Computational Data Analysis

Artificial Neural Networks and Self Organizing Maps

Lars Arvastson
Stand-in: Line Clemmensen

April 25, 2018

Todays Lecture

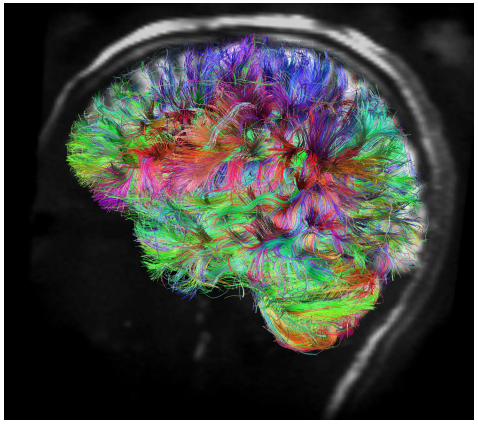
- Artificial Neural Networks
- Autoencoders
- Self Organizing Maps

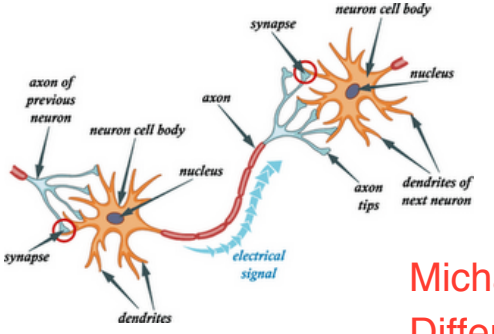
Artificial Neural Networks

- Non-linear regression
- Non-linear classification

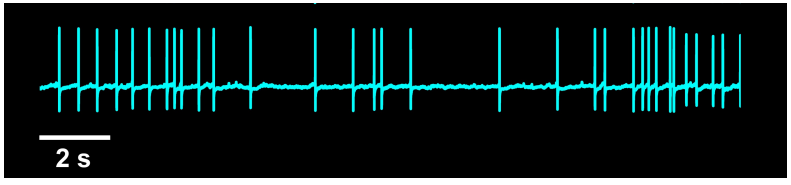
Can be linear if proper link-function is used..

Inspiration from the brain





Michael Irwin Jordan Article
Difference between AI and ML



- The brain is a **network** of neurons that communicate with electrical and chemical signals
- Built around an enormous number of **neurons**, $\sim 10^{11}$, which are relatively similar and simple.
- The typical neuron has many, $\sim 10^{3}$, inputs, dendrites, and one output, axon.
- The signaling pattern looks like a frequency modulated pulse train.

Idea

We would like to have a flexible and general function that maps input to outputs for all kinds of problems.

- Built around **simple functions** taking many inputs and one output.
- These functions are nested in a **network** such that the output from one function is the input to another function.
- For simplicity we put these functions in layers.

Terminology:

Neural networks are biological systems.

Artificial Neural Networks are mathematical functions that are inspired by the function of the human brain.

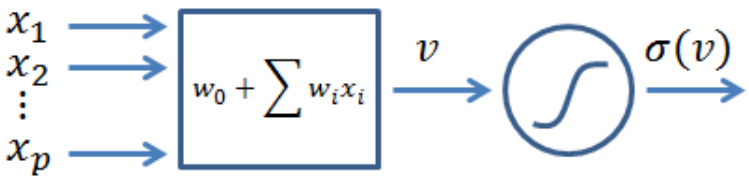
ANN → The correct math term..

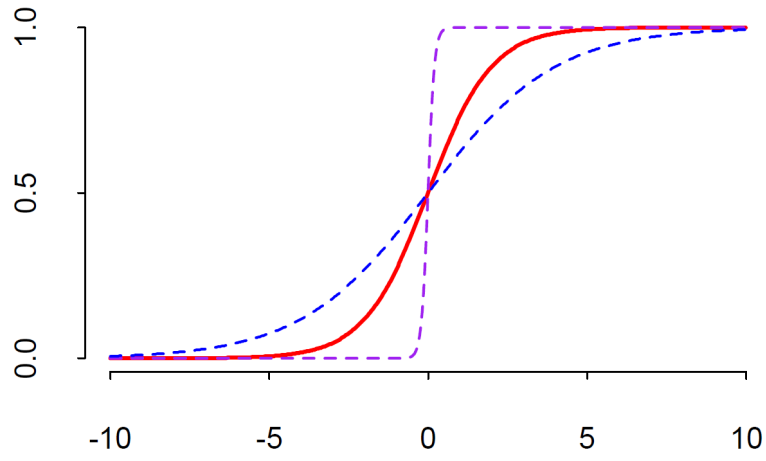
The artificial neuron

As a **building block** in artificial neural networks we introduce the artificial neuron. It takes a weighted sum of the inputs and gives a nonlinear transform as output. This **activation function** is usually chosen to be a sigmoid function,

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

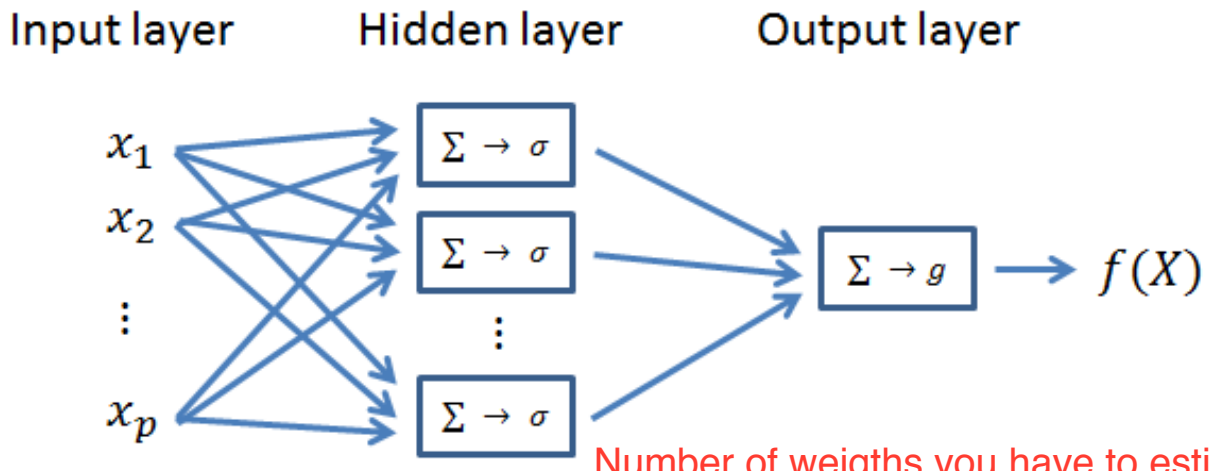
Tekst





One hidden layer feed-forward neural network

Network diagram



Number of weigths you have to estimate…

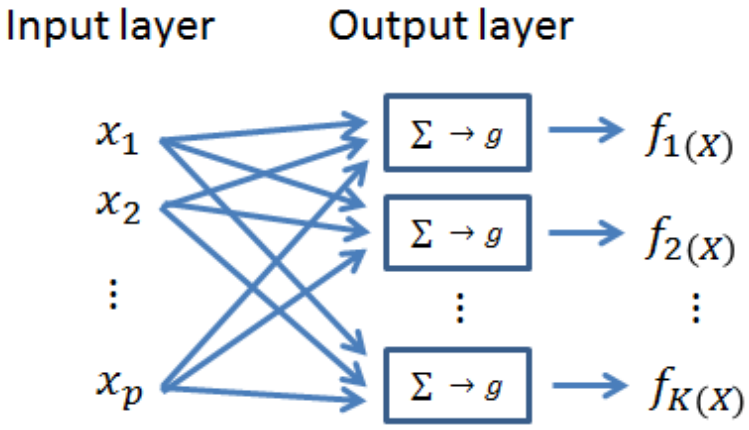
Simple:

- There is a clear connecting between figure and math foulation

Mathematical formulation

$$f(X) = g\left(\sum_{m=1}^{M} w_m^{(2)} \sigma\left(\sum_{p=1}^{P} w_{mp}^{(1)} x_p + w_{p0}^{(1)}\right) + w_0^{(2)}\right)$$

Classification



Use softmax functions in the output layer,

Scaling??

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} eT_l}$$

This is the same transform used in K-class logistic regression which produces positive estimates that sum to one.

Fitting Artificial Neural Networks

The unknown parameters are called **weights** and they should be fitted to training data by minimizing a loss function $R(W)$.

Regression

$$R(W) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2$$

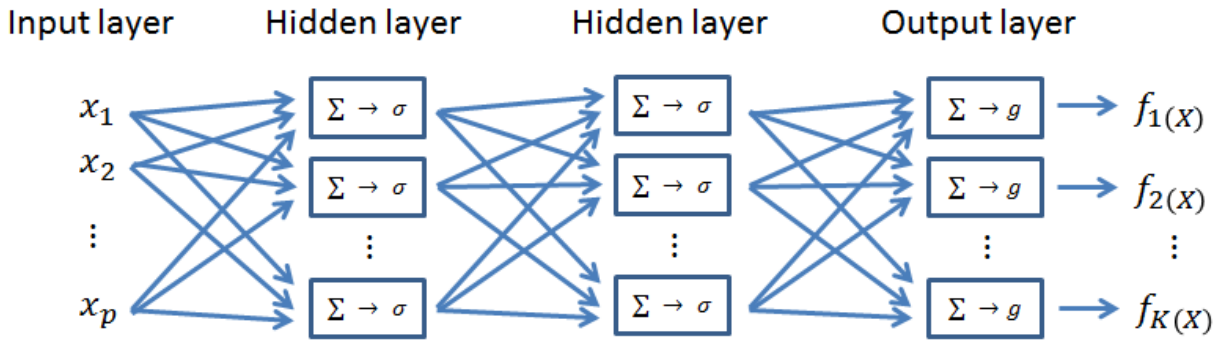
Classification

$$R(W) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log f_k(x_i)$$

This is cross-entropy and the output will be a logistic regression in the hidden layer output.

Design variables

YOU NEED THE think about the number og train sample



Bayes expansion →

Hidden layers, the number of hidden layers has to be decided. Each layer extracts features for the next layer allowing for hierarchical features.

Hidden units, more units mean more flexibility. It is usually better with too many than too few units. The extra weights can be shrunken towards zero with regularization.

Approximates any function

Theorem (Cybenko): An ANN with one hidden layer can approximate any smooth function with arbitrary precision.

However, we might need a VERY large number of hidden nodes...

Weight-space symmetry

There are several equivalent ANNs that will give the same mapping from input to output.

We can change the sign on all weights leading into a hidden layer. This is then reversed by changing sign on the weights into the next layer. For M hidden units this gives 2^M identical networks.

The hidden nodes and their weights in one layer can be permuted with identical result. This gives another $M!$ combinations.

Therefore there are a total of $M!2^M$ **equal optimas** to a (one-layer) network.

multiple Local minima!!!

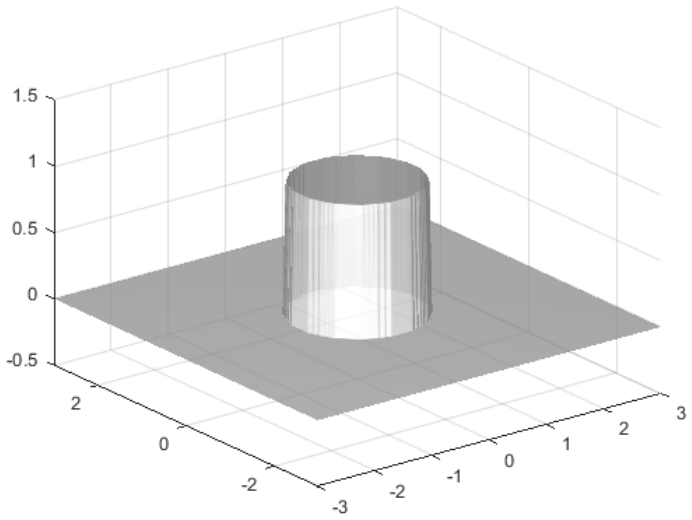
Local minima

The error function is non-convex and possesses many local minima.

- Try different **starting values** and choose the best model
- Use the **average prediction** from several networks as the final prediction.
- Use **bagging**.

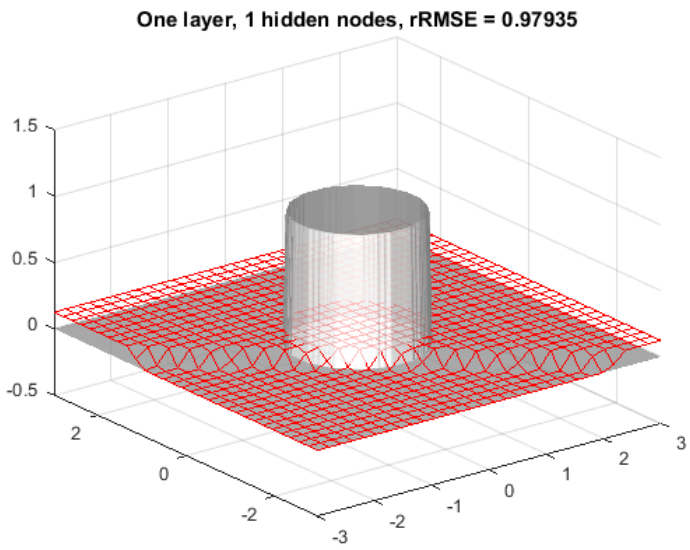
This the best and reconmanded apporach to overcome local minima..

Example with one hidden layer



We will model data to the left using an ANN with one hidden layer and increasing numbers of hidden units.

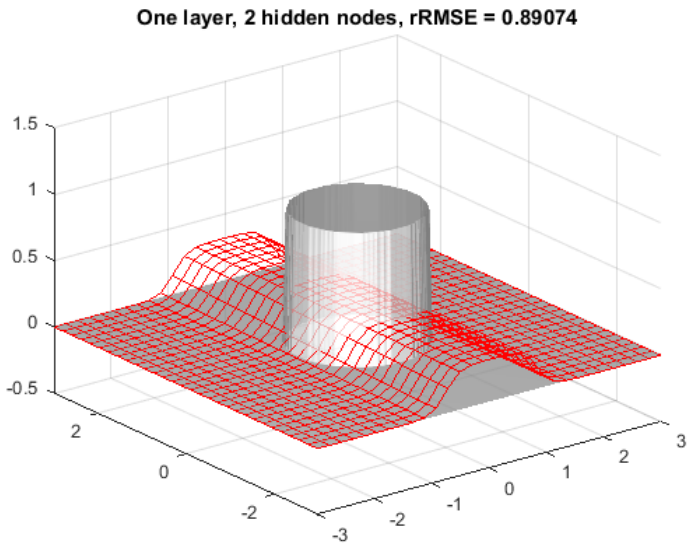
Example with one hidden layer



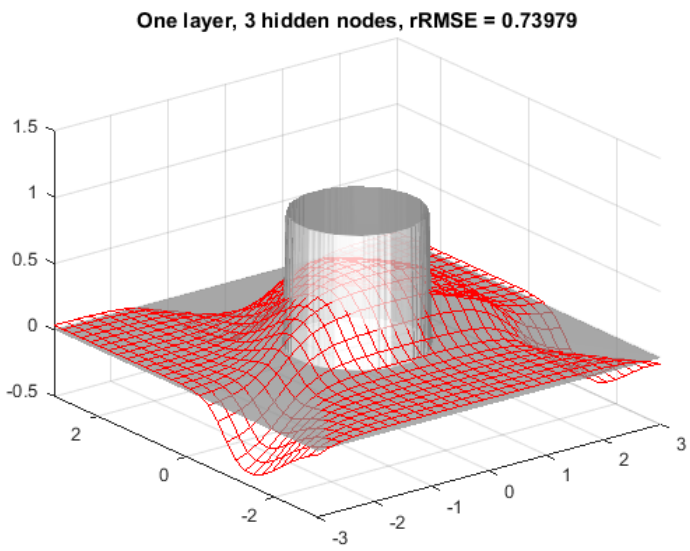
- One hidden layer with 1 node.
- Output is a sigmoid function.

Example with one hidden layer

- One hidden layer with 2 nodes.



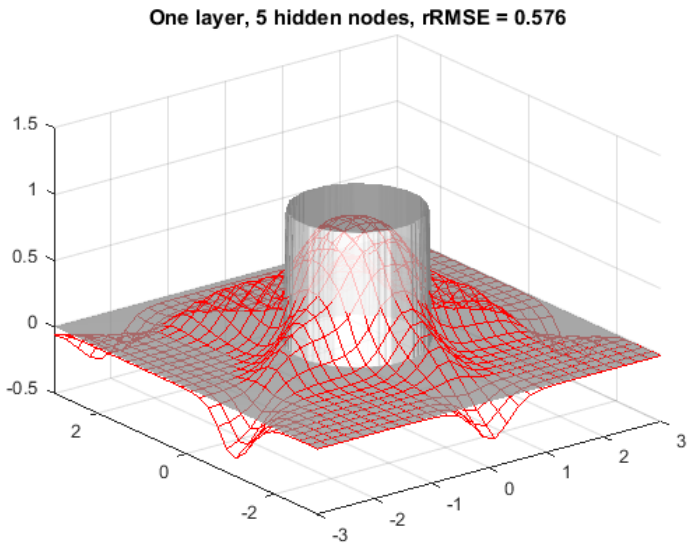
Example with one hidden layer



- One hidden layer with 3 nodes.
- We are beginning to resolve the cylinder in the middle.

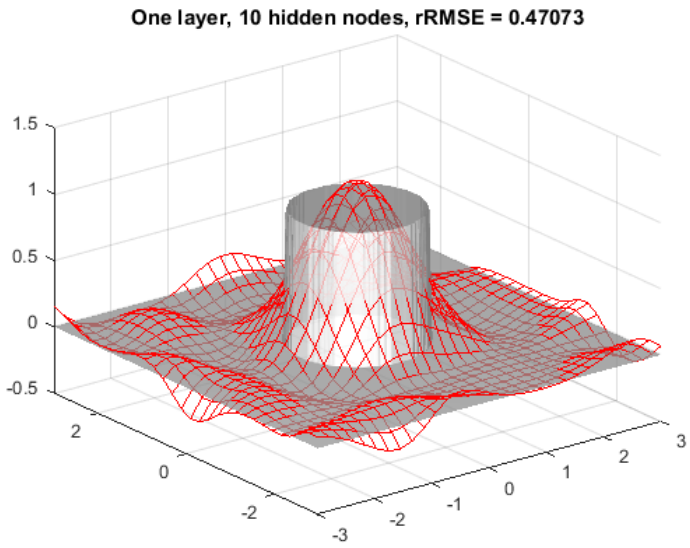
Example with one hidden layer

- One hidden layer with 5 nodes.



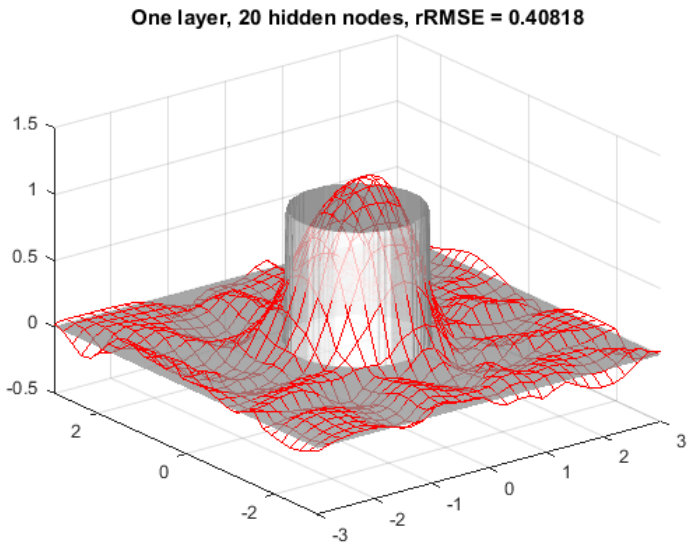
Example with one hidden layer

- One hidden layer with 10 nodes.

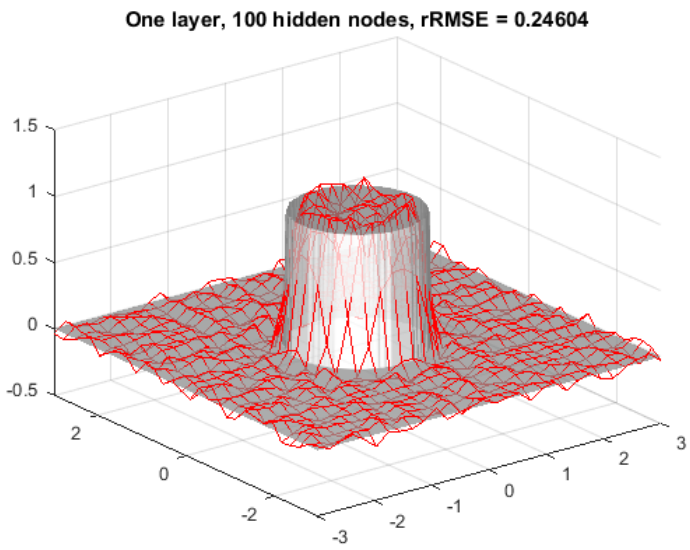


Example with one hidden layer

- One hidden layer with 20 nodes.

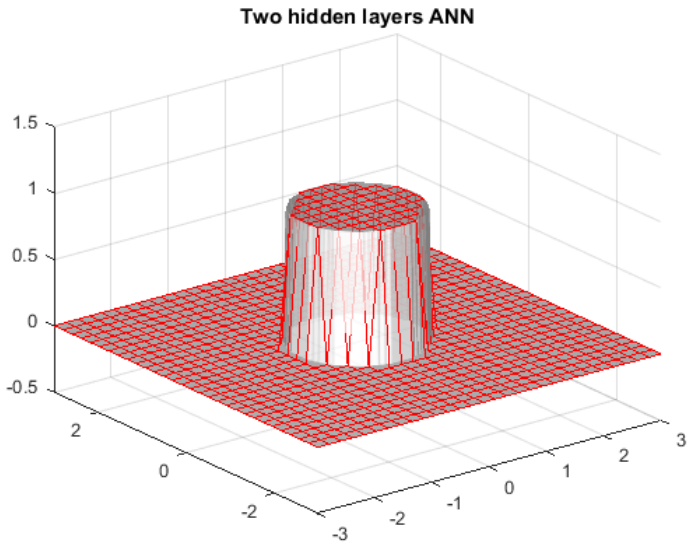


Example with one hidden layer



- One hidden layer with 100 nodes.
- The approximation is not perfect but it resembles data.

Exercise



- Use the function `ANNcylinder.m` and experiment with different **two layer networks**.
- What is the minimum number of nodes in a two layer networks that outperforms one layer with 100 nodes?

Overfitting

ANNs are very flexible models with many parameters to tune. Two techniques are used to avoid overfitting.

You use hueristisk… You have a feeling and using this feeling!!!

- Cross validation / Training and validation set
- Early stopping

Non increase in performance.. → stop

Cross validation

Cross validation and evaluation on a separate validation set can be used to compare different model orders and to tune a **regularization parameter**.

Fitting ANNs are time consuming and thus this might take a lot of time.

Early stopping

A simplified approach to regularization often applied to ANN.

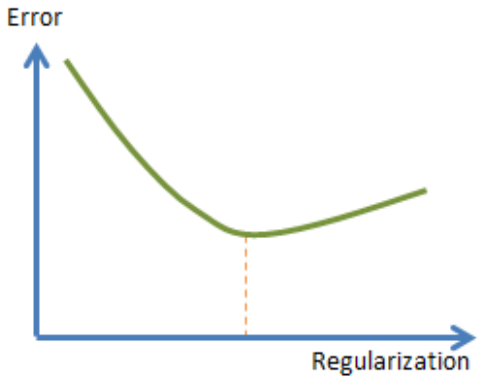
During the optimization, of the weights in the model, performance on a validation set is followed. Instead of having the optimizer finding a minimum we terminate the iteration when performance on validation data does not improve anymore. This approach **saves a lot of time**.

- We do not have to repeat the calculations for different CV folds and values on a regularization parameter.
- We do not even have to run the optimizer until convergence. We stop when the generalization error starts to increase.

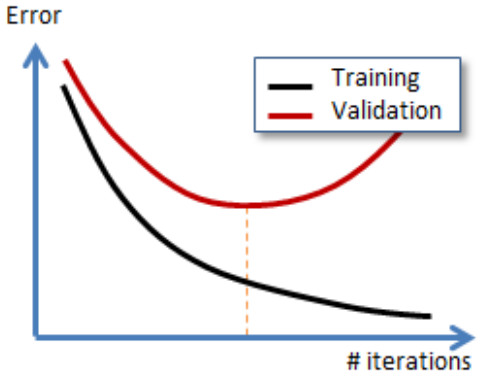
The method does not give the best possible model but the technique is very common when fitting **deep networks**. For a theoretical justification see G. Wahba "Inverse and Ill-posed Problems" 1987.

Overfitting

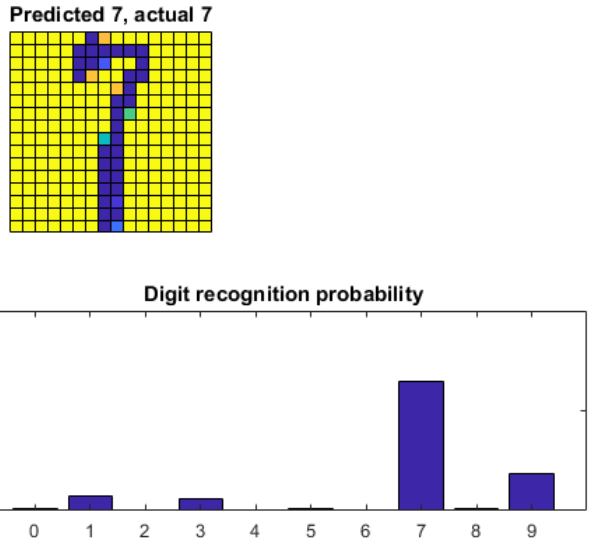
Cross validation



Early stopping



Exercise



- Try ANN classification with `ANNearlystoppingZip.m` and experiment with different models.
- Try regularized ANN with `ANNregularizationZip.m` and different penalty on the weights.

Autoencoders

Unsupervised data reduction with ANN

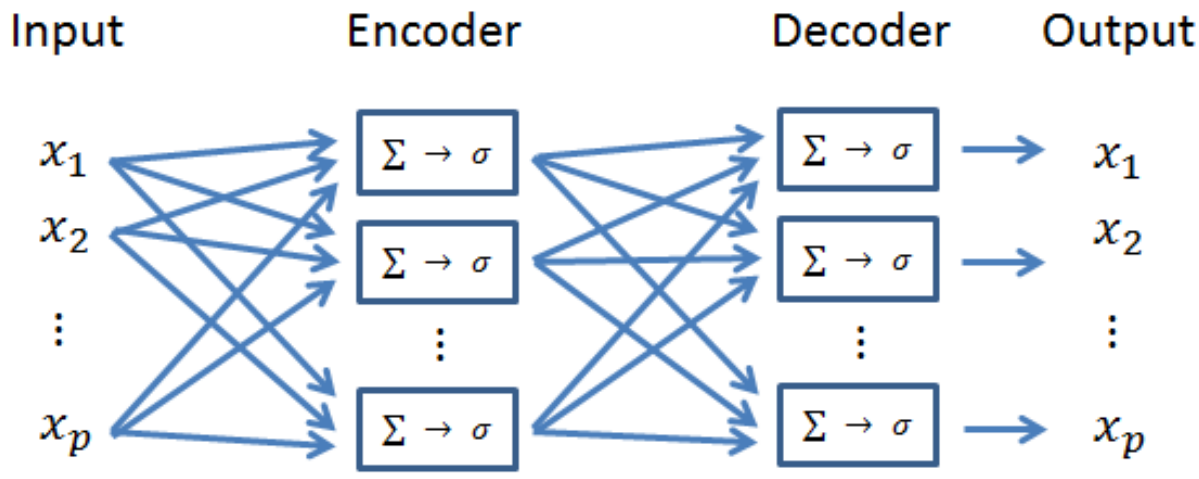
Autoencoders

Latent variables/representation and compression

Train an ANN to reproduce its input at the output layer.

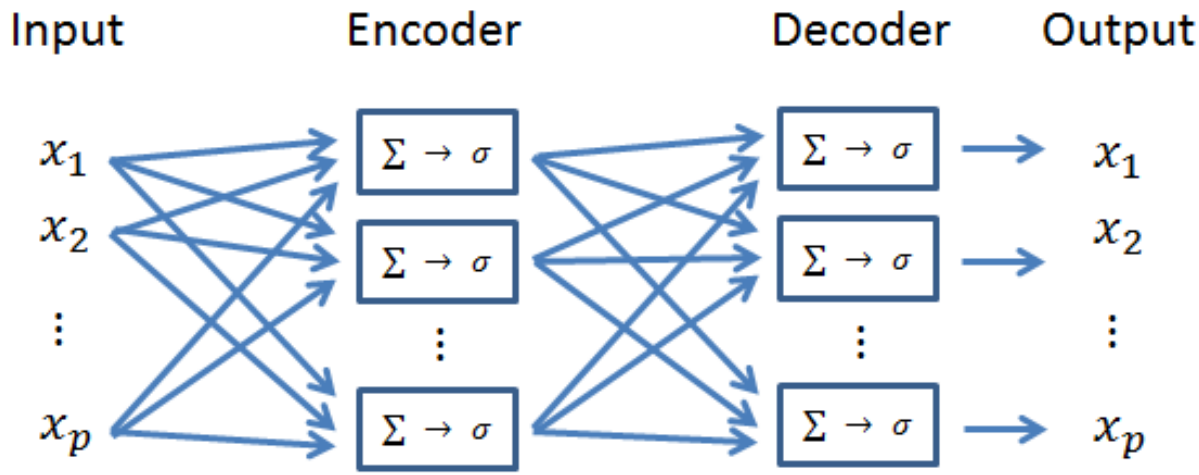
Encoder: The first layer transforms the input into a latent representation.

Decoder: The second layer transforms the latent representation into a reconstruction of the input.



Dimensionality reduction

By using fewer units in the encoder layer than input variables we obtain a dimensionality reduction.



Exercise

Try compressing the the zip data with an autoencoder. You can use `AutoencoderZip.m`

The weights in the encoder represents the pattern used in each latent variable.

Self Organizing Maps

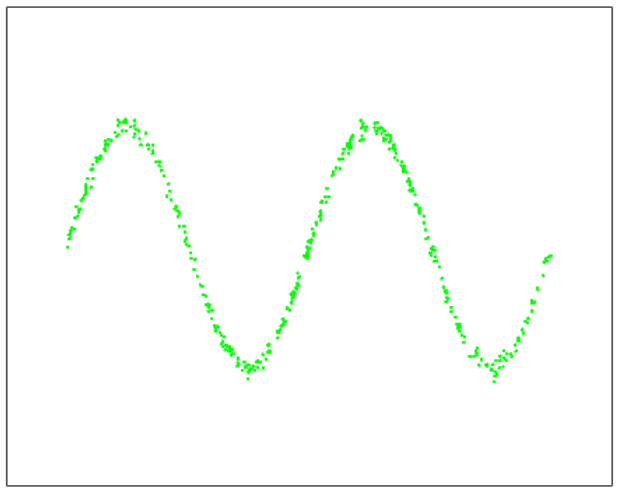
- Unsupervised clustering
- Projection of data to a low dimensional space

Background

- Observations are grouped into clusters much like K-means clustering
- The clusters have neighbors in a one or two dimensional grid.
- Neighbor clusters are enforced to lay close to each other also in feature space.
- This creates a mapping of data down to one or two dimensions.

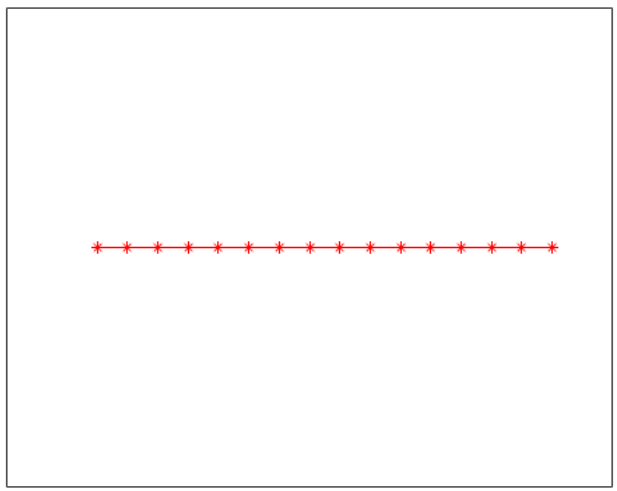
- Great for visualization
  - Exploratory data analysis
  - Overview of large amounts of text documents

Example, data on one dimensional curve



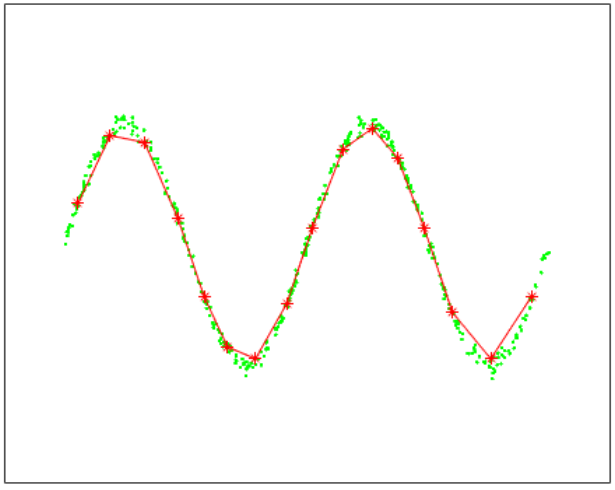
- 400 data points in a 2D-feature space
- Data are laying in a one dimensional curve (plus some noise).

Example, data on one dimensional curve



- Define a one dimensional grid
- We want to map this grid onto data
- Each node is one cluster center
- Data that are close should be clustered to the same node or nodes that are close

Example, data on one dimensional curve



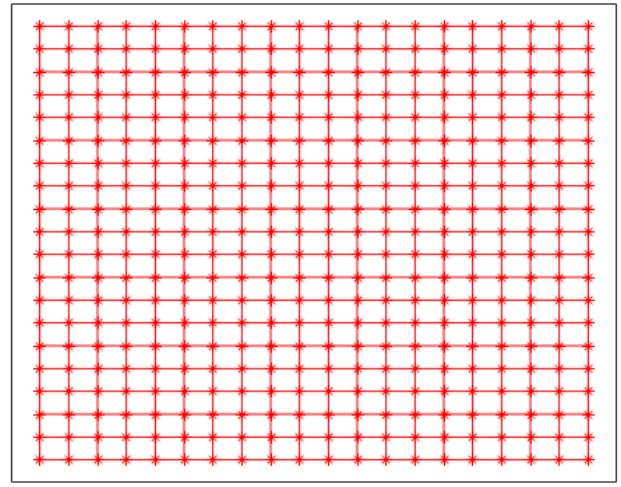
- This is the Self Organizing Map
- Nodes are plotted as cluster centers in **feature space**

Example, self organization of colors

- 10 000 randomly colored pixels.
- Matlab: `rand(10000,3)`
- We will visualize data in 2D using SOM

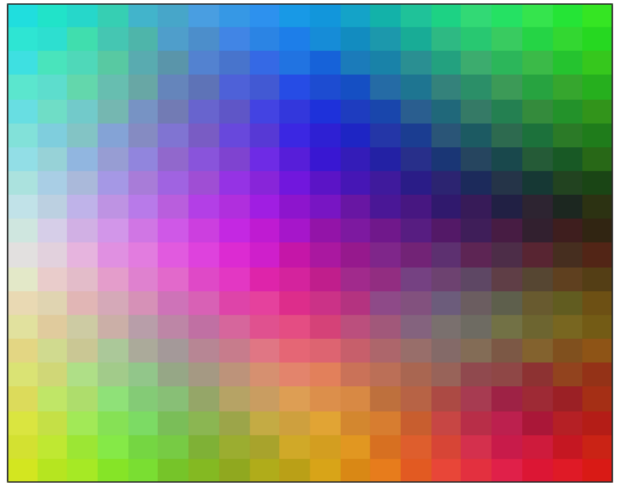
Example, self organization of colors

- Define a grid in 2D
- Train the SOM to data



Example, self organization of colors

- Data are 3 dimensional colors [r g b].
- Each node is a cluster of pixels with similar colors.
- Here we plot the colors of the cluster centers in **grid space** (2D)



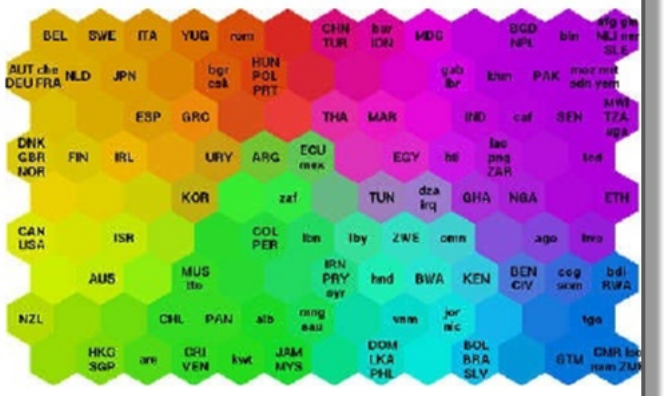
Example, world powerty map

Example Self-Organizing Maps

SOM – Result Example

World Poverty Map

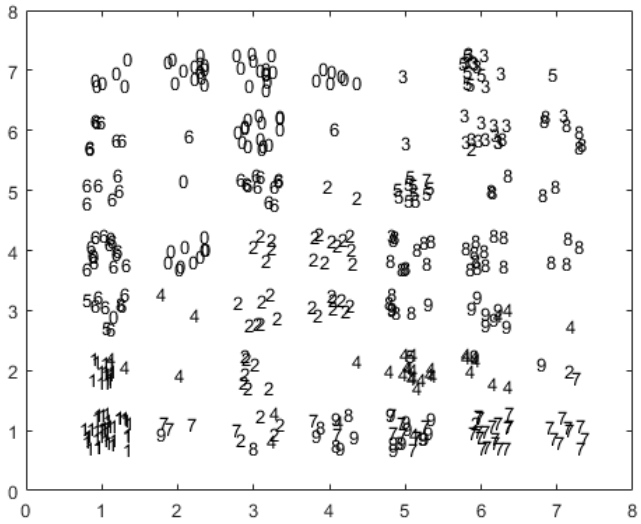
A SOM has been used to classify statistical data describing various quality-of-life factors such as state of health, nutrition, educational services etc. . **Countries with similar quality-of-life factors end up clustered together**. The countries with better quality-of-life are situated toward the upper left and the most poverty stricken countries are toward the lower right.



'Poverty map' based on 39 indicators from World Bank statistics (1992)

Exercise

Map/Grid folding → hard interpretations..



Local distances are presevered..

- Try clustering of zip data using Self Organizing Maps. You can use `SOMzip.m`
- Use two dimensional grids with different number of nodes.
- Can you interpret the result?