Neural Networks: Scheme of Modelling and Peceptron and Multilayer Perceptron

Lecture 4

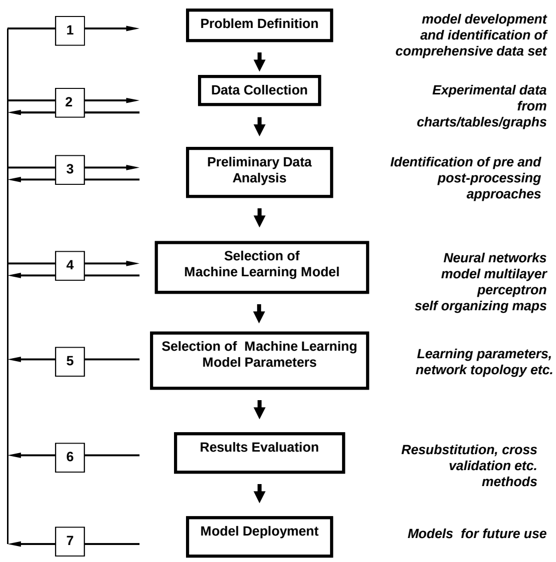
1

Recap and Ahead

- Scheme of Neural Networks Modelling
- Perceptron Model
- Multilayer Perceptron

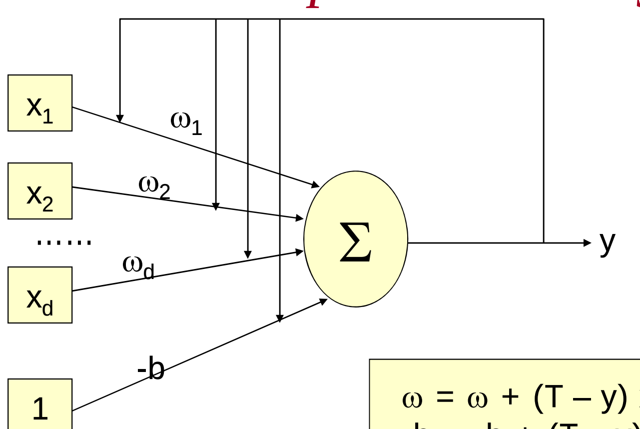
2

Scheme of Modelling



3

Perceptron Learning Rule



$\omega = \omega + (T - y)\ x$

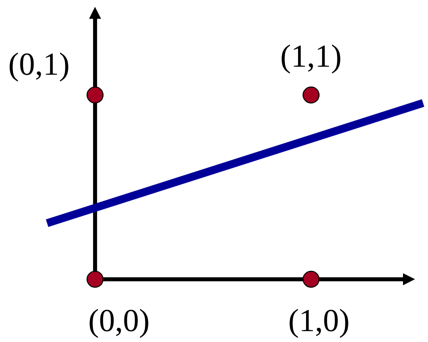
$-b = -b + (T - y)$

T is the expected output

y is the real output

4

XOR Problem

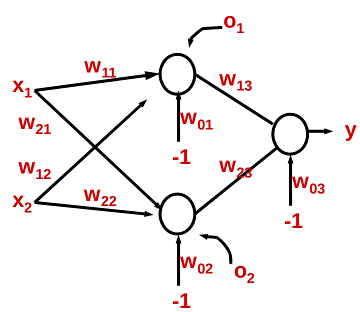


Four Input vectors for the XOR problem. The line $y = w_1x_1 + w_2y_2$ divides the plane into two regions but cannot successfully isolate the set of points (0,0) and (1,1) from the points (0,1) and (1,0)

5

Solving the XOR Problem

Network Topology:
2 hidden nodes
1 output



Desired behavior:

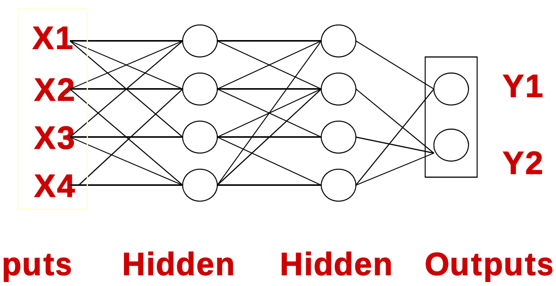
x1	x2	o1	o2	y
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

Weights:
w11= w12=1
w21=w22 = 1
w01=3/2; w02=1/2; w03=1/2
w13=-1; w23=1

6

Multilayer Perceptron

- Inputs: real-valued
- Intermediate “hidden” nodes
- Output(s): one (or more) discrete-valued



7

Features

- **Pro:** More general than perceptrons
  - Not restricted to linear discriminants
  - Multiple outputs: one classification each
- **Con:** No simple, guaranteed training procedure
  - “Gradient descent”, “Backpropagation”

8

MLP as Universal Approximators

- A MLP with one hidden layer can approximate any continuous function to any desired accuracy
- MLP are multivariate nonlinear regression models
- MLP can learn conditional probabilities

9

Data Standardization

- Problem in the units of the inputs
  - Different units cause magnitude of difference
  - Same units cause magnitude of difference
- Standardization – scaling input

10

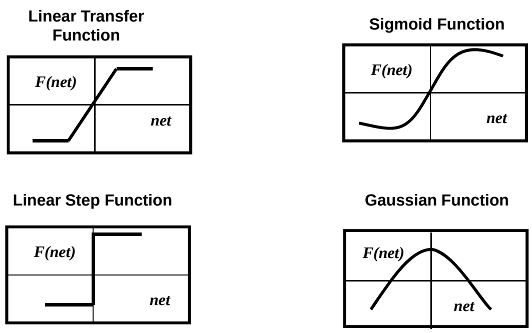
Target Values (output)

- Instead of one-of-c (c is the number of classes), we use +1/-1
  - +1 indicates target category
  - -1 indicates non-target category
- For faster convergence

Number of Hidden Layers

- The **number of hidden layers** governs the expressive power of the network, and also the **complexity of the decision boundary**
- More hidden layers -> higher expressive power -> better tuned to the particular training set -> poor performance on the testing set
- Rule of thumb
  - Choose the number of weights to be roughly **n/10**, where **n is the total number of samples** in the training set
  - Start with a “large” number of hidden units, and “decay”, prune, or eliminate weights

Activation Functions

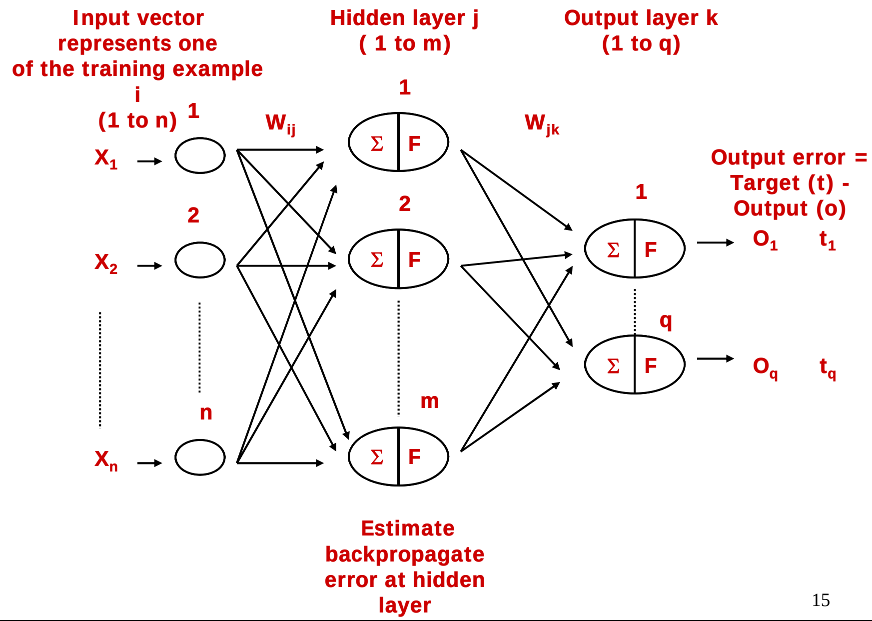


13

Initializing Weight

- Can't start with zero
- Fast and uniform learning
  - All weights reach their final equilibrium values at about the **same time**
  - Choose weights randomly from a **uniform distribution** to help ensure uniform learning
  - **Equal negative and positive** weights
  - Set the weights such that the integration value at a hidden unit is in the range of **–1 and +1**

14



15

General Steps of MLP Learning

- Step 1 : Select the training pair from the training set; apply the input vector to the network input
- Step 2: Calculate the output of the network
- Step 3: Calculate the error between the network output and the desired output (the target vector from the training pair)

16

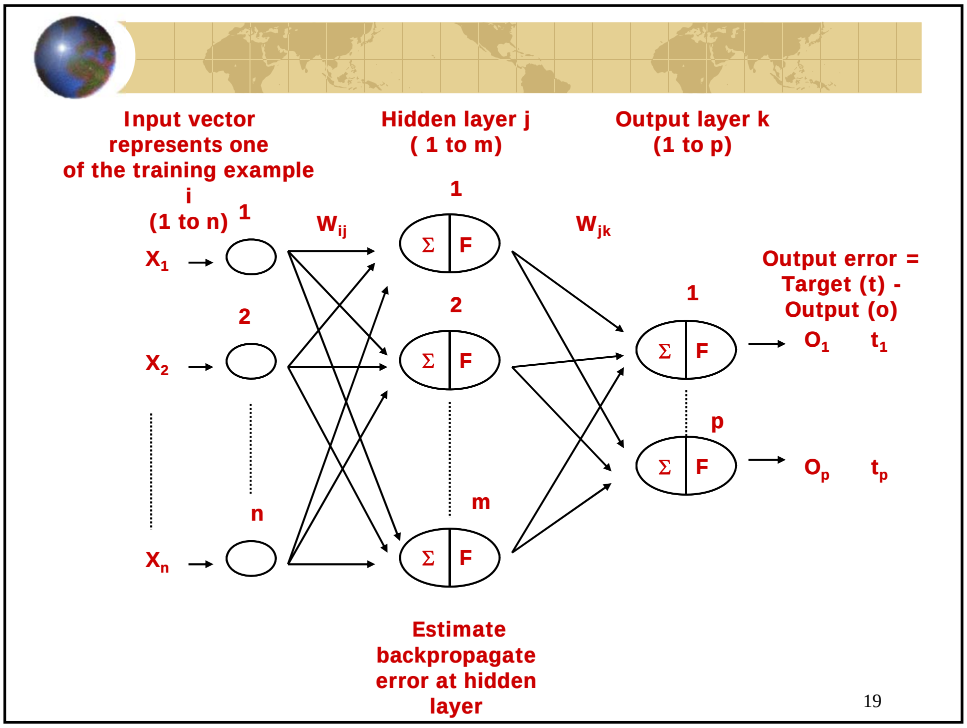
General Steps of MLP Learning

- Step 4: Adjust the weights of the network in a way that minimizes the error
- Step 5: Repeat Steps 1 through 4 for each vector in the training set until the error for the entire set is acceptably low.

17

Back Propagation Algorithm

18



Backpropagation Algorithm Steps

- Step 1: Select
  - Number of input nodes (***n***)
  - Output nodes (***p***)
  - Hidden nodes (***m***) and
  - first training example (***X****i*)
  - Model Parameters

20

- Step 2: Initialize the ***weights*** using random number generator in the range of 0 to 1/-1 to 1.

$$W_{ji} = \begin{bmatrix} w_{11} & w_{12} & \cdots\cdots\cdots & w_{1n} \\ w_{21} & w_{22} & \cdots\cdots\cdots & w_{2n} \\ . & . & \cdots\cdots\cdots & . \\ w_{m1} & w_{m2} & \cdots\cdots\cdots & w_{mn} \end{bmatrix} \qquad W_{kj} = \begin{bmatrix} w_{11} & w_{12} & \cdots\cdots\cdots & w_{1m} \\ w_{21} & w_{22} & \cdots\cdots\cdots & w_{2m} \\ . & . & \cdots\cdots\cdots & . \\ w_{p1} & w_{p2} & \cdots\cdots\cdots & w_{pm} \end{bmatrix}$$

21

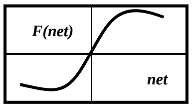
- Step 3: Compute the values of $\{net_j\}$ for the hidden nodes

$$\{net_j\} = \begin{Bmatrix} net_1 \\ net_2 \\ . \\ net_m \end{Bmatrix} = [W_{ji}]\{X_i\}$$

22

- Step 4: Calculate the activation value $\{out_j\}$ for the hidden nodes. Here the Sigmoidal function has recommended. The parameter θ_j is to shift the activation function to left and right along the horizontal axis depending upon its positive or negative values respectively. Similarly, θ_0 is used to modify the shape of the sigmoid

Sigmoid Function



23

$$\{out_j\} = \begin{Bmatrix} out_1 \\ out_2 \\ . \\ . \\ out_m \end{Bmatrix} = \begin{Bmatrix} f_1(net_1) \\ f_2(net_2) \\ . \\ . \\ f_m(net_m) \end{Bmatrix} = \begin{Bmatrix} 1/(1+e^{\frac{(-net_1+\theta_1)}{\theta_0}}) \\ 1/(1+e^{\frac{(-net_2+\theta_2)}{\theta_0}}) \\ . \\ . \\ 1/(1+e^{\frac{(-net_m+\theta_m)}{\theta_0}}) \end{Bmatrix}$$

24

- Step 5: Compute the values of $\{net_k\}$ for the output nodes

$$\{net_k\} = \begin{Bmatrix} net_1 \\ net_2 \\ . \\ net_p \end{Bmatrix} = [W_{kj}]\{out_j\}$$

25

- Step 6: Calculate the activation value $\{out_k\}$ for the output nodes

$$\{out_k\} = \begin{Bmatrix} out_1 \\ out_2 \\ . \\ . \\ out_p \end{Bmatrix} = \begin{Bmatrix} f_1(net_1) \\ f_2(net_2) \\ . \\ . \\ f_p(net_p) \end{Bmatrix} = \begin{Bmatrix} 1/(1+e^{\frac{(-net_1+\theta_1)}{\theta_0}}) \\ 1/(1+e^{\frac{(-net_2+\theta_2)}{\theta_0}}) \\ . \\ . \\ 1/(1+e^{\frac{(-net_m+\theta_p)}{\theta_0}}) \end{Bmatrix}$$

26

- Step 7: Calculate the error [ΔW_{kj}]

$$[\Delta W_{kj}] = \eta\{t_k - o_k\}\{o_k\}\{out_j\}^T + \alpha[\Delta_p W_{kj}]$$

- Learning parameter η
- Momentum Parameter α

These parameters are selected from experience

27

- Step 8: Compute the new values of weights [W_{kj}] between the hidden and output layers

$$[W_{kj}] = [W_{kj}] + [\Delta W_{kj}]$$

28

- Step 9: Calculate the error [ΔW_{ji}] for input to hidden weights

$$[\Delta W_{ji}] = \eta \{out_j\}\{t_k - o_k\}\{o_k\}[W_{kj}]^T\{X_i\}^T + \alpha[\Delta_p W_{ji}]$$

29

- Step 10: Calculate the new values of the weights [W_{ji}] between input and hidden layer

$$[W_{ji}] = [W_{ji}] + [\Delta W_{ji}]$$

30

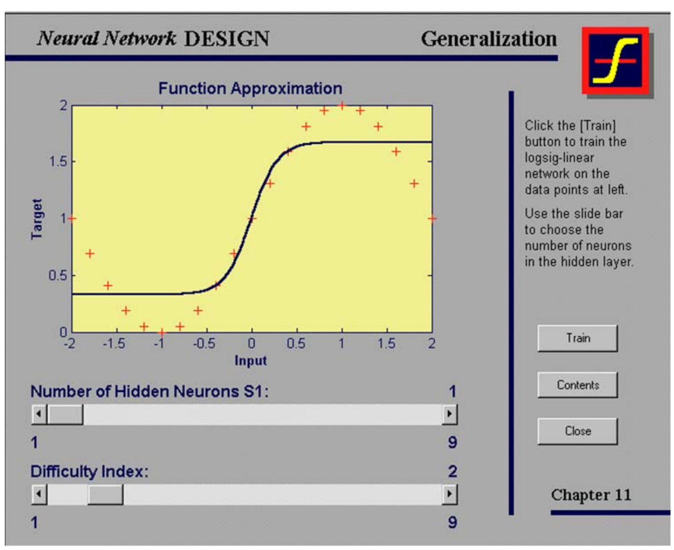
Average System Error (ASE)

- The algorithm continues for all set until the Average System Error (ASE) between the target output and computed output is close to the tolerance specified.

$$ASE = \frac{1}{2P}\sum_{p}\sum_{k}(t_{pk} - o_{pk})^2$$

31

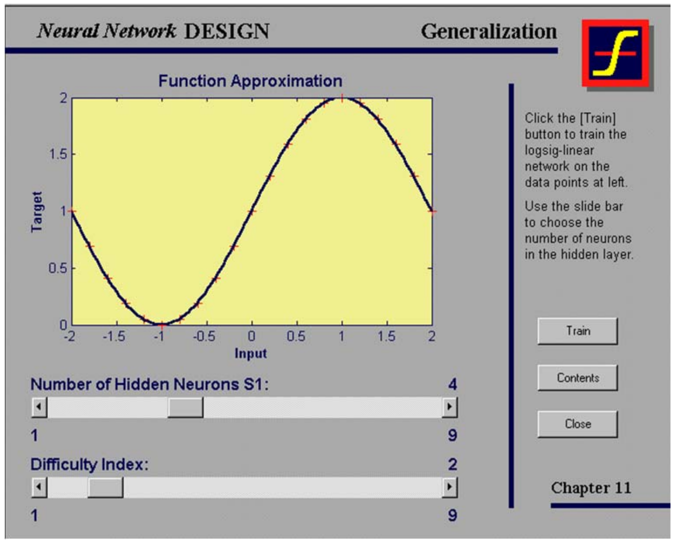
Generalization



21 Examples – Architecture 1-1-1

32

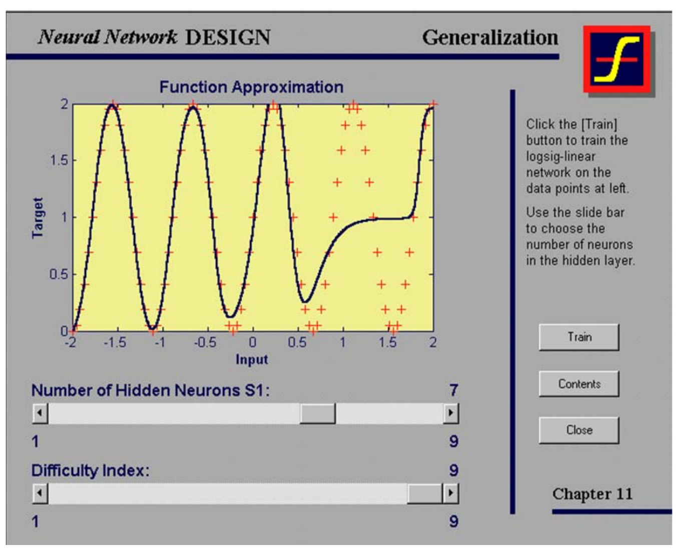
Generalization



21 Examples – Architecture 1-4-1

33

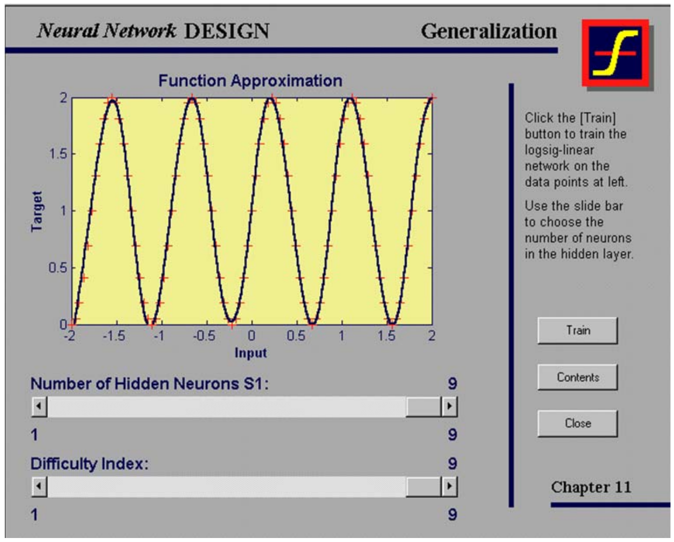
Generalization



90 Examples – Architecture 1-7-1

34

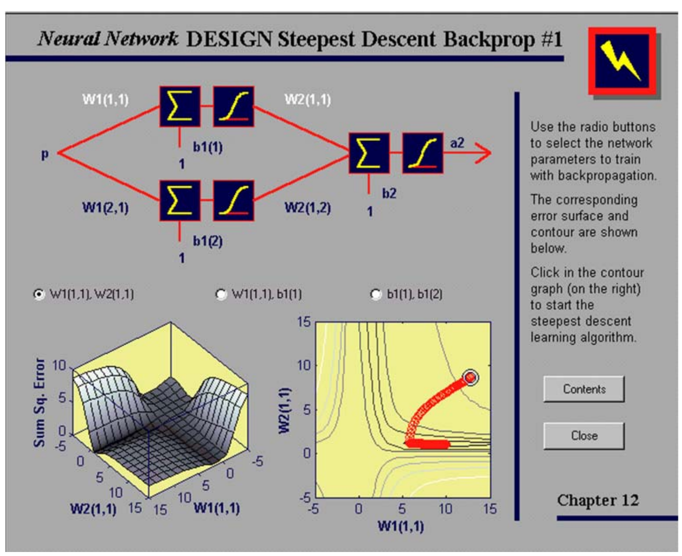
Generalization



90 Examples – Architecture 1-9-1

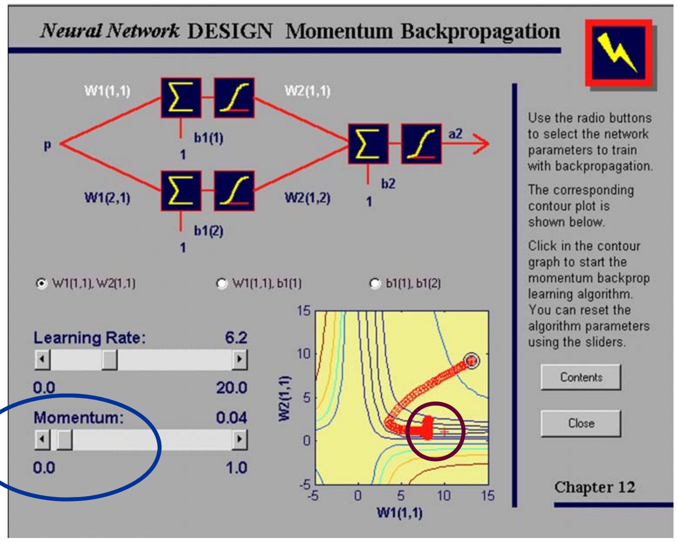
35

Steepest Descent BP



36

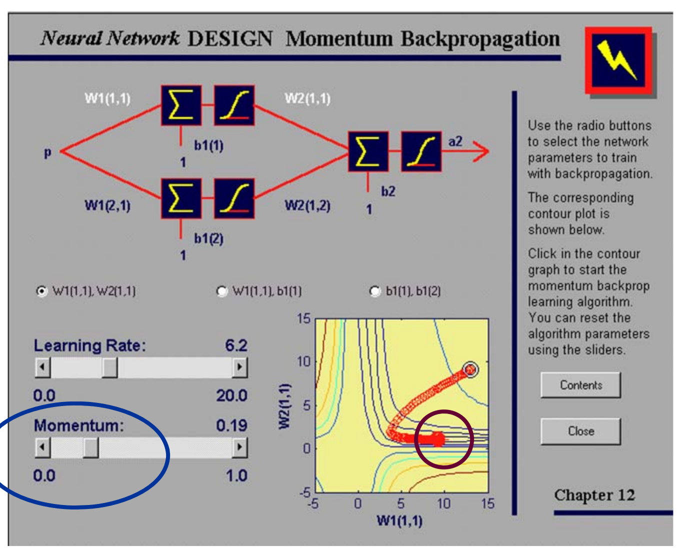
Momentum BP



Non-converged Solution

37

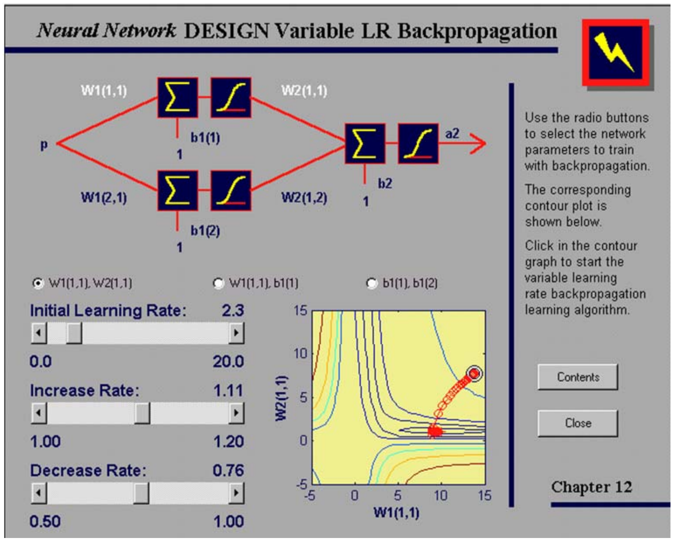
Momentum BP



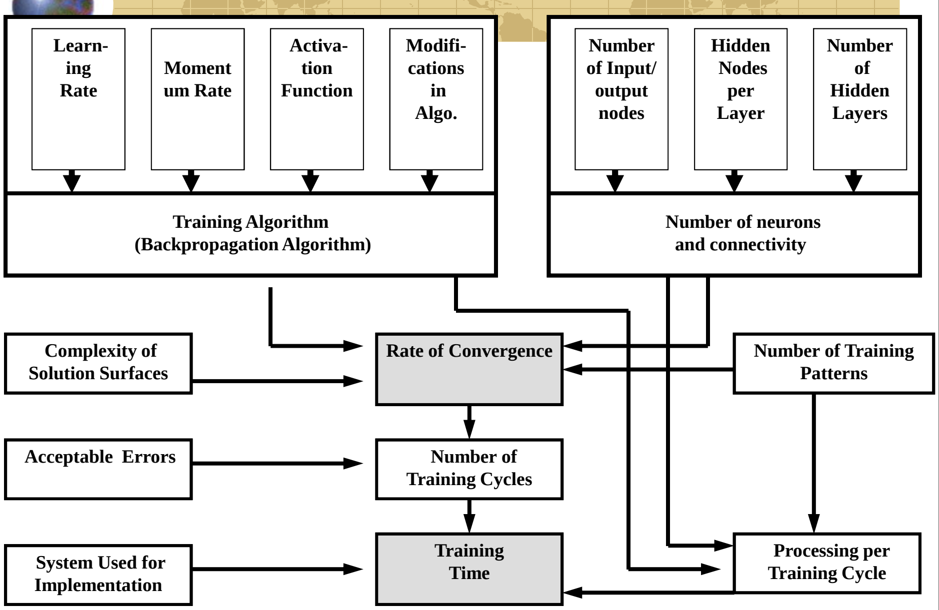
Converged Solution

38

Variable Learning Rate



39



40

Home Work

Perceptron and Multilayer Perceptron

Examples in MATLAB +

Neural Networks Tool Box Environment