

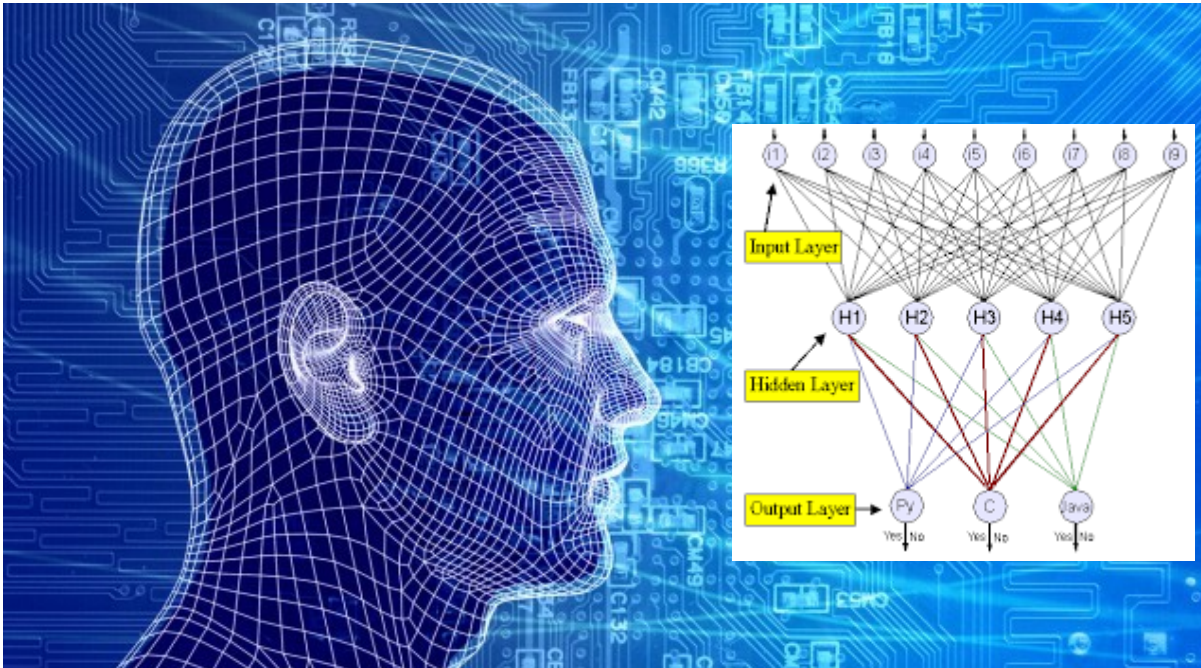
Machine learning

Lecture 4

Marcin Wolter

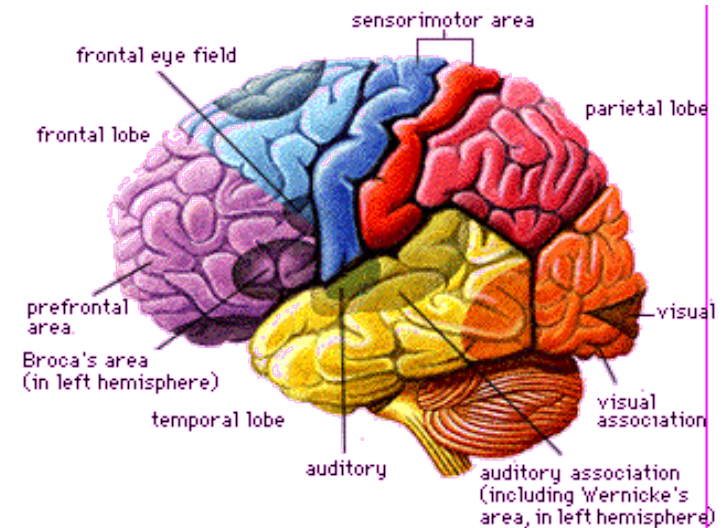
IFJ PAN

20 December 2019



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 brain is working.



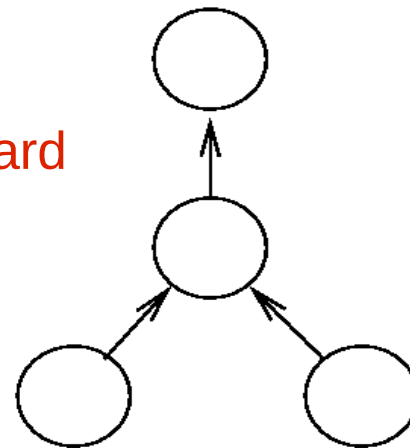
History

- 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-organization in the brain;
- 1982 Kohonen, Self-Organizing Maps
- ...
- 1986 **backpropagation of errors, many application!**
-
- 2010 Deep Neural Networks – great progress in AI!!!!!!**

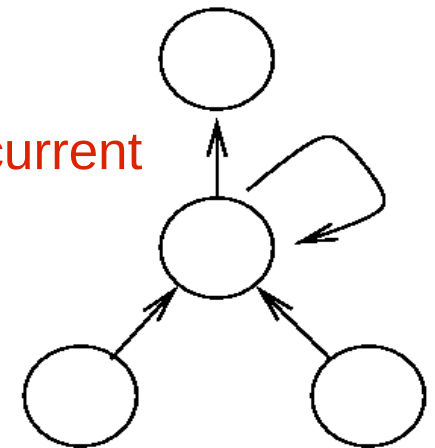
What an Artificial Neural Network is?

- **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



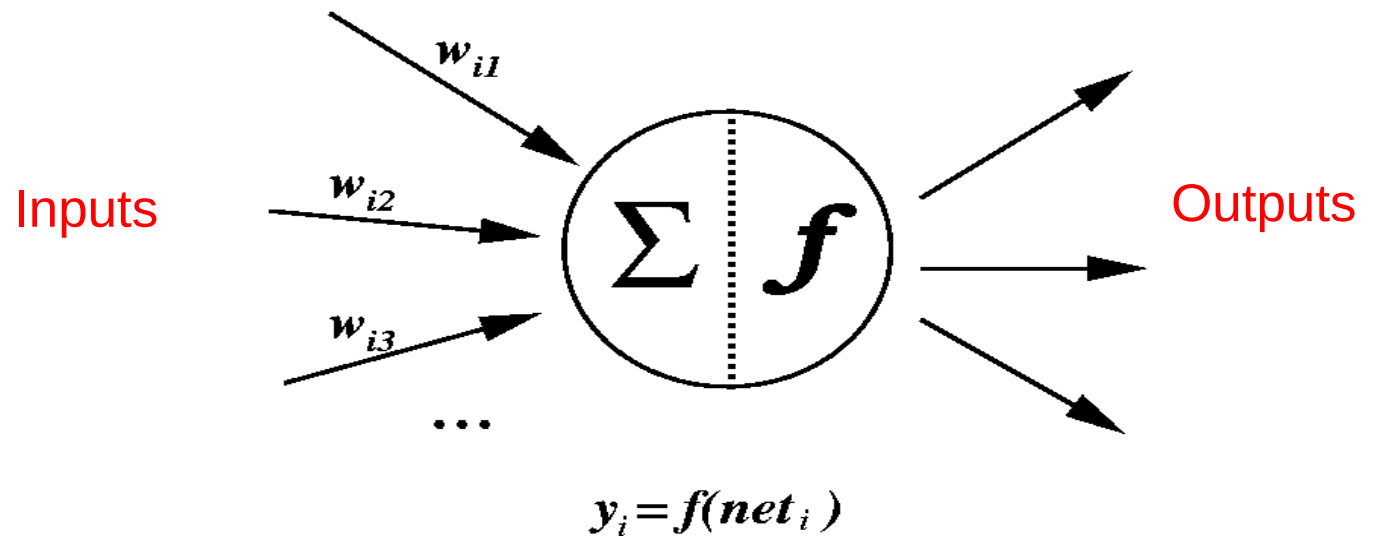
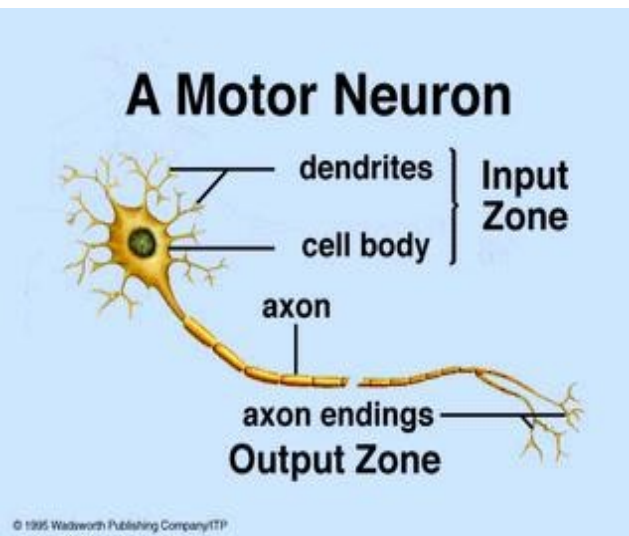


What are they used for?

- Expert systems
- Pattern recognition
- Predictions (meteorology, stock market...)

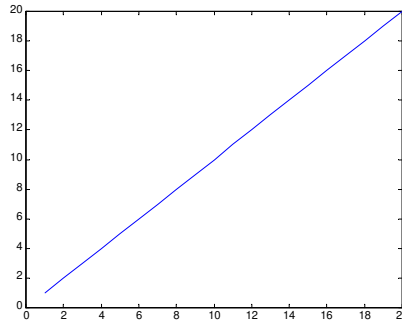
Neuron – the basic element

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



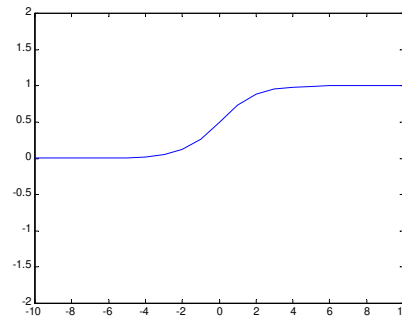
- Function f is called the **activation function**

Typical activation functions



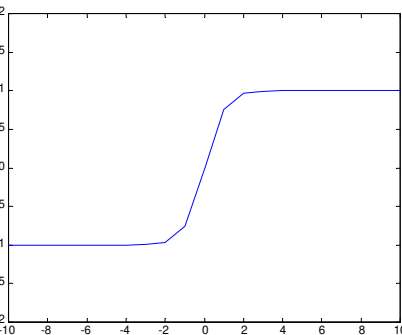
Linear (the whole network is called “linear”)

$$y = x$$



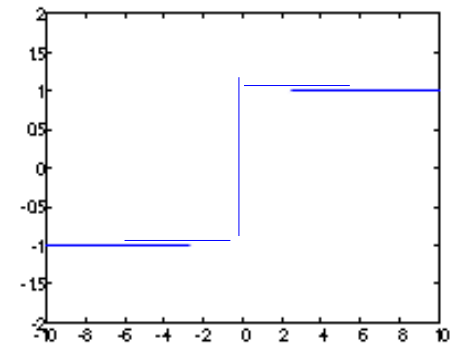
Logistic

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

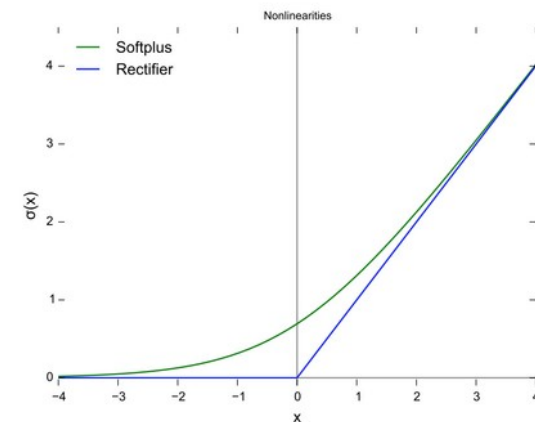


Hyperbolic tangent

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



Step

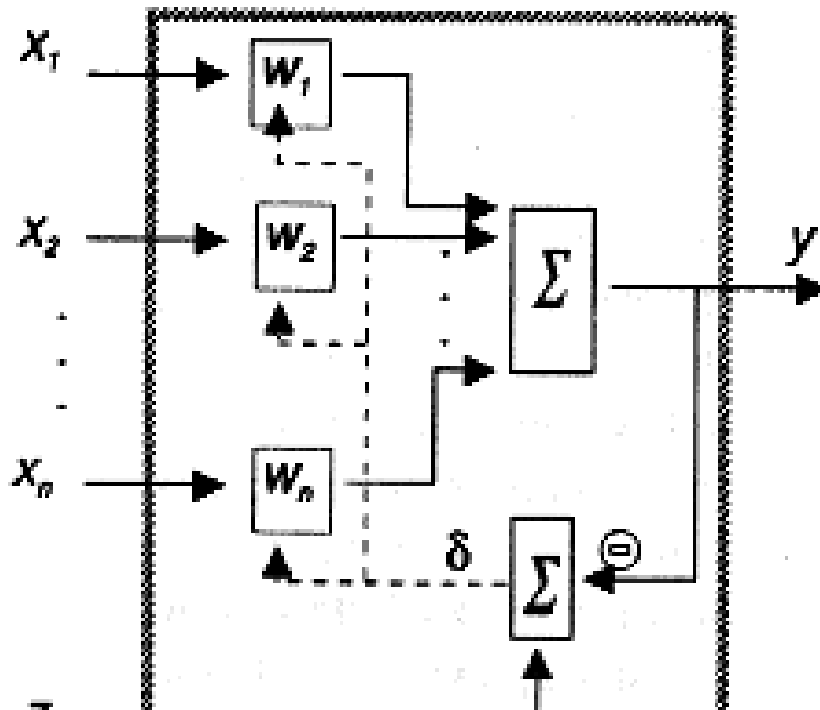


Rectified Linear (Relu), popular for **Deep Neural Networks**

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:

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

- New set of weights:

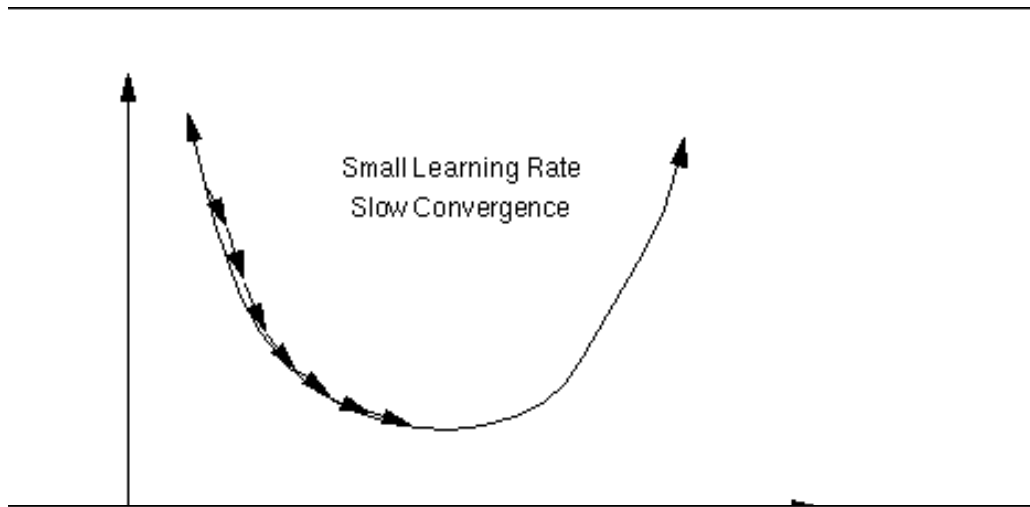
$$\delta = z - y$$

- η - learning speed

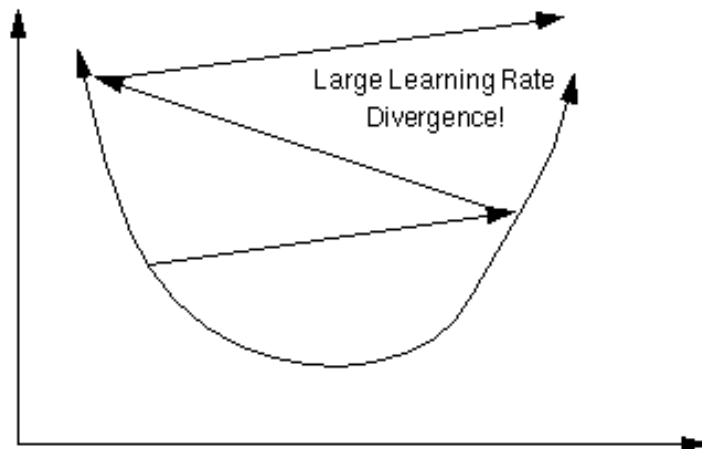
$$W' = W + \eta \cdot \delta \cdot X$$

Speed of learning

- To small

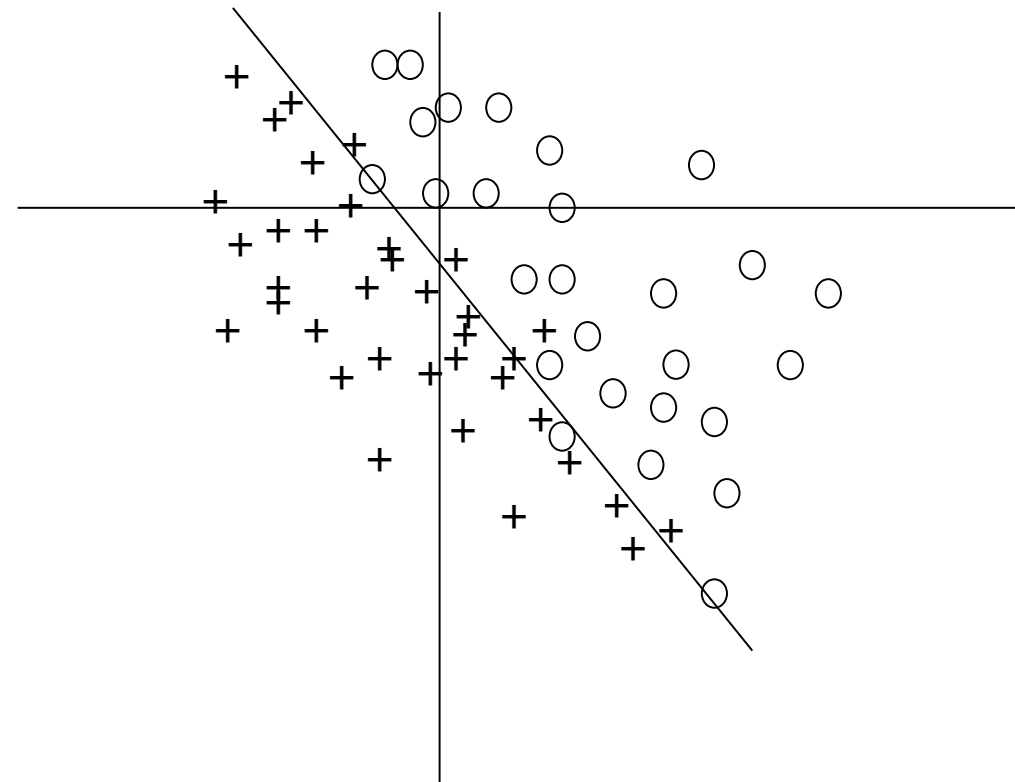
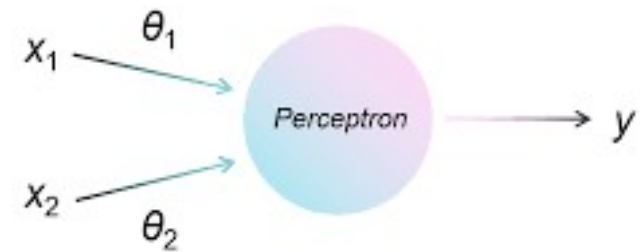


- To big



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.

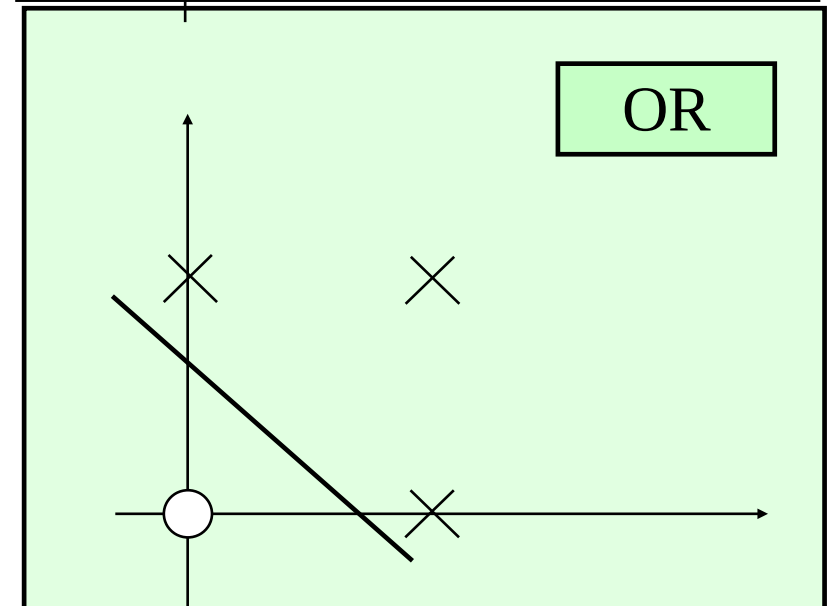
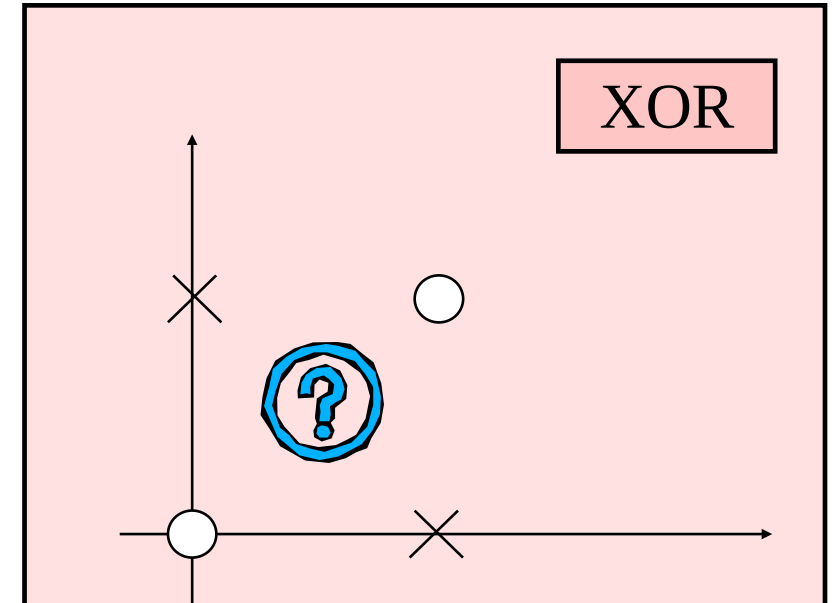
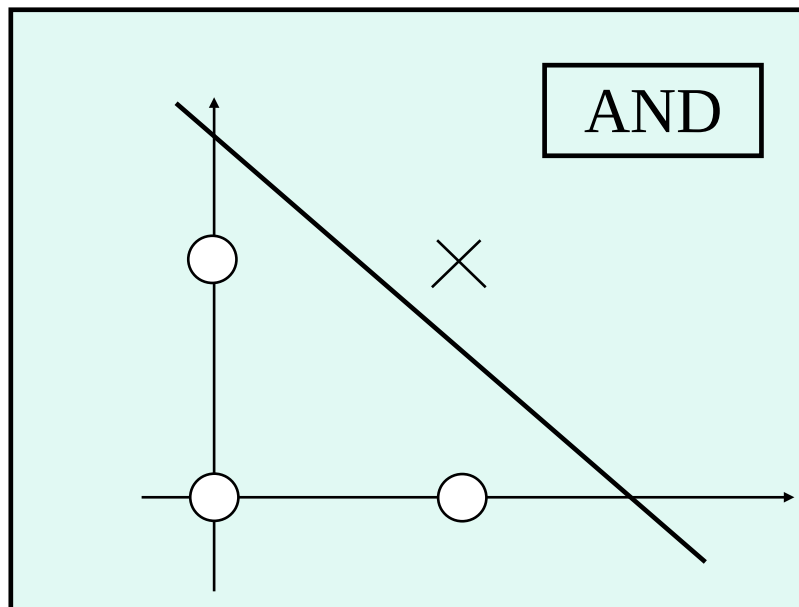


What the perceptron can not do?

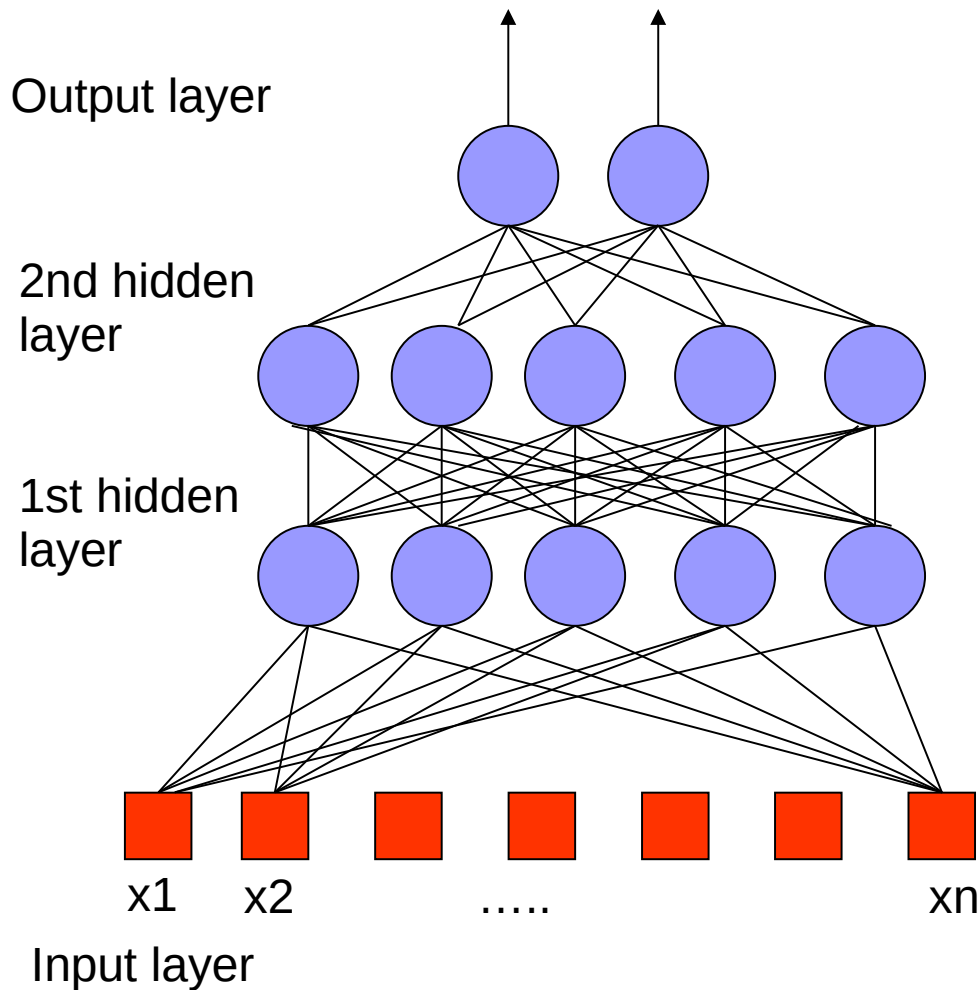


- A single perceptron can't separate the linearly not separable classes, for example the XOR function.
- The discovery of these limitations (1969) stopped the development of Neural Networks for some time.

<https://playground.tensorflow.org>
Nice NN demonstrator



So, maybe a network of perceptrons?

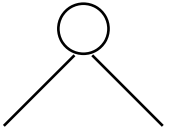
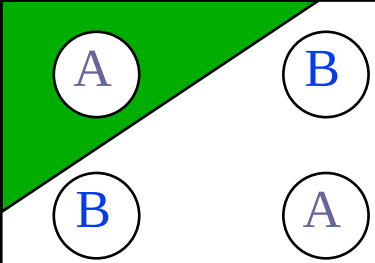
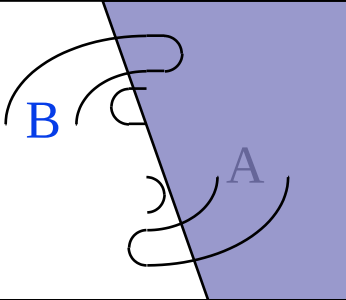
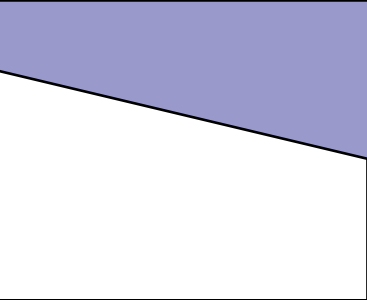
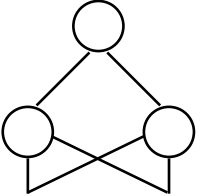
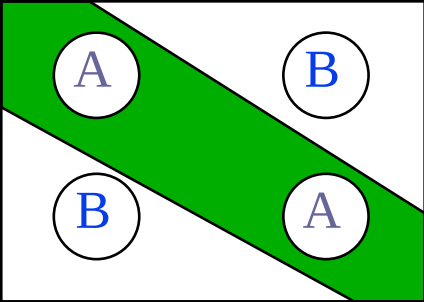
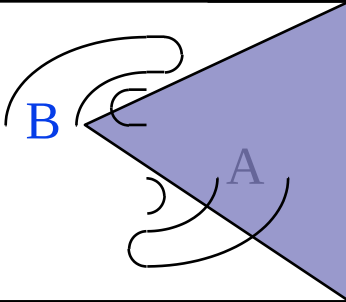
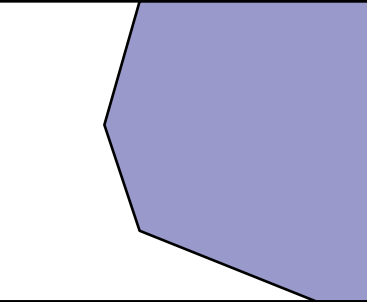
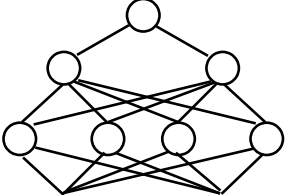
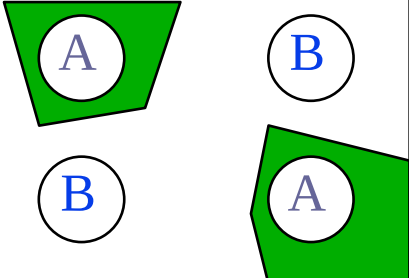
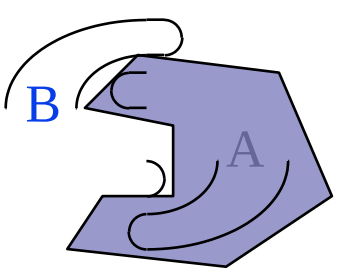
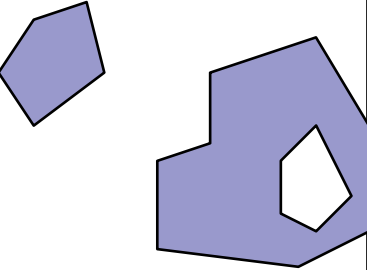


- Feed forward network – the information propagates from input to output.
- The net is the sum of many activation functions (in general non-linear)
- A network complicated enough can reproduce any function.

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 |
|---|---|--|---|---|
| <p>Single-Layer</p>  | <p>Half Plane Bounded By Hyperplane</p> |  |  |  |
| <p>Two-Layer</p>  | <p>Convex Open Or Closed Regions</p> |  |  |  |
| <p>Three-Layer</p>  | <p>Arbitrary (Complexity Limited by No. of Nodes)</p> |  |  |  |

How to train a multilayer network?

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

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

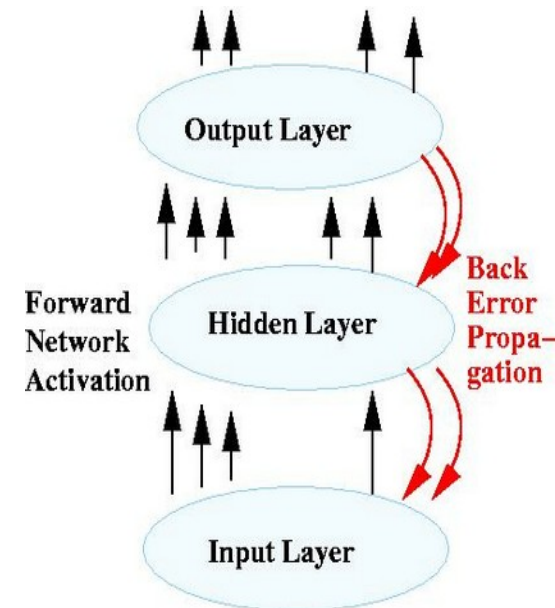
- Problem – how to correct the weights in the deeper layer, while comparing only outputs on the last layer?
- This problem stopped the development of Neural Networks for 30 years, until 80-ties.
- Solution - the **backpropagation** method. Errors $\delta = t - n(x, \omega)$ are propagated backward through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

Typical training procedure

- Two data samples: for training and for tests.
- $\chi^2 = \sum (z - y)^2$ 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:

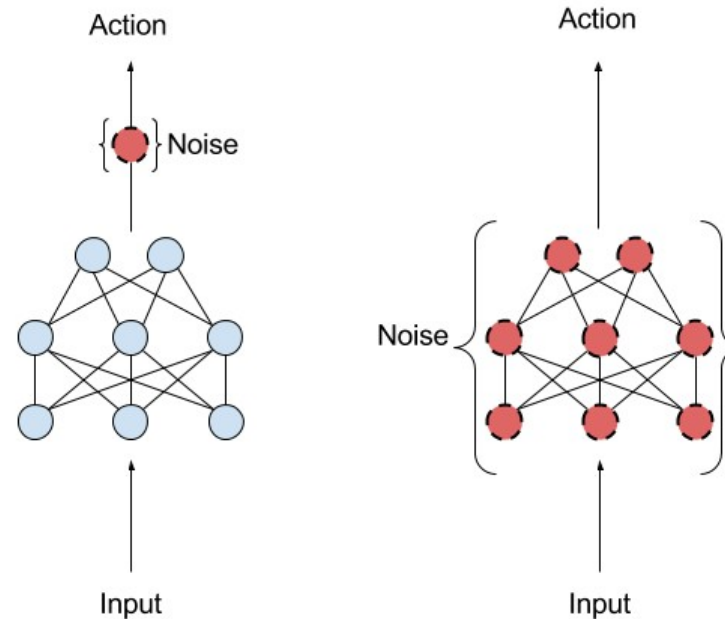
$$dw_{ij} = \rho x_i (t_j - y_j),$$

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 $\chi^2 = \sum (z-y)^2$ 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).



Action-Space-Noise (left) and Parameter-Space-Noise (right)

https://matthiasplappert.com/publications/2017_Plappert_Master-thesis.pdf



Minimization algorithms in sklearn

- Stochastic Gradient Descent (SGD) - updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation, i.e.

$$w \leftarrow w - \eta \left(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w} \right)$$

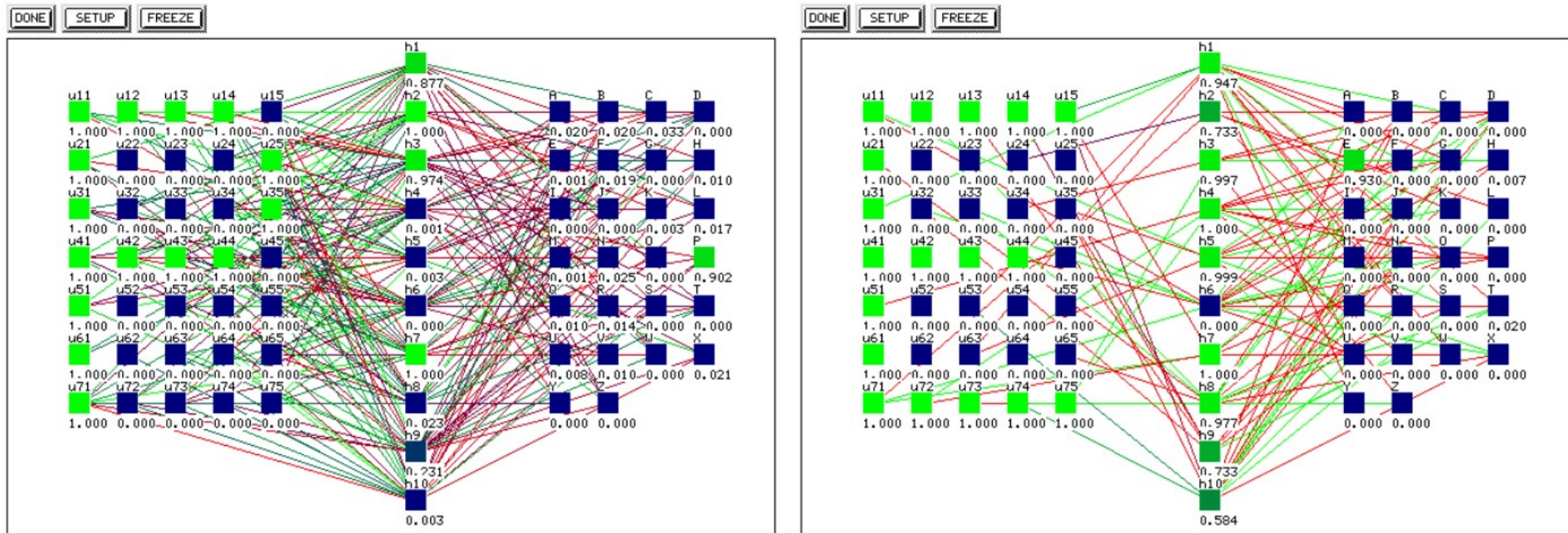
where η is the learning rate which controls the step-size in the parameter space search.

- Adam - similar to SGD, but it can automatically adjust the amount to update parameters based on adaptive estimates of lower-order moments.
- L-BFGS - approximates the Hessian matrix which represents the second-order partial derivative of a function. Further it approximates the inverse of the Hessian matrix to perform parameter updates.

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 pruned using the Optimal Brain Surgeon algorithm. The old package SNNS is used.



Network regularization

- Both MLPRegressor and MLPClassifier use parameter alpha for regularization (L2 regularization) term which helps in avoiding overfitting by penalizing weights with large magnitudes.
- Example with different regularizations:
https://github.com/marcinwolter/MachineLearnin2019/blob/master/simple_classifier_comparison.ipynb



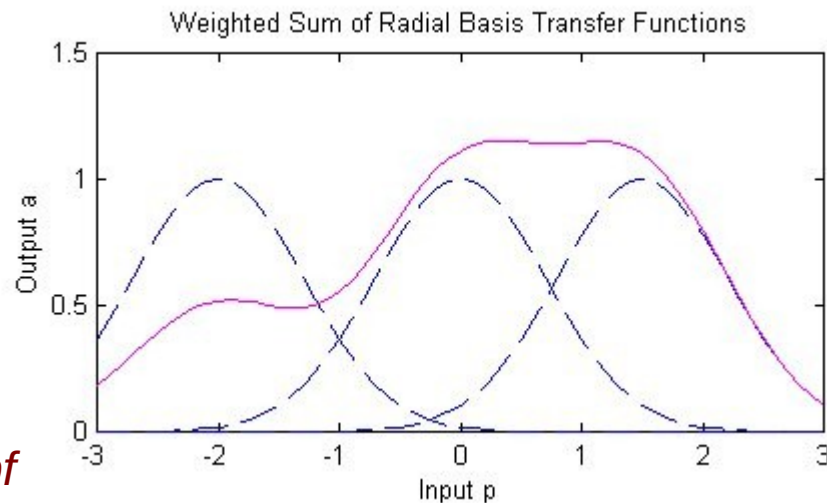
Neural network examples

- Simple comparison of many classifiers
- https://github.com/marcinwolter/MachineLearnin2019/blob/master/simple_classifier_comparison.ipynb
- Neural network for hand-written digits classification
- https://github.com/marcinwolter/MachineLearnin2019/blob/master/plot_digits_classif_mlp.ipynb
- Visualization of MLP weights
- https://github.com/marcinwolter/MachineLearnin2019/blob/master/plot_mnist_filters.ipynb

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:

$$f_i(x) = f_i(\|x - c\|) \text{ - a radial base function.}$$



Applet rbf

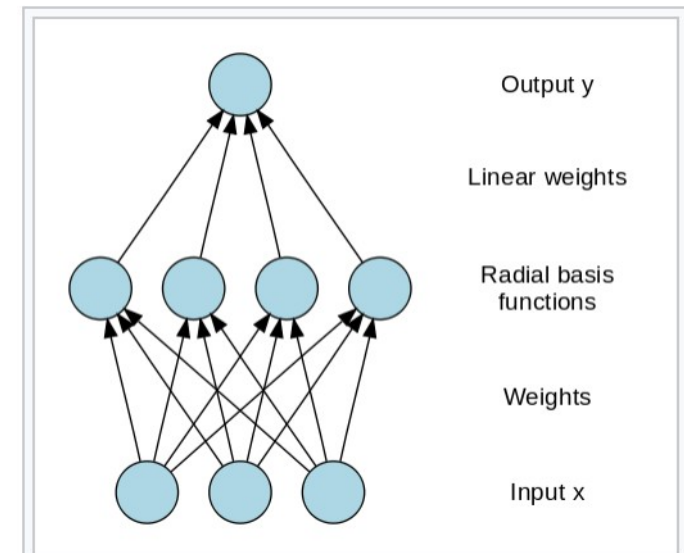


Figure 1: Architecture of a radial basis function network. An input vector x is used as input to all radial basis functions, each with different parameters. The output of the network is a linear combination of the outputs from radial basis functions.

RBF functions

1. Gaussian Functions:

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

width parameter $\sigma > 0$

7. Cubic Function:

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

2. Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^{1/2}$$

parameter $\sigma > 0$

8. Linear Function:

$$\phi(r) = r$$

3. Generalized Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^\beta$$

parameters $\sigma > 0, 1 > \beta > 0$

4. Inverse Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^{-1/2}$$

parameter $\sigma > 0$

5. Generalized Inverse Multi-Quadric Functions:

$$\phi(r) = (r^2 + \sigma^2)^{-\alpha}$$

parameters $\sigma > 0, \alpha > 0$

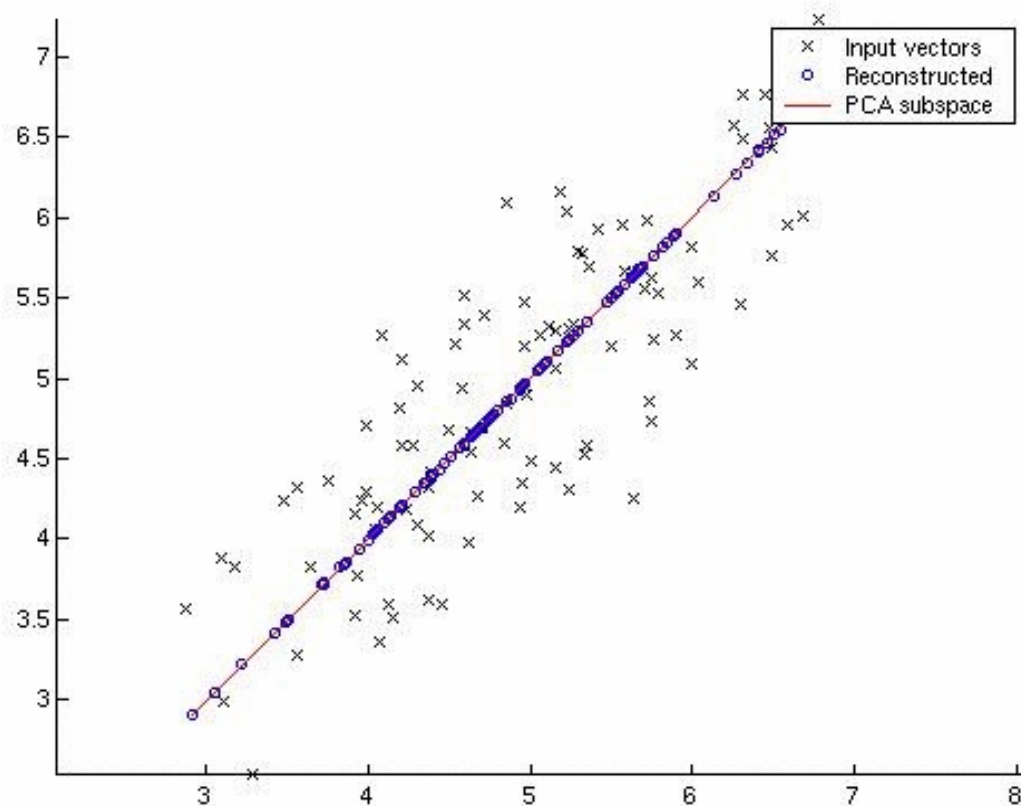
6. Thin Plate Spline Function:

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

Something else...

Non-linear PCA

(Principal Component Analysis)

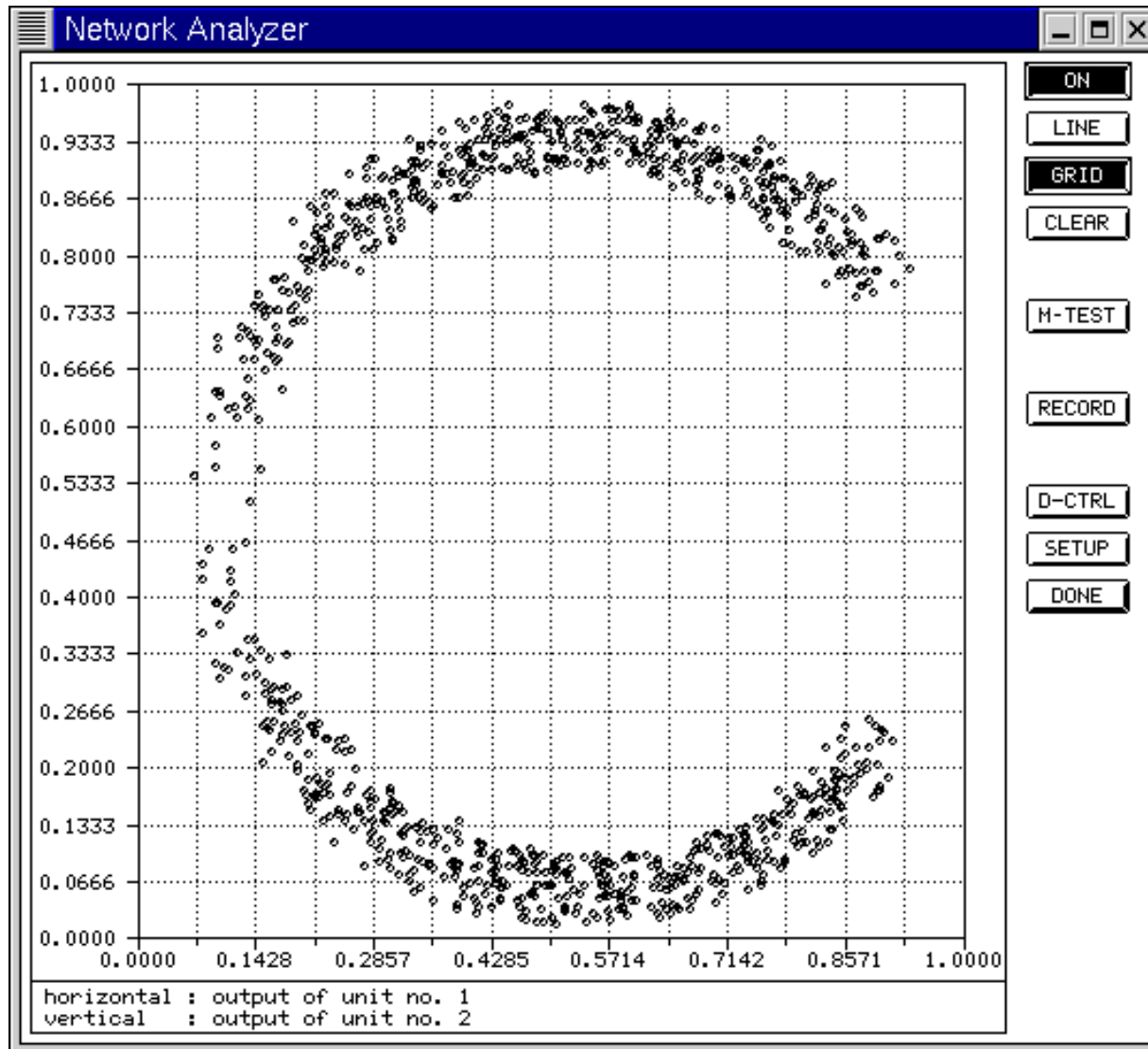


Linear PCA:

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



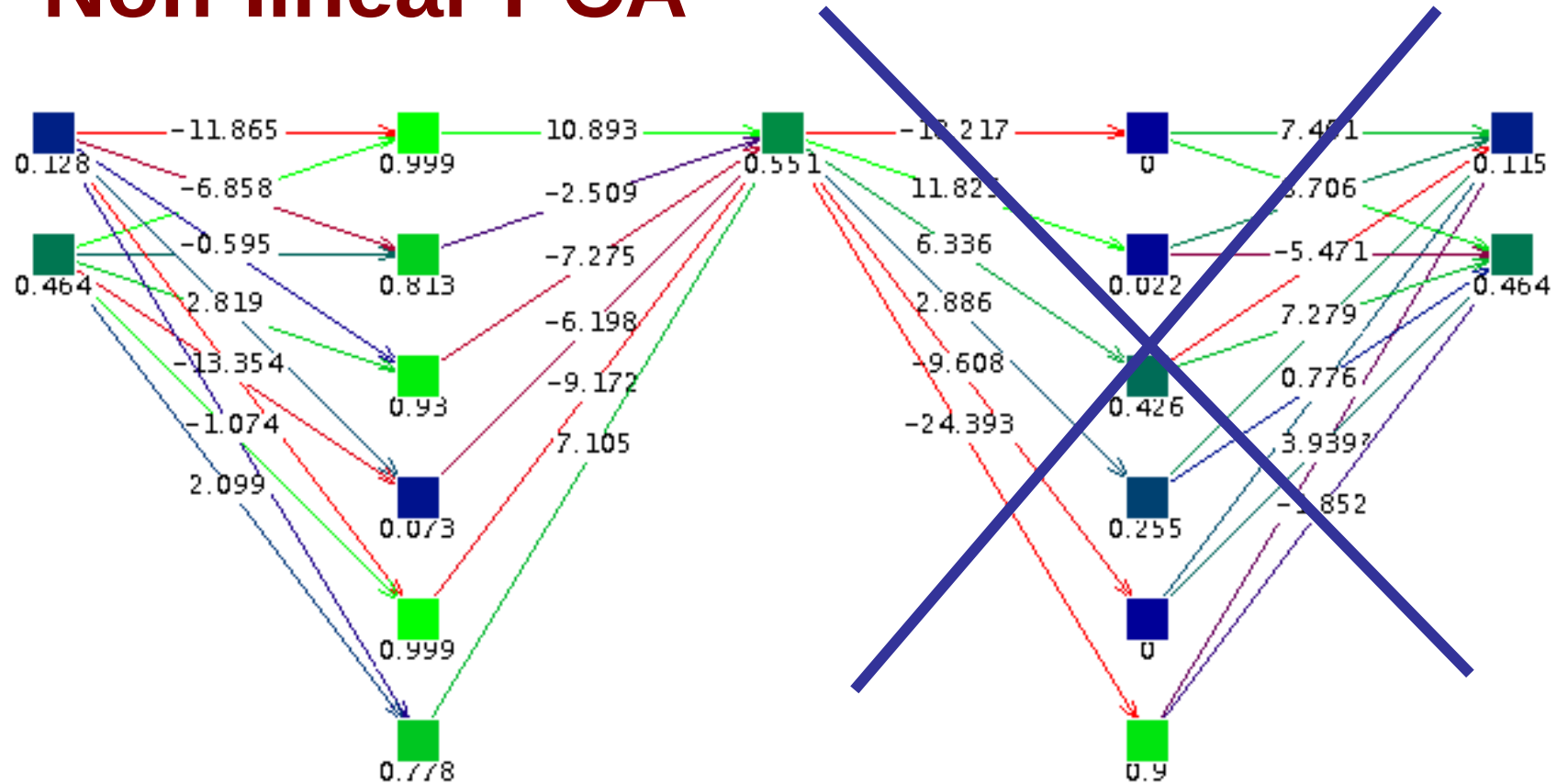
What to do with a non-linear example?



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

AUTOENCODER

Non-linear PCA

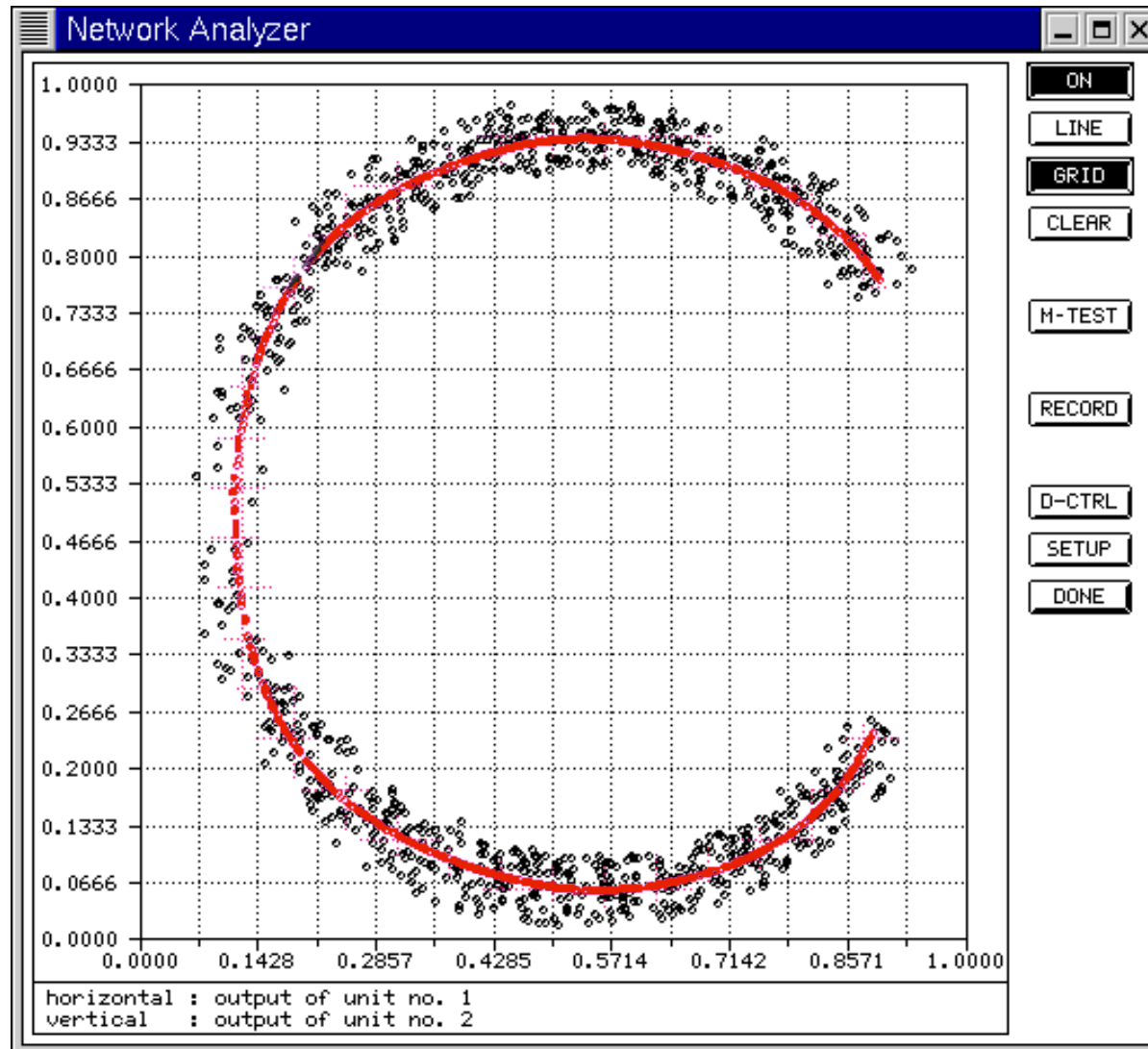


The network is trained by giving the same vectors on input and output. Then it is cut by half.

This is called an AUTOENCODER

Result – transformation into 1 dimension

- Non-linear transformation.



Merry Christmas!!



© marketoonist.com

What are we getting this time for christmas?