Artificial Neural Networks

Felix Dietrich

Agenda

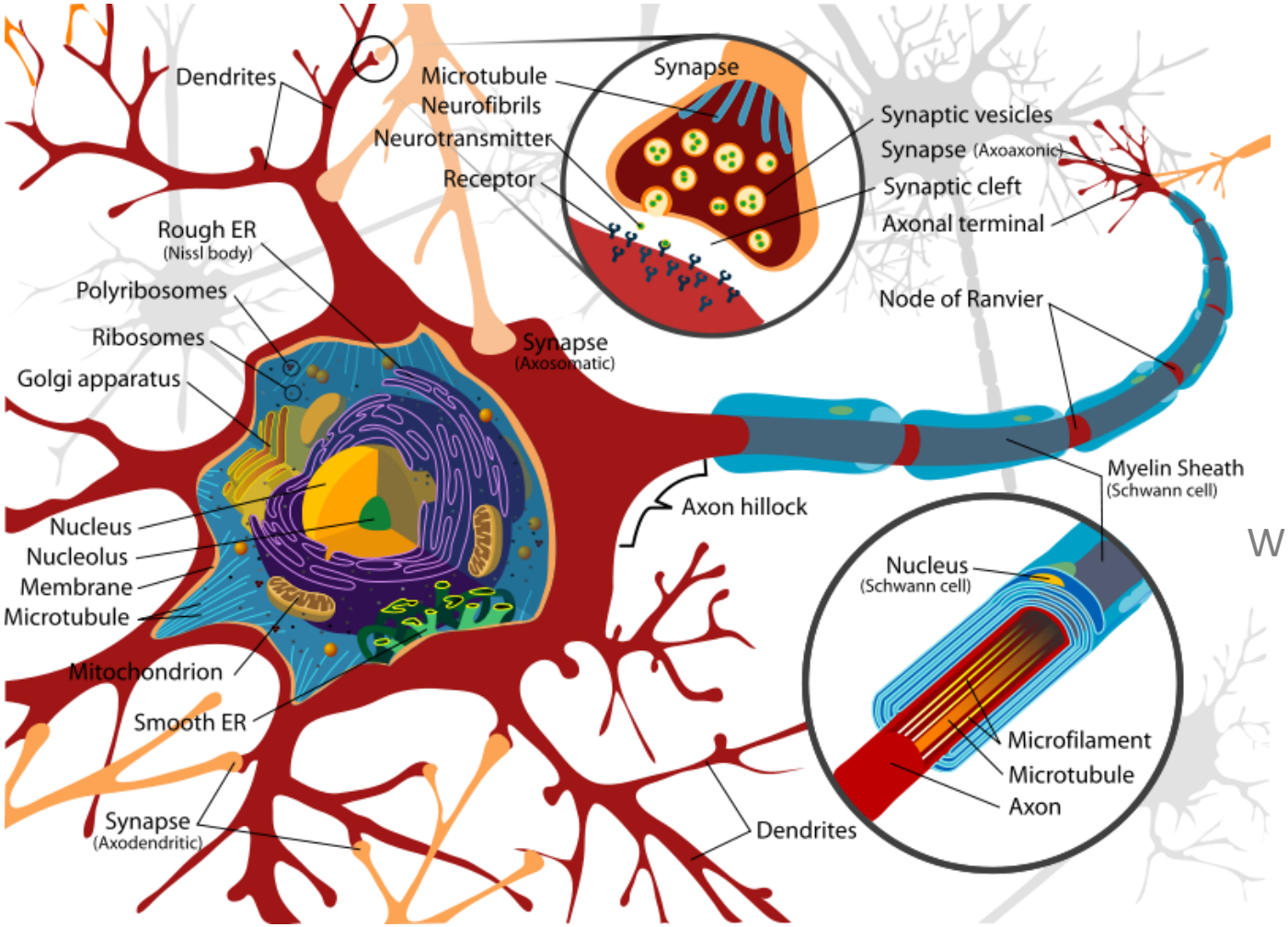
- Introduction
- Neural Network Theory
- Perceptron
- Multi Layer Networks
- Applications
- Handwriting Recognition

WORKSHOP

Introduction

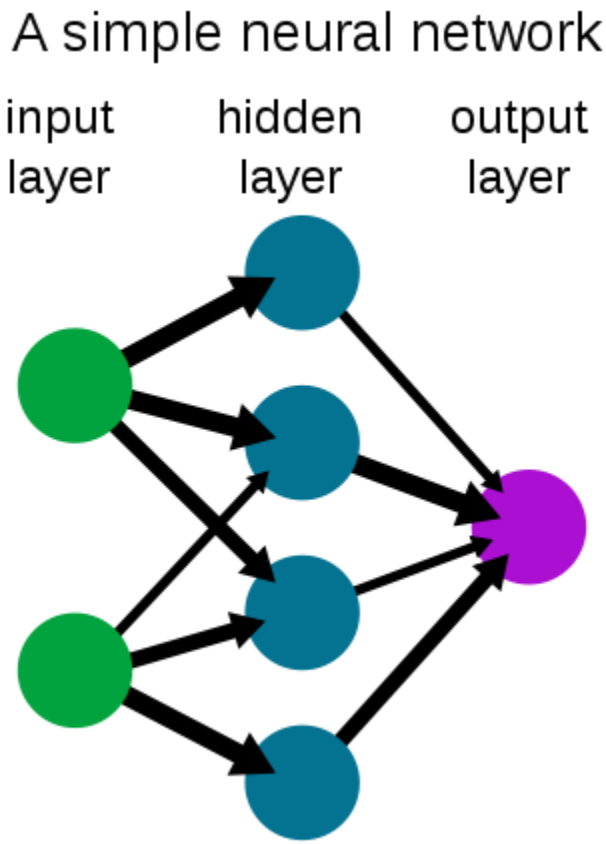
- Neural Networks in Biology
  - Neurons
  - Axons
- Artificial Neural Networks
  - Matrices
  - Weights

Introduction



Wikipedia

Introduction



Wikipedia

Introduction

- Use cases:
  - Classification
  - Generalization
  - Association
  - Principal Component Analysis
  - Optimization
  - Clustering
  - …

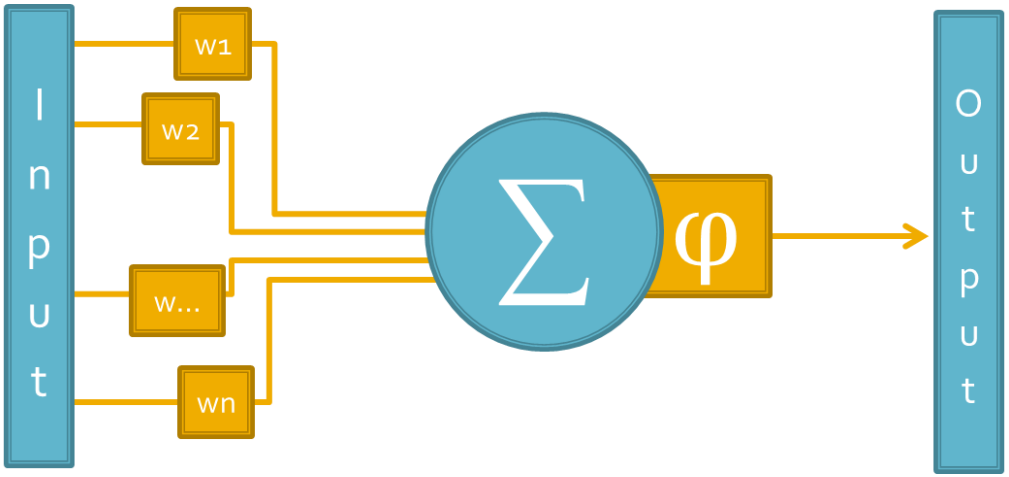
Introduction

- No free lunch!

Averaging over all possible worlds (data sets/problems), there is no single optimal algorithm. But, given a specific domain some algorithms perform better than others.

Neural Network Theory

- Nodes
- Activation function φ
- Learning Rule
- Topology
- Data in
- Data out



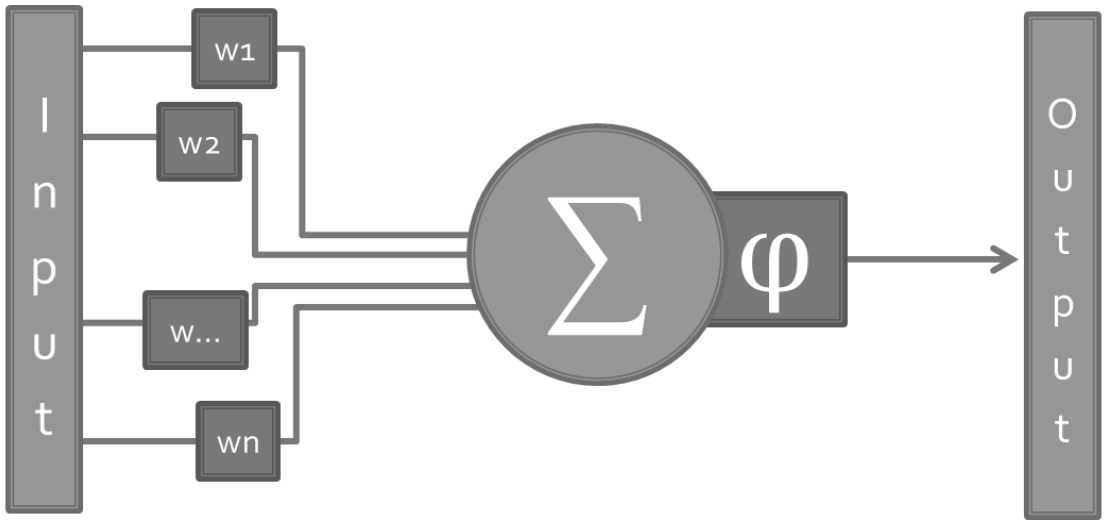
Perceptron

- The perceptron is a **binary classifier** which **maps its input** x (a real-valued vector) **to an output value** f(x) (a single binary value) across the matrix.

$$f(x) = \begin{cases} 1 & if\ w \cdot x + b > 0 \\ 0 & else \end{cases}$$

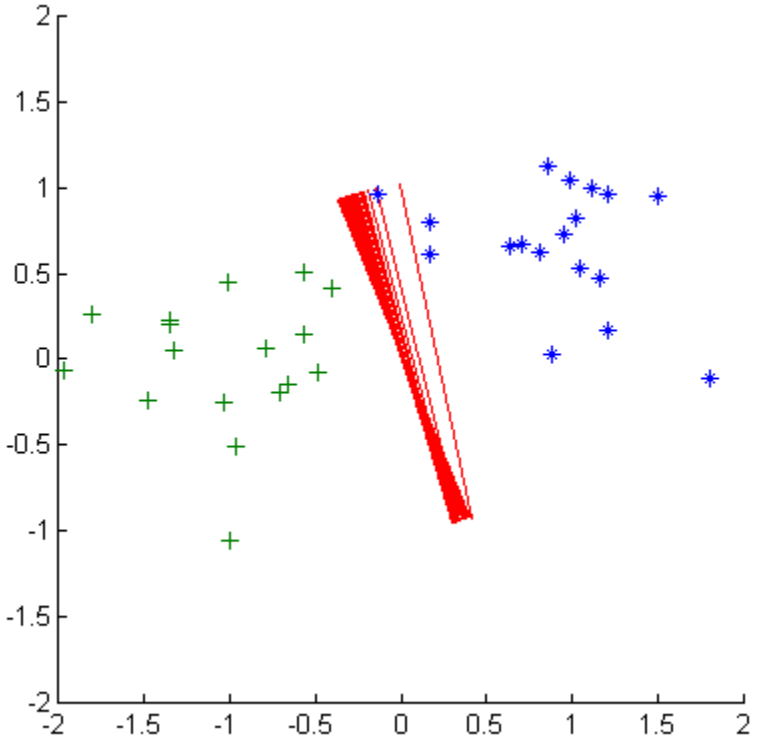
Wikipedia, Perceptron

Perceptron



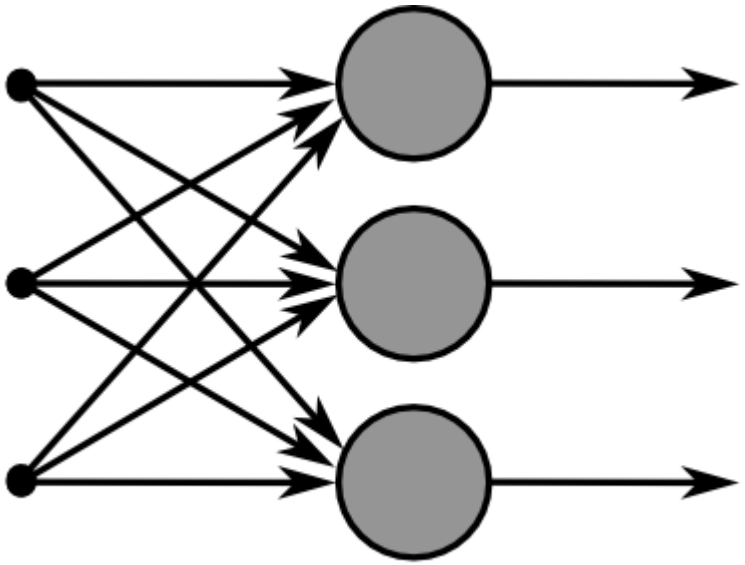
Perceptron

- Linear Classification!



Single Layer Networks

- One Layer
- More than one node
- Feed-Forward
- Thresholded



Wikipedia

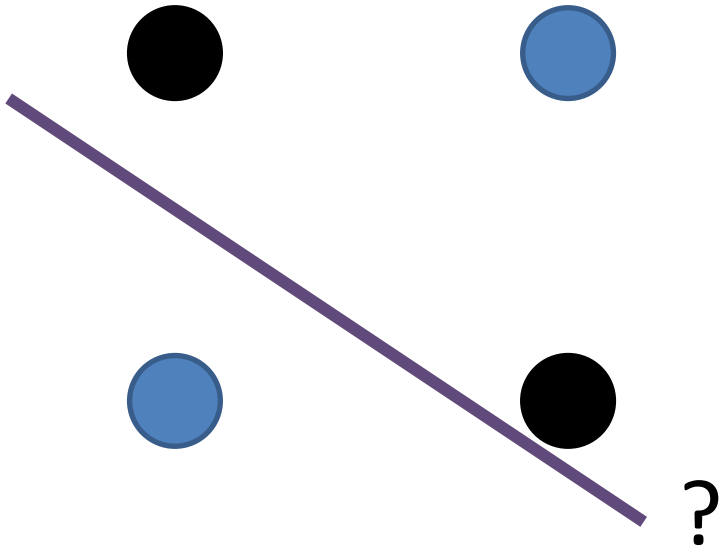
Single Layer Networks

- Learning Rule: Delta Rule

$$\Delta w = \eta e \vec{x} \qquad \text{where} \qquad e = t - \vec{w}^T \vec{x}$$

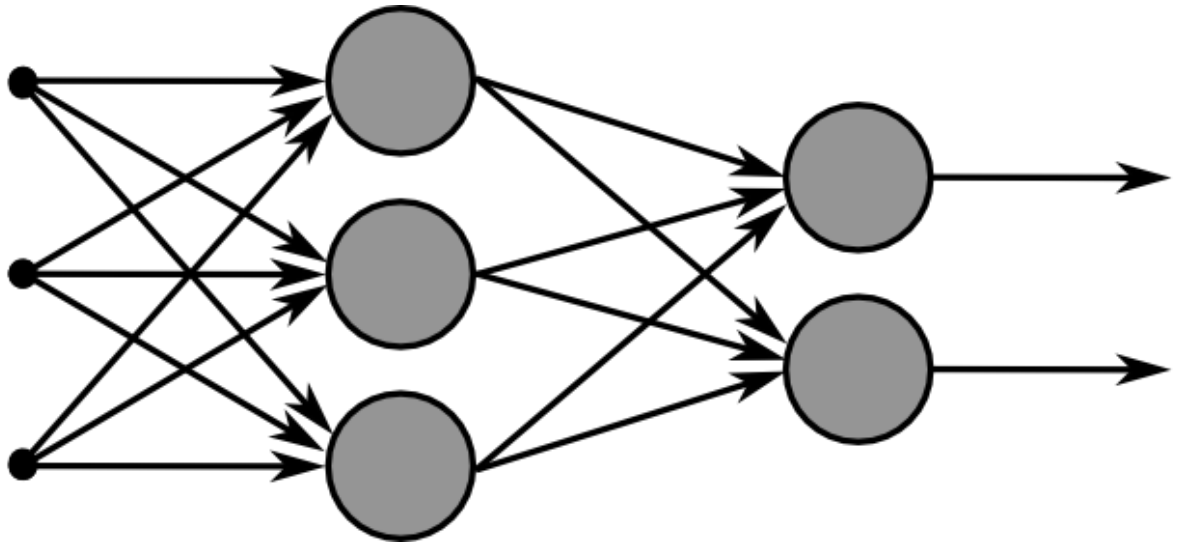
Single Layer Networks

- No XOR problem solvers!



Multi Layer Networks

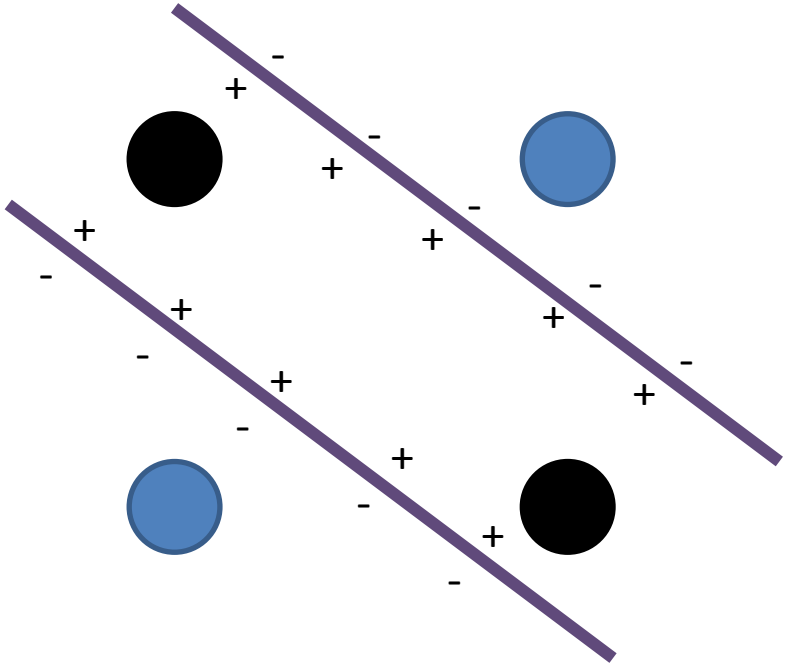
- More Layers
- More than one node
- Feed-Forward
- Thresholded

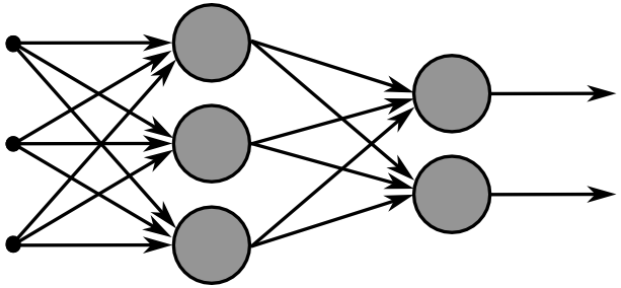


Wikipedia

Multi Layer Networks

- XOR problem solvers





Multi Layer Networks

- Learning Rule: Delta Rule, Back Propagation

Error Back-Propagation

1. Forward Pass: Compute all h_j and y_k

$$h_j = \varphi(\sum_i v_{ji} x_i) \qquad y_k = \varphi(\sum_j w_{kj} h_j)$$

2. Backward Pass: Compute all δ_k and δ_j

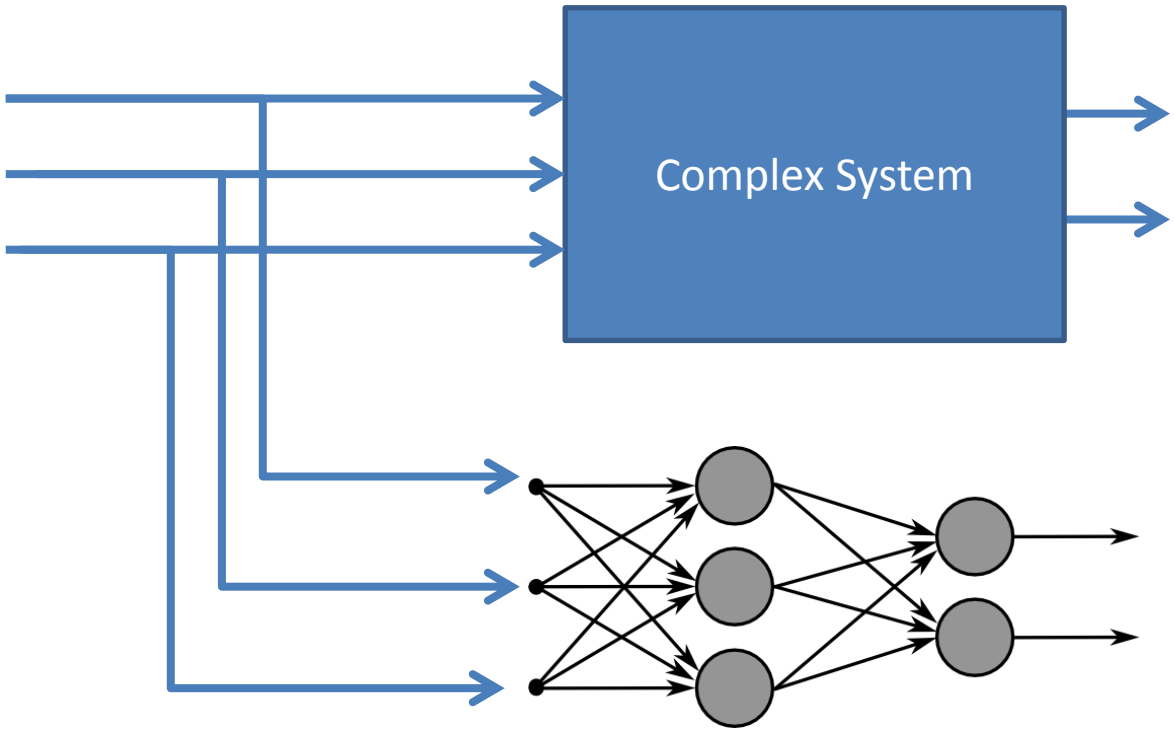
$$\delta_k = (t_k - y_k) \cdot \varphi'(y_k^{\text{in}}) \qquad \delta_j = \sum_k \delta_k \cdot w_{kj} \cdot \varphi'(h_j^{\text{in}})$$

3. Weight Updating:

$$\Delta w_{kj} = \eta \delta_k h_j \qquad \Delta v_{ji} = \eta \delta_j x_i$$

Applications

- Mimic existing systems



Applications

- Principal Component Analysis
  - Recognition
  - Prototype Extraction
  - Clustering
  - Feature Extraction

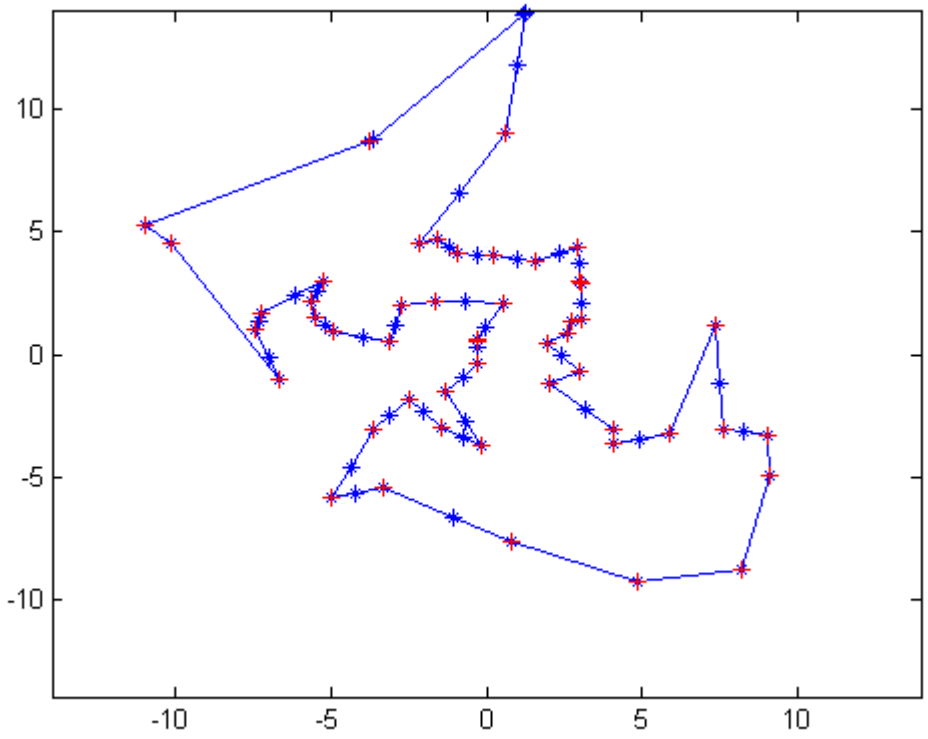
Applications

- Self-Organizing Maps

Dog	Dog	Fox	Fox	Fox	Cat	Cat	Cat	Eagle	Eagle
Dog	Dog	Fox	Fox	Fox	Cat	Cat	Cat	Eagle	Eagle
Wolf	Wolf	Wolf	Fox	Cat	Tiger	Tiger	Tiger	Owl	Owl
Wolf	Wolf	Lion	Lion	Lion	Tiger	Tiger	Tiger	Hawk	Hawk
Wolf	Wolf	Lion	Lion	Lion	Tiger	Tiger	Tiger	Hawk	Hawk
Wolf	Wolf	Lion	Lion	Lion	Owl	Dove	Hawk	Dove	Dove
Horse	Horse	Lion	Lion	Lion	Dove	Hen	Hen	Dove	Dove
Horse	Horse	Zebra	Cow	Cow	Cow	Hen	Hen	Dove	Dove
Zebra	Zebra	Zebra	Cow	Cow	Cow	Hen	Hen	Duck	Goose
Zebra	Zebra	Zebra	Cow	Cow	Cow	Duck	Duck	Duck	Goose

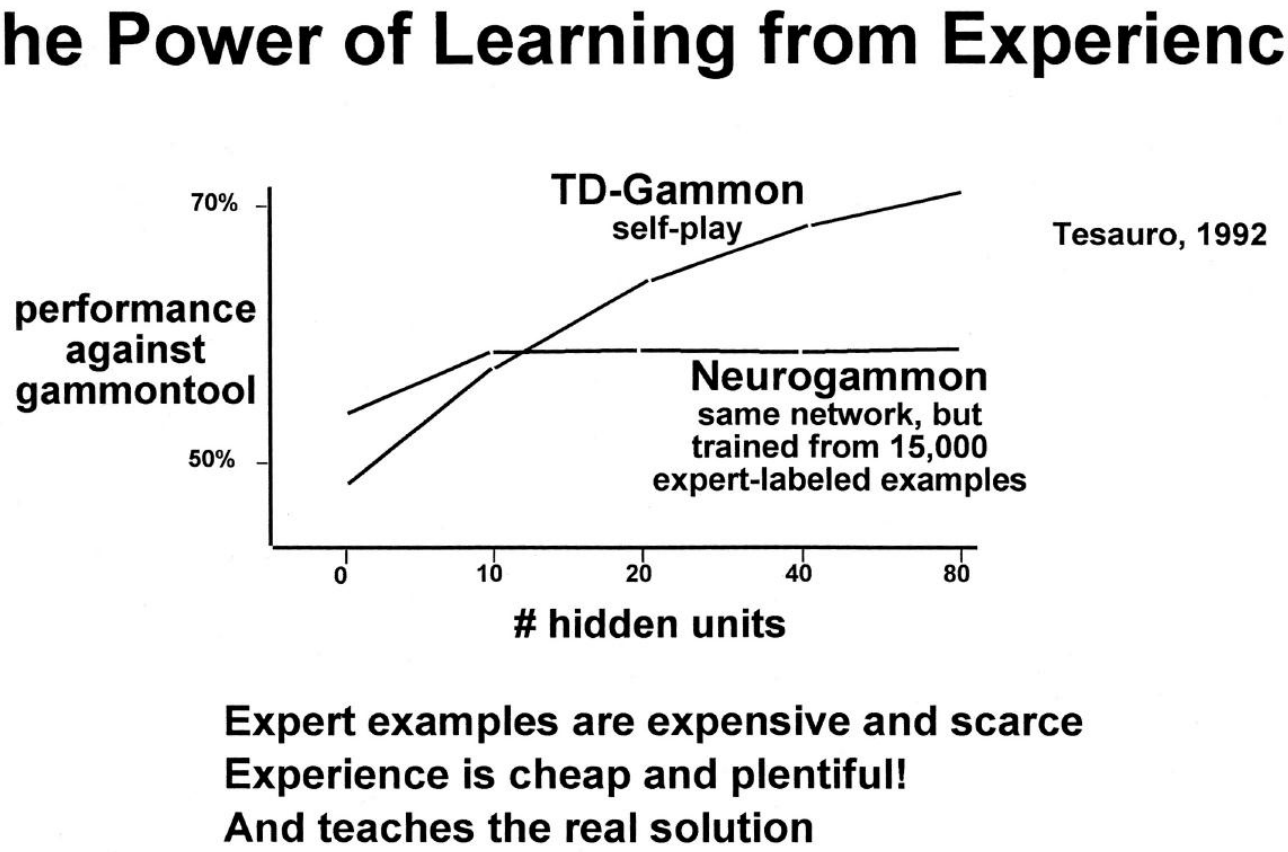
Applications

- Self Organizing Maps, TSP Solver



Applications

- Artificial Intelligence



Applications

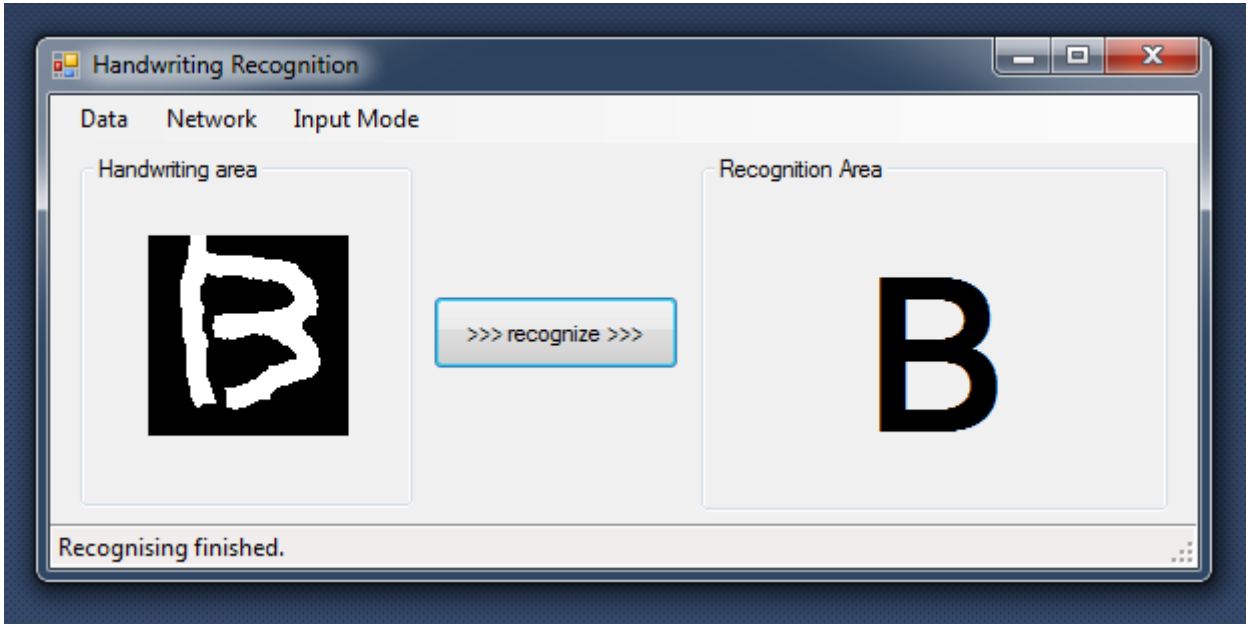
- Artificial Intelligence, Flight Control

http://www.nasa.gov/centers/dryden/news/NewsReleases/2003/03-49.html

Handwriting Recognition

WORKSHOP

- https://github.com/felix11/MSP-Workshops



References

- Artificial Neural Networks and Other Learning Systems, KTH
  http://www.csc.kth.se/utbildning/kth/kurser/DD2432/ann11/

- Perceptron
  http://en.wikipedia.org/wiki/Perceptron

Danke!

Felix Dietrich
felix.dietrich@studentpartners.de