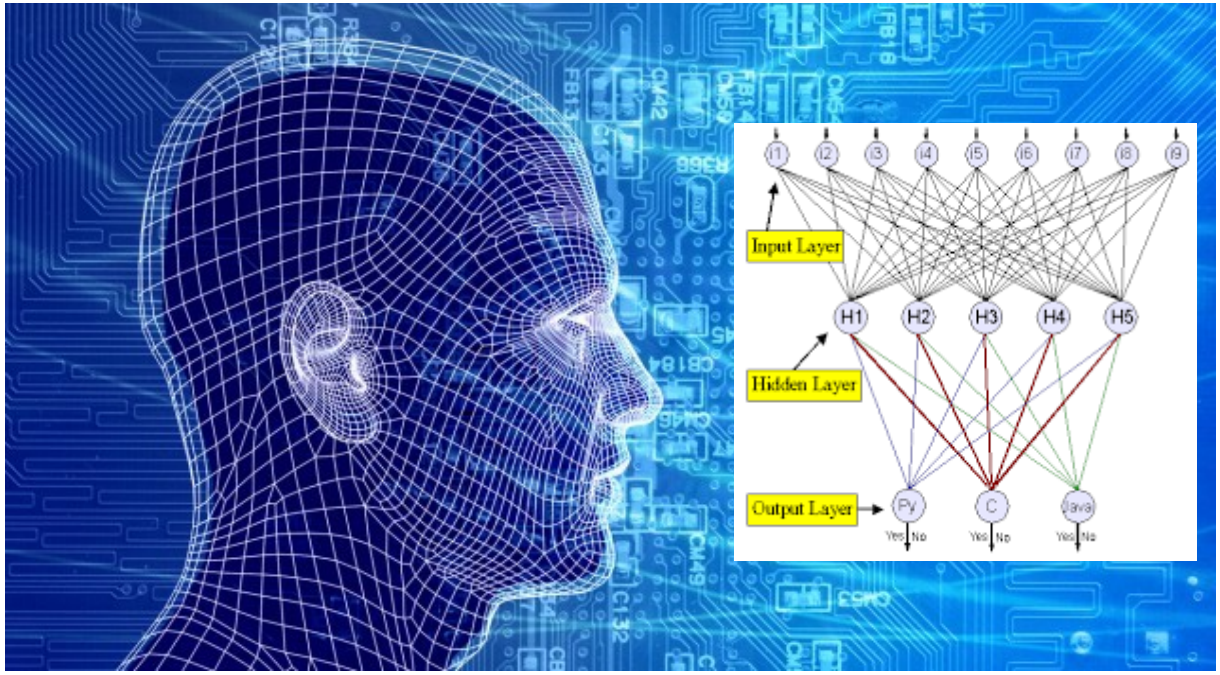
Machine learning

Lecture 3



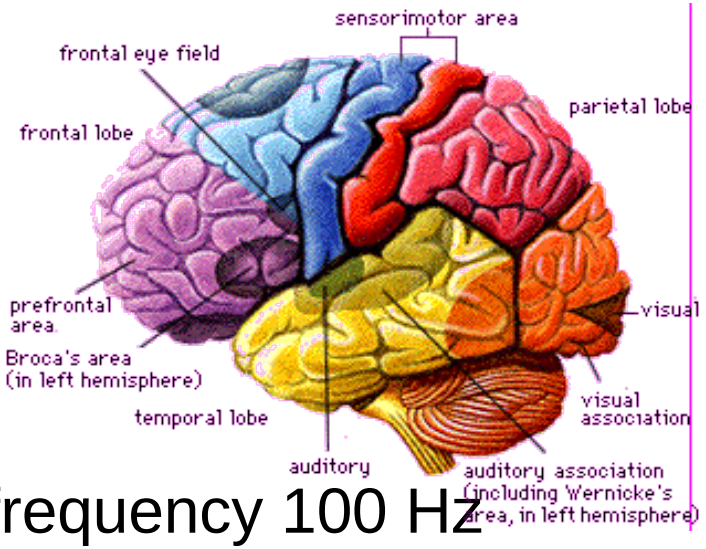
Marcin Wolter

IFJ PAN

14 March 2018

- Neural networks
- Bayesian Neural Networks

Inspired by human brain



- **Human brain:**
  - 10^{14} neurons, frequency 100 Hz
  - Parallel processing of data (complex pattern recognition in 100 ms – 10 steps only!!!)
  - Learns on examples
  - Resistant for errors and damaged neurons
- **Neural Network:**
  - Just an algorithm, which might not reflect the way the bain is working.

History

1938 N. Rashevsky, neurodynamics – neural networks as dynamic systems, recurrent networks;

1943 W. McCulloch, W. Pitts, neural networks=logic systems;

1958 F. Rosenblatt, perceptron, network as a function;

1973 Chr. von der Malsburg, self-orgnization in the brain;

1982 Kohonen, Self-Organizing Maps

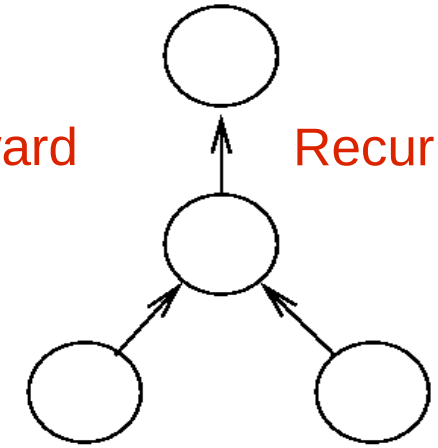
...

1986 **backpropagation of errors, many application!**

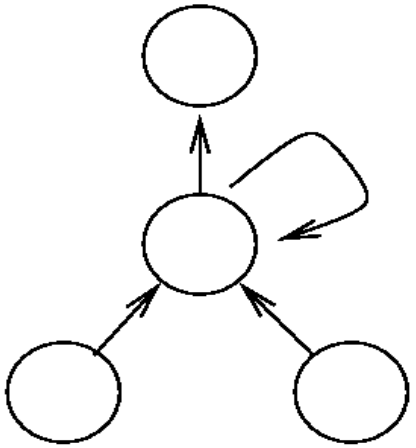
What a Neural Network is?

- Neural Network – a mathematical model which is composed out of many functions (typically nonlinear)
- **Tasks:**
  - **Event classification** – background vs signal classification
  - **Regression** – approximation of a real function
- **Two types of networks:**
  - **Feed forward** – information is sent from input layer to output without any loops
  - Recurrent – recurrent loops.
- **Learning:**
  - supervised
  - unsupervised

Feed-forward



Recurrent

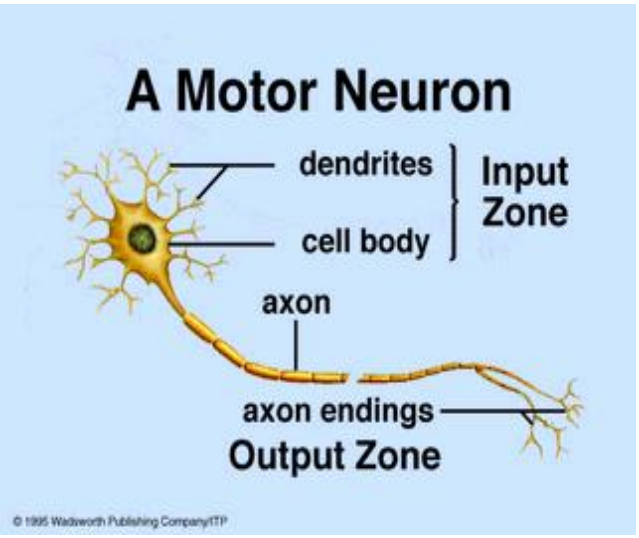


What are they used for?

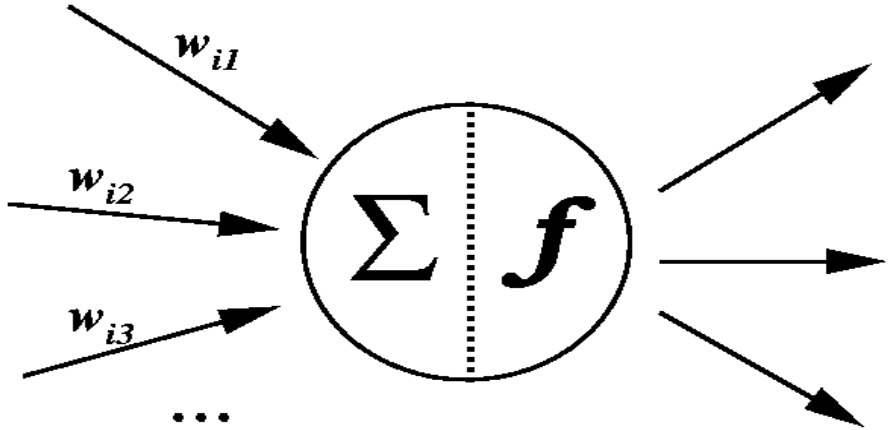
- Expert systems
- Pattern recognition
- Predictions (metereology, stock market...)
- In elementary particle physics:
  - Data analysis (mostly event selection)
  - Trigger systems – times ~ms (dedicated electronics, computer farms)

Neuron – the basic element

- Function of a weighted average of inputs $y_i = f(\sum_j w_{ij}\, y_j)$



Inputs

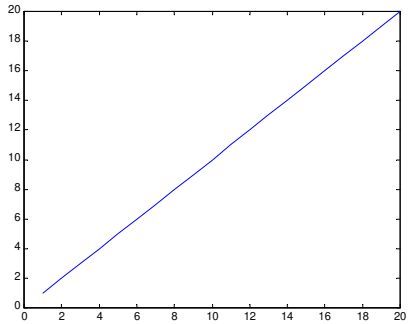


Outputs

$y_i = f(net_i)$

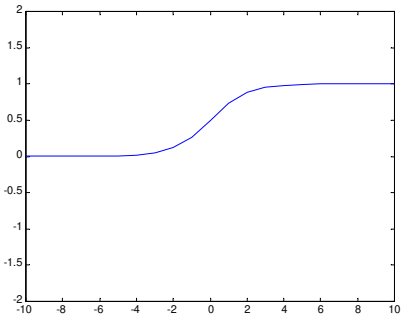
- Function *f* is called the **activation function**

Typical activation functions



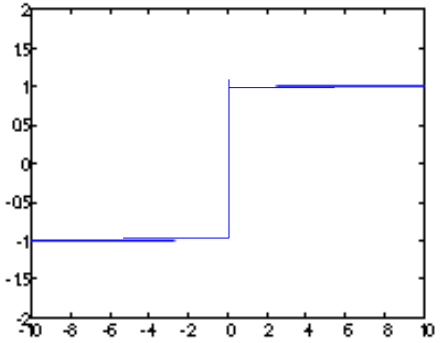
Linear (the whole network is called “linear”)

$$y = x$$

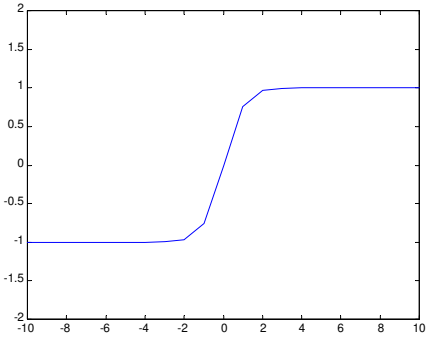


Logistic

$$y = \frac{1}{1 + \exp(-x)}$$



Step

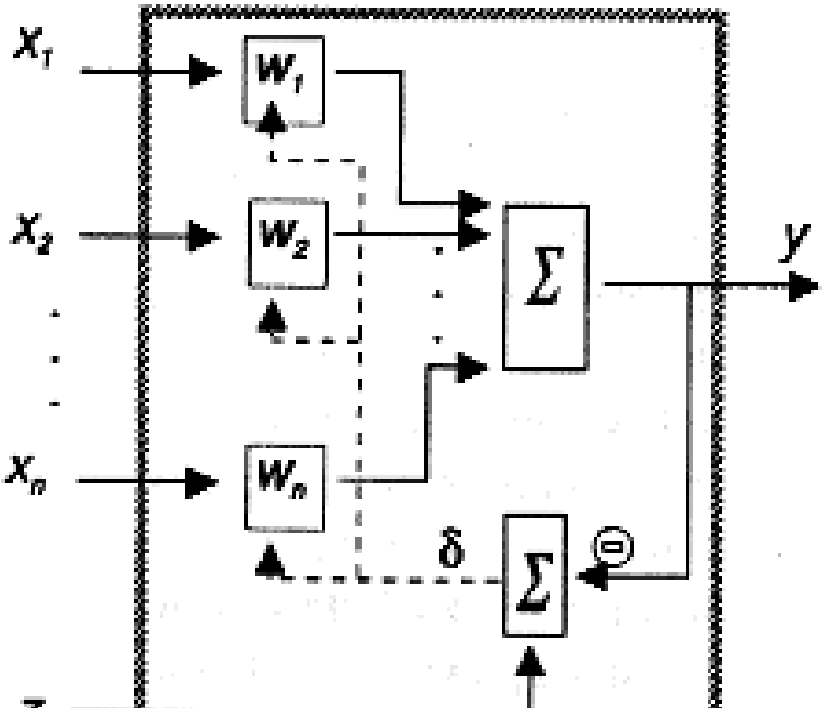


Hyperbolical tangent

$$y = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Training a single neuron

Neuron is trained on examples
Supervised learning – the proper answers are known



ADALINE

(Adaptive Linear Network)

- X_i – input data
- Y - output value
- Z - the true output value (supervised training!)

- TASK – minimize the loss function:

  Minimize: $\chi^2 = \sum (z^{(j)} - y^{(j)})^2$

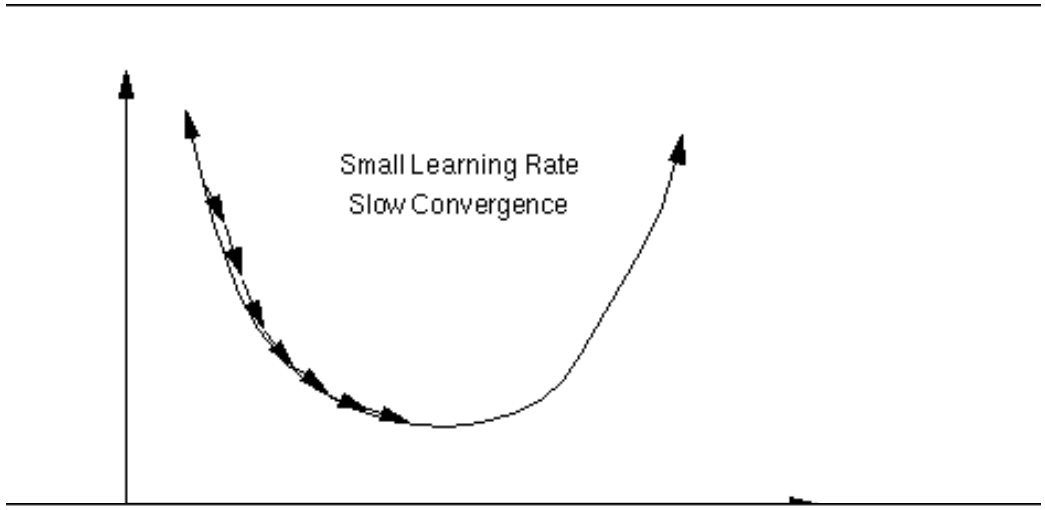
- New set of weights:

$$\delta = z - y$$

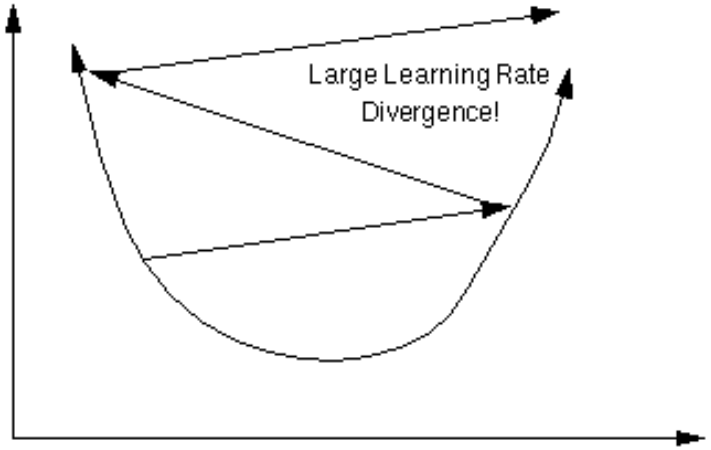
- η - learning speed

$$W' = W + \eta \cdot \delta \cdot X$$

Speed of learning



- To small

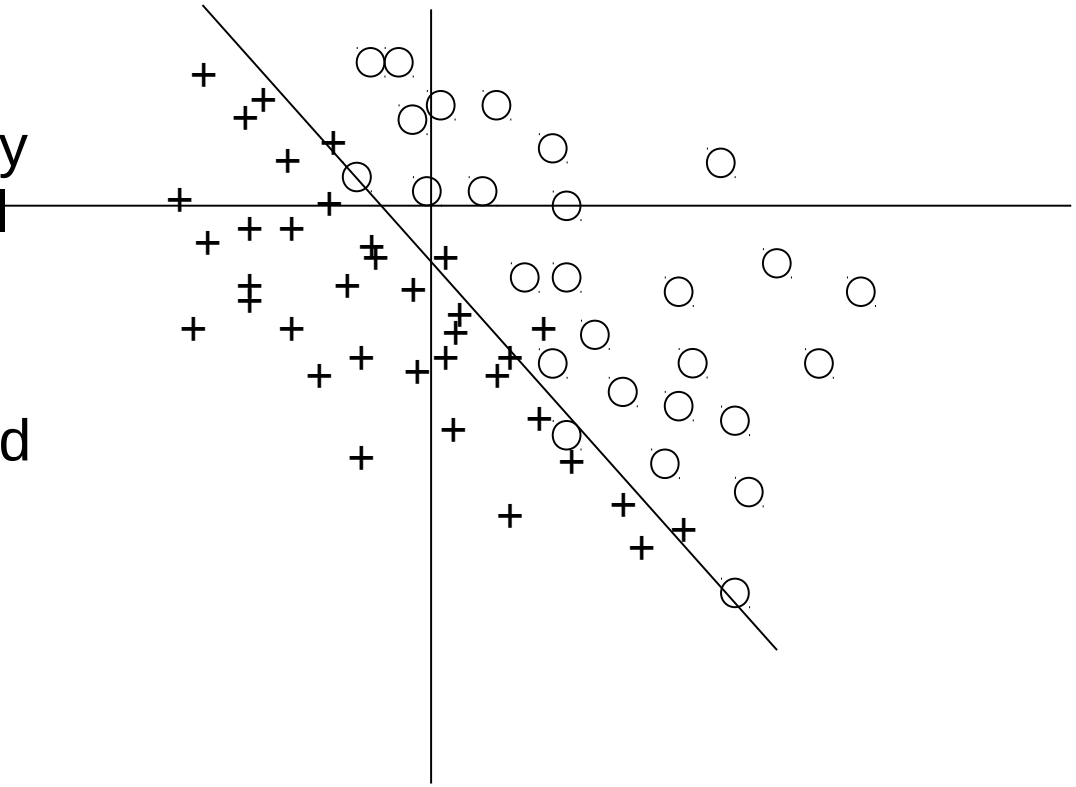


- To big

What can a single neuron (perceptron) do?

- Perceptron (with a step activation function) can divide a plane by a straight line (in general: division by a hyperplane in the n-dimensional space).

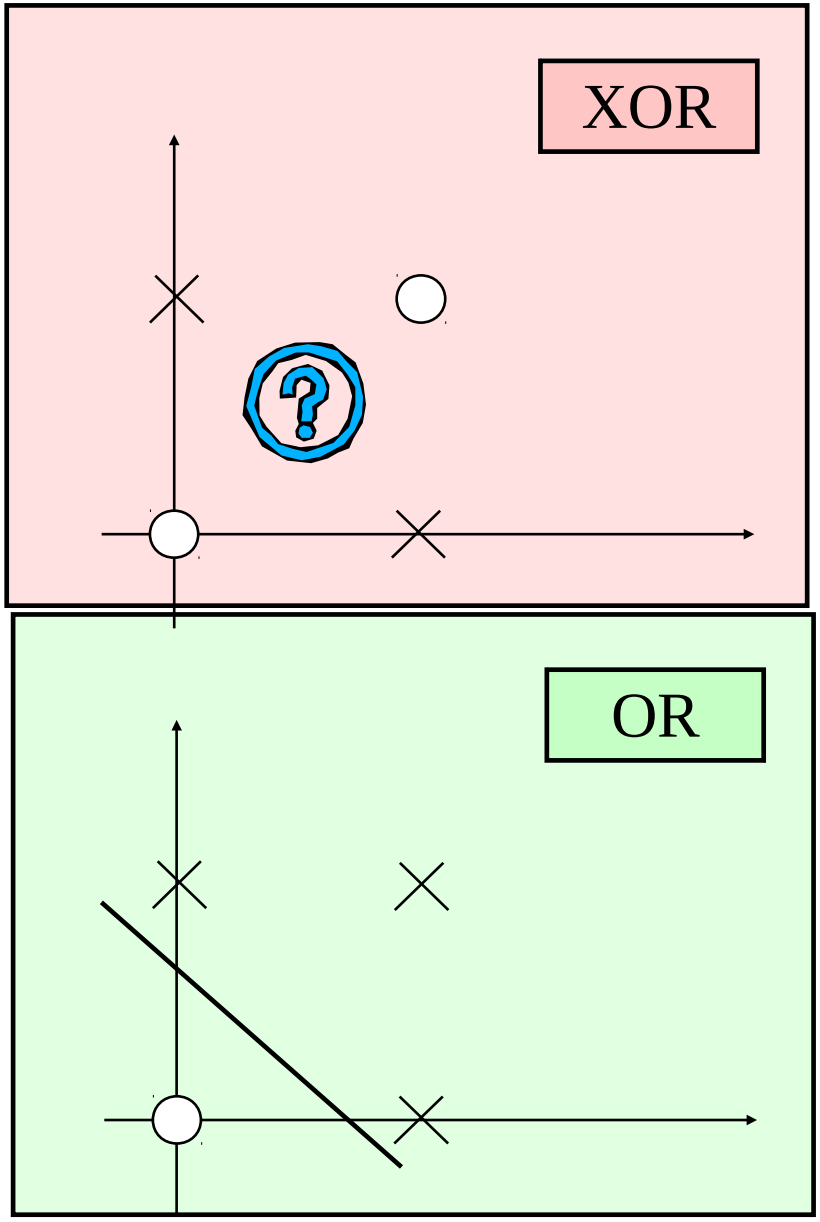
- Points above the line are classified as “1” (signal) and below as “0” - background.

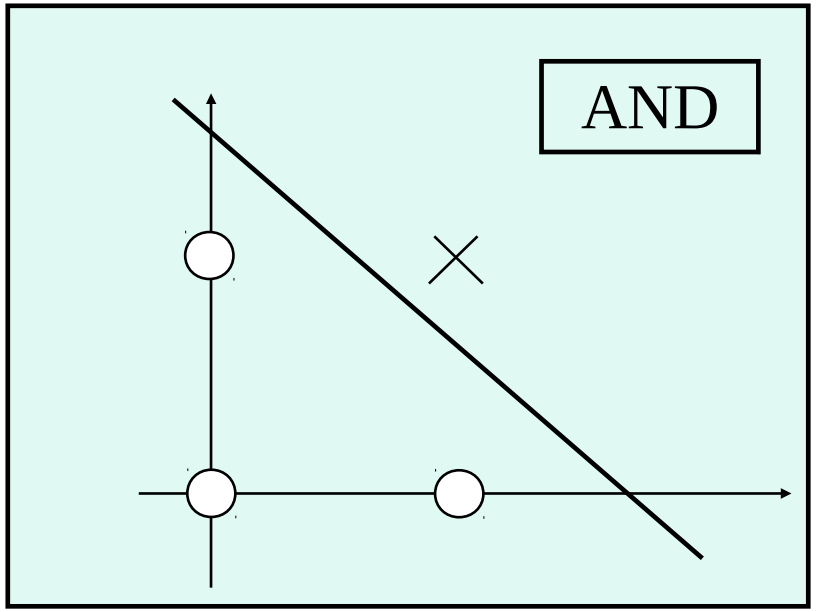


What the perceptron can not do?

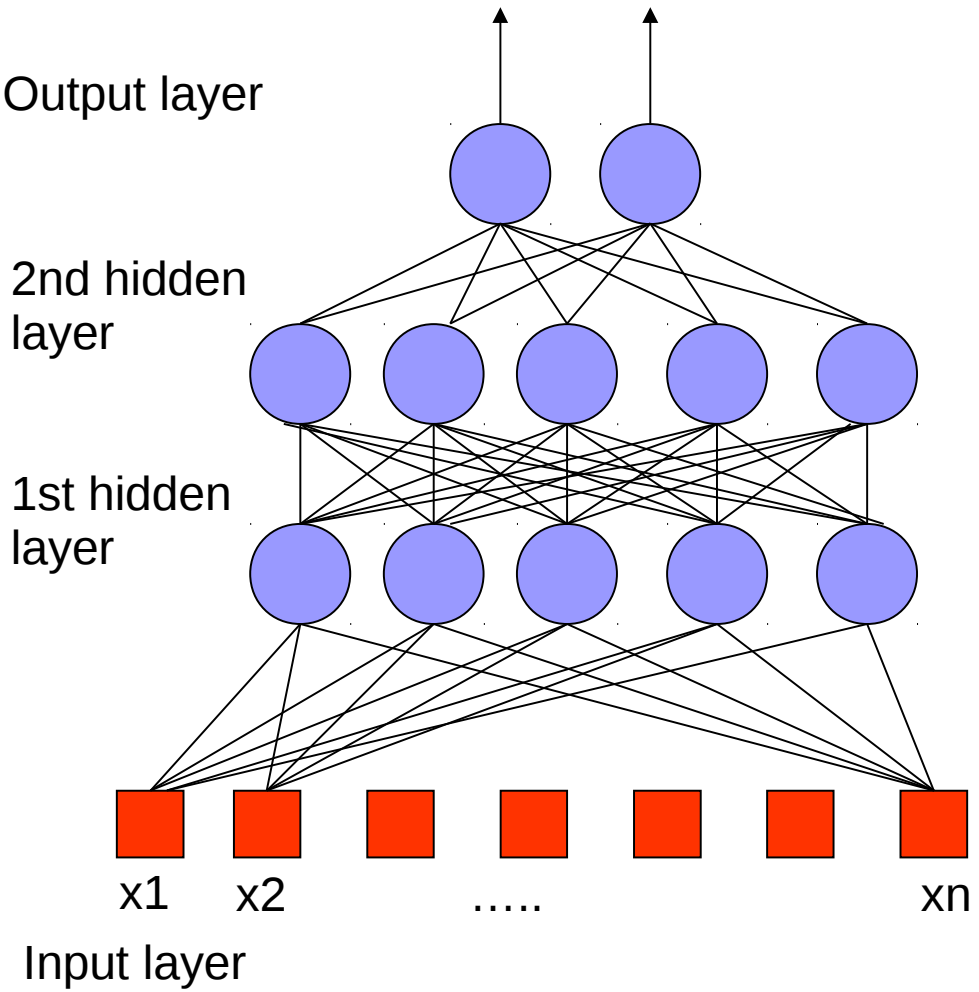
- A single perceptron can’t separate the linearly not separable classes, for example the XOR function.
- The discovery of these limitations (1969) stopped the development of Neural Networks for some time.

Applet perceptron





So, maybe a network of perceptrons?

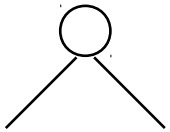
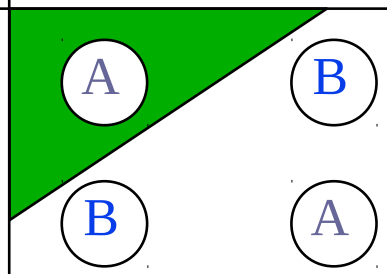
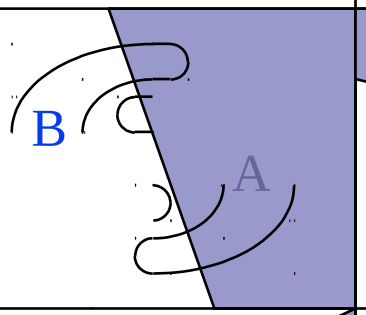
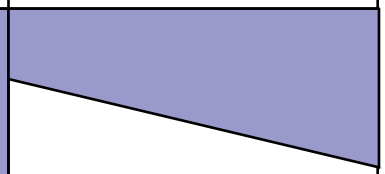
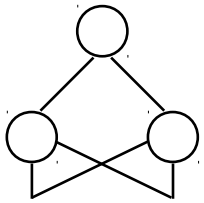
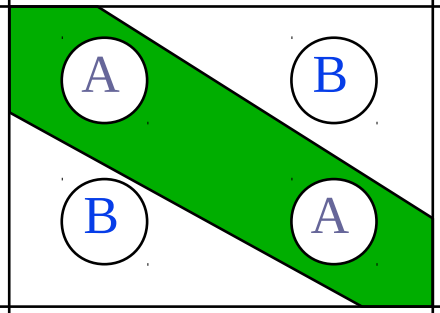
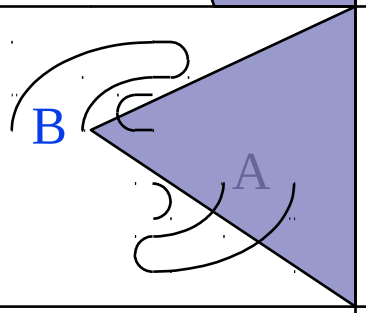
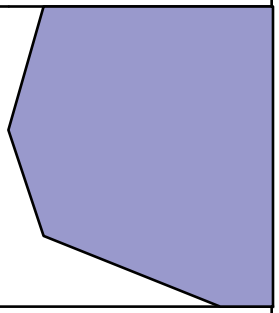
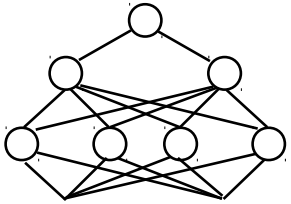
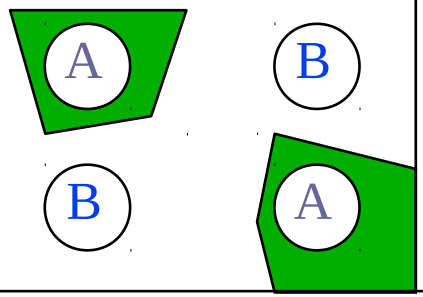
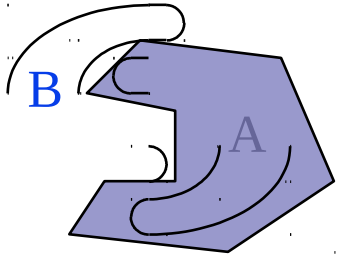
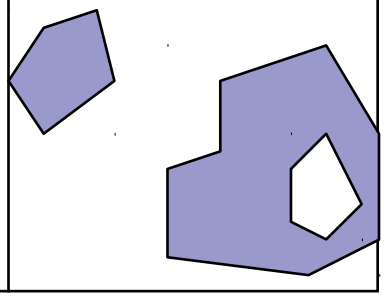


- Feed forward network – the information propagates from input to output.
- The net is the sum of many activation functions (in general non-linear)
- A network complicated enough can reproduce any function.

What a network can do?

(step activation function)

applet general1

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Abitrary (Complexity Limited by No. of Nodes)			

How to train a multilayer network?

- Minimize the loss function by choosing a set of weights ω:

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

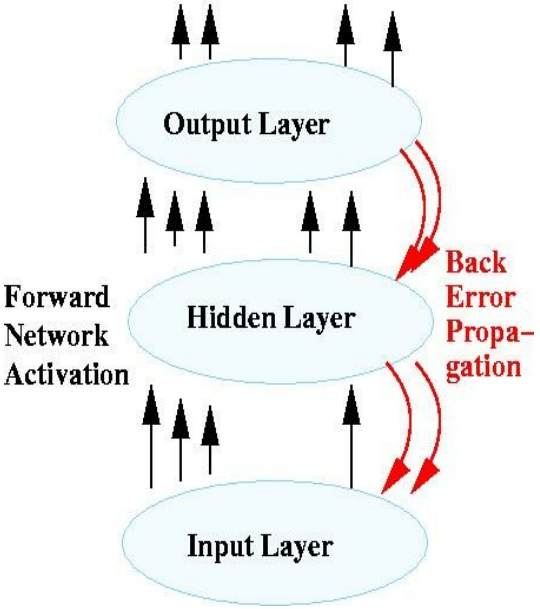
- Problem – how to correct the weights in the deeper layer, while comparing only outputs on the last layer?
- This problem stopped the development of Neural Networks for 30 years, until 80-ties.
- Solution - the **backpropagation** method. Errors $\delta = t - n(x, \omega)$ are propagated backward through the net using the actual weights.

Typical training procedure

- Two data samples: for training and for tests.
- $\chi^2=\Sigma(z-y)^2$ is calculated for both samples and compared to avoid **overtraining.**
- ***Backpropagation:*** difference between the expected and calculated value on output *y-f(x,w)* is propagated backward through the net using the actual weights:

$$dw_{ij} = \rho x_i (t_j - y_j),$$

where ρ is a speed of learning, t_j the true value on the output *j*, y_j calculated by the net, and x_i is an actual value on the neuron *i* in the layer preceding the output layer.



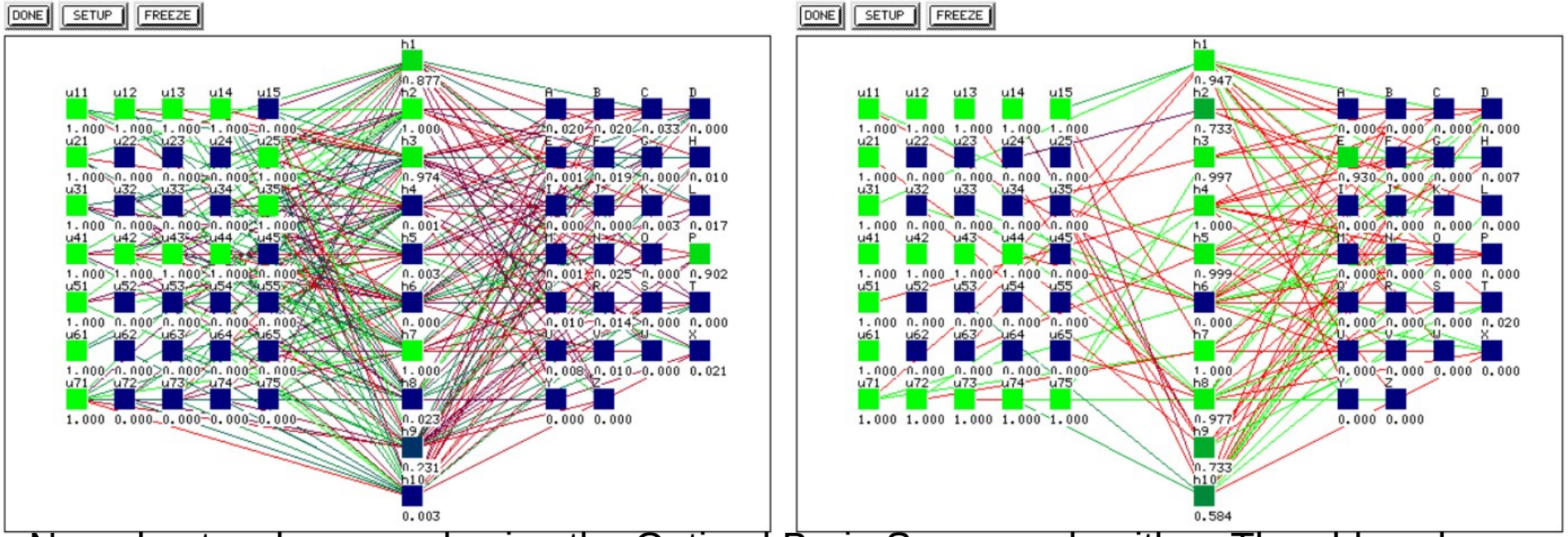
Finding the minimum

- We never know, whether the global or a local minimum of the loss function $\chi^2=\Sigma(z-y)^2$ was found.
- Mechanisms preventing stopping in a local minimum:
  - Using random initial weights, repetition of training,
  - Addition of noise, so the minimizing algorithm can jump out of a local minimum (jittering).

Network pruning

Algorithms removing the less important connections
- Simplifying the net they increase the speed
- Avoid overtraining

Alternative – gradually build the net adding new neurons (or layers)until it reaches the optimal size.



Neural network prunned using the Optimal Brain Surgeon algorithm. The old package SNNS is used.

A Strategy for Discovering a Low-Mass Higgs Boson at the Tevatron

Pushpalatha Bhat, Russell Gilmartin and Harrison B. Prosper

FERMILAB-Pub-00/006

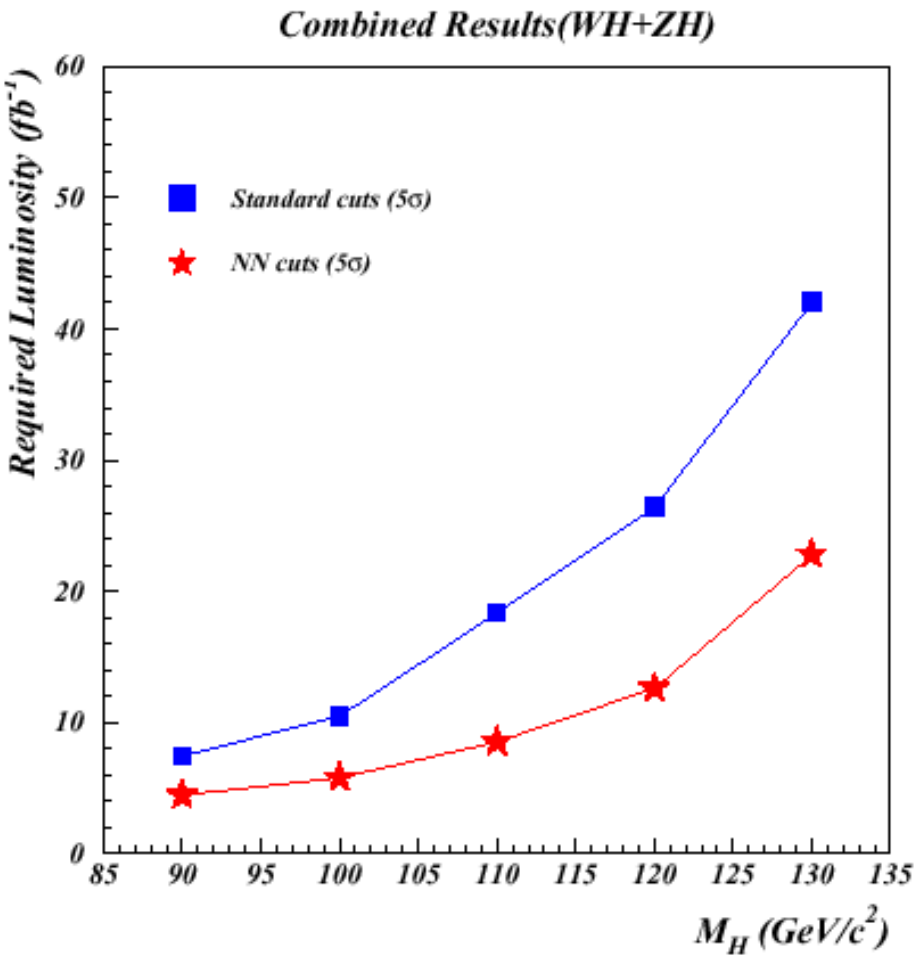
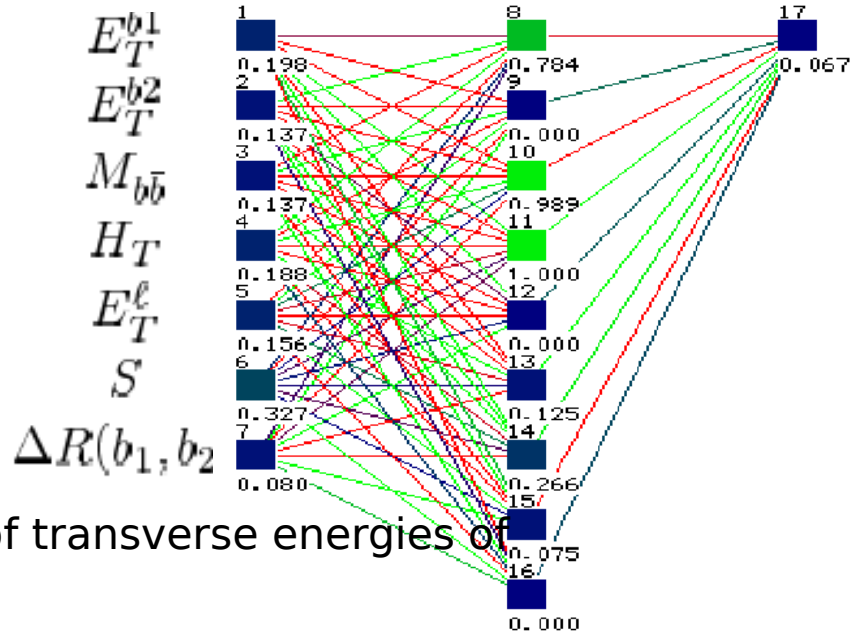


FIG. 8. Required integrated luminosity for a 5σ observation with all channels combined. The luminosities given are for a *single* Tevatron experiment, as in the previous plots.



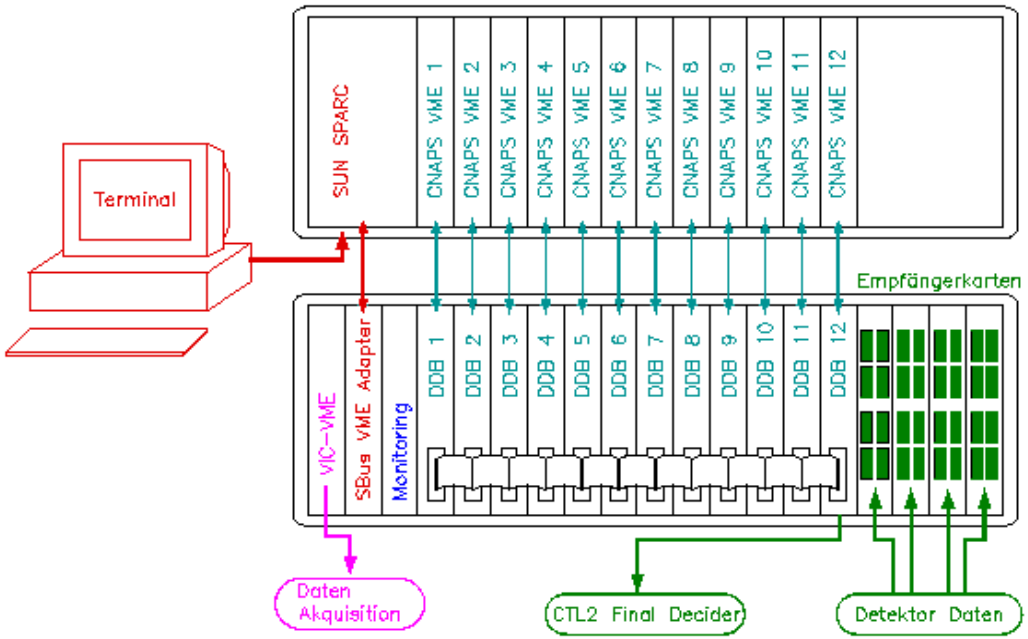
H_T – sum of transverse energies of all tracks.

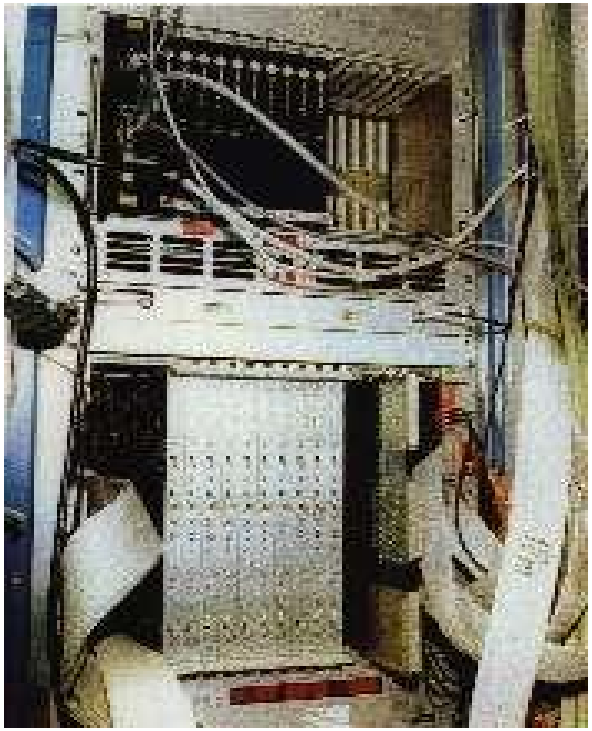
Example:

Prediction, how many events are needed to discover (that time hypothetical) Higgs boson at Fermilab.

The trigger system at the H1 experiment at DESY (1996)

- The trained NN is fast – can be used in the trigger system.
- Secend trigger selection step (L2) – hardware implementation of the NN.
- Implementation on parallel processors (CNAPS from Adaptive Solutions).
- Decision time – (L1 hardware – 2.3 µs, L2 Neural Network - 20 µs, L3 microprocessors - 800 µs).

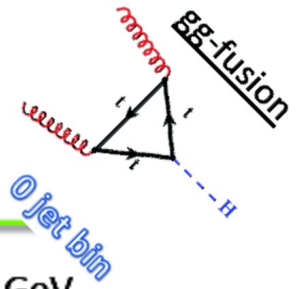




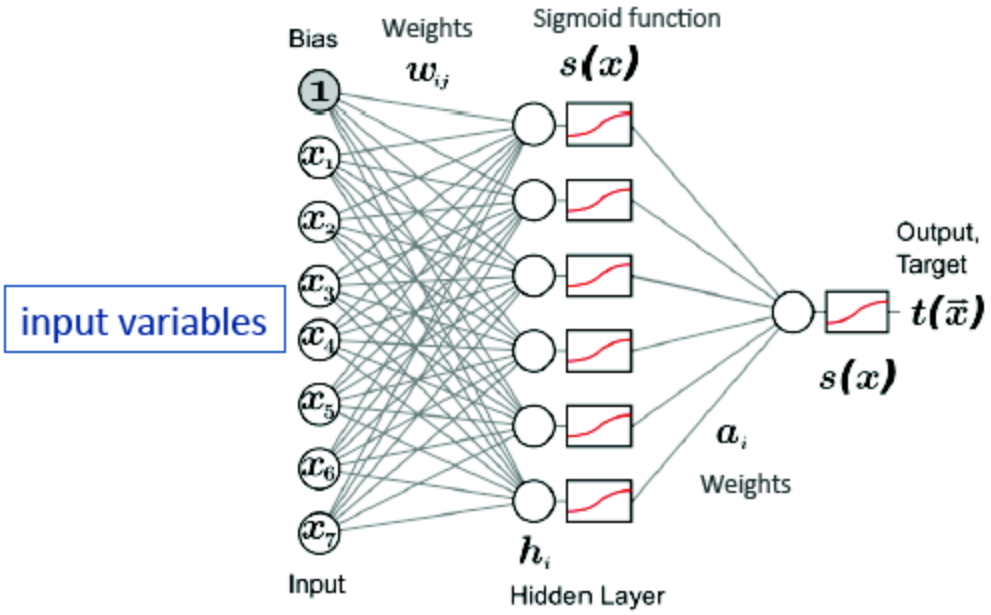
J. K. Kohn et al,
Realization of a second level neural network trigger for the H1 experiment at HERA,
Nucl. Instrum. Meth. A389 (1997)

Neural network for signal selection in the Higgs searches at ATLAS experiment

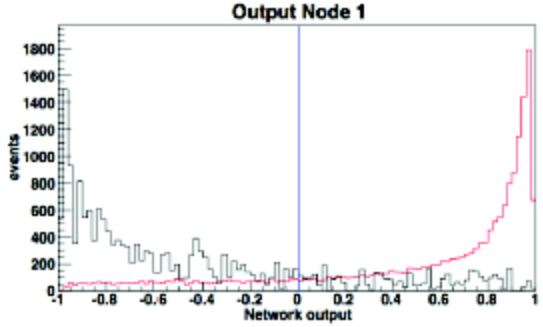
Using Neural Networks to improve sensitivity
Input variables for the training of NN

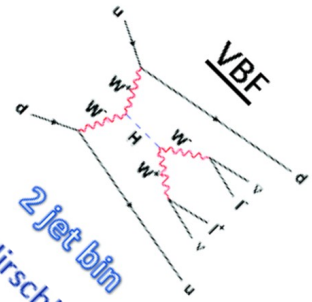


Neural Networks are trained in the 0-jet bin and 2-jet bin for m_H = 150 GeV and 180 GeV.



Training @
m_H = 150 GeV
m_H = 180 GeV



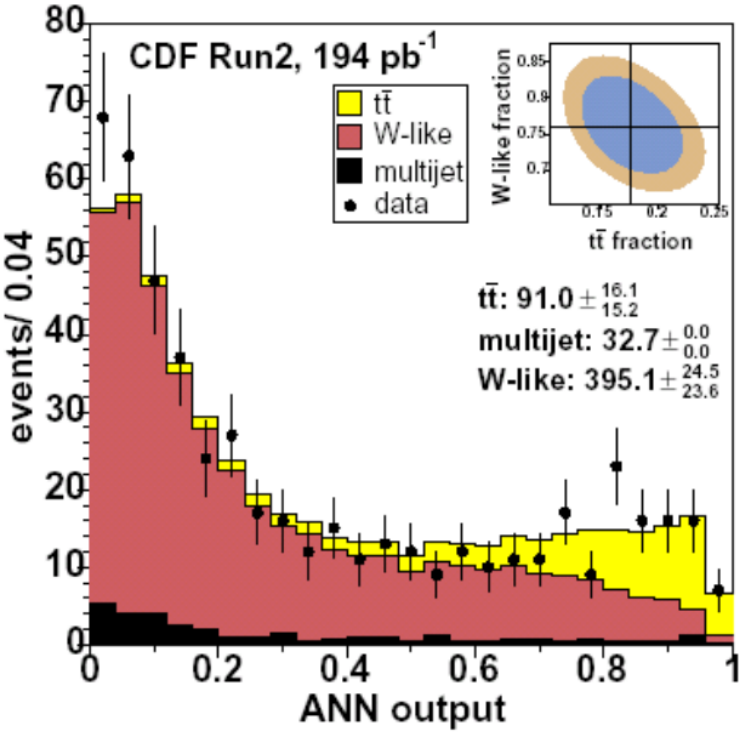


Simon Köhlmann, Georg Sartisohn, Wolfgang Wagner, Dominic Hirschbühl, Gunar Ernis,

Search for Higgs bosons predicted in Two-Higgs-Doublet Models in the WW→lvlv decay channel

<phi-t> *Implementation with NeuroBayes*

Identification of *t* quarks at CDF experiment

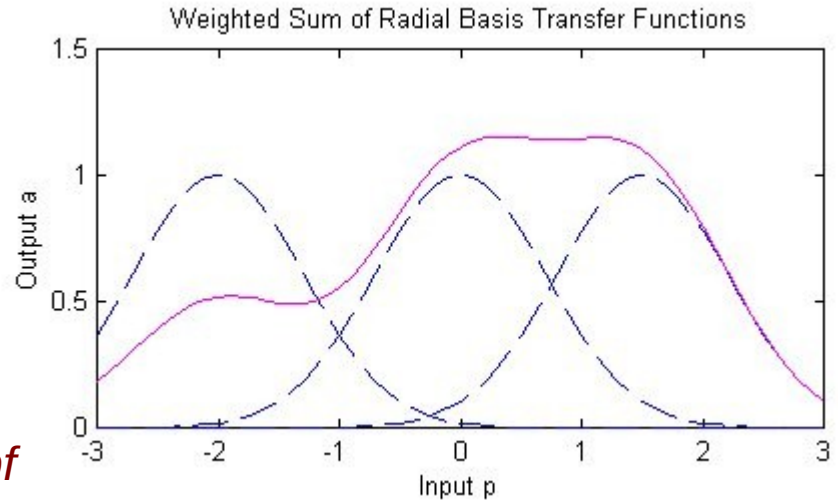


Rysunek 4.5: Rozkład odpowiedzi sieci neuronowej dla stanu końcowego $W+ \geq 3$ dżety (bozon W oraz co najmniej 3 dżety) w eksperymencie CDF, porównany z wynikami dopasowania funkcji [69].

Radial Base Functions (RBF)

- A neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Formulated in a 1988 paper by Broomhead and Lowe.
- Neuron in a hidden layer – the radial function, which is non-zero around the center c only:

  $f_i(x) = f_i(||x - c||)$ - a radial base function.



Applet rbf

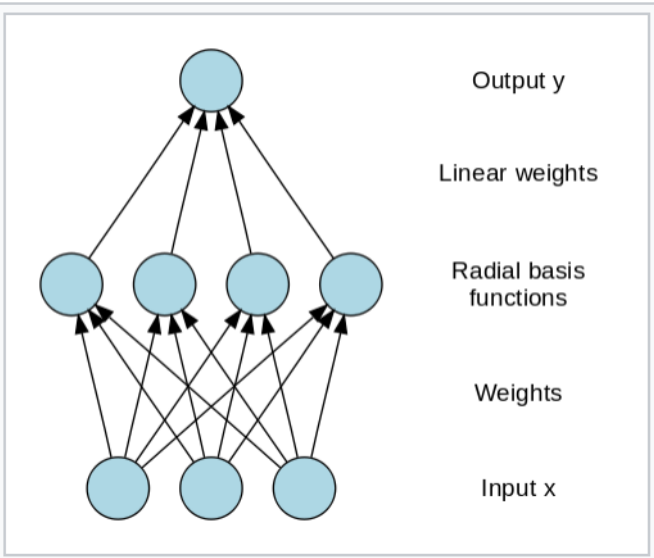


Figure 1: Architecture of a radial basis function network. An input vector x is used as input to all radial basis functions, each with different parameters. The output of the network is a linear combination of the outputs from radial basis functions.

RBF functions

1. **Gaussian Functions:**

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad \text{width parameter } \sigma > 0$$

2. **Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{1/2} \qquad \text{parameter } \sigma > 0$$

3. **Generalized Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{\beta} \qquad \text{parameters } \sigma > 0,\ 1 > \beta > 0$$

4. **Inverse Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{-1/2} \qquad \text{parameter } \sigma > 0$$

5. **Generalized Inverse Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{-\alpha} \qquad \text{parameters } \sigma > 0,\ \alpha > 0$$

6. **Thin Plate Spline Function:**

$$\phi(r) = r^2 \ln(r)$$

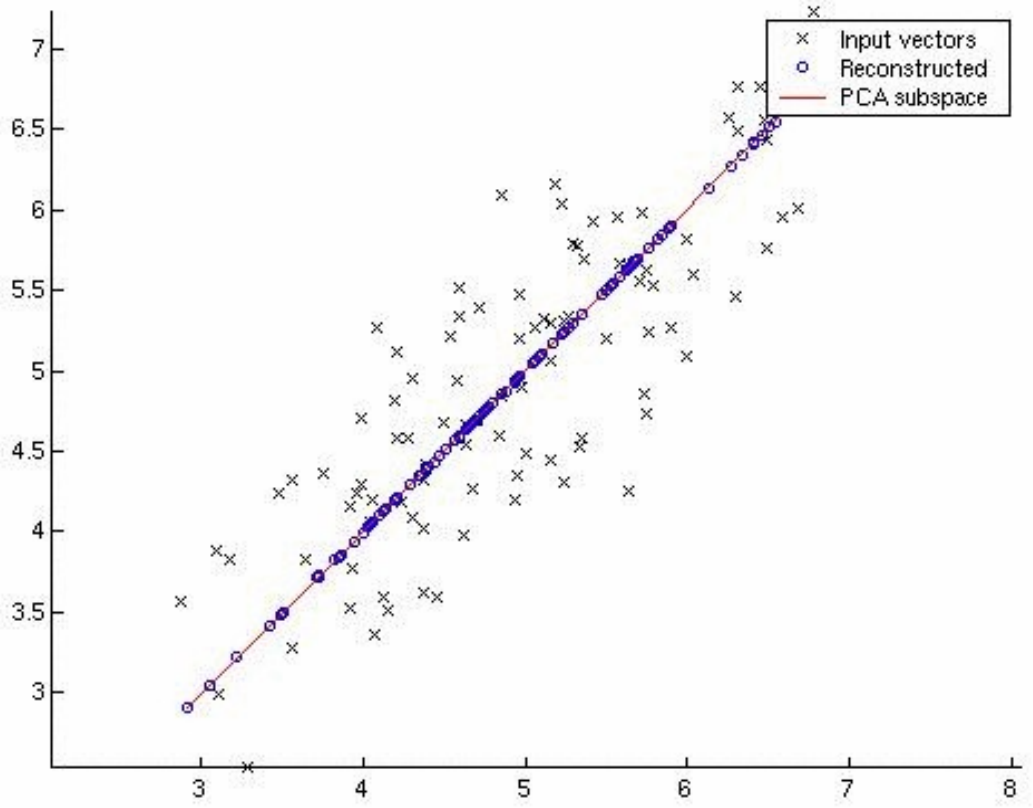
7. **Cubic Function:**

$$\phi(r) = r^3$$

8. **Linear Function:**

$$\phi(r) = r$$

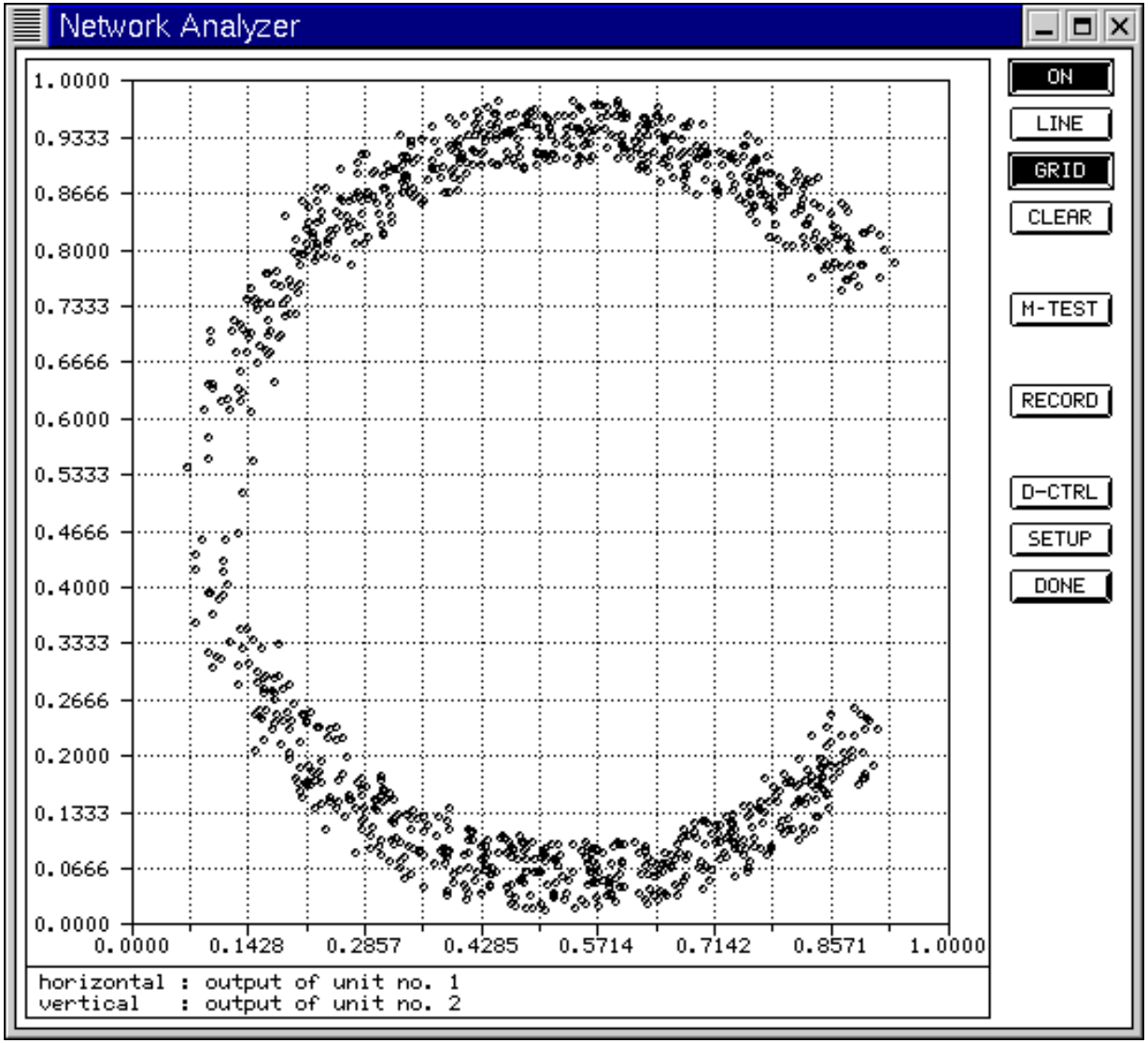
Something else...
Non-linear PCA
(Principal Component Analysis)



Linear PCA:

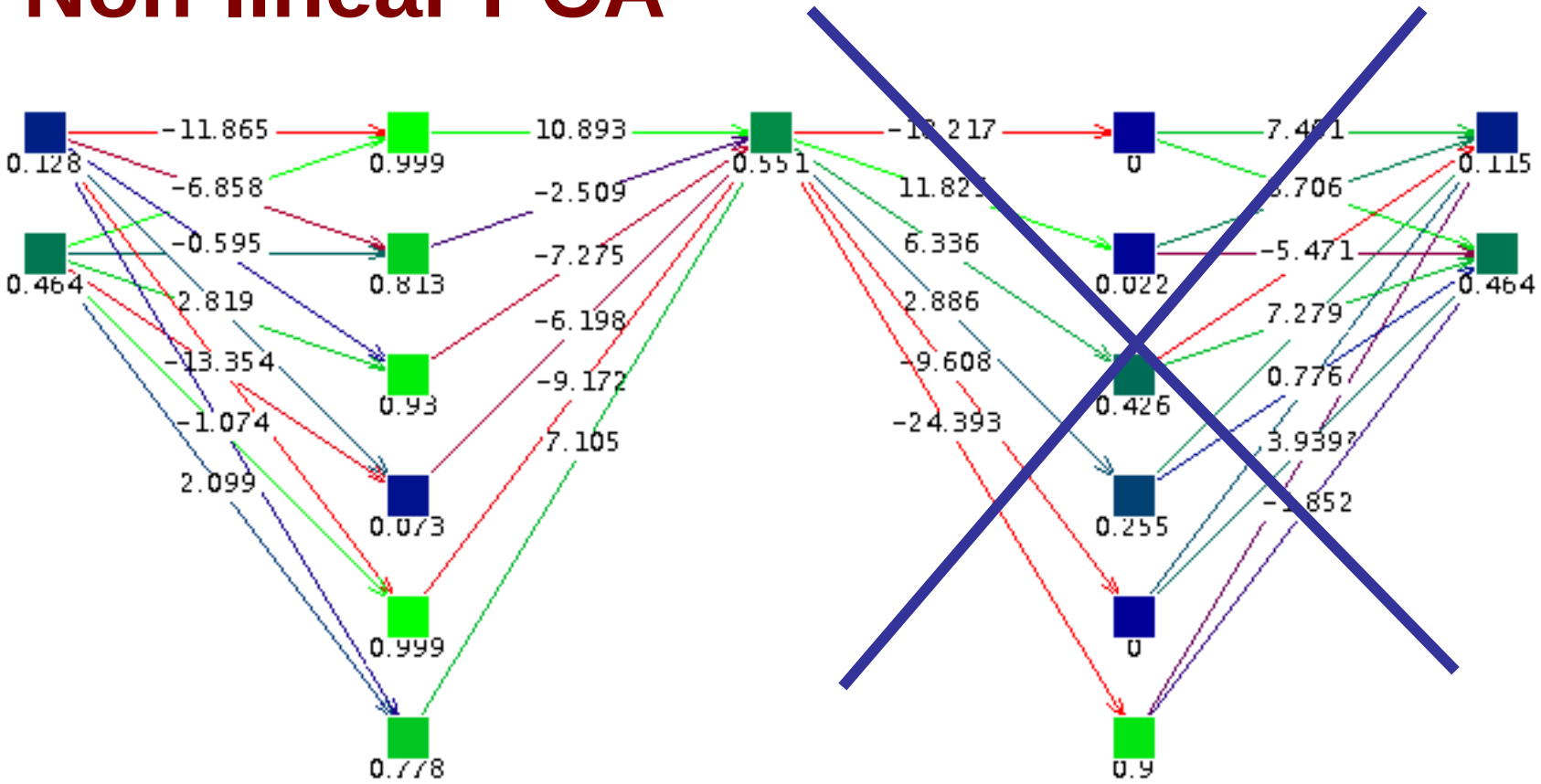
- Dimensionality reduction (here from 2 dimensions to 1)in such, that a loss of information is minimized.
- Finds the orthogonal base of covariant matrices, eigenvectors with smallest eigenvalues are skipped.

What to do with a non-linear example?



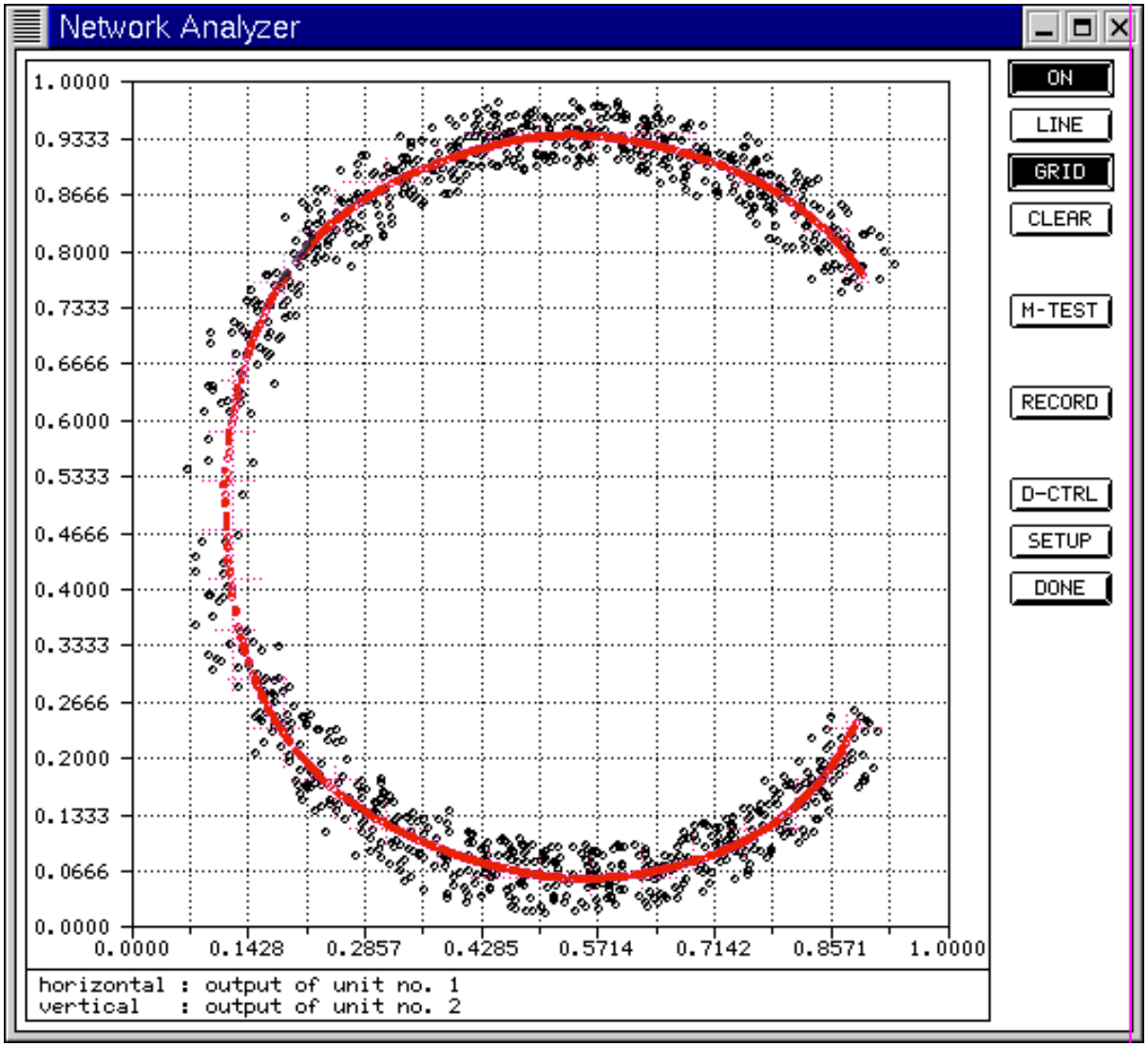
- How to transform (in optimal way) into 1-dim?

Non-linear PCA



The network is trained by giving the same vectors on input and output. Then it is cut by half.

Result – transformation into 1 dimension



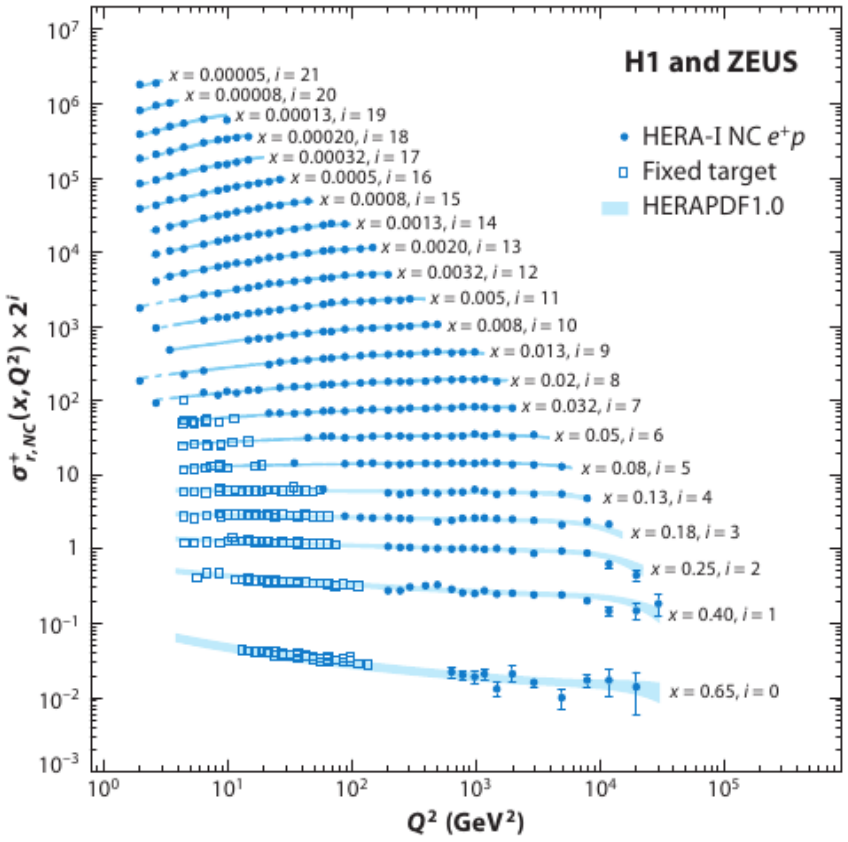
- Non-linear transformation.

Neural networks approximating the proton structure functions

- Inclusive cross-section for electron scattering on protons can be parametrized by the structure functions $F_1(x,Q^2)$ and $F_2(x,Q^2)$, where x – fraction of momentum carried by a parton, Q^2 – the four-momentum transfer.

- Structure functions are measured in various experiment and in various kinematical ranges. Taking all the available data and parametrizing them we can get the parametrized structire functions.

- **NNPDF** collaboration uses Neural Networks for approximation of the structure functions (or the distribution of partons in proton):
  - unbiased estimator (we do not choose the approximating function),
  - we do not histogram events (better use of available information).



NNPDF – structure function approximation

- **Create many replicas of data**

  All available data are used to create pseudo-data: replication of means, errors, correlations.

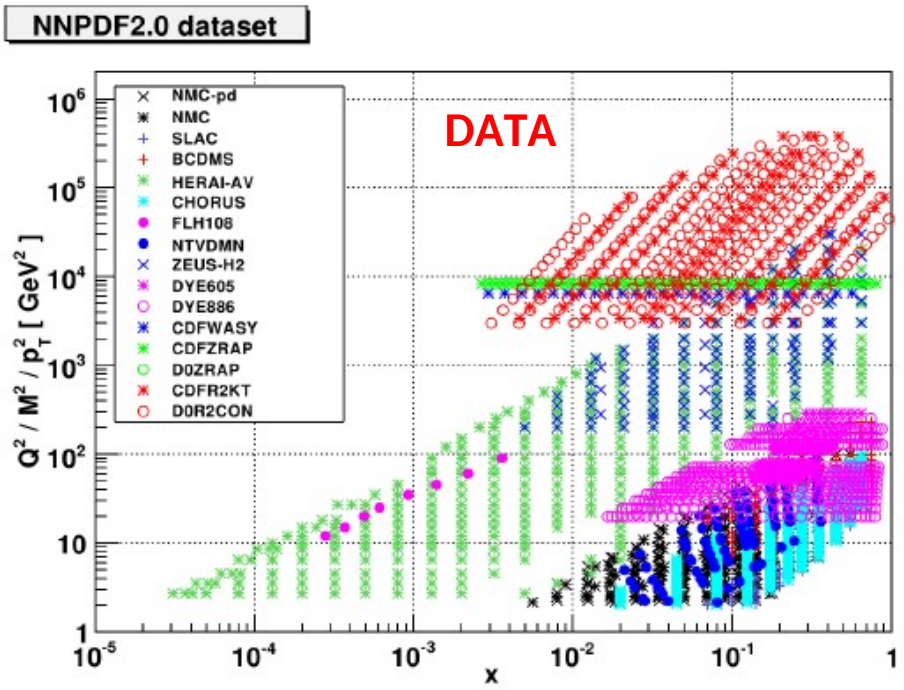
- **Create the probability densities of partons**

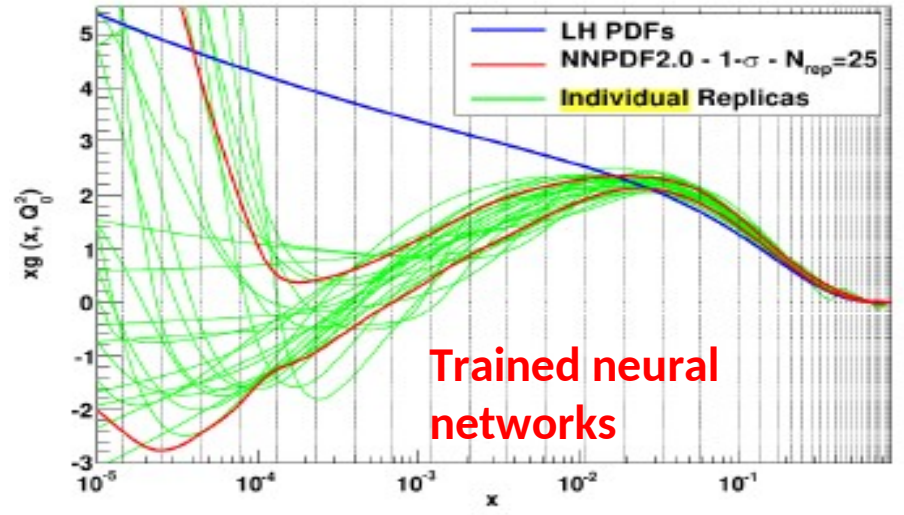
  Fitting of neural networks, one for each data replica.

- **Statistical reliability**

  A set of trained NN is used to reproduce the observables, including errors and correlations.

  Remark: the result is obtained from many neural networks – an error estimation.





Machine learning vs Bayesian learning

Machine learning

Teaching the function $y = f(x)$ giving **training data** $\boldsymbol{T} = (\boldsymbol{x}, \boldsymbol{y}) = (x,y)_1, (x,y)_2, \ldots (x,y)_N$ and **bonds** by giving a class of this function.

Bayesian learning

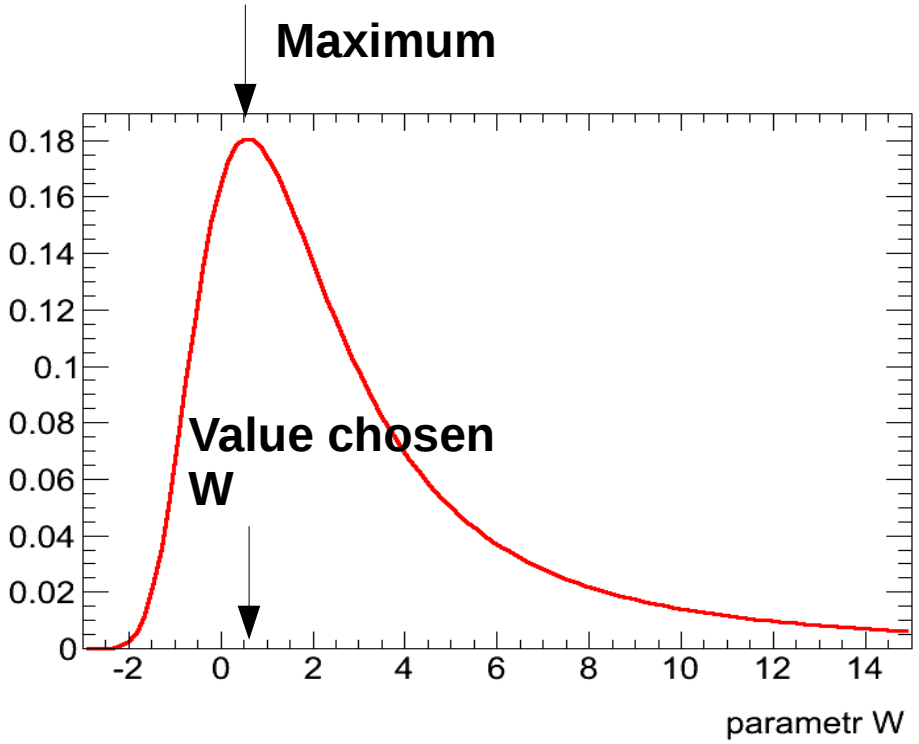
For each function f(x) from the function space F we find an *a posteriori* probability $\boldsymbol{p(f \mid T)}$ using the training sample **T= (*x*, *y*).**

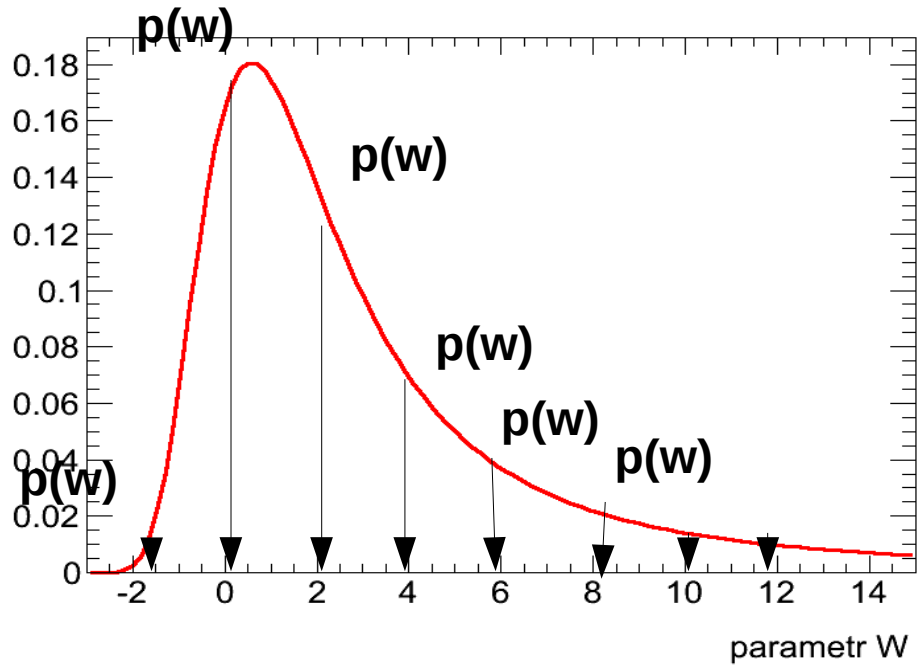
In the bayesian learning we DO NOT find the one, best function, but we use many functions weighted by their probability.

Probability *a posteriori* – probability calculated using the results of the experiment.

Training sample $\boldsymbol{T} = (\boldsymbol{x}, \boldsymbol{y})$: a set of input vectors ***x*** and results ***y.***

Machine and bayesian learning





Machine learning

We chose one function (or a value of a parameter describing the function).

Bayesian learning

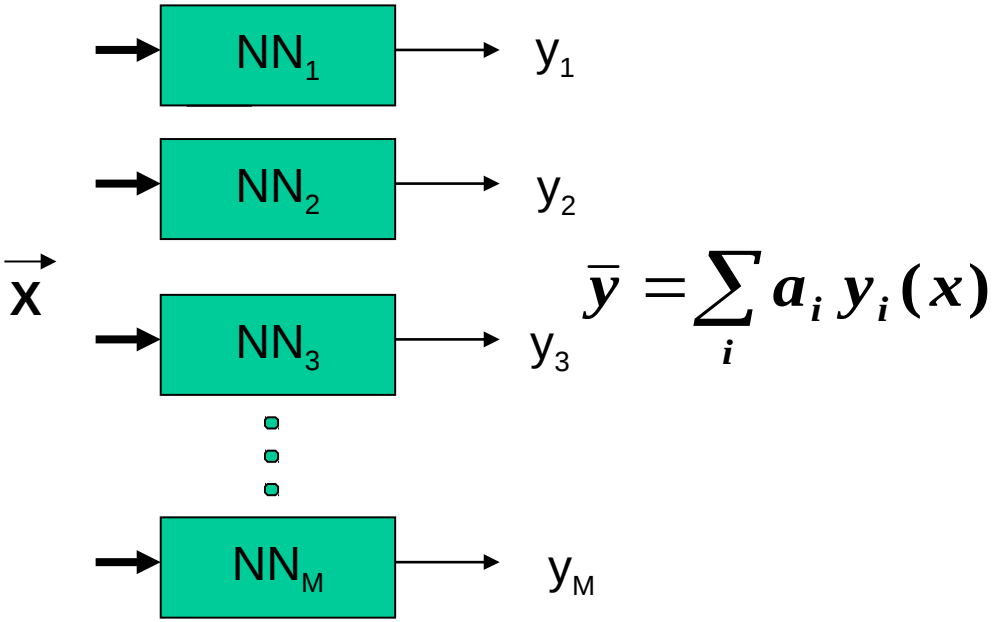
Each function (or a parameter value) is given some probability (weight).

Implementation of bayesian networks:

Instead of choosing a single set of weights describing the NN we should find the probability density for the entire space of weights.

Many neural networks.

Having many NN we can get the weighted mean or he most probable network and also the estimation error



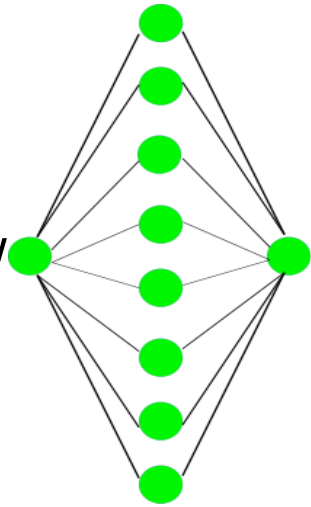
C.M. Bishop
"Neural Networks for Pattern Recognition",
Oxford 1995

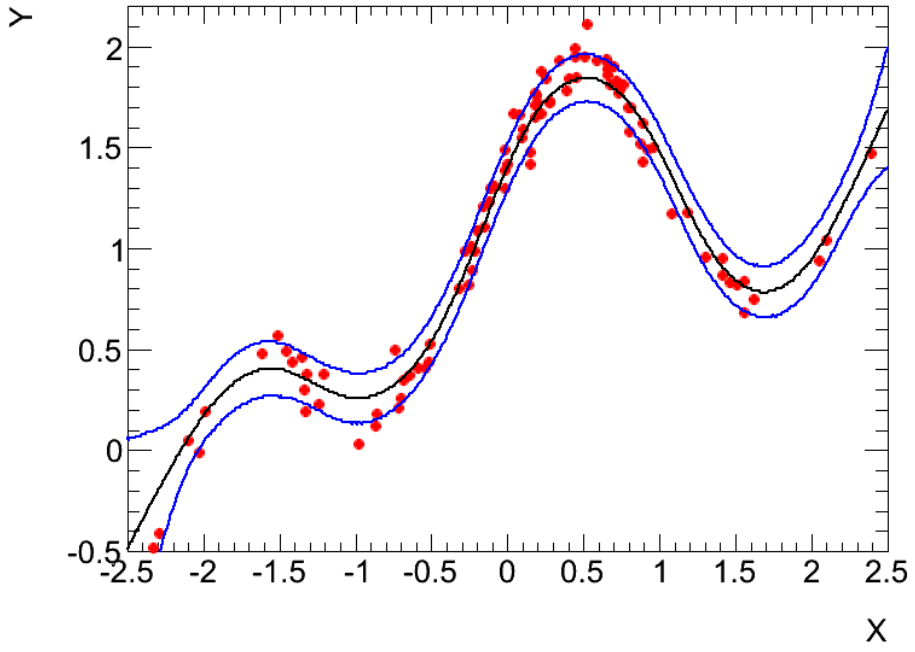
Free software:
Radford Neal, http://www.cs.toronto.edu/~radford/fbm.software.html

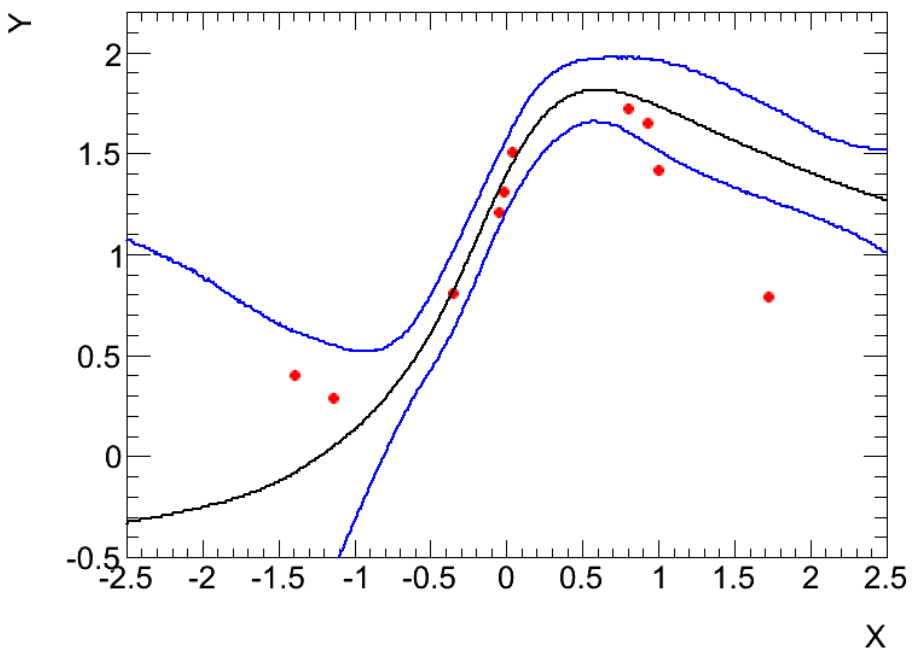
Example of BNN

How does the bayesian network with 8 neurons works for different amount of data?

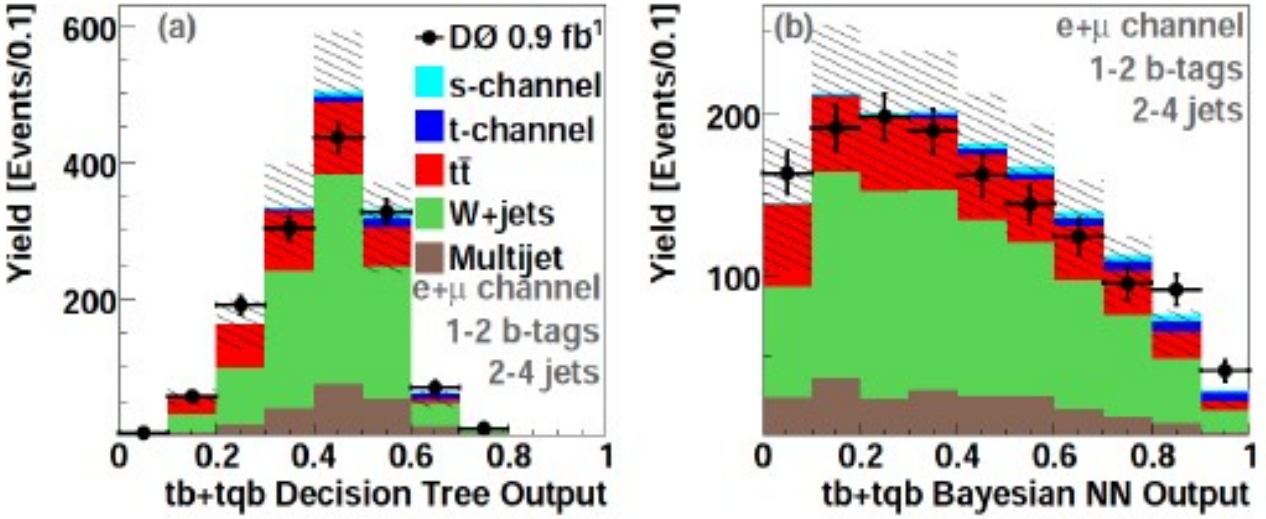
- Data generated using function: $y = 0.3 + 0.4x + 0.5\sin(2.7x) + 1.1/(1+x^2)$ with a gaussian noise (sigma = 0.1)
- 400 neural networks, from the distribution of answers we get: median and 10% Qnt i 90% Qnt (10% of networks gave answers below lower blue line and 10% above the upper line).
- When only 10 training points used, we got much higher errors (as they should be).





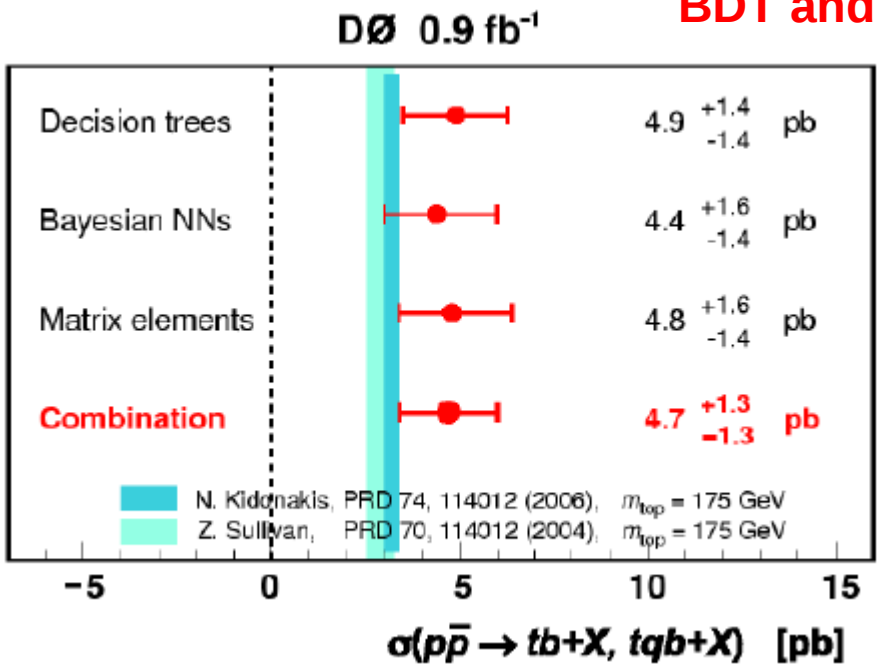


Example – search for a single top quark in D0 experiment



Similar performance of BDT and BNN

Analysis using:
Boosted Decision Trees (BDT) and bayesian neural networks



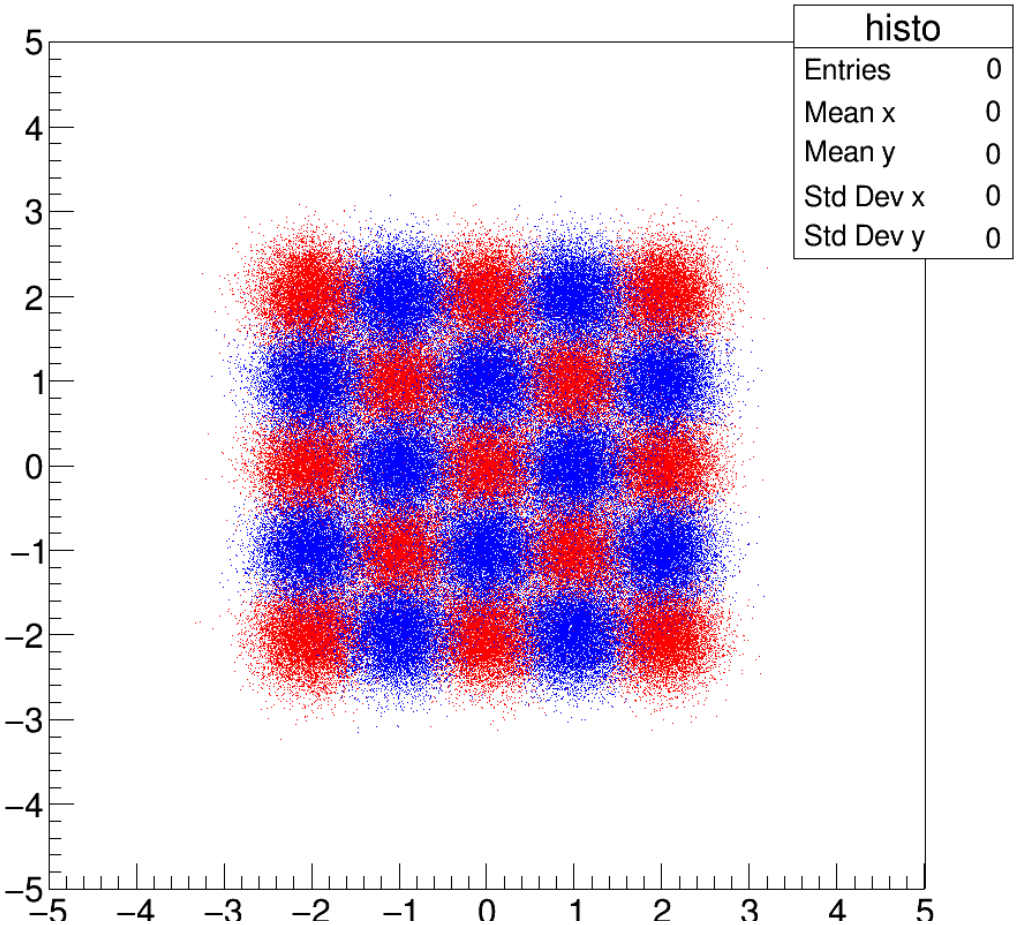
D0 Collaboration,
PRD 78 012005, 2008

Bayesian Neural Network Revival (some recent papers)

- A. Honkela and H. Valpola. Variational learning and bits-back coding: An information-theoretic view to Bayesian learning. IEEE Transactions on Neural Networks, 15:800-810, 2004.
- Alex Graves. Practical variational inference for neural networks. In NIPS 2011.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In ICML, 2015.
- José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In ICML, 2015.
- José Miguel Hernández-Lobato, Yingzhen Li, Daniel Hernández-Lobato, Thang Bui, and Richard E Turner. Black-box alpha divergence minimization. In Proceedings of The 33rd International Conference on Machine Learning, pages 1511-1520, 2016.
- Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. ICML, 2016.
- Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. NIPS, 2016.

Exercise

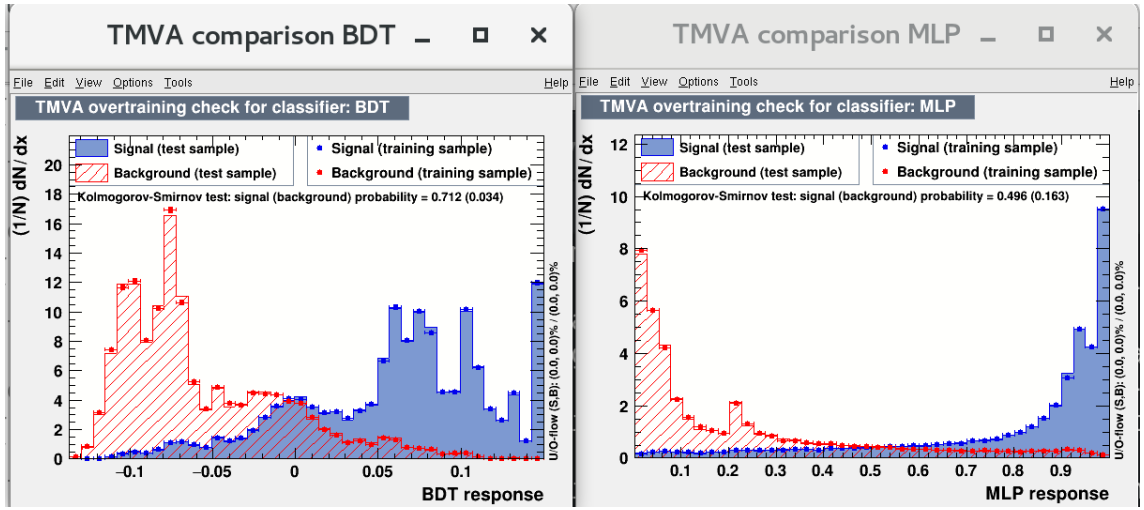
- To the last example of “chess board” classification add also Neural Network (MLP – Multilayer Perceptron)

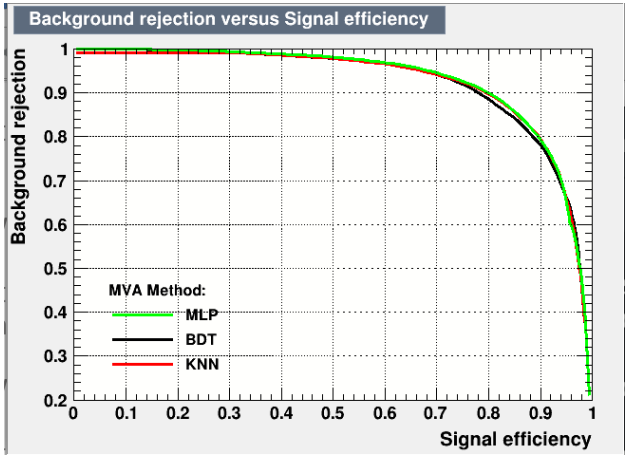


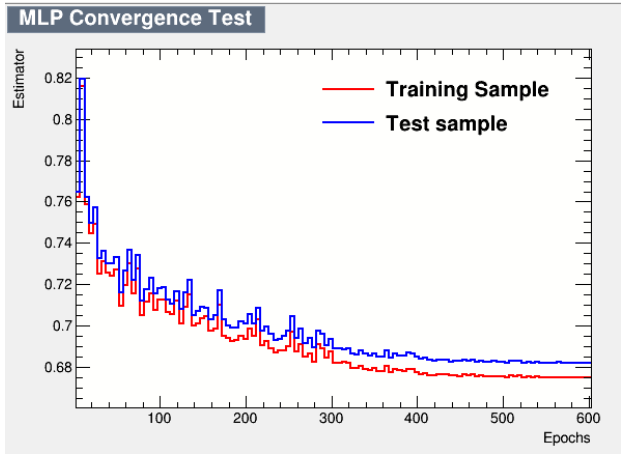
What I did:

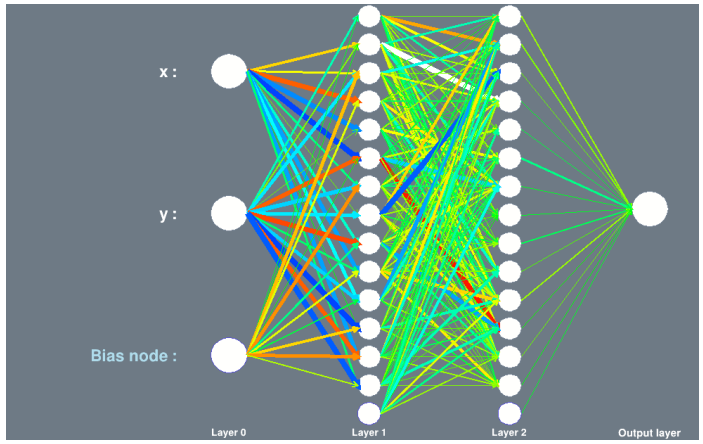
- Added MLP to the training script:

factory.BookMethod(dataloader, ROOT.TMVA.Types.kMLP, "MLP", "H:!V:NeuronType=tanh:VarTransform=N:NCycles=600:HiddenLayers=N+12,N+12:TestRate=5:!UseRegulator")









Challenge for machine learning experts

https://www.kaggle.com/c/higgs-boson

Open Higgs challenge!!!!!!!!!!!!!!

Machine Learning and HEP

- 90'ies - Neural Nets used by LEP experiments
- BDT (Adaboost) invented in 97
- Machine Learning used extensively at D0/CDF (mostly BDT, also Neural Nets) in the 00'ies
- Last years – mostly BDT built in TMVA ROOT package (popular among physicists). Neural Nets and other techniques treated as obsolete.

- **Not much work within LHC experiments on studying possible better MVA techniques.**
- **Enormous development of Machine Learning in the outside world in the last 10 years ("Big Data", "Data Science", even "Artificial Intelligence" is back).**

- **We have to catch up and learn from computer scientists:**

Make an open Higgs challenge!

- **Task: identify H->tau tau signal out of background in the simulated data.**

How did it work ?

- People register to Kaggle web site hosted https://www.kaggle.com/c/higgs-boson . (additional info on https://higgsml.lal.in2p3.fr).
- …download training dataset (with label) with 250k events
- …train their own algorithm to optimize the significance (à la s/sqrt(b))
- …download test dataset (without labels) with 550k events
- …upload their own classification
- The site automatically calculates significance. Public (100k events) and private (450k events) leader boards update instantly. (Only the public is visible)
- 1785 teams (1942 people) have participated
- most popular challenge on the Kaggle platform (until a few weeks ago)
- 35772 solutions uploaded

Funded by: Paris Saclay Center for Data Science, Google, INRIA

Final leaderboard

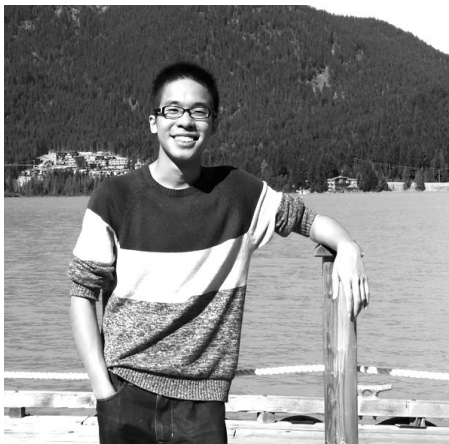
#	Δrank	Team Name ‡ model uploaded * in the money		Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑1	Gábor Melis ‡ *	7000$	3.80581	110	Sun, 14 Sep 2014 09:10:04 (-0h)
2	↑1	Tim Salimans ‡ *	4000$	3.78913	57	Mon, 15 Sep 2014 23:49:02 (-40.6d)
3	↑1	nhlx5haze ‡ *	2000$	3.78682	254	Mon, 15 Sep 2014 16:50:01 (-76.3d)
4	↑38	ChoKo Team		3.77526	216	Mon, 15 Sep 2014 15:21:36 (-42.1h)
5	↑35	cheng chen		3.77384	21	Mon, 15 Sep 2014 23:29:29 (-0h)
6	↑16	quantify		3.77086	8	Mon, 15 Sep 2014 16:12:48 (-7.3h)
7	↑1	Stanislav Semenov & Co (HSE Yandex)		3.76211	68	Mon, 15 Sep 2014 20:19:03
8	↓7	Luboš Motl's team	Best physicist	3.76050	589	Mon, 15 Sep 2014 08:38:49 (-1.6h)
9	↑8	Roberto-UCIIIM		3.75864	292	Mon, 15 Sep 2014 23:44:42 (-44d)
10	↑2	Davut & Josef		3.75838	161	Mon, 15 Sep 2014 23:24:32 (-4.5d)
45	↑5	crowwork ‡	HEP meets ML award XGBoost authors Free trip to CERN	3.71885	94	Mon, 15 Sep 2014 23:45:00 (-5.1d)
782	↓149	Eckhard		3.49945	29	Mon, 15 Sep 2014 07:26:13 (-46.1h)
991	↑4	Rem.		3.20423	2	Mon, 16 Jun 2014 21:53:43 (-30.4h)

The winners

- See http://atlas.ch/news/2014/machine-learning-wins-the-higgs-challenge.html
- 1 : **Gabor Melis** (Hungary) software developer and consultant : wins 7000$.
- 2 : **Tim Salimans** (Neitherland) data science consultant: wins 4000$
- 3 : **Pierre Courtiol** (nhlx5haze) (France) ? : wins 2000$
- HEP meets ML award: (team crowwork), **Tianqi Chen** (U of Washington PhD student in Data Science) and **Tong He** (graduate student Data Science SFU). Provided **XGBoost public software** used by many participants.

https://github.com/dmlc/xgboost

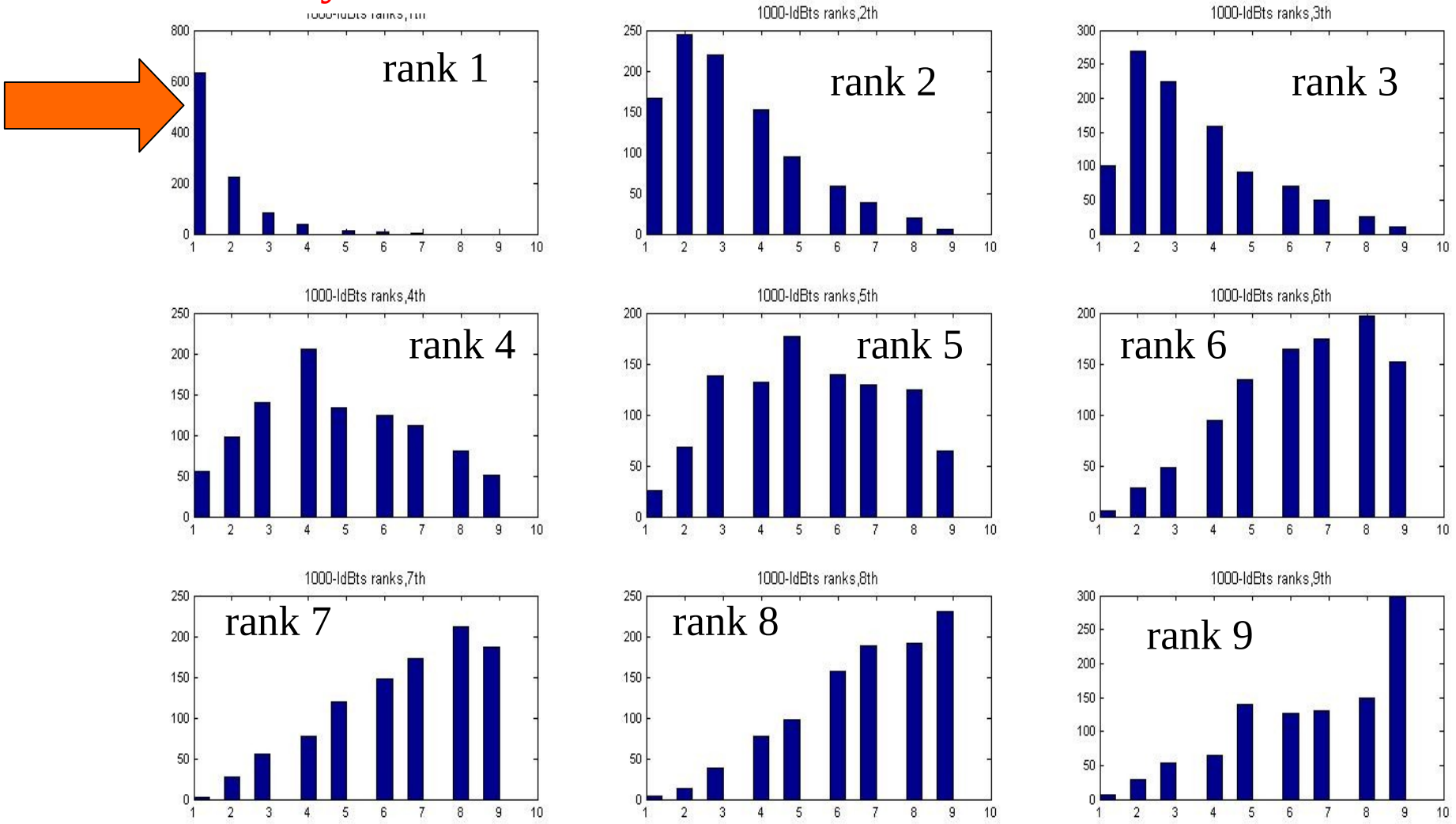
?



Rank distribution after bootstrap

Distribution of rank of participant of rank i after 1000 bootstraps of the test sample.

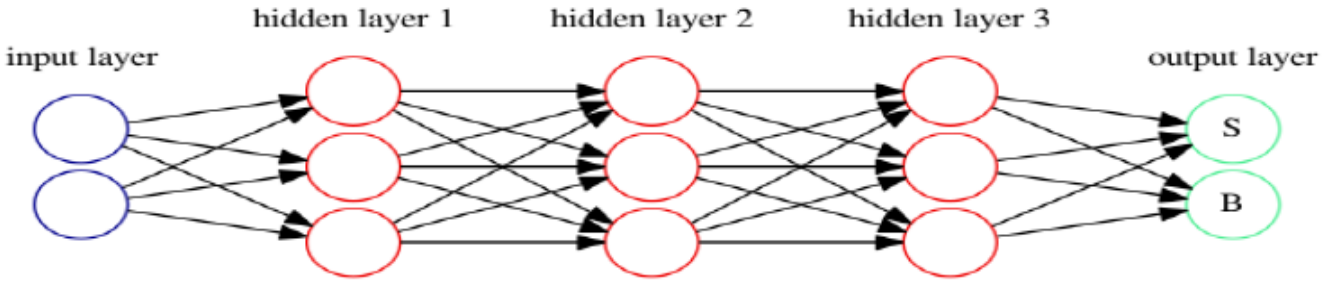
! Gabor clearly better




Who are the
XGBoost)

Deep neural network

- Hierarchical feature extraction – first build abstract objects, than find dependencies between them.
- Deep neural network (DNN)- an artificial neural network with multiple hidden layers of units between the input and output layers.
- Extra layers - composition of features from lower layers, potential of modeling complex data with fewer units than a similarly performing shallow network.



- inputs: normalized features (~30), some log transformed
- 3 hidden layers of 600 neurons each
- output layer: 2 softmax units (one for signal, one for background)
- activation function: “max channel” in groups of 3
- trained to minimize cross entropy
- regularization: dropout on hidden layers, $L_1 + L_2$ penalty and a mild sparsity constraint input weights

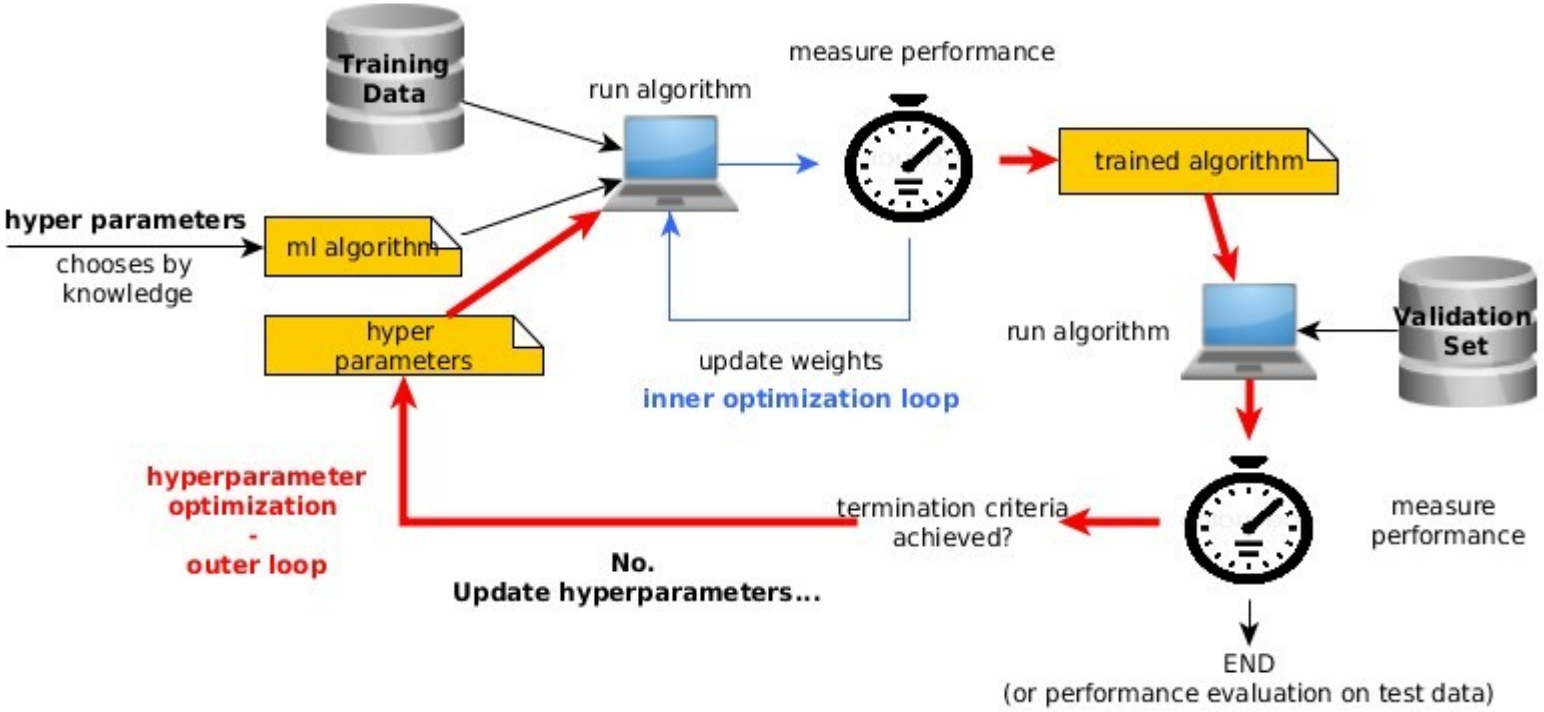
Challenge winning
Gabor's deep neural network
(from Gabor's presentation)

- CV bagged NNs: 3.83
- CV bagged xgboost: 3.79

Remark:
Few years ago some experts claimed neural networks are an obsolete tool :)

Automatic optimization of hyperparameters

- Manual optimization of NN (or any other method) is time consuming.
- Fortunately the Bayesian optimization methods can rival and surpass human domain experts in finding good hyperparameter settings.
- SMAC, SPEARMINT, TPE (and others) are doing that with great success: http://www.cs.ubc.ca/~hutter/papers/13-BayesOpt_EmpiricalFoundation.pdf



Backup

Analiza podczas praktyk studenckich

- Próbowaliśmy powtórzyć HiggsChallenge podczas praktyk studenckich.
- Udało się za pomocą TMVA (konwersja danych do formatu root) oraz pakietu XGBoost
- Optymalizacja parametrów XGBoost za pomocą programu hyperopt

A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss (2009)
TMVA 4 Package Documentation
https://tmva.sf.net

Tianqi Chen, Tong He, Bing Xu and Michael Benesty (2014)
XGBoost Package Documentation
https://github.com/dmlc/xgboost

James Bergstra, Dan Yamins, and David D. Cox (2013)
Hyperopt Package Documentation
https://github.com/hyperopt

Rozwiązania Kaggle-Higgs vs Hyperopt

Porównianie wyników uzyskanych przez nas automatycznie z wynikami z najlepszymi znalezionymi parametrami dla XGBoost.

Kto	9. K-H	M. Wolter	Nasze obliczenia
Maks. głębokość	9	10	9
Wsp. uczenia	0.01	0.089	0.059
Liczba drzew	3000	150/250/500	300
Liczba testów	-	300	100
Sub_sample	0.9	1	0.9
Maks. ROC	0.987	0.933/0.934/0.933	0.934

Sub_sample - jaka cześć danych brana jest do procesu uczenia - wprowadza pewną losowość i zapobiega przeuczaniu

Jak widać wyniki przez nas osiągnięte są znacznie słabsze. Prowadziliśmy poszukiwania w innym regionie parametrów.

One slide on Bayesian machine learning

Everything follows from two simple rules:

Sum rule: $P(x) = \sum_y P(x,y)$

Product rule: $P(x,y) = P(x)P(y|x)$

Learning:

$$P(\theta|\mathcal{D},m) = \frac{P(\mathcal{D}|\theta,m)P(\theta|m)}{P(\mathcal{D}|m)}$$

$P(\mathcal{D}|\theta,m)$	likelihood of parameters θ in model m
$P(\theta|m)$	prior probability of θ
$P(\theta|\mathcal{D},m)$	posterior of θ given data $\mathcal{D}$

Prediction:

$$P(x|\mathcal{D},m) = \int P(x|\theta,\mathcal{D},m)P(\theta|\mathcal{D},m)d\theta$$

Model Comparison:

$$P(m|\mathcal{D}) = \frac{P(\mathcal{D}|m)P(m)}{P(\mathcal{D})}$$

Bayesowskie sieci neuronowe

- Powinniśmy liczyć średnią ze wszystkich możliwych sieci neuronowych...
- Ponieważ sieć neuronowa jest funkcją nieliniową, można posłużyć się rozwinięciem w szereg wokół zestawu parametrów dającym sieć o najmniejszej funkcji straty, czyli takiej, jaką otrzymalibyśmy trenując klasyczną sieć neuronową [Bishop].
- Użycie metod Monte Carlo. Musimy wygenerować zbiór punktów w przestrzeni wag według pewnej gęstości prawdopodobieństwa. Stosowanym rozwiązaniem jest ich generacja z użyciem symulacji Monte Carlo posługującej się łańcuchami Markowa (ang. Markov Chain Monte Carlo, MCMC) [O'Neil].

Bayesowskie sieci neuronowe

Każda sieć opisana przez wektor parametrów **w.**

Dla danych treningowych T= {y,x}, gęstość prawdopodobieństwa w punkcie **w** dana jest przez:

$$p(w,T)=\frac{p(T,w)\,p(w)}{p(T)}$$

równanie Bayesa

p(w) – prawdopodobieństwo a priori, musi być wybrane wcześniej

- Odpowiedzią jest średnia po wszystkich sieciach neuronowych (wartościach **w**)**:**

$$\bar{y}(x)=\int f(x,w)\,p(w,T)\,dw \qquad f(x,w)-neural\ network$$

- Obliczanie średniej: próbkowanie z użyciem łańcuchów Markowa.
- **Zalety:**
  - Otrzymujemy błąd estymowanej funkcji,
  - Zwiększona odporność na przetrenowanie i fluktuacje.