Neural Networks

(Chapter 9)

Kai Goebel, Bill Cheetham

GE Corporate Research & Development

goebel@cs.rpi.edu

cheetham@cs.rpi.edu

1

Stuff we talked about

Categories

- Unsupervised Learning
- Reinforcement Learning
- Unsupervised Learning

Hopfield Net

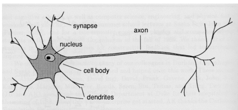
Perceptron

- Single Layer
- Multi Layer

Radial Basis Function Network

2

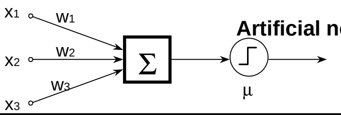
Neurons



Real neuron

≠

Artificial neuron

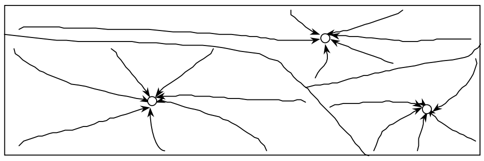


3

Hopfield Network Operation

Picture is pattern; stored as attractor in the configuration space.

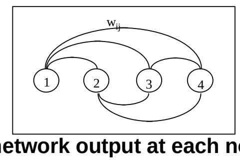
From arbitrary starting points, one attractor will be found



4

Hopfield Architecture

- Recurrent Network
- Symmetric Architecture
- Evaluation until no more changes are observed i.e., network settles into local minimum config.



- network output at each node:

$$y_i = \operatorname{sgn}\left(\sum_i w_{ij} x_j\right)$$

updating rule:

$$w_{ij}^{(n+1)} = w_{ij}^{(n)} + x_i x_j$$

$$w_{ii} = 0$$

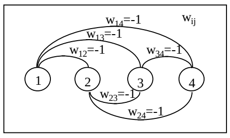
5

Hopfield Example

Learn x=[1 1 -1 -1]

which gives us the weight matrix

w=[0 1 -1 -1
1 0 -1 -1
-1 -1 0 1
-1 -1 1 0]



Now let's check the slightly corrupted pattern

p=[1 1 -1 1]

which will restore the pattern found close

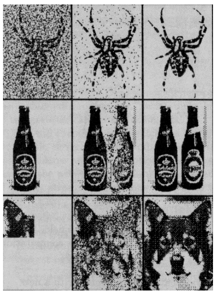
y=[1 1 -1 -1]

with an energy level of E=-6

6

More Complex Hopfield Examples

Reconstruction of Images



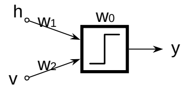
binary images are 130x180 pixels

7

Single-Layer Perceptron

Example: Gender classification (according to Jang)

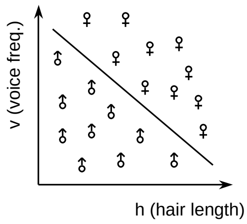
Network Arch.



y = signum($hw_1+vw_2+w_0$)

= -1 if female
1 if male

Training data



Learning:

select an input vector

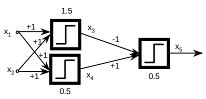
if the response is incorrect, modify all weights

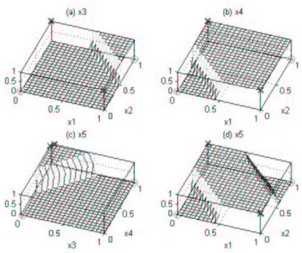
$\Delta w_i = \eta\, t_i x_i$

8

Two-Layer Perceptron: XOR

Node output as surface of their two inputs



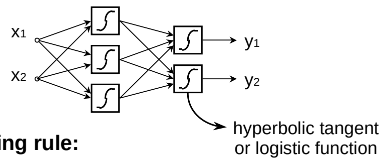


note location of “o” and “x”

9

Multilayer Perceptrons (MLPs)

Network architecture



Learning rule:

- **Steepest descent (Backprop)**
- **Conjugate gradient method**
- **All optim. methods using first derivative**
- **Derivative-free optim.**

Activation function $f(x) = \frac{1}{1+e^{-x}}$ $\quad f(x) = \frac{1-e^{-x}}{1+e^{-x}} = \tanh\left(\frac{x}{2}\right)$

10

Back-prop procedure

Make incremental change in the direction dE/dw to decrease the error.

The learning rule for each node can be derived using the chain rule...

…to propagate the error back through a multi-layer perceptron 1. Initialize weights to small random values

2. Choose a pattern and apply it to input layer

3. Propagate the signal forward through the network

4. Compute the deltas for the output layer

5. Compute the deltas for the preceding layers by propagating the error backwards

6. Update all weights

7. Go back to step 2 and repeat for next pattern

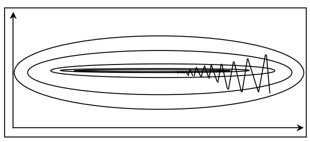
8. Repeat until error rate is acceptable

11

Momentum

If error minimum in long narrow valley,

then updating can happen to zig-zag down the valley



smoothes weight updating

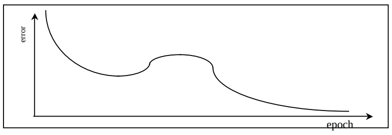
can speed learning up $\Delta w = -\eta \nabla_w E + \alpha \Delta w_{prev}$

12

Soft Computing: Neural Networks

Local Minima

There is no guarantee that the algorithm converges to a global minimum



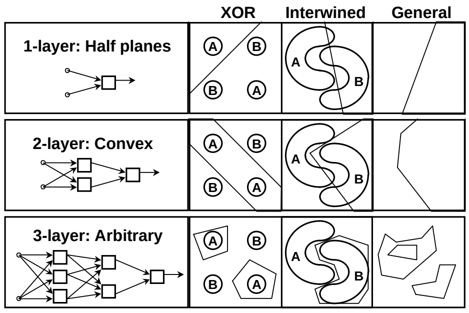
- check with different initial conditions (different weights, etc.)

- perturb the system (data) with noise to improve result

13

Soft Computing: Neural Networks

MLP Decision Boundaries

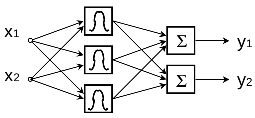


14

Soft Computing: Neural Networks

Radial Basis Function (RBF) Networks

Network architecture



Each node is described by a bell shaped function

$$o_i = \exp\left(-\frac{\|x - c_i\|^2}{2\sigma_i^2}\right)$$

where

c_i is the center of the curve

Output:

weighted sum

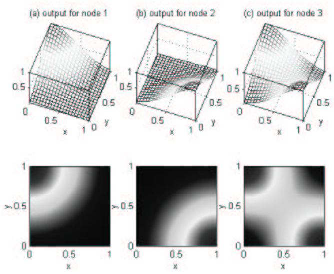
linear combination

15

Soft Computing: Neural Networks

XOR, revisited

x	y	output
0	0	0
0	1	1
1	0	1
1	1	0

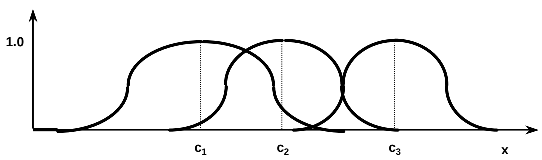


16

Soft Computing: Neural Networks

RBF and FIS

Consider the radial basis functions:



and a linear combination of the output variables

$y_i = \vec{a}_i \vec{o} + b_i$

then the response is equivalent to ... FIS

17

Soft Computing: Neural Networks

last slide

18