

JavaNNS - a practical introduction to neural networks

Denise Peters

Overview

- Cognitive Architecture vs. Connectionisms
 - Theory
 - Basic elements
 - Network structure
 - Learning
 - JavaNNS
 - First steps
 - XOR network
 - Creating a network
-

Cognitive Architecture vs. Connectionsms

Cognitive Architecture

- Production System : Physical Symbol System
 - Memory, Symbols, Operations, Interpretation, Capacities
 - Production: Condition + Action
 - **If** production is true, **than** do action
 - Recognize-act cycle
 - PS is a cognitive information processing model
 - Soar, EPIC, ACT-R
-

Cognitive Architecture vs. Connectionisms

Connectionisms

- Knowledge is distributed over network by each node = **neuron metaphor**
 - Single node has no meaning - Pattern of nodes represent knowledge = **subsymbolic representation**
 - Information processing: modification of connection weights
 - **Firing:** sum of input high than activation level
 - Connectionisms Systems: basal processes e.g.: perception
 - SNNS (JavaNNS), NeuralWorks Professional II, Brainmaker
-

Pro's of networks

- Learning
 - Parallelism
 - Distributed knowledge
 - High error tolerance
 - Associative memory
-

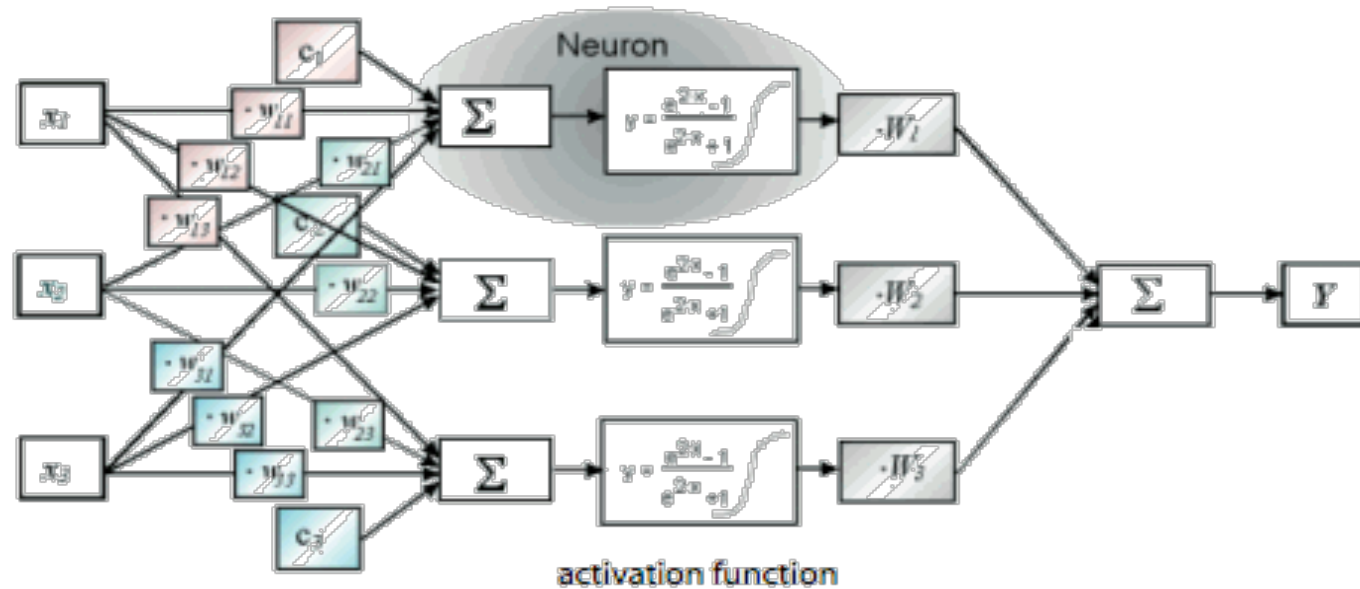
Contra's of networks

- Knowledge acquisition only by learning
 - No introspection possible
 - Learning is slow
-

Overview

- Cognitive Architecture vs. Connectionisms
 - Theory
 - Basic elements
 - Network structure
 - Learning
 - JavaNNS
 - First steps
 - XOR network
 - Creating a network
-

Theory - What is a network



Theory - Basic elements

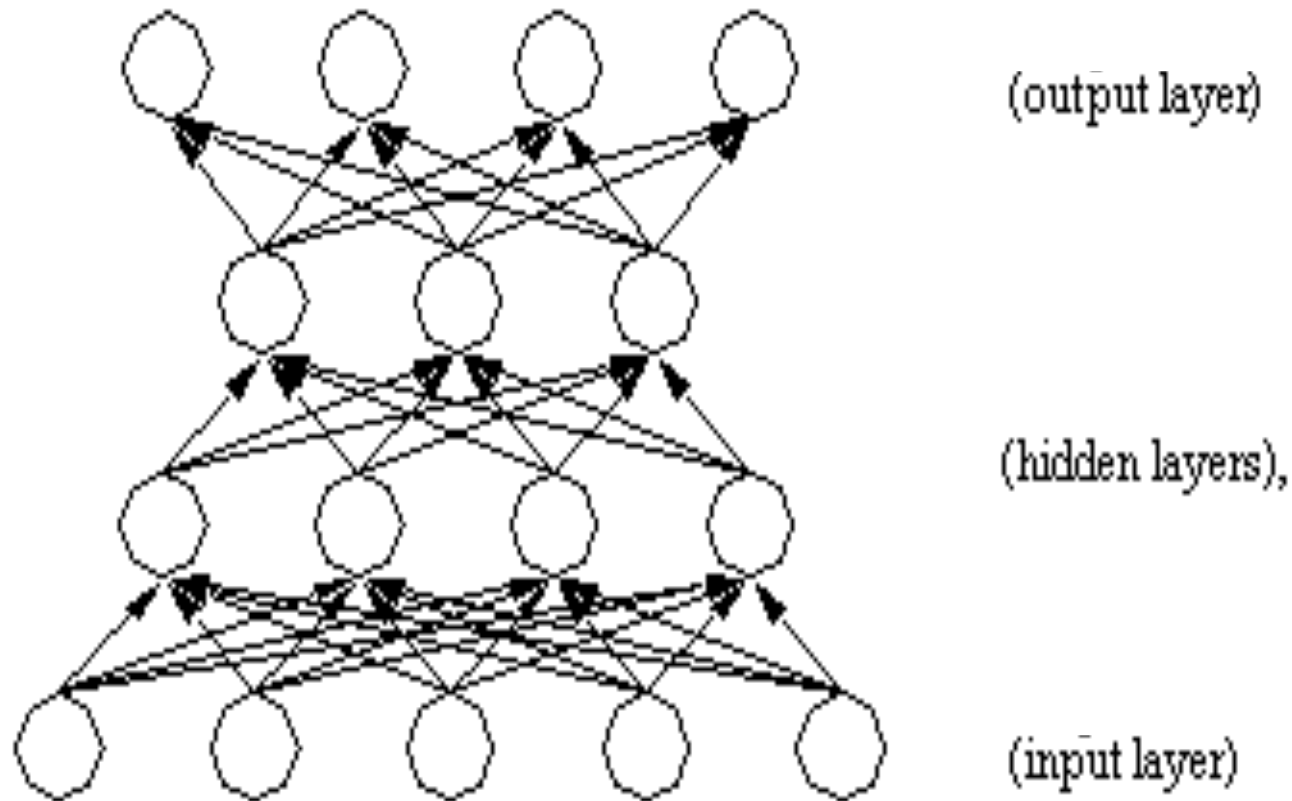
- Neuron
 - Activation/threshold
 - Activation function
 - Output function
- Weight network
 - Directed weighted graph
 - Weight matrix
- Propagation function
- Learning
 - Supervised learning
 - Reinforcement learning
 - Unsupervised learning

$$a_j(t+1) = f_{\text{act}}(a_j(t), \text{net}_j(t), \theta_j)$$

$$o_j = f_{\text{out}}(a_j)$$

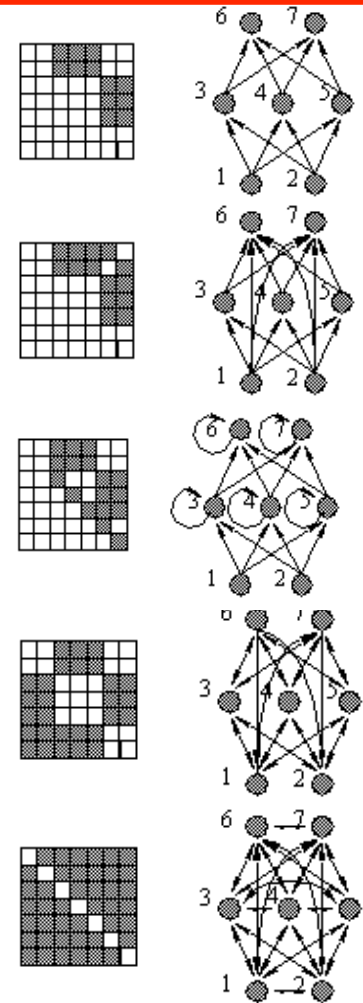
$$\text{net}_j(t) = \sum_i o_i(t) w_{ij}$$

Theory - Network structure



Theory - networks types

- Feedforward:
- Feedforward with shortcuts:
- Direct feedback:
- Indirect feedback:
- Complete connected:



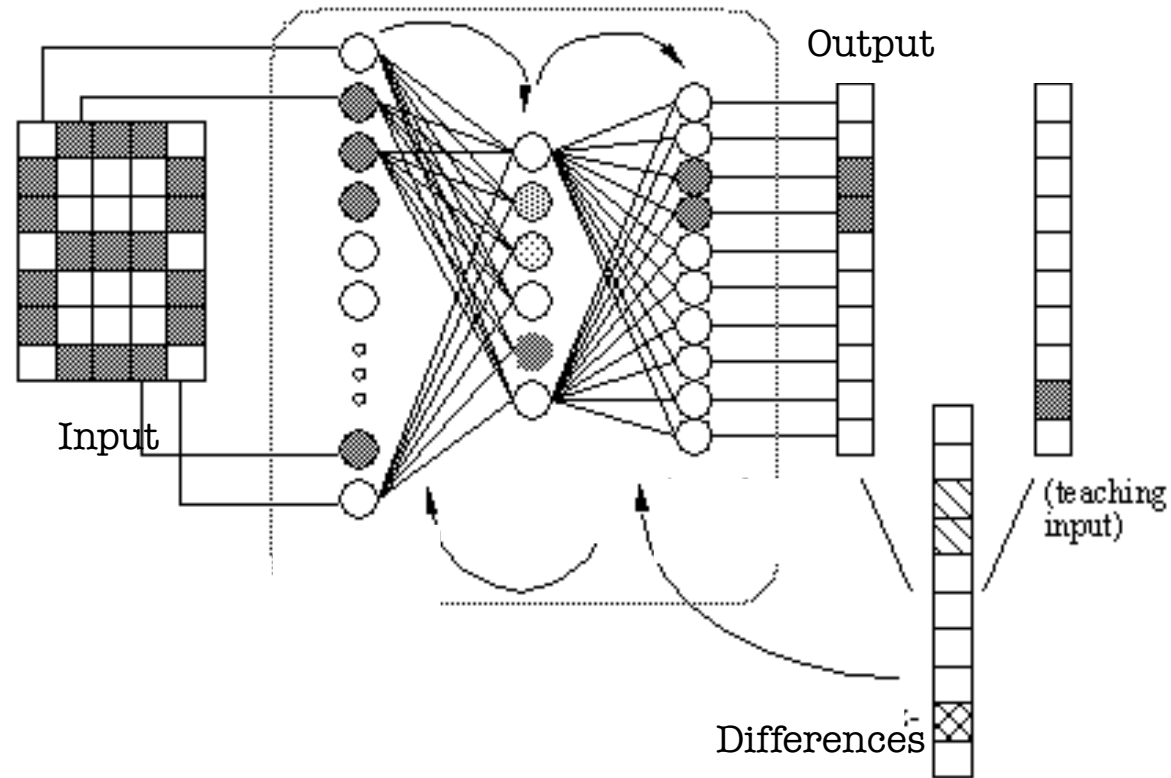
Theory - Learning

- Possible ways of learning
 - Creating new connections
 - Deleting new connections
 - Modification of weight strength
 - Modification of threshold
 - Modification of activation-/propagation function
 - Adding/deleting neurons
-

Theory - Learning

- Supervised learning
 - Most networks
 - Reinforcement learning
 - Unsupervised learning
 - Learning vector quantization
 - Self organizing maps
-

Theory - Supervised Learning



What do you need for learning

- Supervised learning
 - Training pattern: Input + output
 - Validation pattern
- Unsupervised learning
 - Training pattern: Input

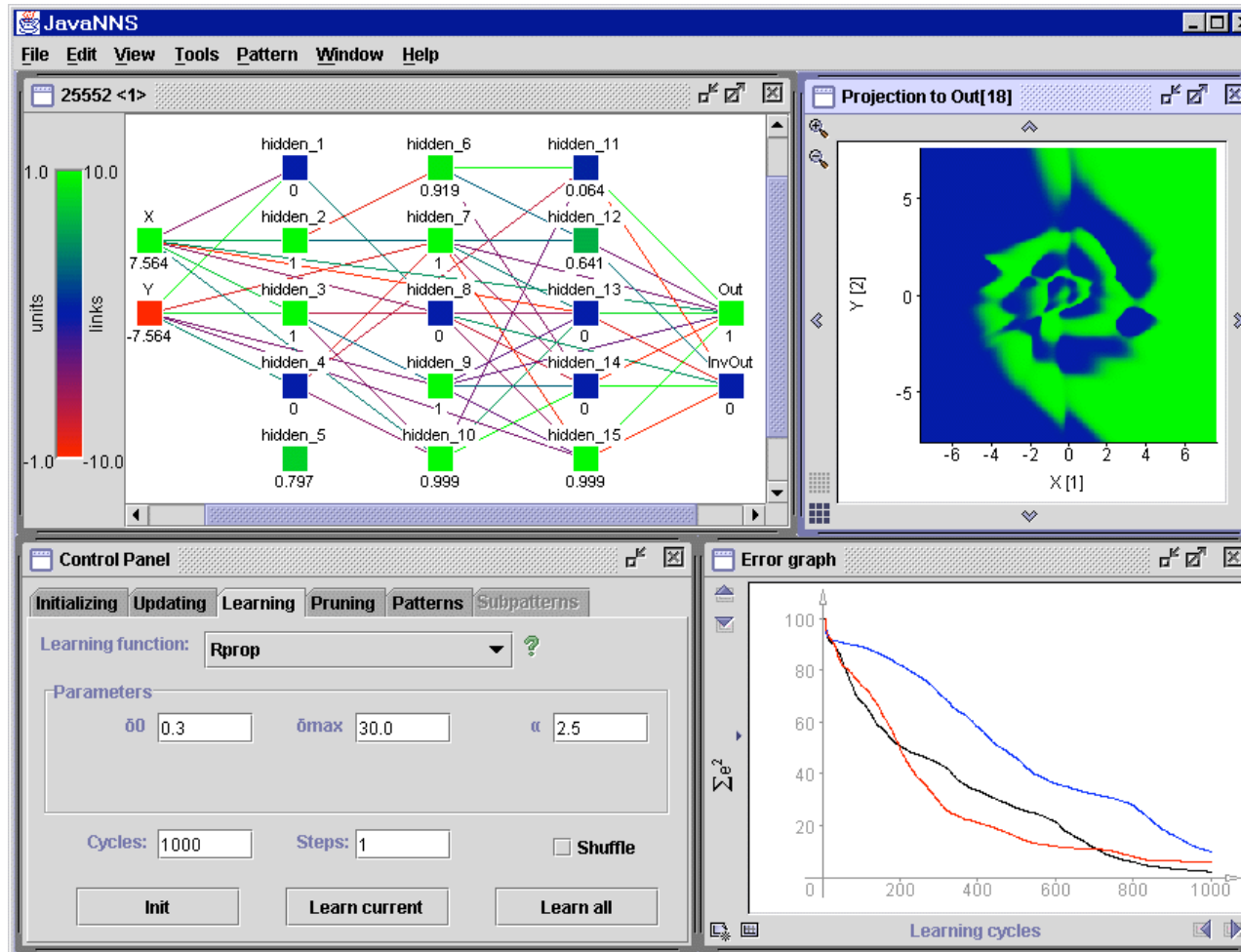
Analysing a network

- Parameters:
 - Learning rule
 - Structure of the network
 - Activation function
 - Time
 - How many learning cycles
 - Accuracy
 - How many false positive/negative hits
 - Generalisation
 - Overlearning vs accuracy
-

Overview

- Cognitive Architecture vs. Connectionisms
 - Theory
 - Basic elements
 - Network structure
 - Learning
 - **JavaNNS**
 - First steps
 - XOR network
 - Creating a network
-

JavaNNS - intro



JavaNNS - first steps

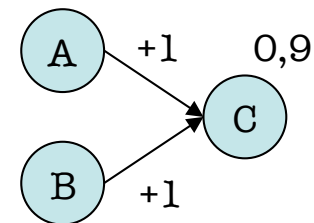
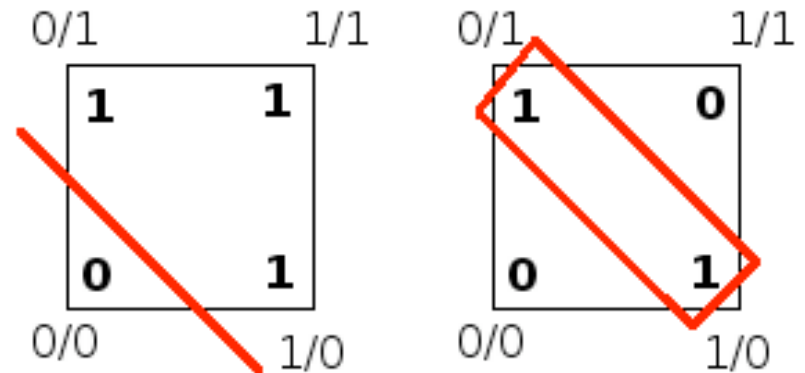
- Java Runtime Environment is needed
 - Download e.g. JavaNNS-Win.Zip
 - <http://www.ra.cs.uni-tuebingen.de/downloads/JavaNNS/>
 - Unpack it
 - JavaNNS.jar
 - Examples
 - Manual
 - First time: ask you where to install the library
 - Running JavaNNS:
 - `java -jar JavaNNS.jar`
-

Starting JavaNNS

- Load a file
 - xor_untrained.net
 - xor.pat
 - Train the network
 - Use Rprop as learning rule
 - Open error graph and log file
-

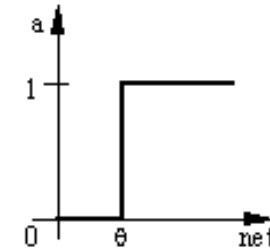
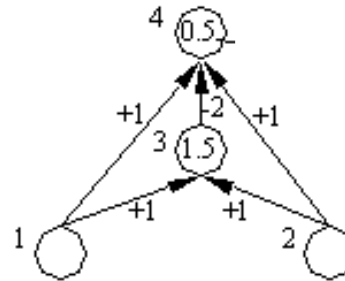
XOR

- Exclusive OR
 - If A and not B than C
 - If B and not A than C
- Linear separability
 - Perceptron
 - Additional hidden layer



XOR - Theory

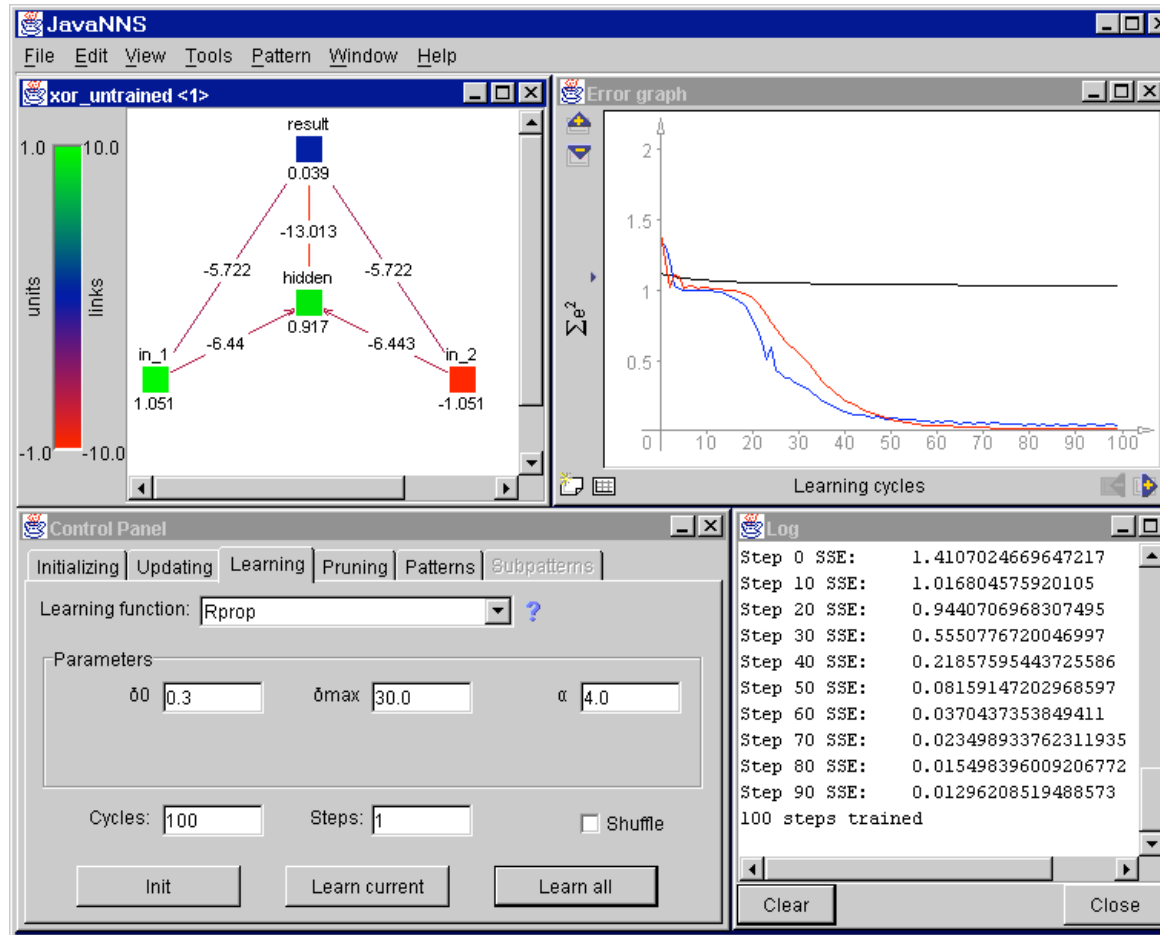
- Network



- Weight matrix

$$W = \begin{bmatrix} W_{11} & \dots & W_{14} \\ \dots & \dots & \dots \\ W_{41} & \dots & W_{44} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

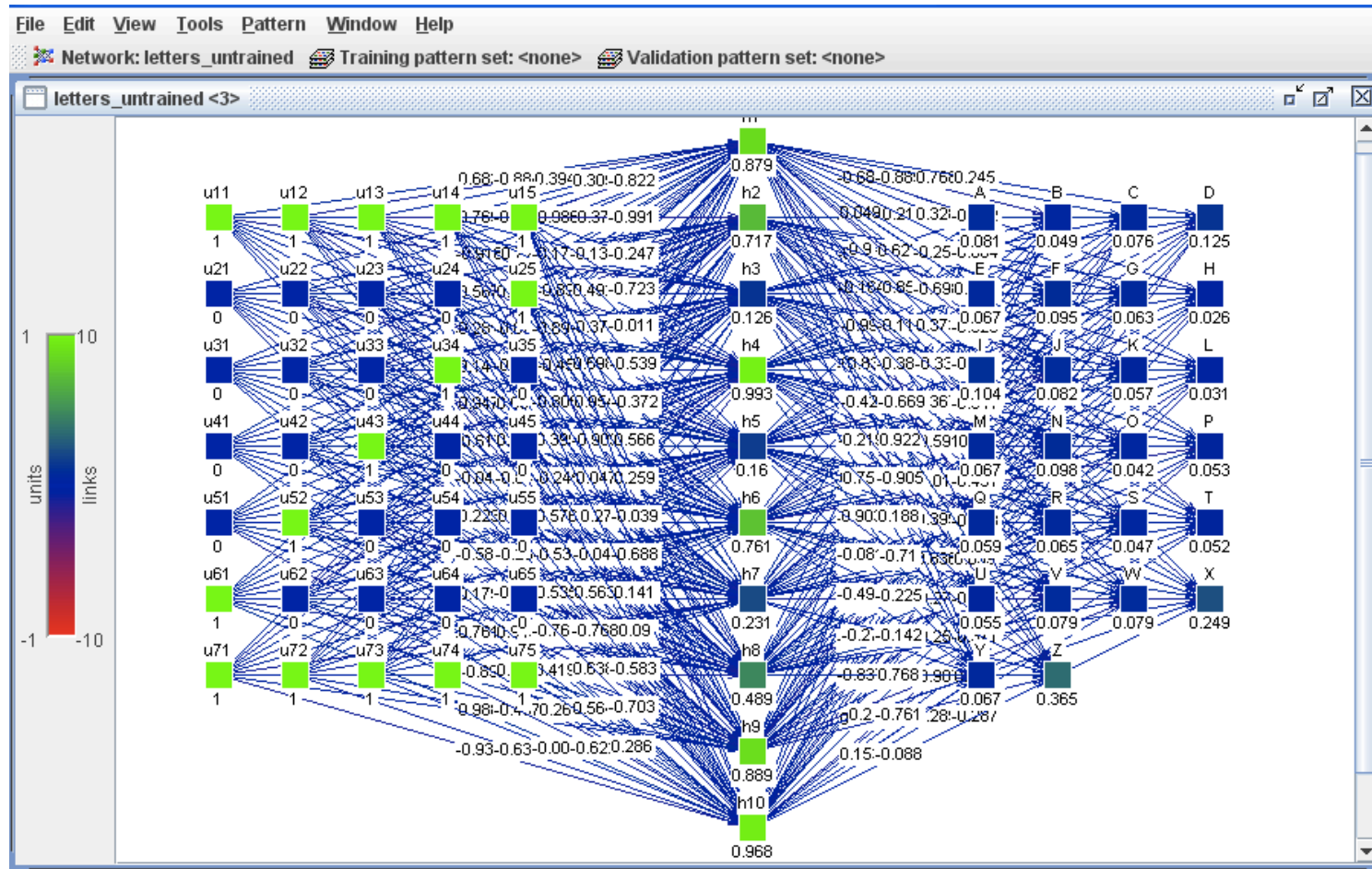
Learned XOR Network



Other Example: Letter.net

- Letter recognition network
 - 35 input neurons, 10 hidden neurons, 26 output neurons
 - letter.pat + letter.cfg
 - Simplified letter recognition
-

Letter.net



Other Examples

- Seeing examples
 - With Read-Me files:
 - Description
 - Parameters
 - Training examples x.pat
 - Additional config files
-

Creating Networks

Create layers

Size & Position

Width: Height:

Top left position:

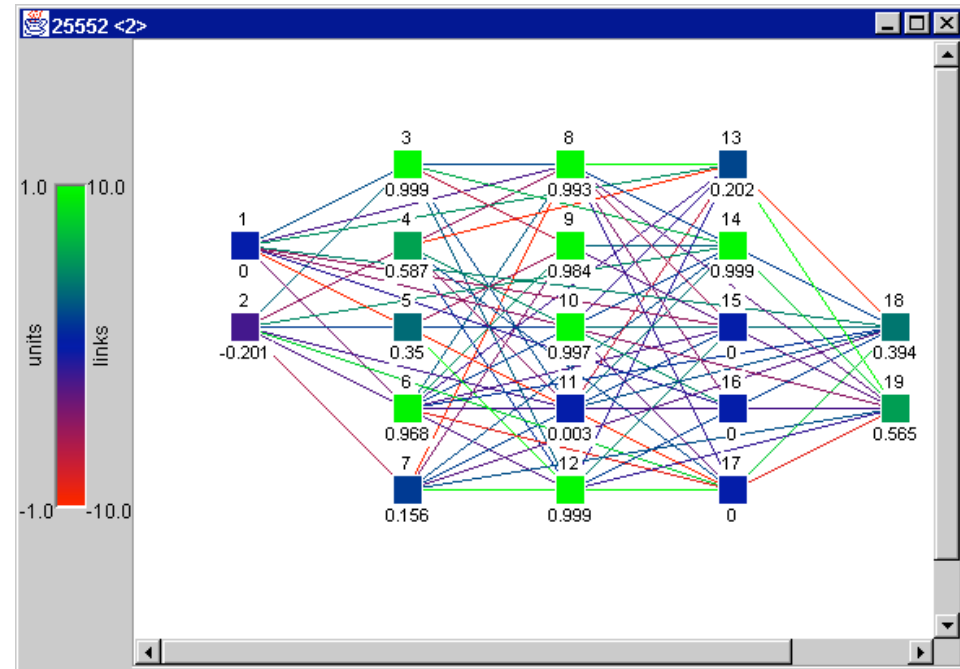
Unit detail

Unit type:

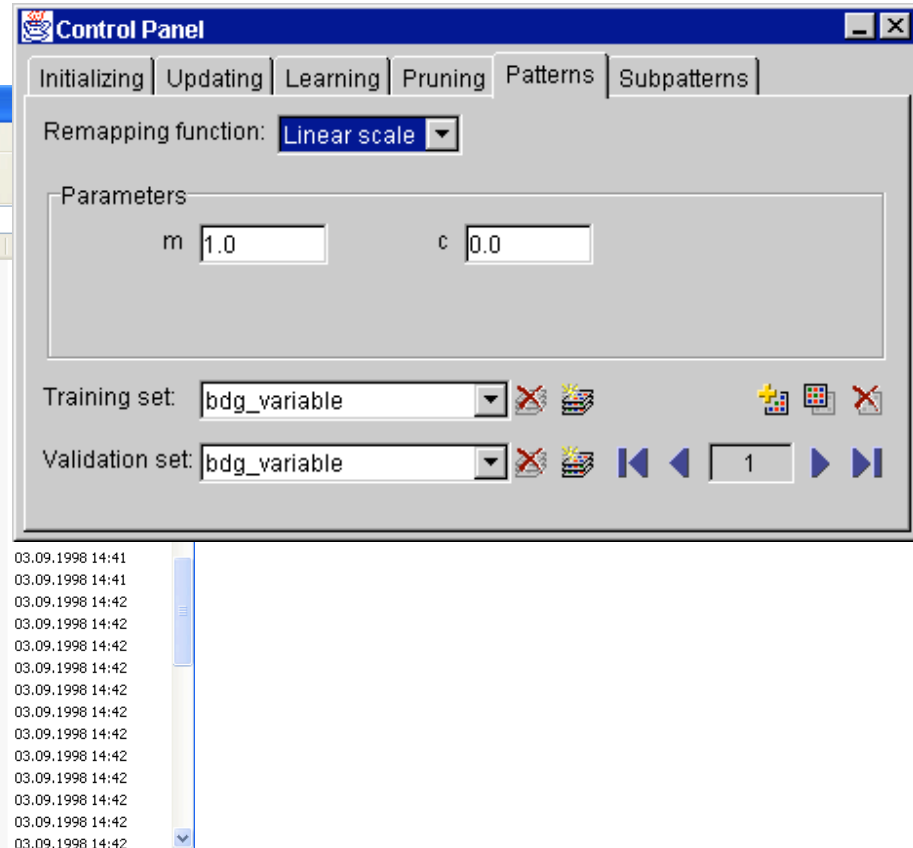
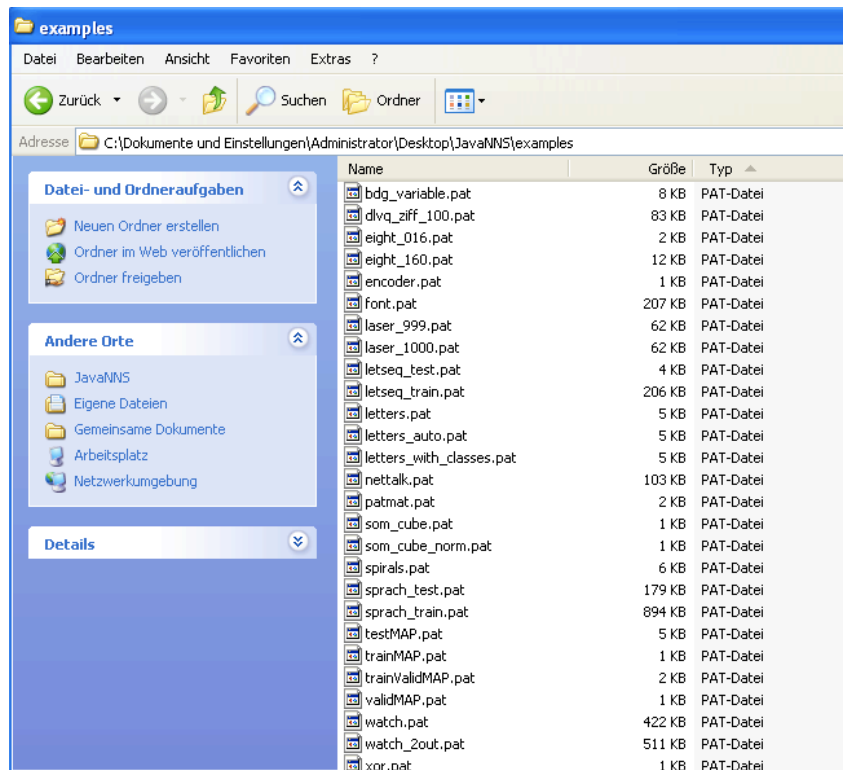
Activation function:

Output function:

Layer number: Subnet number:



Training Pattern Set



Supervised Learning: Backpropagation

- Formulated 1974 by Paul Werbos
 - Backpropagation of errors:
 - Forward propagation of input
 - Calculating differences of output and target value
 - Backpropagation of error by changing each weight of neurons
-

Hebbian Learning

- Postulated by Donald O. Hebb
 - Biological motivated
 - “Fire together wire together”
-

Literature

- <http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/manual/JavaNNS-manual.html>
 - <http://www.ra.cs.uni-tuebingen.de/SNNS/UserManual/UserManual.html>
-

- 
- Thank you !!!!
- 

- 
- Questions
- 

Homework

- A) First install JavaNNS:
 - <http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS>
 - B)
 - Get it to run and open the Error Graph Window and the Log Window, to see what you are training.
 - Open the "spirals.pat" from the "examples" folder and generate a feedforward network with the following layer setup:
 - 2-5-5-5-2 (these are the layers)
 - Train the network (for this you will have to find the appropriate parameters and should use only 10.000 steps for training). Use all two rules as learning rules, one after the other:
 - 1.) Backpropagation
 - 2.) Hebbian
-

Homework

- C)
 - For the discussion of the result you need to explain each learning rule in detail:
 - Provide the learning equation, a thorough discussion of the theoretical backgrounds of the learning rules
 - origins, design rationales, how do they compare to one another?
 - Quote all literature and other sources that you use.
 - You need to also submit protocols with chosen parameters and snapshots of the learning curves for all rules.
-