Machine Learning

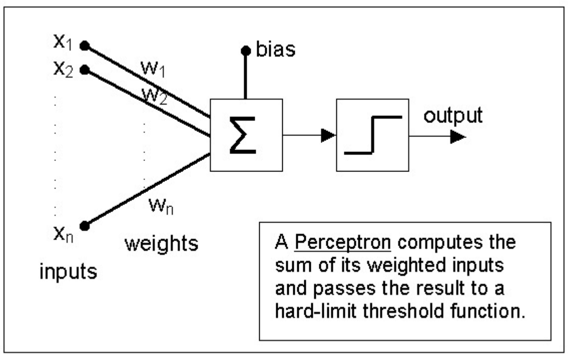
Neural Networks (I)
Perceptron

1

1

Perceptron - Basic

- Perceptron is a type of artificial neural network (ANN)



2

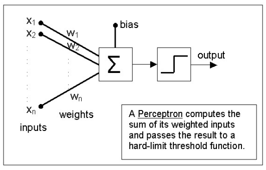
2

Perceptron - Operation

- It takes a vector of real-valued inputs, calculates a linear combination of these inputs, then output 1 if the result is greater than some threshold and -1 otherwise

$$R = w_0 + w_1x_1 + w_2x_2, \cdots, w_nx_n = w_0 + \sum_{i=1}^{n} w_ix_i$$

$$o = sign(R) = \begin{cases} +1; & \text{if } \ R > 0 \\ -1, & \text{otherwise} \end{cases}$$



3

Perceptron – Decision Surface

- Perceptron can be regarded as representing a hyperplane decision surface in the n-dimensional **feature space** of instances.

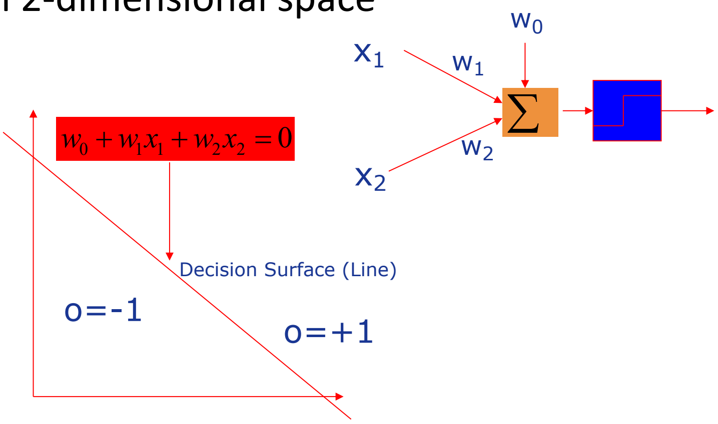
- The perceptron outputs a 1 for instances lying on one side of the hyperplane and a -1 for instances lying on the other side.

- This hyperplane is called the **Decision Surface**

4

Perceptron – Decision Surface

- In 2-dimensional space

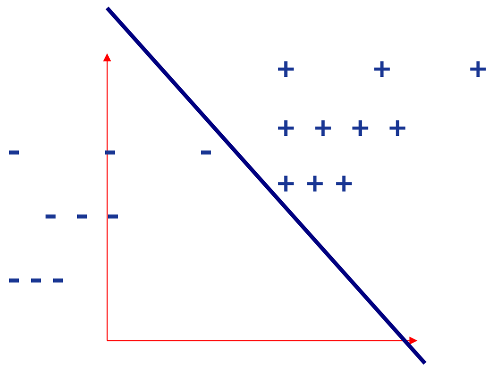


5

5

Perceptron – Representation Power

- The Decision Surface is linear
- Perceptron can only solve **Linearly Separable Problems**



6

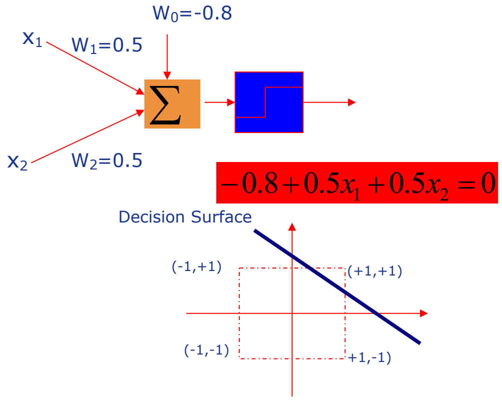
6

Perceptron – Representation Power

- Can represent many boolean functions: Assume boolean values of 1 (true) and -1 (false)

AND

x1	x2	D
-1	-1	-1
-1	+1	-1
+1	-1	-1
+1	+1	+1



7

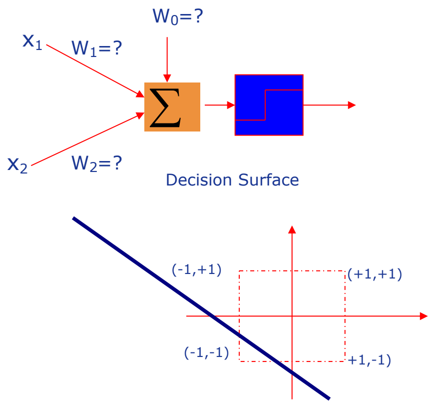
7

Perceptron – Representation Power

- Can represent many boolean functions: Assume boolean values of 1 (true) and -1 (false)

OR

x1	x2	D
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	+1

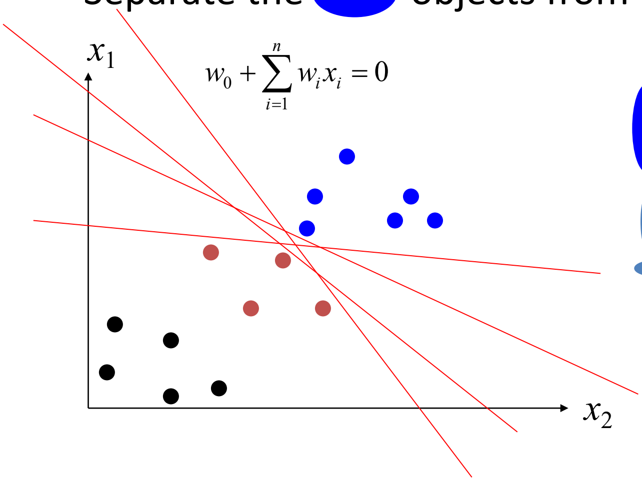


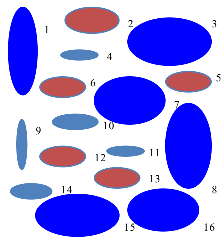
8

8

Perceptron – Representation Power

- Separate the objects from the rest





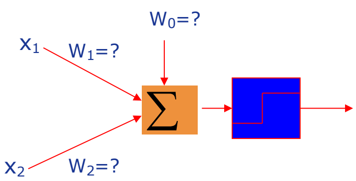
Elliptical blobs (objects)

Perceptron – Representation Power

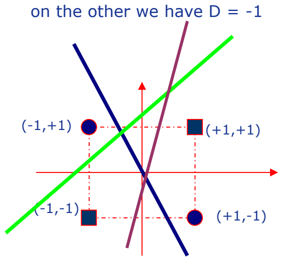
- Some problems are linearly non-separable

XOR

x1	x2	D
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1



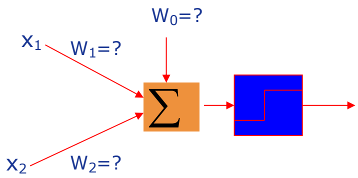
Decision Surface:
It doesn't matter where you place the line (decision surface), it is impossible to separate the space such that on one side we have D = 1 and on the other we have D = -1

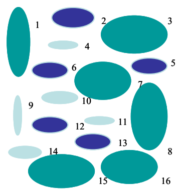


Perceptron Cannot Solve such Problem!

Perceptron – Training Algorithm

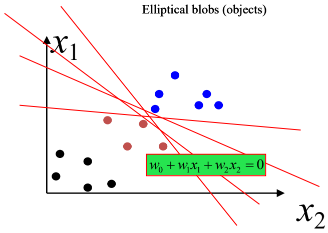
- Separate the objects from the rest





Elliptical blobs (objects)

We are given the training sample (experience) pairs (X, D), how can we determine the weights that will produce the correct +1 and -1 outputs for the given training samples?



11

11

Perceptron – Training Algorithm

- Training sample pairs (X, d), where X is the input vector, d is the input vector's classification (+1 or -1) is iteratively presented to the network for training, *one at a time*, until the process converges

12

12

Perceptron – Training Algorithm

- The Procedure is as follows

1. Set the weights to small random values, e.g., in the range (-1, 1)
2. Present X, and calculate

$$R = w_0 + \sum_{i=1}^{n} w_i x_i \qquad o = sign(R) = \begin{cases} +1; & \text{if} \quad R > 0 \\ -1, & \text{otherwise} \end{cases}$$

3. Update the weights

$$w_i \leftarrow w_i + \eta(d - o)x_i, i = 1, 2, \cdots, n$$

$0 < \eta < 1$ *is the training rate* $x_0=1$ (constant)

4. Repeat by going to step 2

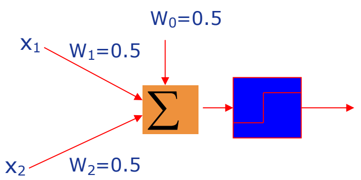
13

13

Perceptron – Training Algorithm

- Example

x1	x2	D
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	+1



$$w_i \leftarrow w_i + \eta(d - o)x_i, i = 1, 2, \cdots, n$$

14

14

Perceptron – Training Algorithm

- Convergence Theorem
  - The perceptron training rule will converge (finding a weight vector correctly classifies all training samples) within a finite number of iterations, **provided the training examples are linearly separable** and provided a sufficiently small η is used.

15

Further Reading

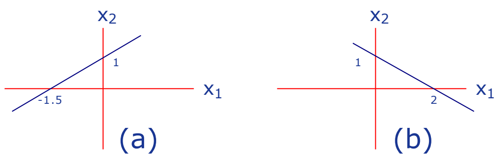
- T. M. Mitchell, Machine Learning, McGraw-Hill International Edition, 1997

Chapter 4

16

Tutorial/Exercise Questions

1. What is the weight values of a perceptron having the following decision surfaces



2. Design two-input perceptrons for implementing the following boolean functions

   AND, OR, NAND, NOR

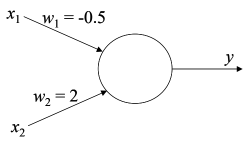
3. A single layer perceptron is incapable of learning simple functions such as XOR (exclusive OR). Explain why this is the case (hint: use the decision boundary)

17

17

Tutorial/Exercise Questions

4. A single layer Perceptron is as follows



a) Write down and plot the equation of the decision boundary of this device
b) Change the values of w1 and w2 so that the Perceptron can separate following two-class patterns

   Class 1 Patterns: (1, 2), (1.5. 2.5), (1, 3)
   Class 2 Patterns: (2, 1.5), (2, 1)

18

18

Machine Learning

Neural Networks (II)

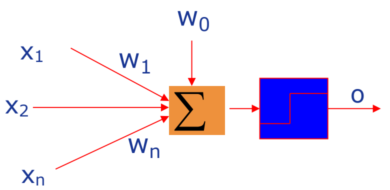
ADLINE and Delta Rule

19

19

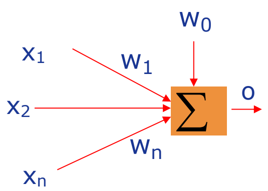
The ADLINE and Delta Rule

- Adaptive Linear Element (ADLINE) VS Perceptron



$R = w_0 + \sum_{i=1}^{n} w_i x_i$

$o = sign(R) = \begin{cases} +1; & \text{if } R > 0 \\ -1, & \text{otherwise} \end{cases}$



$o = w_0 + \sum_{i=1}^{n} w_i x_i$

20

20

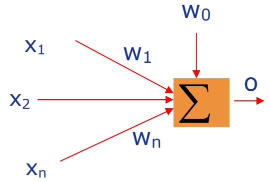
The ADLINE and Delta Rule

- Adaptive Linear Element (ADLINE) VS Perceptron
  - When the problem is not linearly separable, perceptron will fail to converge
  - ADLINE can overcome this difficulty by finding a best fit approximation to the target.

21

The ADLINE Error Function

- We have training pairs (X(k), d(k), k =1, 2, …, K), where K is the number of training samples, the training error specifies the difference between the output of the ALDLINE and the desired target
- The error is defined as



$$E(W) \equiv \frac{1}{2}\sum_{k=1}^{K}(d(k)-o(k))^2$$

$$o = w_0 + \sum_{i=1}^{n} w_i x_i$$

$$o(k) = W^T X(k)$$

is the output of presenting the training input X(k)

22

The ADLINE Error Function

- The error is defined as

$$E(W) \equiv \frac{1}{2}\sum_{k=1}^{K}(d(k)-o(k))^2$$

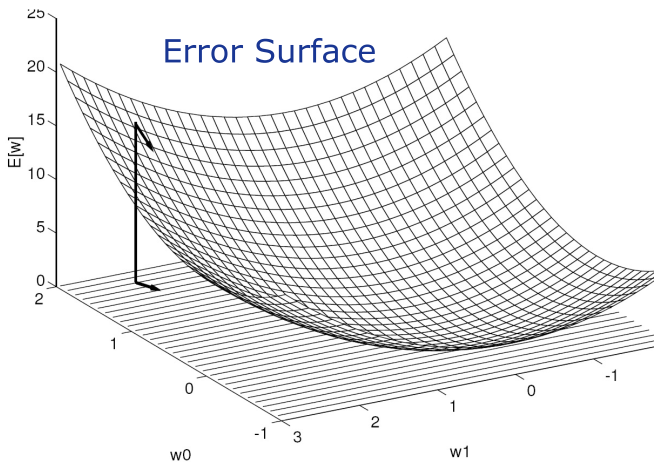
- The smaller E(W) is, the closer is the approximation

- We need to find W, based on the given training set, that minimizes the error E(W)

23

23

The ADLINE Error Function

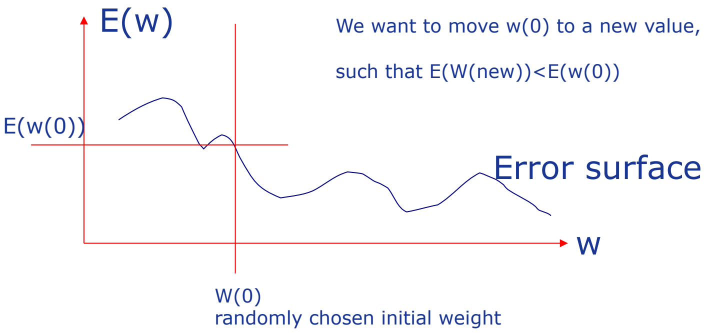


24

24

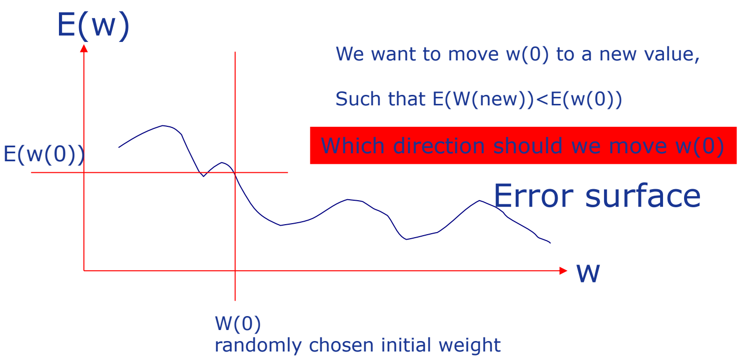
The Gradient Descent Rule

- An intuition
  - Before we formally derive the gradient decent rule, here is an intuition of what we should be doing



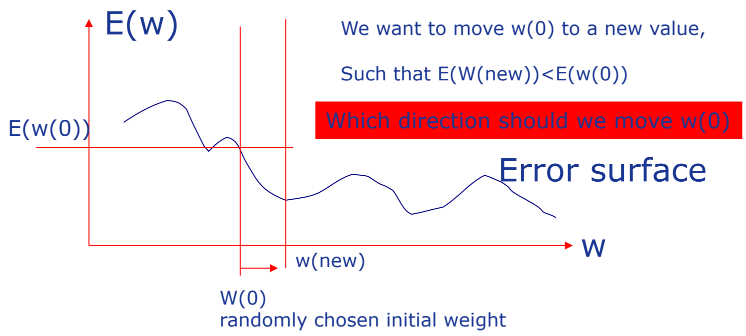
The Gradient Descent Rule

- An intuition
  - Before we formally derive the gradient decent rule, here is an intuition of what we should be doing



The Gradient Descent Rule

- An intuition
  – Before we formally derive the gradient decent rule, here is an intuition of what we should be doing

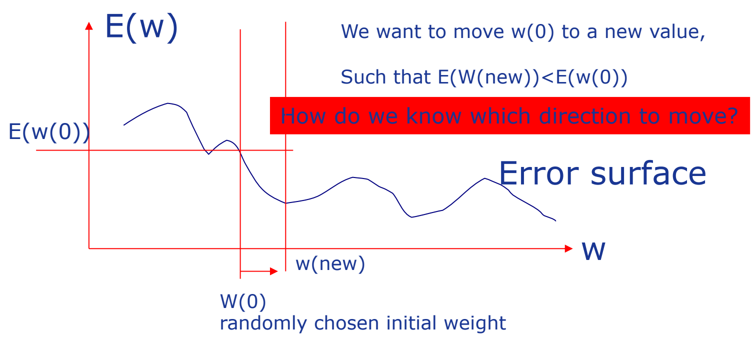


27

27

The Gradient Descent Rule

- An intuition
  – Before we formally derive the gradient decent rule, here is an intuition of what we should be doing

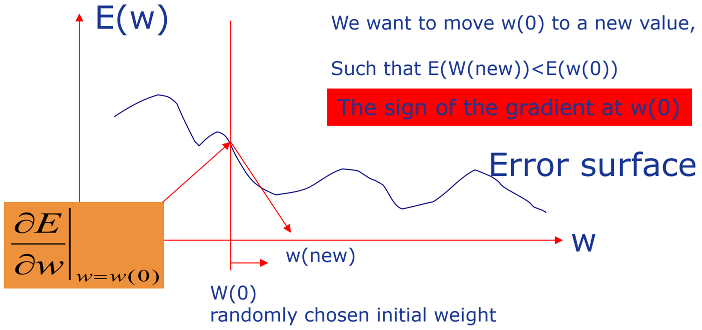


28

28

The Gradient Descent Rule

- An intuition
  - Before we formally derive the gradient decent rule, here is an intuition of what we should be doing

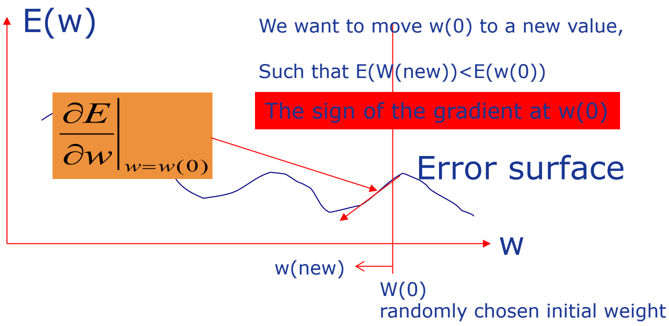


29

29

The Gradient Descent Rule

- An intuition
  - Before we formally derive the gradient decent rule, here is an intuition of what we should be doing



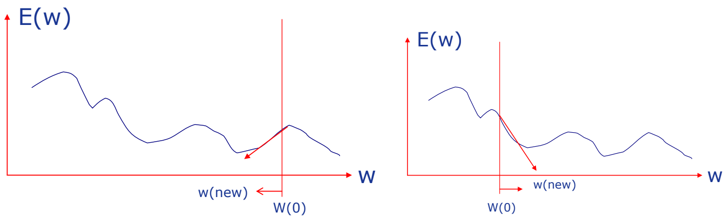
30

30

The Gradient Descent Rule

- An intuition
  - The intuition leads to

$$w(new) \leftarrow w(old) - \Delta sign\left(\left.\frac{\partial E}{\partial w}\right|_{w=w(old)} \right)$$



The Gradient Descent Rule

- Formal Derivation of Gradient Descent

$$\nabla E(W) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots \frac{\partial E}{\partial wn_n} \right]$$

  - The gradient of E is a vector, whose components are the partial derivatives of E with respect to each of the wi
  - The gradient specifies the direction that produces the speepest increase in E.
  - Negative of the vector gives the direction of steepest decrease.

The Gradient Descent Rule

- The gradient training rule is

$$W \leftarrow W - \eta \nabla E(W)$$

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

η is the training rate

The Gradient Descent Rule

- Gradient of ADLINE Error Functions

$$E(W) \equiv \frac{1}{2}\sum_{k=1}^{K}(d(k)-o(k))^2$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial w_i}\left(\frac{1}{2}\sum_{k=1}^{K}(d(k)-o(k))^2\right) \\
&= \frac{1}{2}\sum_{k=1}^{K}\left(\frac{\partial E}{\partial w_i}(d(k)-o(k))^2\right) \\
&= \frac{1}{2}\sum_{k=1}^{K}\left(2(d(k)-o(k))\frac{\partial E}{\partial w_i}(d(k)-o(k))\right) \\
&= \sum_{k=1}^{K}\left((d(k)-o(k))\frac{\partial E}{\partial w_i}\left(d(k)-w_0-\sum_{i=1}^{n}w_i x_i(k)\right)\right) \\
&= \sum_{k=1}^{K}((d(k)-o(k))(-x_i(k))) \\
&= -\sum_{k=1}^{K}(d(k)-o(k))x_i(k)
\end{aligned}$$

The Gradient Descent Rule

- ADLINE weight updating using gradient descent rule

$$w_i \leftarrow w_i + \eta \sum_{k=1}^{K} (d(k) - o(k)) x_i(k)$$

35

35

The Gradient Descent Rule

- Gradient descent training procedure
  - Initialise w_i to small vales, e.g., in the range of (-1, 1), choose a learning rate, e.g., η = 0.2
  - Until the termination condition is met, Do
    - For all training sample pair (X(k), d(k)), input the instance X(k) and compute

$$\delta_i = -\sum_{k=1}^{K} (d(k) - o(k)) x_i(k)$$

    - For each weight w_i, Do

$$w_i \leftarrow w_i - \eta \delta_i$$

Batch Mode:

gradients accumulated over **ALL** samples first

Then update the weights

36

36

Stochastic (Incremental) Gradient Descent

- Also called online mode, Least Mean Square (LMS), Widrow-Hoff, and Delta Rule
  - Initialise w_i to small vales, e.g., in the range of (-1, 1), choose a learning rate, e.g., $\eta = 0.01$ (should be smaller than batch mode)
  - Until the termination condition is met, Do
    - For EACH training sample pair (X(k), d(k)), compute

$$\delta_i = -\left(d(k) - o(k)\right)x_i(k)$$

    - For each weight w_i, Do

$$w_i \leftarrow w_i - \eta\delta_i$$

Online Mode:

Calculate gradient for **EACH** samples

Then update the weights

37

37

Training Iterations, Epochs

- Training is an iterative process; training samples will have to be used repeatedly for training
- Assuming we have K training samples [(X(k), d(k)), k=1, 2, …, K]; then an epoch is the presentation of all K sample for training once
  - First epoch: Present training samples: (X(1), d(1)), (X(2), d(2)), … (X(K), d(K))
  - Second epoch: Present training samples: (X(K), d(K)), (X(K-1), d(K-1)), … (X(1), d(1))
  - Note the order of the training sample presentation between epochs can (and should normally) be different.
- Normally, training will take many epochs to complete

38

38

Termination of Training

- To terminate training, there are normally two ways
  - When a pre-set number of training epochs is reached
  - When the error is smaller than a pre-set value

$$E(W) \equiv \frac{1}{2}\sum_{k=1}^{K}(d(k)-o(k))^2$$

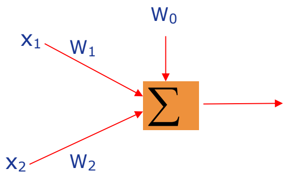
39

39

Gradient Descent Training

- A worked Example

x1	x2	D
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	+1



Initialization

$W_0(0)=0.1$; $W_1(0)=0.2$; $W_2(0)=0.3$;

$\eta=0.5$

40

40

Further Readings

- T. M. Mitchell, Machine Learning, McGraw-Hill International Edition, 1997

Chapter 4

- Any other relevant books/papers

41

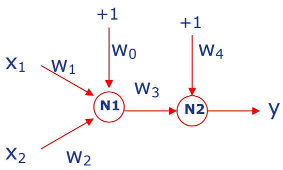
41

Tutorial/Exercise Questions

1. Derive a gradient descent training rule for a single unit with output y

$$y = w_0 + w_1 x_1 + w_1 x_1^2 + w_2 x_2 + w_2 x_2^2 + \cdots + w_n x_n + w_n x_n^2$$

2. A network consists of two ADLINE units N1 and N2 is shown as follows. Derive a delta training rule for all the weights



42

42

Tutorial/Exercise Questions

3. The connection weights of a two-input ADLINE at time n have following values:

   w_0 (n) = -0.5 w_1 (n) = 0.1 w_2 (n) = -0.3.

   The training sample at time n is:

   x_1 (n) = 0.6 x_2 (n) = 0.8

   The corresponding desired output is d(n) = 1

   a) Base on the Least-Mean-Square (LMS) algorithm, derive the learning equations for each weight at time n

   b) Assume a learning rate of 0.1, compute the weights at time (n+1):

      w_0 (n+1), w_1 (n+1), and w_2 (n+1).