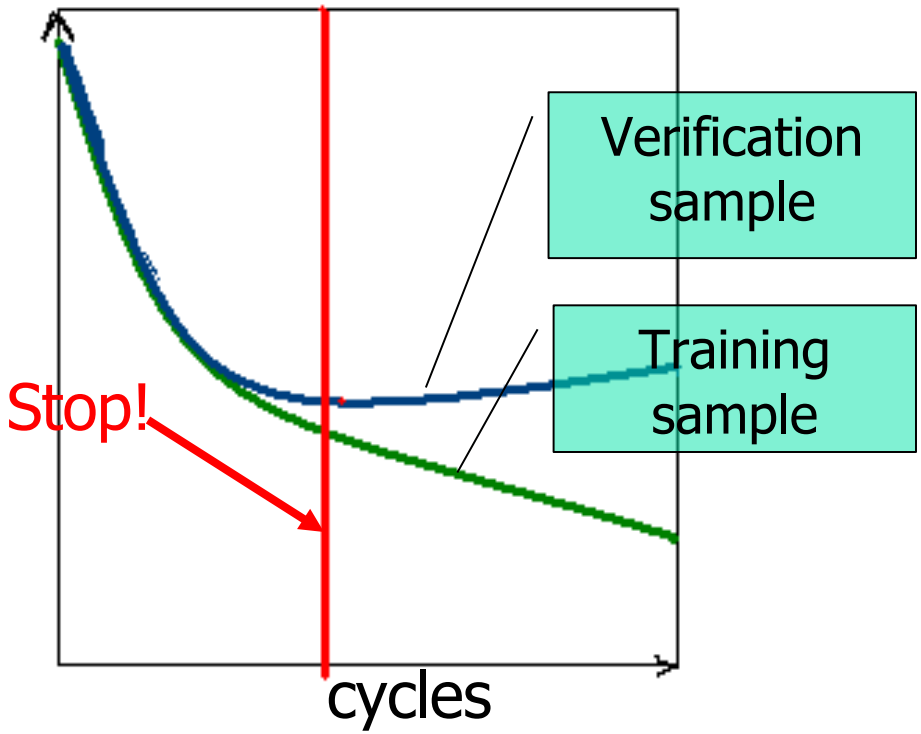
- Overtraining - network learns single events instead of general rules.
- Similar overtraining effect – too narrow kernels in the PDE method.



Proper

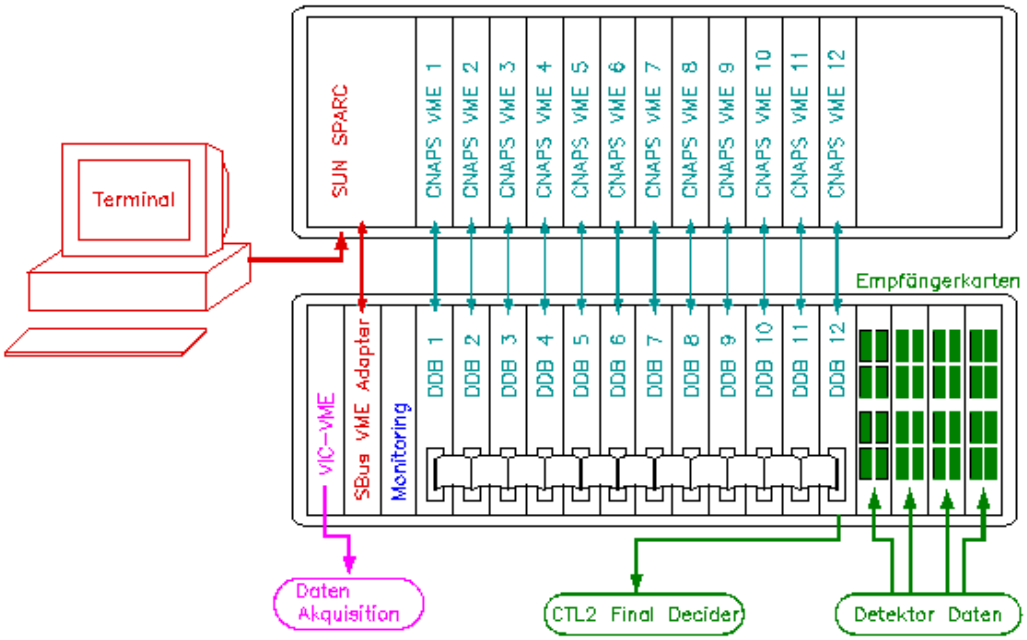


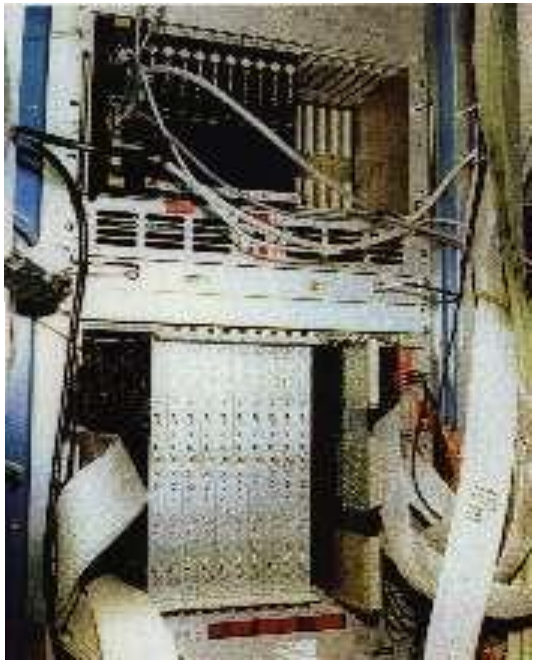
Overtrained



Neural Network Trigger at the H1 Experiment

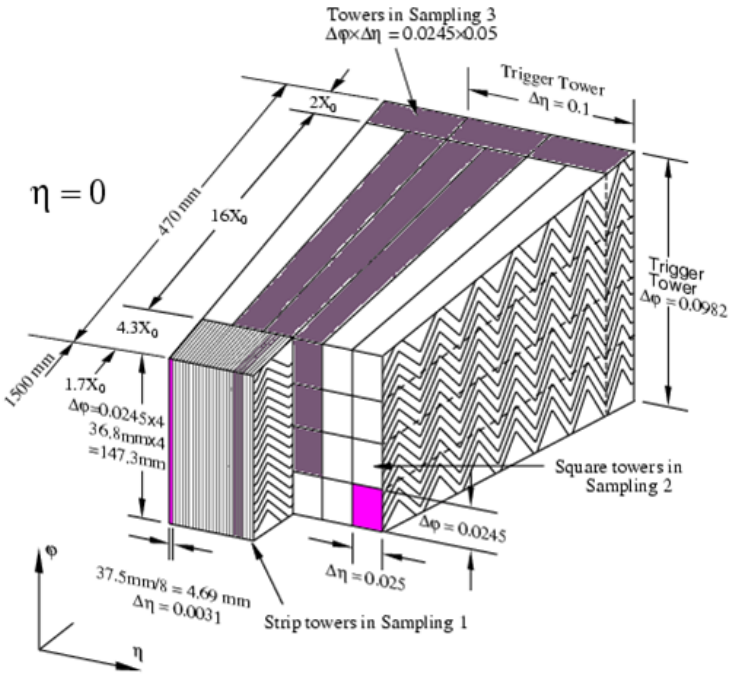
- Works since 1996.
- Implemented on several parallel computers (CNAPS from Adaptive Solutions).
- 20μs decision time.





Application: Identification of low energetic electrons in ATLAS experiment

- Each track extrapolated to the EM calorimeter
- For ech layer a cell with maximum energy deposit is found
- Eight e/π separation variables based on
  - cascade shapes - calorimeter
  - joined Inner Detector/Calo information
  - inner Detector information



|eta|<2.4, PT>2 GeV/c

Track quality cuts:

# of hits in silicon detectors NSi>8

# of hits in pixel detector NPix>1

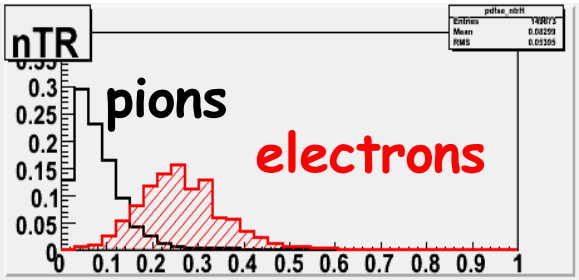
hits in B-layer >= 1

Transverse impact parameter A0<1mm

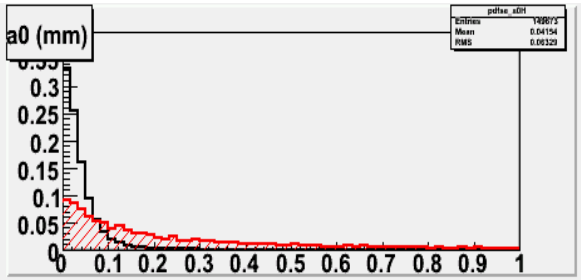
A. Kaczmarska (IFJ)
T. Bołd (IFJ)
J. Cochran (ISU)
F. Derue (LPNHE, Paris)
K. Facius (The Niels Bohr Institute, Copenhagen)
P. Schwemling (LPNHE, Paris)
E. Stanecka (IFJ)
M. Wolter (IFJ)

Discriminating variables

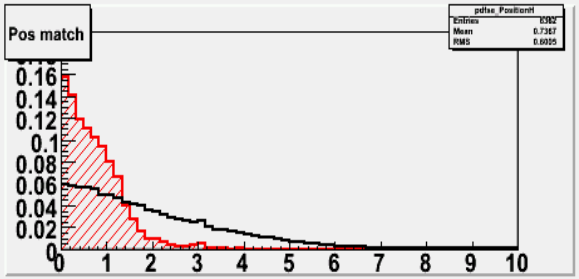
TR hits

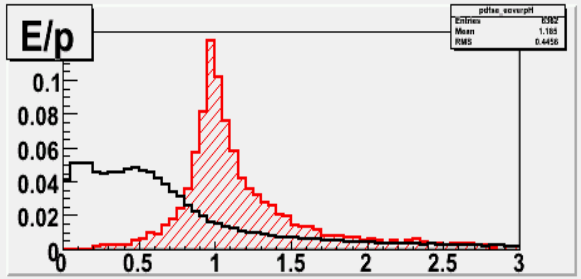


Transverse impact parameter



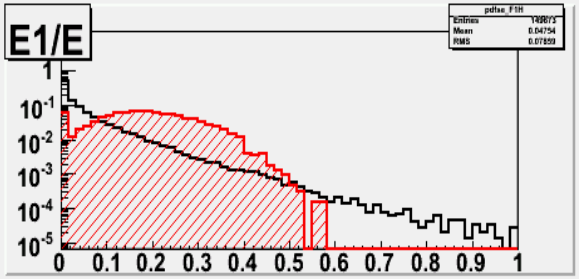
Difference cascade-track extrapolation

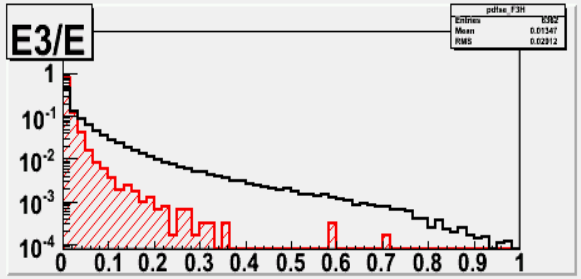




Et(calo)/pt

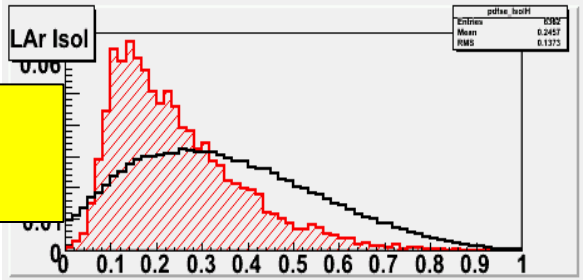
E1/E

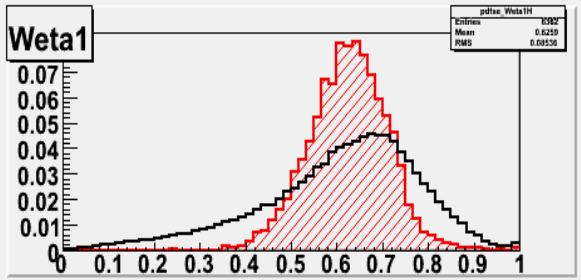




E3/E

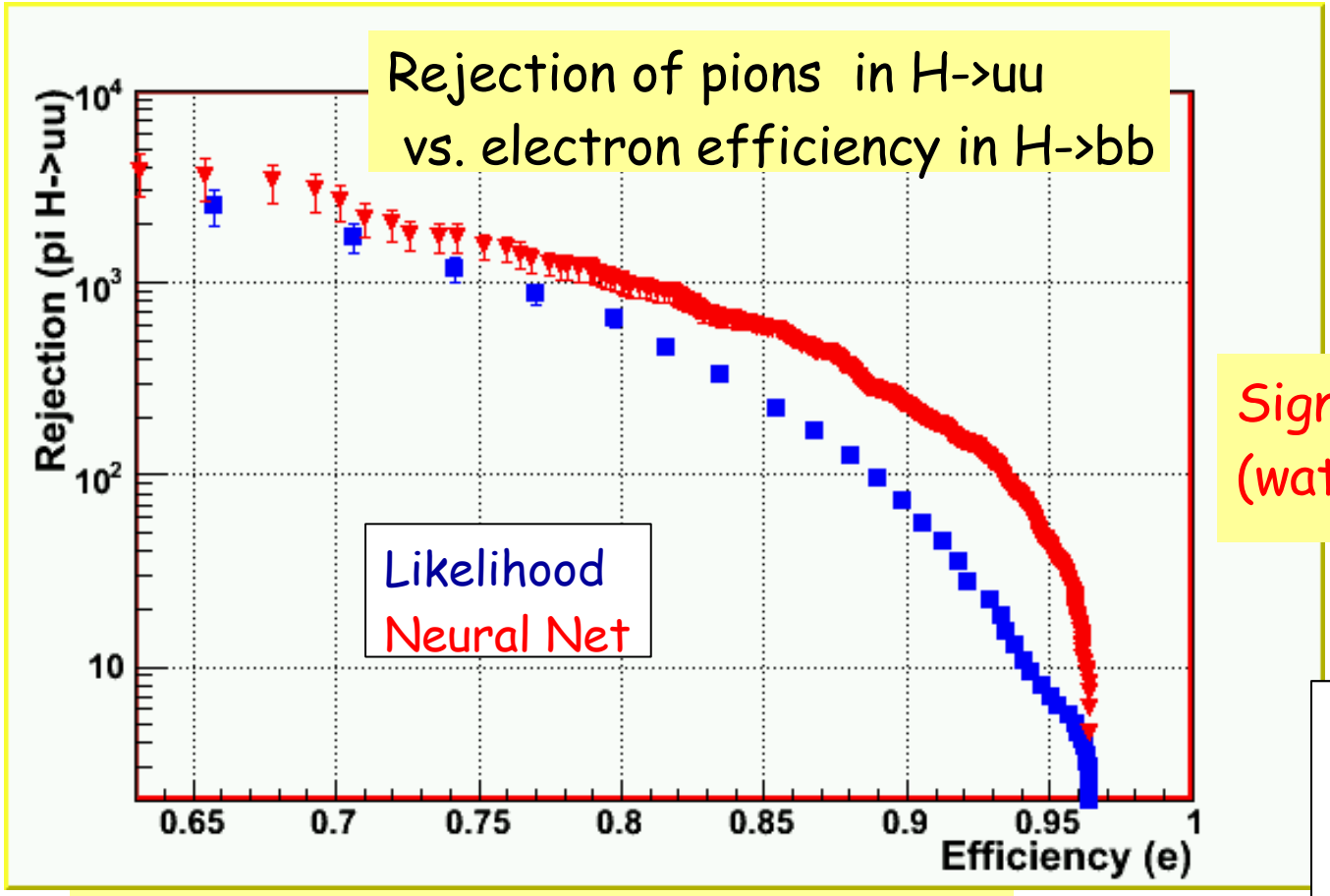
Cascade isolation ~E(3*7)/E(7*7)





Cascade width

Neural network vs. standard method



Significant improvement (watch the log scale)

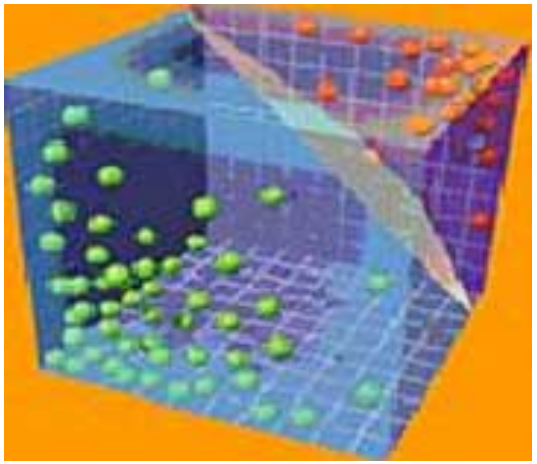
Standard method - **ratio of likelihood:**

XRL= log (Π PDF(e)/Π PDF(e))

No correlations taken into account!

Stuttgart Neural Network Simulator
Feed-forward network:
-input layer – 8 nodes
-2 hidden layers, 10 nodes in each

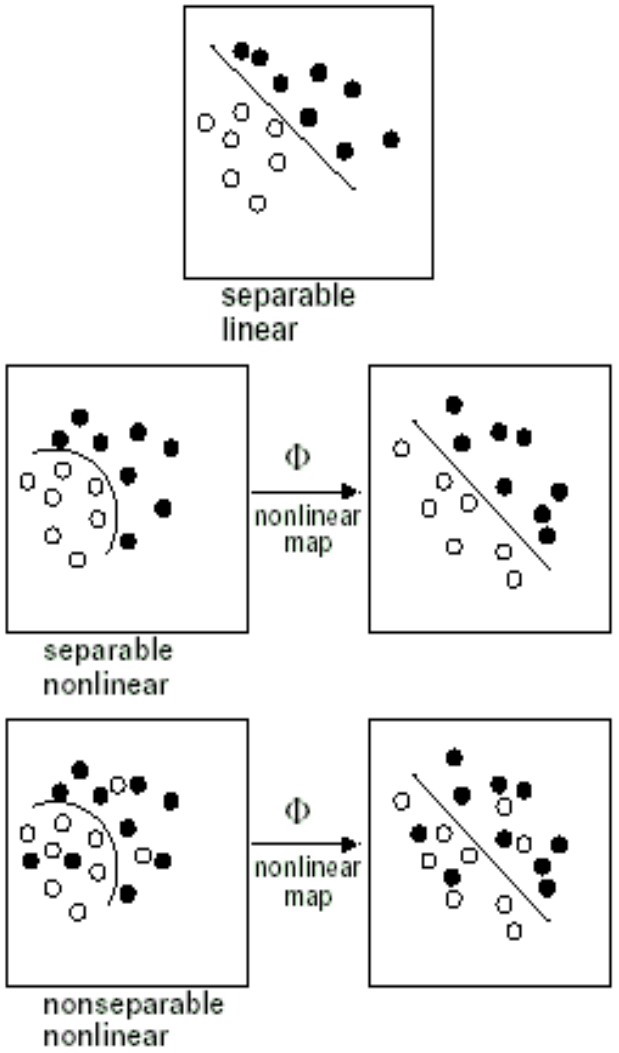
Support Vector Machines



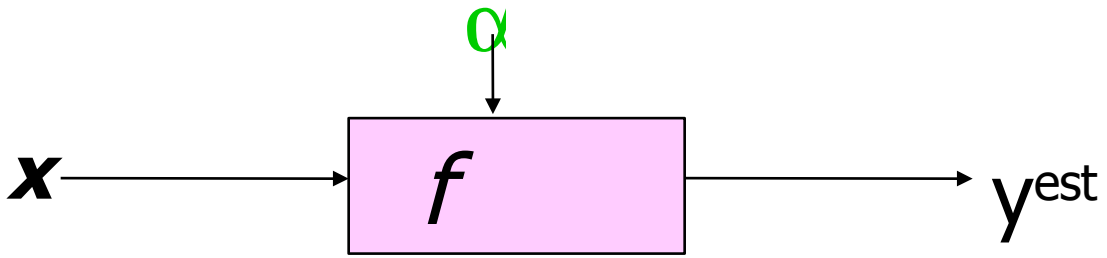
- **Relatively new algorithm**
- **Basic idea**:
  - build a separating hyperplane using the minimal number of vectors from the training sample **(Support Vectors).**
  - should be able to model any arbitrary function.
- *Functionally algorithm similar to Neural Network*

Support Vector Machine

- **Early sixties:** – “support vectors” method developed for pattern recognition using the separating hyperplane (Vapnik and Lerner 1963, Vapnik and Czervonenkis 1964).
- **Early 1990:** method extended to the non-linear separation (Boser 1992, Cortes and Vapnik 1995)
- **1995:** further development to perform regression (Vapnik 1995)



Linear classifier



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} - b)$

- means $y_i=+1$
- means $y_i=-1$

Support Vectors

-points defining the margin.

$$\forall_i \quad y_i(\vec{w}\cdot\vec{x}_i+b)-1\geqslant 0$$

$$\vec{w}\cdot\vec{x}+b=0 \quad straight\ line$$

$$margin=\frac{2}{|\vec{w}|}$$

$$minimize|\vec{w}|^2$$

Maximum margin linear classifier

- Intuitionally is the best.
- Not sensitive on errors in the hyperplane position.
- **Separation depends on support vectors only.**
- Works in practice!

Maximize margin = minimize $|w|^2$

If data are not separable

- Additional slack variable:

$\xi_i = 0 \qquad x_i \, corectly \, classified$

$\xi_i = distance \quad x_i \, incorrectly \, classified$

- And we get:

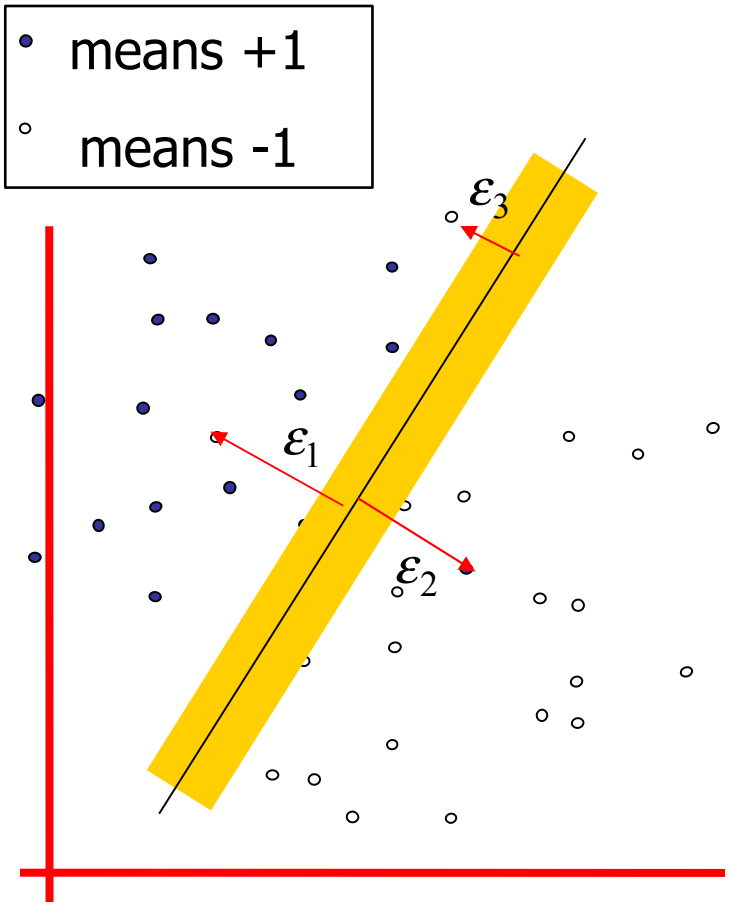
$\forall_i \quad y_i(\vec{w}\cdot\vec{x}_i + b) - 1 + \xi_i \geqslant 0$

- For the linear classifier:

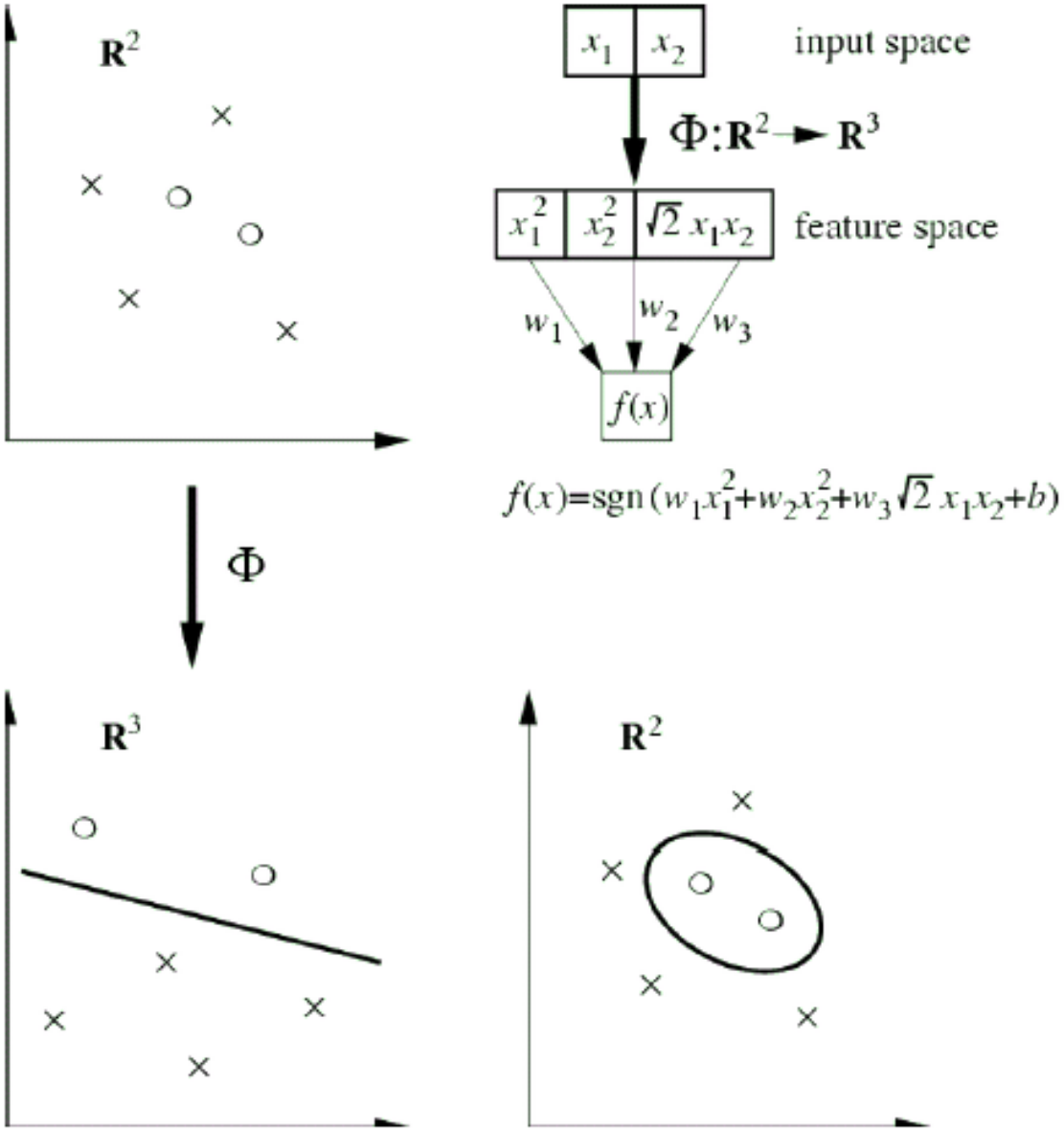
$\vec{w}\cdot\vec{x} + b = 0$

$$minimize : \frac{1}{2}|\vec{w}|^2 + C\sum_i \xi_i$$

- C is a free parameter - cost variable.



Problem – data separable but non-linear

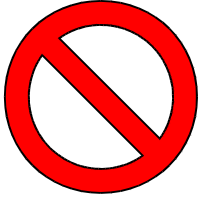


Transformation to higher dimensional feature space, where data ARE linearly separable.

In this example data separable by elliptic curve in R^2 are linearly separable in R^3.

We need a universal method of transformation to the higher dimensional space!

Some mathematics



- Lagrangian and Lagrange multipliers:

$$L(\vec{w}, b, \vec{\alpha}) = 1/2|w|^2 - \sum_i \alpha_i (y_i[\langle \vec{w}, \vec{x} \rangle + b] - 1)$$

$$\alpha_i \geq 0$$

$$y_i[\langle \vec{w}, \vec{x} \rangle + b] - 1 \geq 0 \quad \textit{the bonds}$$

- Lagrangian should be minimized in respect to **w** and b and maximized in respect to α_i.

$$y_i[\langle \vec{w}, \vec{x} \rangle + b] - 1 > 0 \;\rightarrow\; \alpha_i = 0 \quad (\textit{not contributing})$$

$$y_i[\langle \vec{w}, \vec{x} \rangle + b] - 1 = 0 \qquad \textit{support vectors}$$

- in the extremum:

$$\frac{\partial L}{\partial b} = 0 \;,\; \frac{\partial L}{\partial \vec{w}} = 0$$

$$\rightarrow \sum_i \alpha_i y_i = 0 \quad \vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

- we substitute into L and we have maximized W in the space of α_i

$$W(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \vec{x}_i, \vec{x}_j \rangle$$

$$\alpha_i \geq 0 \;,\; \sum_i y_i \alpha_i = 0$$

$$f(\vec{x}) = sgn(\langle \vec{w}, \vec{x} \rangle + b) = sgn\left(\sum_i \alpha_i y_i \langle \vec{x}_i, \vec{x} \rangle + b \right)$$

In this space W is a function of products $x_i * x_j$

Kernel trick

- Margin **w** maximization using Laplace multipliers α_i:

$$W(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \vec{x}_i, \vec{x}_j \rangle$$

- Margin depends on **dot products** $x_i * x_j$ only.
- Function Φ transforms the input space to the higher dimensional feature space, where data are linearly separable:

$$\Phi(x) : input\ \Re^n \to \Re^N \quad (N \geq n)$$

  - We can replace:

$$\langle x_i, x_j \rangle \to \langle \Phi(x_i), \Phi(x_j) \rangle = K(x_i, x_j)$$

- The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.

$$W(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$$

- **No need to know function Φ(x), enough to know kernel $K(x_i, x_j)$.**

Commonly used kernels

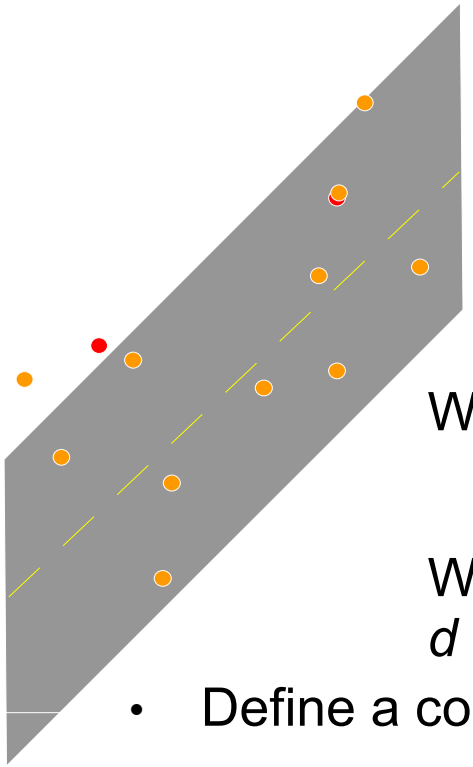
- Polynomial
$$K(\mathbf{x}_i, \mathbf{x}_j) = \left(\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c\right)^d$$
- Sigmoid
$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \theta\right)$$
- Gaussian
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

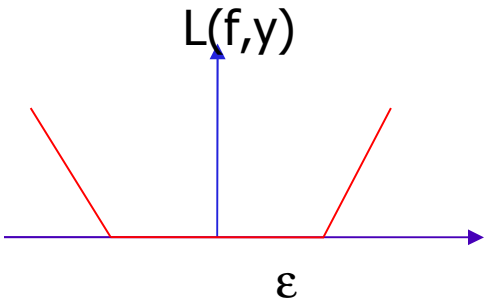
Must be symmetric:

$$K(x_i, x_j) = K(x_j, x_i)$$

If the kernel used is a radial base function (Gaussian) the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results.

Regression – "ε insensitive loss"





We have input data:

$X = \{(\underline{x}_1, d_1), \ldots, (\underline{x}_N, d_N)\}$

We want to find f(x), which has small deviation from *d* and which is maximally smooth.

- Define a cost function:

$$|y - f(\mathbf{x})|_\varepsilon := \max\{0, |y - f(\mathbf{x})| - \varepsilon\}$$

- Minimize:

$$\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\varepsilon$$

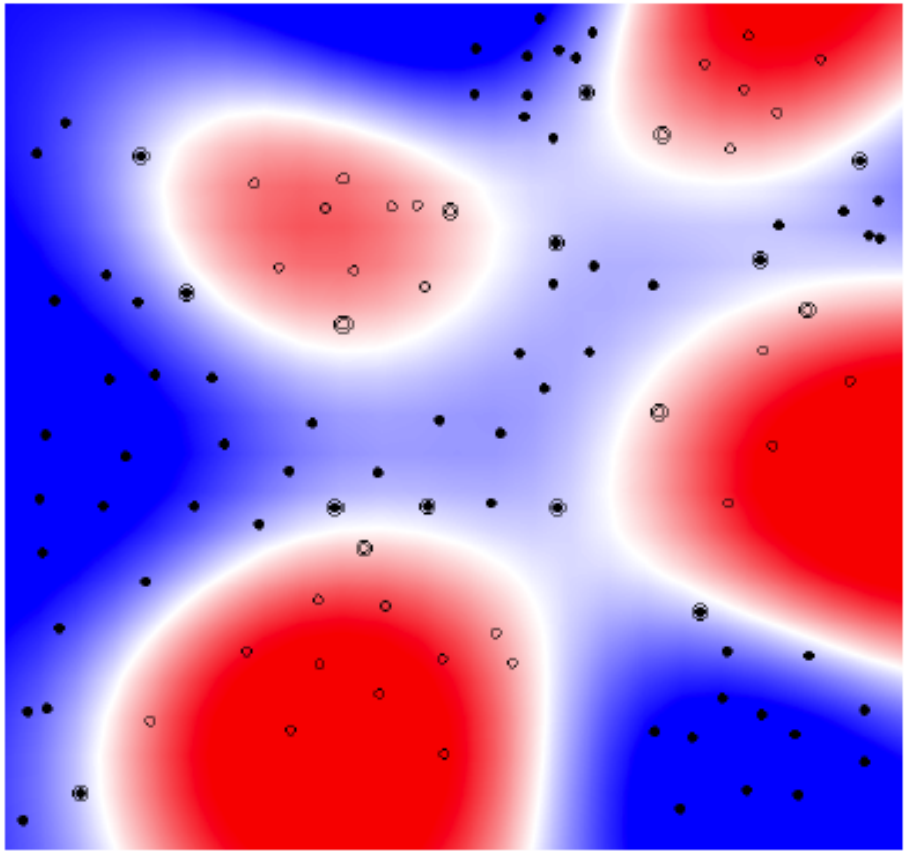
- And repeat the kernel trick

Non-linear Kernel example

Gaussian kernel

Kernel: $K(\vec{x}_i,\vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2/\sigma^2)$

plot by Bell SVM applet



SVM and feed-forward neural network
A comparison

NN – complexity controlled by a number of nodes.

SVM – complexity doesn't depend on the dimensionality.

NN – can fall into local minima.

SVM – minimization is a quadratic programming problem, always finds minimum.

SVM – discriminating hyperplane is constructed in a high dimensionality space using a kernel function.

Jianfeng Feng, Sussex University

SVM strength

- Statistically well motivated => Can get bounds on the error, can use the structural risk minimization (theory which characterizes generalization abilities of learning machines).

- Finding the weights is a quadratic programming problem - guaranteed to find a minimum of the error surface. Thus the algorithm is efficient and SVM generates near optimal classification and is quite insensitive to overtraining.

- Obtain good generalization performance due to high dimension of the feature space.

Jianfeng Feng, Sussex University

SVM weakness

- Slow training (compared to neural network) due to computationally intensive solution to QP problem especially for large amounts of training data => need special algorithms.
- Slow classification for the trained SVM.
- Generates complex solutions (normally > 60% of training points are used as support vectors), especially for large amounts of training data.

E.g. from Haykin: increase in performance of 1.5% over MLP. However, MLP used 2 hidden nodes, SVM used 285

- Difficult to incorporate prior knowledge.

Jianfeng Feng, Sussex University

Applications in physics

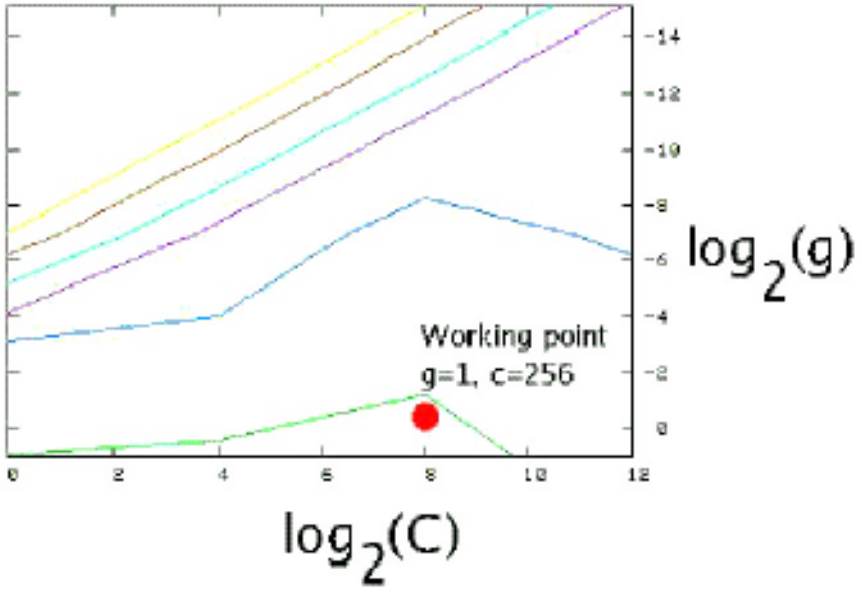
- While neural networks are quite commonly used, SVM are rarely applied.
  - LEP compared NN and SVM

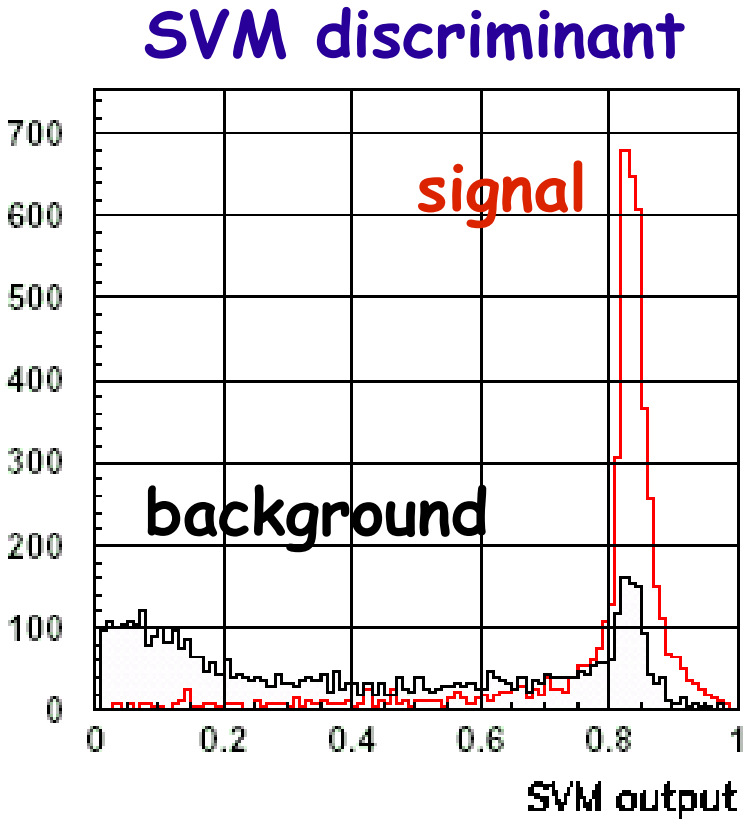
    *"**Classifying LEP Data with Support Vector Algorithms**"*
    P. Vannerema K.-R. Muller B. Scholkopf A. Smola S. Soldner-Rembold
  - There were some tries in D0 and CDF (Tufts Group – top quark identification)
- SVM works well in other areas (for example handwriting recognition).

- **One should look carefully, the method might be worth trying!**

Application of SVM to tau identification

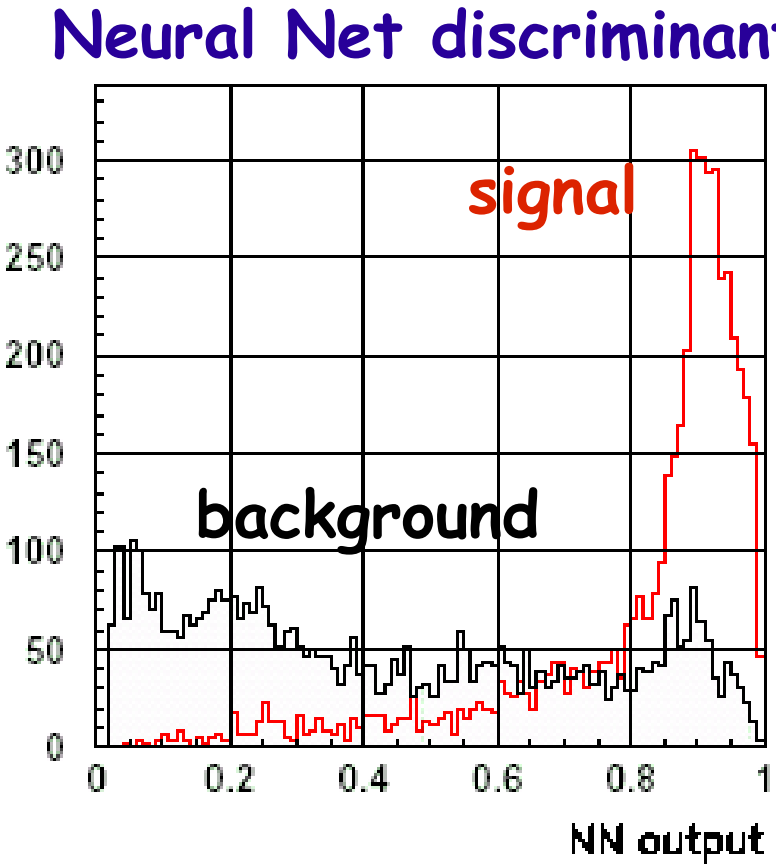
Grid search to find SVM working point
– optimal C and width of Gaussian kernel





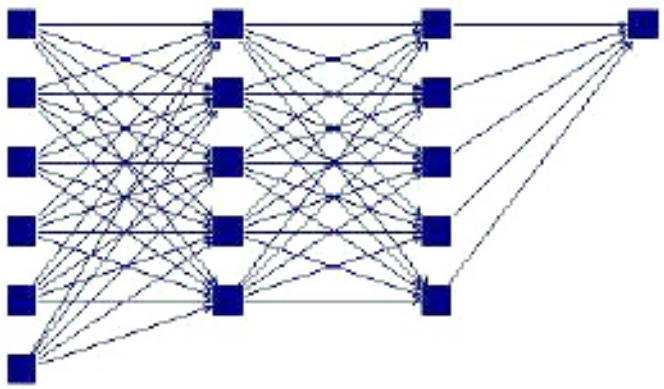
- SVM with Gaussian kernel.
- Grid search in C-g space to optimize parameters.
- Discriminant – probability of an event to be a signal - *p(signal|x).*

Neural Network in ATLAS tau identification

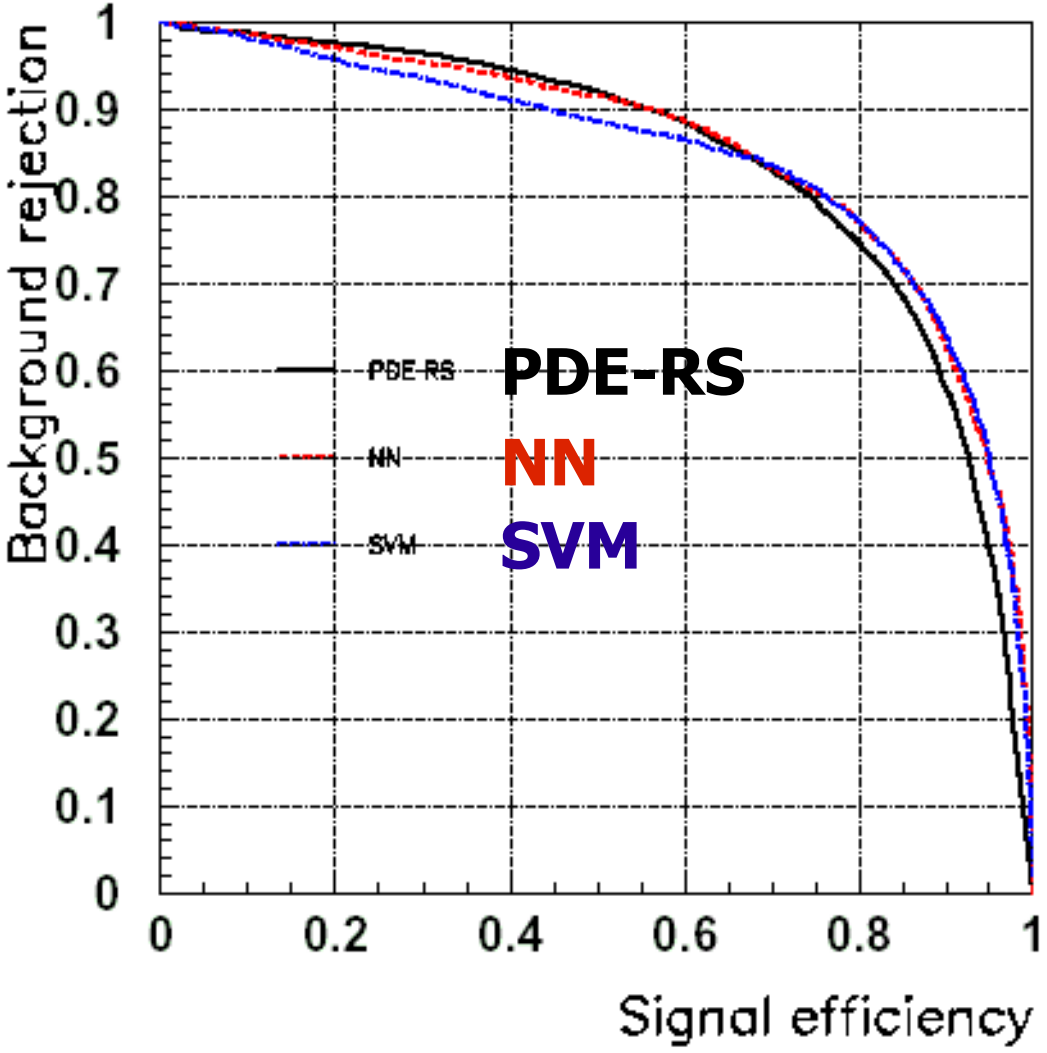


- *T. Szymocha, M. Wolter com-phys-2006-019*

- feed-forward network
- same discriminating as used by PDE_RS method.
- NN: 6 inputs, 2 hidden layers



Comparison of three methods in tau identification.

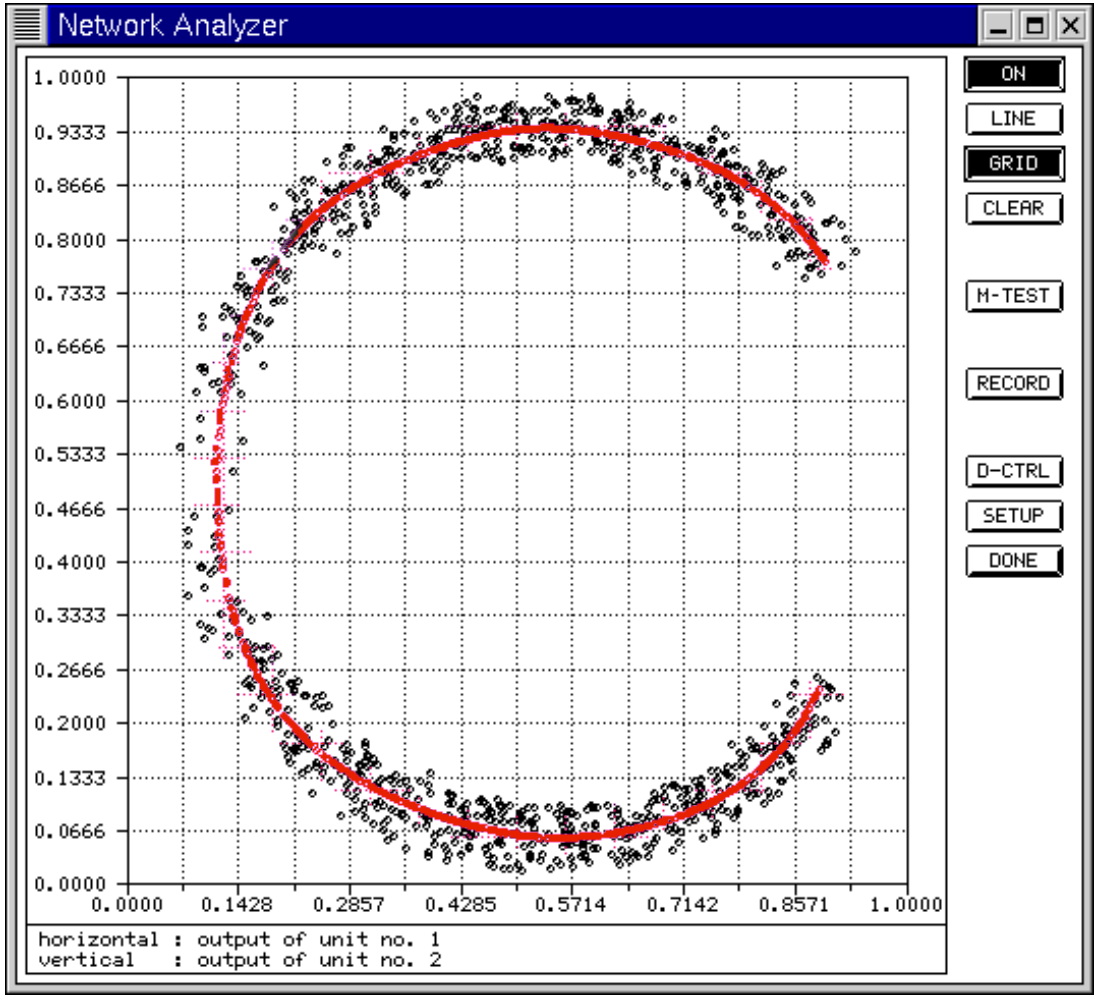


Method	Efficiency	Rejection
	90 %	58% ± 1%
PDE-RS	80 %	75% ± 1%
(6 variables)	70 %	83% ± 1%
	90 %	64% ± 1%
NN	80 %	77% ± 1%
(6 variables)	70 %	84% ± 1%
	90 %	64% ± 1%
SVM	80 %	77% ± 1%
(6 variables)	70 %	83% ± 1%

- Similar results, probably classification efficiency close to the statistical limit.
- Performance of methods may change by different tuning.

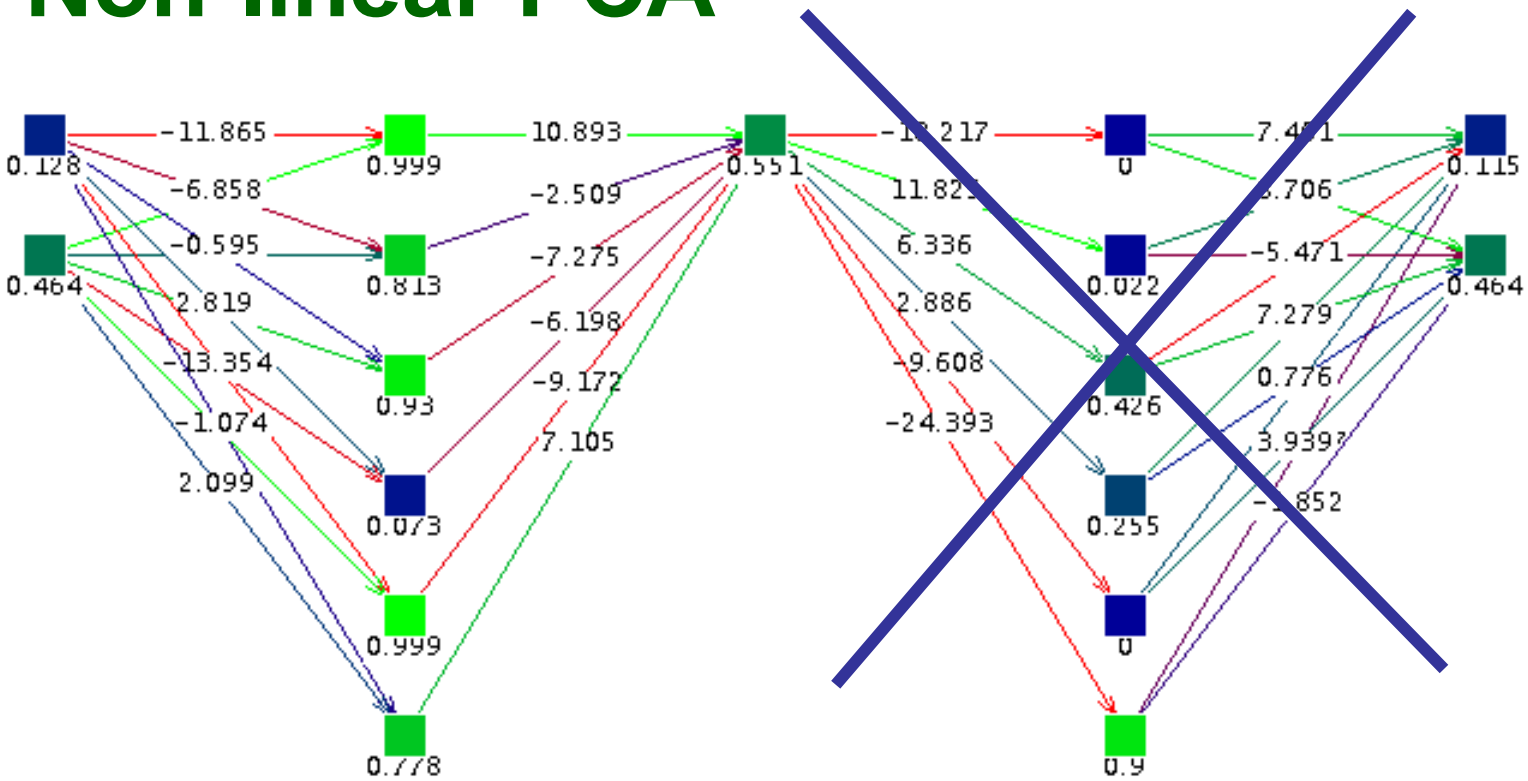
Unsupervised NN Methods

Non-linear PCA using neural network



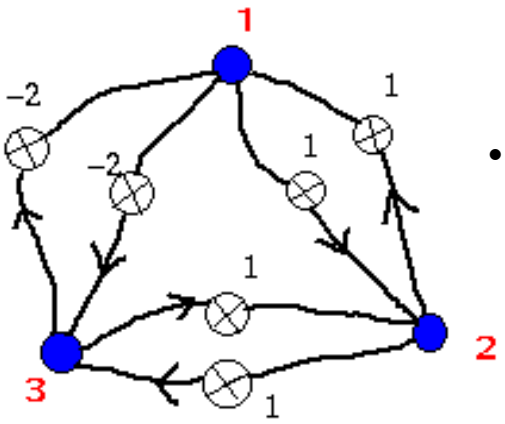
- Data can be reduced to one dimension.
- Non-linear transformation is needed.

Non-linear PCA

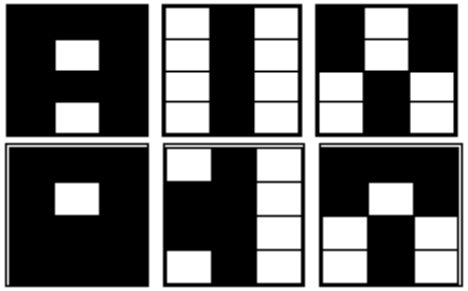


The network is trained by presenting the same vectors to the input and the output. Than the network is cut into two parts.

Hopfield recurrent network

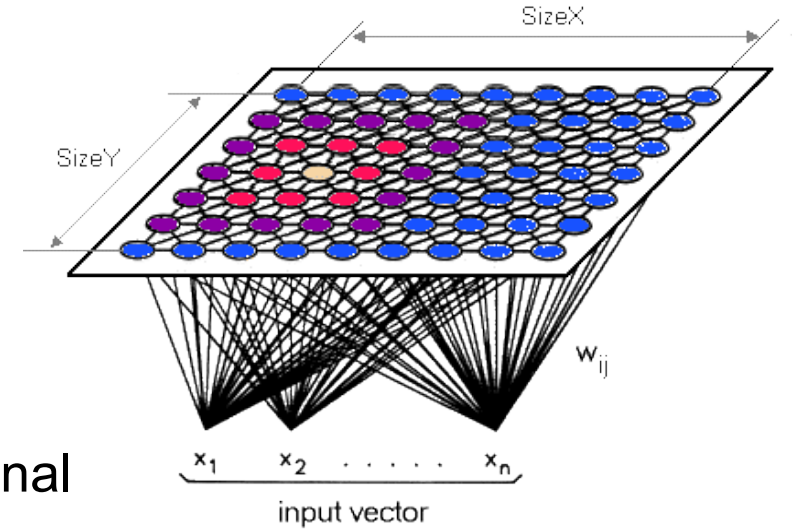


- **Recurrent network** – binary self coupled system. Output signals are at the same time input signals for the next training cycle $x_i(k)=y_i(k-1)$.
- The network works as an autoassociative memory:
  - memorizes the presented vectors
  - after presenting the unknown vector return the memorized vector closest to it.
- Example – a network trained on 3 letters:
- recognizes patterns:



Used for pattern used for pattern recognition in tracking algorithms assigning track segments to the tracks – ALEPH, HERA, DELPHI. Mostly used for TPC tracking (clean 3D hits).

Kohonen network - Self Organizing Map (SOM)



- Unsupervised learning
- Transforms vectors from an n-dimensional space into (typically) 2D map in such a way, that similar input vectors are close to each other on the map.
- Training:
  - Random initial neuron weights
  - Input vectors *x* are presented, an output neuron having a vector of weights *w* closest to *x* is a winner.
  - Vector *w* (and similar vectors) are corrected to be closer to *x.*
  - The procedure is repeated until the system is stable.
- Application: automated classification, groups similar vectors into clusters.

Comments on classification

- Every classification task tries to solve the *same* fundamental problem:
  - After adequately pre-processing the data
  - …find a ***good***, and ***practical***, approximation to the *Bayes decision rule*: Given ***X***, if ***P(S|X) / P(B|X)>cut*** , choose hypothesis ***S*** otherwise choose ***B***.
- If we knew the densities ***p(X|S)*** and ***p(X|B)*** and the priors ***p(S)*** and ***p(B)*** we could compute the *Bayes Discriminant Function* (BDF):

  D(X) = P(S|X)/P(B|X)
- All presented methods are simply different algorithms to approximate the Bayes discriminant function ***D(X)***.

- **It follows that if a method is already close to the Bayes limit, then no other method, however sophisticated, can be expected to yield dramatic improvements.**

Harrison B. Prosper

Summary

- Multivariate analysis is useful, if it is important to extract as much information from the data as possible.
- For classification problems, the common methods provide different approximations to the Bayes discriminant.
- There is considerably empirical evidence that, as yet, no uniformly most powerful method exists. Therefore, be wary of claims to the contrary!

Appendix

Literature and links to SVMs

- http://www.autonlab.org/tutorials/ - wykłady Andrew Moore'a.
- http://www.cs.colorado.edu/~grudic/teaching/CSCI4202_2004/ - wykłady Greg Grudic
- http://www.informatics.sussex.ac.uk/users/jianfeng/ -strona Janfeng Feng
- http://www.kernel-machines.org/papers/Burges98.ps.gz - “A Tutorial on SVM ...”, C. Burges (b. dobry)
- http://www.kernel-machines.org/ -bardzo intersująca strona
- http://www.csie.ntu.edu.tw/~cjlin/libsvm/ - software
- http://www.cs.cornell.edu/People/tj/svm_light/ - software

AN INTRODUCTION TO SUPPORT VECTOR MACHINES (and other kernel-based learning methods)**.**

N. Cristianini and J. Shawe-Taylor, Cambridge University Press. 2000. ISBN: 0 521 78019 5

Literature – Neural Networks

Christopher M. Bishop „Neural Networks for Pattern Recognition”

Andreas Zell „Simulation neuronaler Netze”

Ryszard Tadeusiewicz „Sieci neuronowe”

S. Osowski, “Sieci Neuronowe w ujęciu algorytmicznym”

Tools:

MLP in ROOT and in PAW

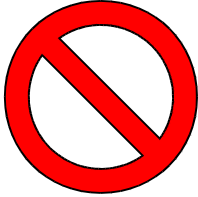
SNNS:

http://www-ra.informatik.uni-tuebingen.de/SNNS/

PDP++

http://www.cnbc.cmu.edu/Resources/PDP++//PDP++.htn

Tau identification variables



- Number of strips, N^{τ}_{strips}, with energy deposition above a threshold in strip layer of the electromagnetic calorimeter
- The width of the energy deposition in strips, W^{τ}_{strips}, calculated as the variance of the η coordinate, weighted by the transverse energy deposition in a given strip
- Fraction of the transverse energy deposited in the radius $0.1 < \Delta R < 0.2$ with respect to the total energy in the cone $\Delta R = 0.2$.
- Electromagnetic radius, R^{τ}_{em}, weighted by the transverse energy deposition for given cell

In addition, as an identification variable we use also:

- Ratio of the track transverse momenta and energy deposited in the hadronic calorimeter in the vicinity of the track, $\frac{E_T^{chrgHAD}}{p_T^{track}}$, and for 3 prong data we take sum of p_T of all three tracks.
- Ratio of energy deposited in a radius $0.2 < \Delta R < 0.4$, $\frac{E_T^{otherEM} + E_T^{otherHAD}}{E_T^{cand}}$, which represents isolation criteria.

Demonstration applets

- SVM in action:

  http://wiesiek.ifj.edu.pl/SVM/java/test_applet.html

- Neural network applets:

  http://wiesiek.ifj.edu.pl/talks/NeuralNetUJ/applets/tutorial/