MultiLayer Perceptrons (Çok Katmanlı Algılayıcılar)
Backpropagation (Geriye Yayınım Alg)

1

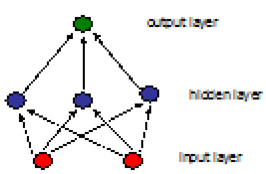
1

Capabilities of Multilayer Perceptrons

2

2

Multilayer Perceptron



In Multilayer perceptrons, there may be one or more hidden layer(s) which are called hidden since they are not observed from the outside.

3

3

Multilayer Perceptron

Each layer may have different number of nodes and different activation functions

Commonly:

- Same activation function within one layer
- Typically,
  - sigmoid activation function is used in the hidden units, and
  - sigmoid or linear activation functions are used in the output units depending on the problem (classification or function approximation)

In feedforward networks, activations are passed only from one layer to the next

4

4

Backpropagation

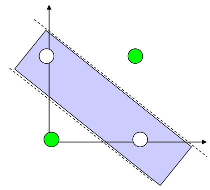
Capabilities of multilayer NNs were known, but a learning algorithm was introduced by Werbos (1974); made famous by Rumelhart and McClelland (mid 1980s - the PDP book)

- Started massive research in the area

XOR problem

Learning Boolean functions: 1/0 output can be seen as a 2-class classification problem

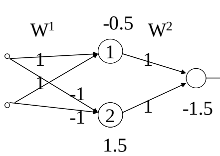
Xor can be solved by a 1-hidden layer network

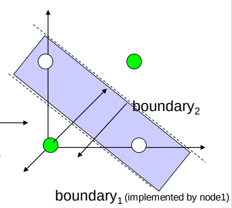


XOR problem

$W^1_1 = [1 \;\; 1 \; -0.5]$
$W^1_2 = [-1 \; -1 \;\; 1.5]$
$W^2 = [1 \;\; 1 \; -1.5]$

Notice how each node implements a decision boundary and the output node combines (AND) their result.





Capabilities (Hardlimiting nodes)

Single layer
- Hyperplane boundaries

1-hidden layer
- Can form any, possibly unbounded convex region

2-hidden layers
- Arbitrarily complex decision regions

Capabilities: Decision Regions

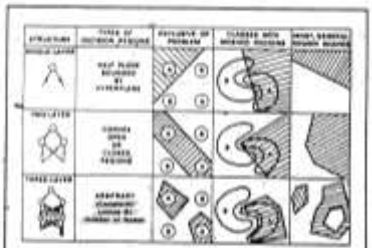


Figure 14. Types of decision regions that can be formed by single- and multi-layer perceptrons with one and two layers of hidden units and two inputs. Shading denotes decision regions for class A. Smooth closed contours bound input distributions for classes A and B. Nodes in all nets use hard limiting nonlinearities.

From Lippman's NN tutorial

When the hardlimiting non-linearities are replaced with sigmoidal nonlinearities, similar behavior is observed except that the decision regions are replaced by smooth curves instead of straight lines.

9

9

Capabilities (Hardlimiting nodes)

2-hidden layers (see Lippman, 1987):

- First hidden layer computes regions
- 2nd hidden layer computes an AND operation (one for each hypercube - worst case: #of disconnected regions)
  - about 1/3 the # of nodes in the first hidden layer
- Output layer computes an OR operation

No more than 2-hidden layers is ever required

10

10

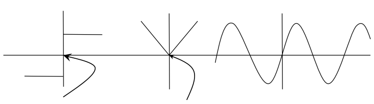
Capabilities

Every bounded continuous function can be approximated arbitrarily accurately by 2 layers of weights (1-hidden layer) and sigmoidal units (Cybenko 1989, Hornik et al. 1989)

- Discontinuities can be theoretically tolerated for most real life problems. Also, functions without compact support can be learned under some conditions.

All other functions can be learned by 2-hidden layer networks (Cybenko 1988)

- *Proff is based on Kolmogorov's Thm.*



Discontinuity

Continuous with discontinuous first derivative

11

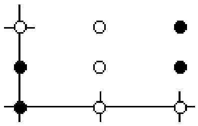
11

Self Study
Classification Example

12

12

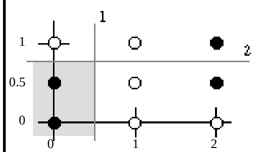
Example-Self Study



13

13

Elementary Decision Boundaries



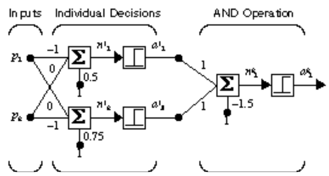
First Boundary:

$$a_1^1 = hardlim(\begin{bmatrix}-1 & 0\end{bmatrix}\mathbf{p} + 0.5)$$

Second Boundary:

$$a_2^1 = hardlim(\begin{bmatrix}0 & -1\end{bmatrix}\mathbf{p} + 0.75)$$

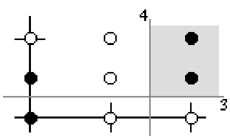
First Subnetwork



14

14

Elementary Decision Boundaries



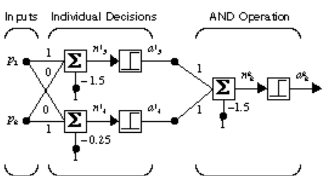
Third Boundary:

$$a_3^1 = hardlim(\begin{bmatrix}1 & 0\end{bmatrix}\mathbf{p} - 1.5)$$

Fourth Boundary:

$$a_4^1 = hardlim(\begin{bmatrix}0 & 1\end{bmatrix}\mathbf{p} - 0.25)$$

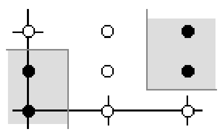
Second Subnetwork



15

15

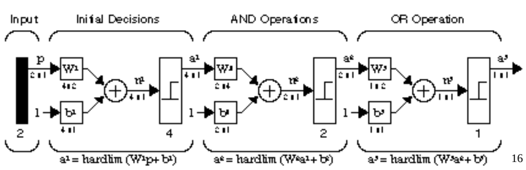
Total Network



$$\mathbf{W}^1 = \begin{bmatrix}-1 & 0\\ 0 & -1\\ 1 & 0\\ 0 & 1\end{bmatrix} \quad \mathbf{b}^1 = \begin{bmatrix}0.5\\ 0.75\\ -1.5\\ -0.25\end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\end{bmatrix} \quad \mathbf{b}^2 = \begin{bmatrix}-1.5\\ -1.5\end{bmatrix}$$

$$\mathbf{W}^3 = \begin{bmatrix}1 & 1\end{bmatrix} \quad \mathbf{b}^3 = \begin{bmatrix}-0.5\end{bmatrix}$$



16

16

Function Approximation & Network Capabilities

17

17

Function Approximation

Neural Networks are intrinsically function approximators:

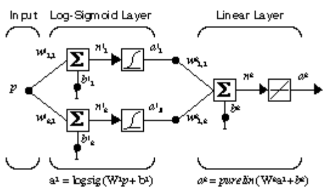
- we can train a NN to map real valued vectors to real-valued vectors

Function approximation capabilities of a simple network, in response to its parameters (weights and biases) are illustrated in the next slides

18

18

Function Approximation: Example



$$f^1(n) = \frac{1}{1 + e^{-n}}$$

$$f^2(n) = n$$

Layer number as superscripts

Nominal Parameter Values

$w^1_{1,1} = 10 \quad b^1_1 = -10 \quad w^2_{1,1} = 1$

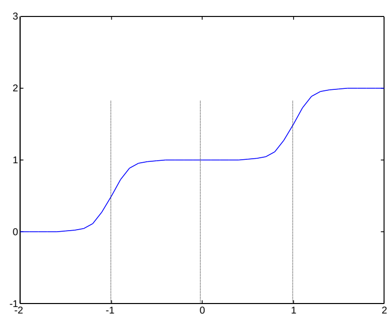
$w^1_{2,1} = 10 \quad b^1_2 = 10 \quad w^2_{1,2} = 1$

$b^2 = 0$

19

19

Nominal Response

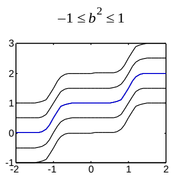


20

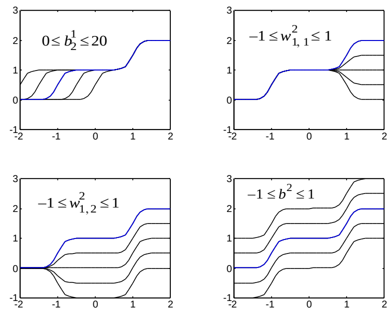
20

Parameter Variations

What would be the effect of varying the bias of the output neuron?



Parameter Variations



Performance Learning

Performance Learning

A learning paradigm where we adjust the network parameters (weights and biases) so as to optimize the "performance" of the network

- Need to define a performance index (e.g. mean square error)
- Search the parameter space to minimize the performance index with respect to the parameters

Performance Index Example

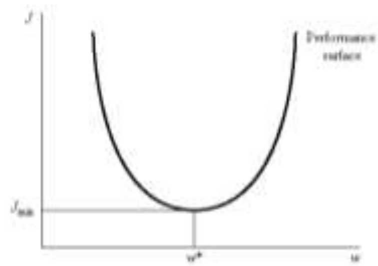
Example: Performance index of a linear perceptron defined to be the mean square error, over the input samples x_i is:

$$J = \frac{1}{2N}\sum_{i}\left(d_i - wx_i\right)^2$$

25

25

Performance surface

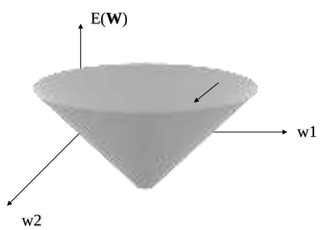


27

27

Performance surface with 2 weights

The idea is to find a minimum in the space of weights and the error function E:



28

28

Performance Optimization

Iterative minimization techniques:

- Define E(.) as the performance index
- Starting with an initial guess W(0), find W(n+1) at each iteration such that

  E(W(n+1)) < E(W(n))

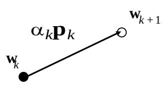
29

29

Basic Optimization Algorithm

Start with initial guess $\mathbf{w}_0$ and update the guess in each stage moving along the search direction:

$$\boxed{\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k}$$ or $$\boxed{\Delta\mathbf{w}_k = (\mathbf{w}_{k+1} - \mathbf{w}_k) = \alpha_k \mathbf{p}_k}$$



$\mathbf{p}_k$ - Search Direction

α_k - Learning Rate

30

30

Performance surface

The **gradient of the performance surface** is a vector (with the dimension of w) that:

- points toward the direction of maximum change,
- with a magnitude equal to the slope of the tangent of the performance surface.

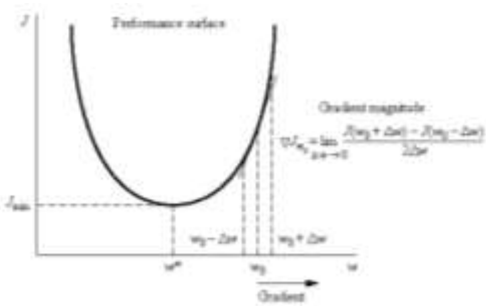
A ball rolling down the hill will always attempt to roll in the direction **opposite to the gradient arrow** (steepest descent).

- The slope at the bottom is zero, so the gradient is also zero (that is the reason the ball stops there).

31

31

Performance surface



32

32

Performance Optimization

Iterative minimization techniques: Steepest Descent

- Successive adjustments to W are in the direction of the steepest descent (direction opposite to the gradient vector)

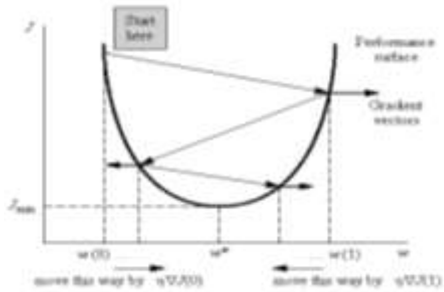
$$W(n+1) = W(n) - \eta g(n)$$

where $g(n) = \nabla E(W(n))$

33

33

Steepest Descent



34

34

Matrix Form

Gradient

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2}F(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n}F(\mathbf{x}) \end{bmatrix}$$

Hessian

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2}F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_1 \partial x_n}F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2}F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_2 \partial x_n}F(\mathbf{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2}F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_n^2}F(\mathbf{x}) \end{bmatrix}$$

35

35

Directional Derivatives

*i*th element of gradient is the first derivative (slope) of $F(\mathbf{x})$ along x_i axis:

$$\partial F(\mathbf{x}) / \partial x_i$$

i,i element of Hessian is the second derivative (curvature) of $F(\mathbf{x})$ along x_i axis:

$$\partial^2 F(\mathbf{x}) / \partial x_i^2$$

What is the derivative of a function along an arbitrary direction?

36

36

Directional Derivatives

First derivative of $F(\mathbf{x})$ along vector **p** is the projection of the gradient onto p:

$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}$$

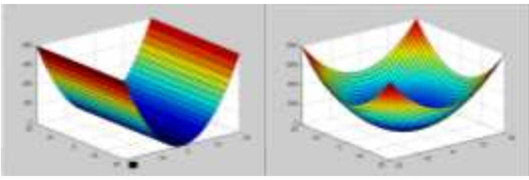
Which direction has the greatest slope?

- When the inner product of the direction vector and the gradient is a maximum
- I.e. When the direction vector is the same as the gradient

37

37

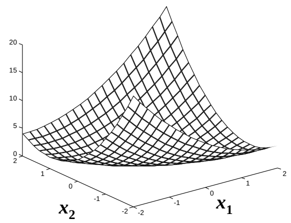
Two simple error surfaces (for 2 weights)

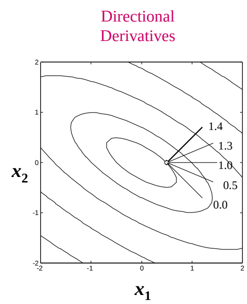


38

Plots

$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2$





39

Example

$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2$ For $\mathbf{x}^* = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$ $\mathbf{p} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} = \left.\begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2}F(\mathbf{x}) \end{bmatrix}\right|_{\mathbf{x}=\mathbf{x}^*} = \left.\begin{bmatrix} 2x_1 + 2x_2 \\ 2x_1 + 4x_2 \end{bmatrix}\right|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|} = \frac{\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}}{\left\|\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\|} = \frac{\begin{bmatrix} 0 \end{bmatrix}}{\sqrt{2}} = 0$$

40

Performance Optimization: Iterative Techniques Summary

Choose the next step so that the function decreases:

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$$

For small changes in $\mathbf{x}$ we can approximate $F(\mathbf{x})$ using the Taylor Series Expansion:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k$$

where

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$$

If we want the function to decrease, we must choose $\mathbf{p}_k$ such that:

$$\mathbf{g}_k^T \Delta\mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0$$

We can maximize the decrease by choosing:

$$\mathbf{p}_k = -\mathbf{g}_k$$

$$\boxed{\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k}$$

41

Example

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

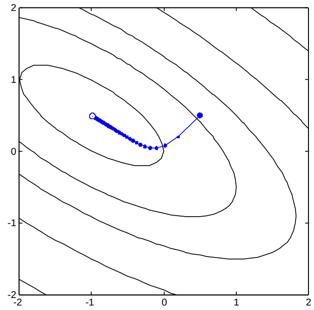
$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \qquad \alpha = 0.1$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2}F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \qquad \mathbf{g}_0 = \nabla F(\mathbf{x})\big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha\mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.1\begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha\mathbf{g}_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} - 0.1\begin{bmatrix} 1.8 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.08 \end{bmatrix}$$

Plot



Steepest Descent

Show that steepest descent satisfies the condition for iterative descent:

E(W(n+1)) < E(W(n))

Using Taylor series expansion:

E(W(n+1)) $\cong$ E(W(n)) - ηg^T(W(n)) g(W(n))

= E(W(n)) - η||g(W(n))||^2

Result follows since η||g(W(n))||^2 is >0 for $\eta > 0$