Artificial Neural Networks

1. Artificial Neural Networks
   - Properties
   - Applications
   - Classical Examples
   - Biological Background
2. Single Layer Networks
   - Limitations
   - Training Single Layer Networks
3. Multi Layer Networks
   - Possible Mappings
   - Backprop Algorithm
   - Practical Problems
4. Generalization

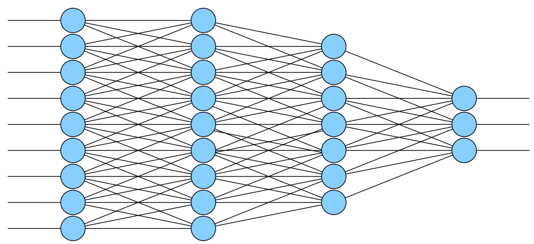
1. Artificial Neural Networks
   - Properties
   - Applications
   - Classical Examples
   - Biological Background
2. Single Layer Networks
   - Limitations
   - Training Single Layer Networks
3. Multi Layer Networks
   - Possible Mappings
   - Backprop Algorithm
   - Practical Problems
4. Generalization

Artificial Neural Networks (ANN)

- Inspired from the nervous system
- Parallel processing

We will focus on one class of ANNs:

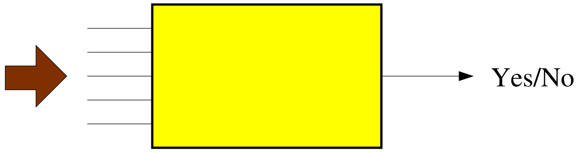
Feed-forward Layered Networks



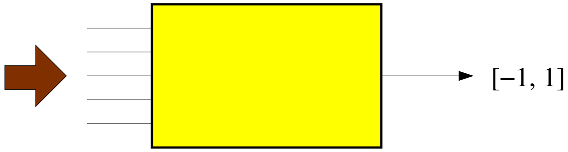
Applications

Operates like a general "Learning Box"!

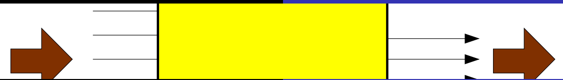
Classification



Function Approximation



Multidimensional Mapping



Classical Examples

ALVINN

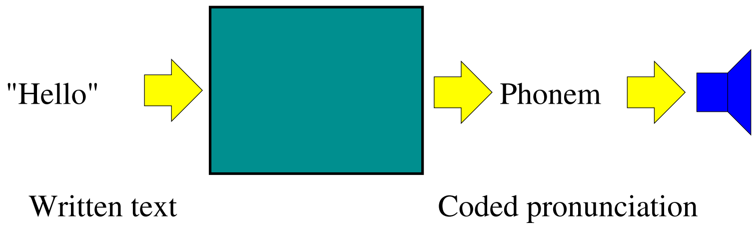
Autonomous driving

Trained to mimic the behavior of human drivers

Classical Examples

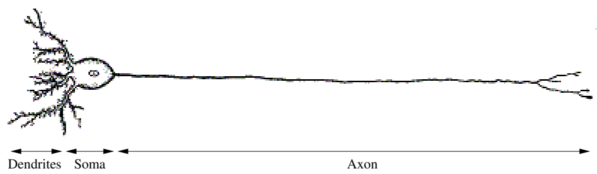
NetTalk

Speech Synthesis



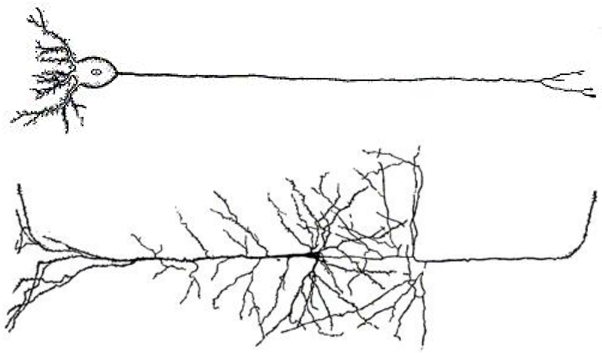
Trained using a large database of spoken text

How do real neurons (nerve cells) work?



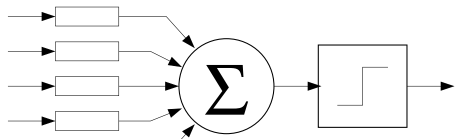
- Dendrites
  Passive reception of (chemical) signals
- Soma (Cell Body)
  Summing, Thresholding
- Axon
  Aktive pulses are transmitted to other cells

Nerve cells can vary in shape and other properties



ANN-caricatures

(simplified view of the neural information processing)



- Weighted input signals
- Summing
- Thresholded output

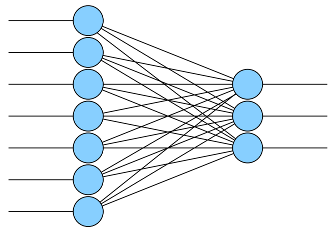
1 Artificial Neural Networks
- Properties
- Applications
- Classical Examples
- Biological Background

2 Single Layer Networks
- Limitations
- Training Single Layer Networks

3 Multi Layer Networks
- Possible Mappings
- Backprop Algorithm
- Practical Problems

4 Generalization

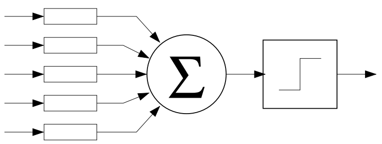
What do we mean by a *Single Layer Network*?



Each cell operates independently of the others!

It is sufficient to understand what one cell can compute

What can a single "cell" compute?

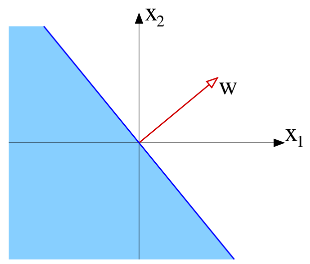


$\vec{x}$ Input in vector format
$\vec{w}$ Weights in vector format
o Output

$$o = \text{sign}\left(\sum_i x_i w_i\right)$$

$$o = \text{sign}\left(\sum_i x_i w_i\right)$$

Geometrical interpretation



Separating hyper plane
Linear separability

Learning in ANNs

What does learning mean here?

The network structure is normally fixed

Learning means finding the best weights w_i

Two good algorithms for single layer networks:

- Perceptron Learning
- Delta Rule

Perceptron Learning

- Incremental learning
- Weights only change when the output is wrong
- Update rule: $w_i \leftarrow w_i + \eta(t - o)x_i$
- Always converges if the problem is solvable

Delta Rule (LMS-rule)

- Incremental learning
- Weights always change
- $w_i \leftarrow w_i + \eta(t - \vec{w}^T\vec{x})x_i$
- Converges only in the mean
- Will find an optimal solution even if the problem can not be fully solved

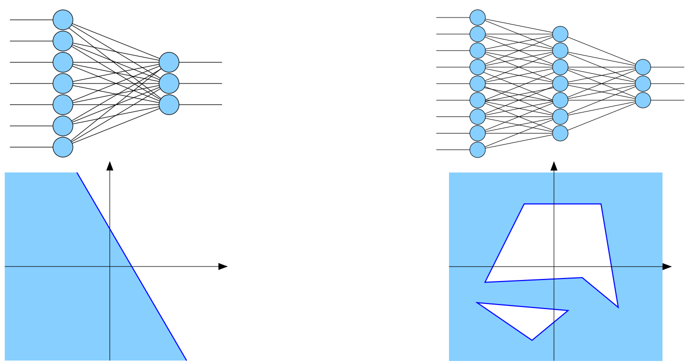
1 Artificial Neural Networks
- Properties
- Applications
- Classical Examples
- Biological Background

2 Single Layer Networks
- Limitations
- Training Single Layer Networks

3 Multi Layer Networks
- Possible Mappings
- Backprop Algorithm
- Practical Problems

4 Generalization

What is the point of having multiple layers?



A two layer network can implement arbitrary decision surfaces
...provided we have *enough hidden units*

Will it be even better with more layers?

- Two layers can describe any classification
- Two layers can approximate any "continuous" function
- Three layers can sometimes do the same thing more efficiently
- More than three layers are rarely used

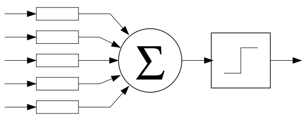
How can we train a multi layer network?

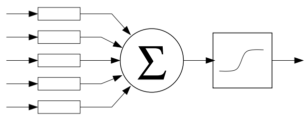
Neither perceptron learning, nor the delta rule can be used

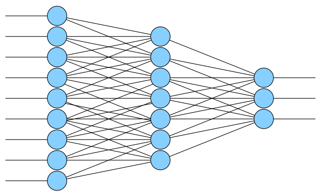
Fundamental problem:

When the network gives the wrong answer
there is no information on in which direction
the weights need to change to improve the result

Fundamental trick:
Use threshold-like, but continuous functions







Basic idéa:

Minimize the error (E) as a function of all weights ($\vec{w}$)

1. Compute the direction in weight space where the error increases the most $\mathrm{grad}_{\vec{w}}(E)$
2. Change the weights in the opposite direction

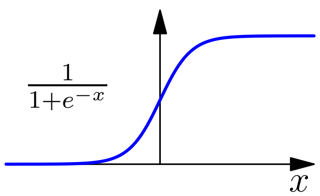
$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

Normally one can use the error from each example separately

$$E = \frac{1}{2} \sum_{k \in \mathrm{Out}} (t_k - o_k)^2$$

A common "threshold-like function" is

$$\varphi(y) = \frac{1}{1 + e^{-y}}$$



The gradient can be expressed as a function of a *local generalized error* δ

$$\frac{\partial E}{\partial w_{ji}} = -\delta_i x_j \qquad w_{ji} \leftarrow w_{ji} + \eta \delta_i x_j$$

Output layer:

$$\delta_k = o_k \cdot (1 - o_k) \cdot (t_k - o_k)$$

Hidden layers:

$$\delta_h = o_h \cdot (1 - o_h) \cdot \sum_{k \in \mathrm{Out}} w_{kh} \delta_k$$

The error δ propagates backwards through the layers
Error backpropagation (BackProp)

Things to think about when using BackProp

- Sloooow
  Normal to require thousands of iterations through the dataset
- Gradient following
  Risk of getting stuck in local minima
- Many parameters
  - Step size η
  - Number of layers
  - Number of hidden units
  - Input and output representation
  - Initial weights

1 Artificial Neural Networks
- Properties
- Applications
- Classical Examples
- Biological Background

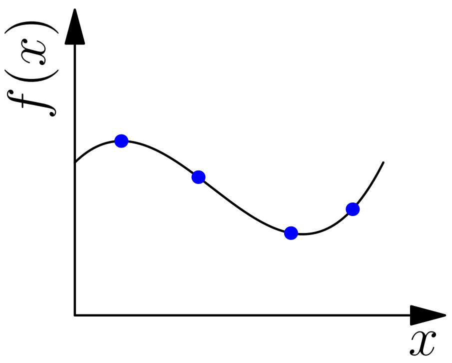
2 Single Layer Networks
- Limitations
- Training Single Layer Networks

3 Multi Layer Networks
- Possible Mappings
- Backprop Algorithm
- Practical Problems

4 Generalization

Generalization

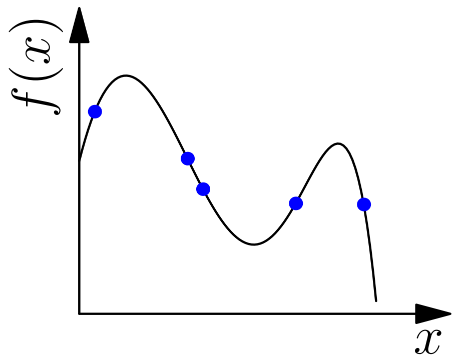
The net normally *interpolates* smoothly between the data points



Results in good generalization

Risk for overfitting!

If the network has too many degrees of freedom (weights), the risk increases that learning will find a "strange" solution



Limiting the number of hidden units tends to improve generalization