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

DEVAM

1

1

Minima and Maxima

go to next section

7

7

Minima

Strong (Local) Minimum

The point $\mathbf{x}^*$ is a strong minimum of $F(\mathbf{x})$ if a scalar $\delta > 0$ exists, such that $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$ for all $\Delta\mathbf{x}$ such that $\delta > \|\Delta\mathbf{x}\| > 0$.

Global Minimum

The point $\mathbf{x}^*$ is a unique global minimum of $F(\mathbf{x})$ if $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$ for all $\Delta\mathbf{x} \neq 0$.

Weak Minimum

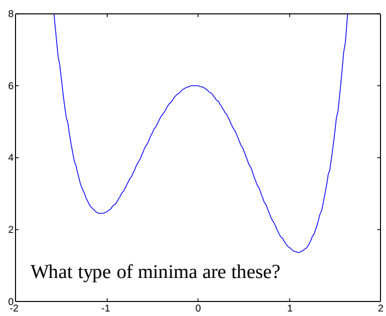
The point $\mathbf{x}^*$ is a weak minimum of $F(\mathbf{x})$ if it is not a strong minimum, and a scalar $\delta > 0$ exists, such that $F(\mathbf{x}^*) \leq F(\mathbf{x}^* + \Delta\mathbf{x})$ for all $\Delta\mathbf{x}$ such that $\delta > \|\Delta\mathbf{x}\| > 0$.

8

8

Scalar Example

$$F(x) = 3x^4 - 7x^2 - \frac{1}{2}x + 6$$

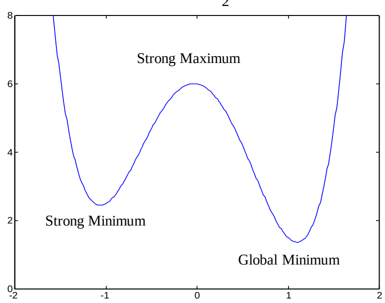


9

9

Scalar Example

$$F(x) = 3x^4 - 7x^2 - \frac{1}{2}x + 6$$

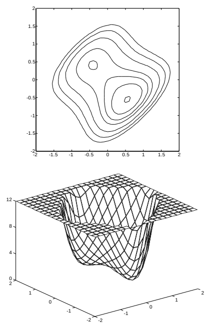


10

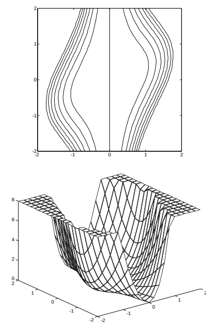
10

Vector Example

$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$$



$$F(\mathbf{x}) = (x_1^2 - 1.5x_1x_2 + 2x_2^2)x_1^2$$



11

11

Optimality Conditions

What are the conditions that need to be satisfied for minima?

Show using the Taylor series that the necessary condition for a minimum point (strong or weak) is:

$$\nabla F(\mathbf{x})\big|_{\mathbf{x} = \mathbf{x}^*} = \mathbf{0}$$

12

12

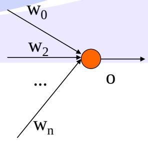
Gradient Descent

Delta Rule for Adaline (Linear Activation)

Backpropagation for MLP

15

15



To understand, consider simpler *linear unit*, where

$$o = w_0 + w_1x_1 + \cdots + w_nx_n$$

Let's learn w_i's that minimize the squared error

$$E[\vec{w}] \equiv \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

Where D is set of training examples

16

16

Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

17

17

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}$$

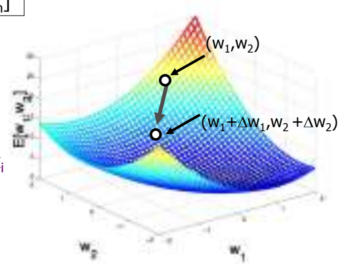
18

18

Gradient Descent: another slide explaining the same thing

Gradient:
$\nabla E[w]=[\partial E/\partial w_0, \ldots\ \partial E/\partial w_n]$

where
$\partial E/\partial w_i$
$=\partial/\partial w_i\ \Sigma_p(t_p-o_p)^2$
$= \Sigma_p 2(t_p - o_p)\partial/\partial w_i\ o_p$
$= \Sigma_p 2(t_p - o_p)\partial/\partial w_i\ \Sigma_i w_i x_i$
$= \Sigma_p 2(t_p - o_p)(-x_i)$



$\Delta w=-\eta\ \nabla E[w]$

19

19

Stochastic Approximation to Steepest Descent

Instead of updating every weight until all examples have been observed, we update on every example:

$\nabla w_i \cong \eta\,(t-o)\,x_i$ (not summing through all the patterns!)

In this case we update the weights "incrementally".

Remarks:

-When there are multiple local minima stochastic gradient descent may avoid the problem of getting stuck on a local minimum.

-Standard gradient descent needs more computation but can be used with a larger step size.

Learning algorithm using the Delta Rule

Algorithm for learning using the delta rule:

1. Assign random values to the weight vector
2. Continue until the stopping condition is met
   a) Initialize each ∇w_i to zero
   b) For each example p:
      Update ∇w_i:
      $\nabla w_i \mathrel{+}= (t_p - o_p)\, x_i$
   c) Update w_i:
      $w_i = w_i + \eta \nabla w_i$
3. Until error is small

Difficulties with Gradient Descent

There are two main difficulties with the gradient descent method:

1. Convergence to a minimum may take a long time.

2. There is no guarantee we will find the global minimum.

Backpropagation Algorithm

General Activation Function

Chain Rule

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw}$$

Example: $\boxed{f(n(w)) = \cos(e^{2w})}$ $\quad f(n) = \cos(n)$, $n = e^{2w}$

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} = (-\sin(n))(2e^{2w}) = (-\sin(e^{2w}))(2e^{2w})$$

Application to Gradient Calculation

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{m}} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \qquad \frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}$$

25

Transfer Function Derivatives

$$f'(n) = \frac{d}{dn}\left(\frac{1}{1+e^{-n}}\right) = \frac{e^{-n}}{(1+e^{-n})^2} = \left(1 - \frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right) = (1-a)(a)$$

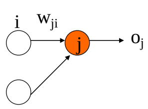
$$f'(n) = \frac{d}{dn}(n) = 1$$

26

Backpropagation

To calculate the partial derivative of E_p (error on pattern p) w.r.t a given weight w_{ji}, we have to consider whether this is the weight of an output or hidden node:

If w_{ji} is an **output** node weight:



$$\frac{dE_p}{dw_{ji}} = \frac{dE}{do_j} \times \frac{do_j}{dnet_j} \times \frac{dnet_j}{dw_{ji}}$$

$$\frac{dE_p}{dw_{ji}} = -(t_j - o_j) \times f'(net_j) \times o_i$$

$$net_j = \sum_i o_i w_{ji}$$

Note that o_i is the input to node j.

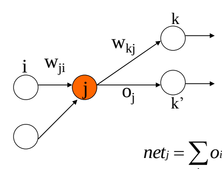
$$o_j = f(net_j)$$

$E_p = (t_p - o_p)^2$

27

Backpropagation

If w_{ji} is a **hidden** node weight:



$$\frac{dE_p}{dw_{ji}} = \frac{dE}{do_j} \times \frac{do_j}{dnet_j} \times \frac{dnet_j}{dw_{ji}}$$

$$= \frac{dE}{do_j} \times f'(net_j) \times o_i$$

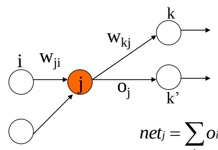
$$net_j = \sum_i o_i w_{ji}$$

$$o_j = f(net_j)$$

28

Backpropagation

If w_{ji} is a **hidden** node weight:



$$\frac{dE_p}{dw_{ji}} = \frac{dE}{do_j} \times \frac{do_j}{dnet_j} \times \frac{dnet_j}{dw_{ji}}$$

$$= \frac{dE}{do_j} \times f'(net_j) \times o_i$$

$$net_j = \sum_i o_i w_{ji}$$

$$o_j = f(net_j)$$

Note that as j is a hidden node, **we do not know its target.**
Hence, dE/do_j can only be calculated through **j's contribution to the derivative of E w.r.t net$_k$ at the output nodes:**

$$\frac{dE}{do_j} = \sum_k w_{kj} \times \frac{dE}{dnet_k}$$

29

29

Backpropagation Algorithm
Matrix Format
(skip)

go to next section

30

30

Network Complexity

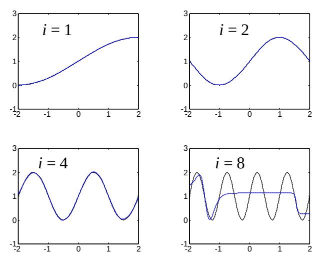
43

43

Choice of Architecture

$$g(p) = 1 + \sin\left(\frac{i\pi}{4}p\right)$$

1-3-1 Network (1 input, 3 hidden, 1 output nodes)

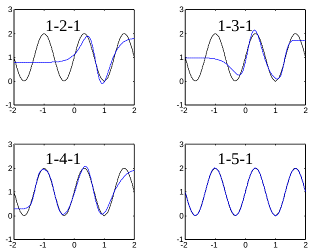


44

44

Choice of Network Architecture

$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right)$$



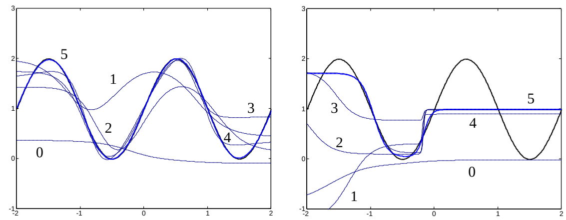
Residual error decreases with O(1/M) where M is the number of hidden units

45

45

Convergence in Time

$$g(p) = 1 + \sin(\pi p)$$



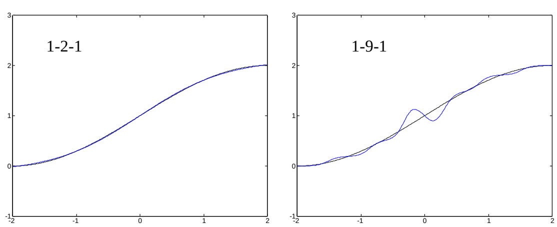
46

46

Generalization

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right) \qquad p = -2, -1.6, -1.2, \ldots, 1.6, 2$$



47

47

Next: Issues and Variations on Backpropagation

48

48