Machine intelligence

6th lecture

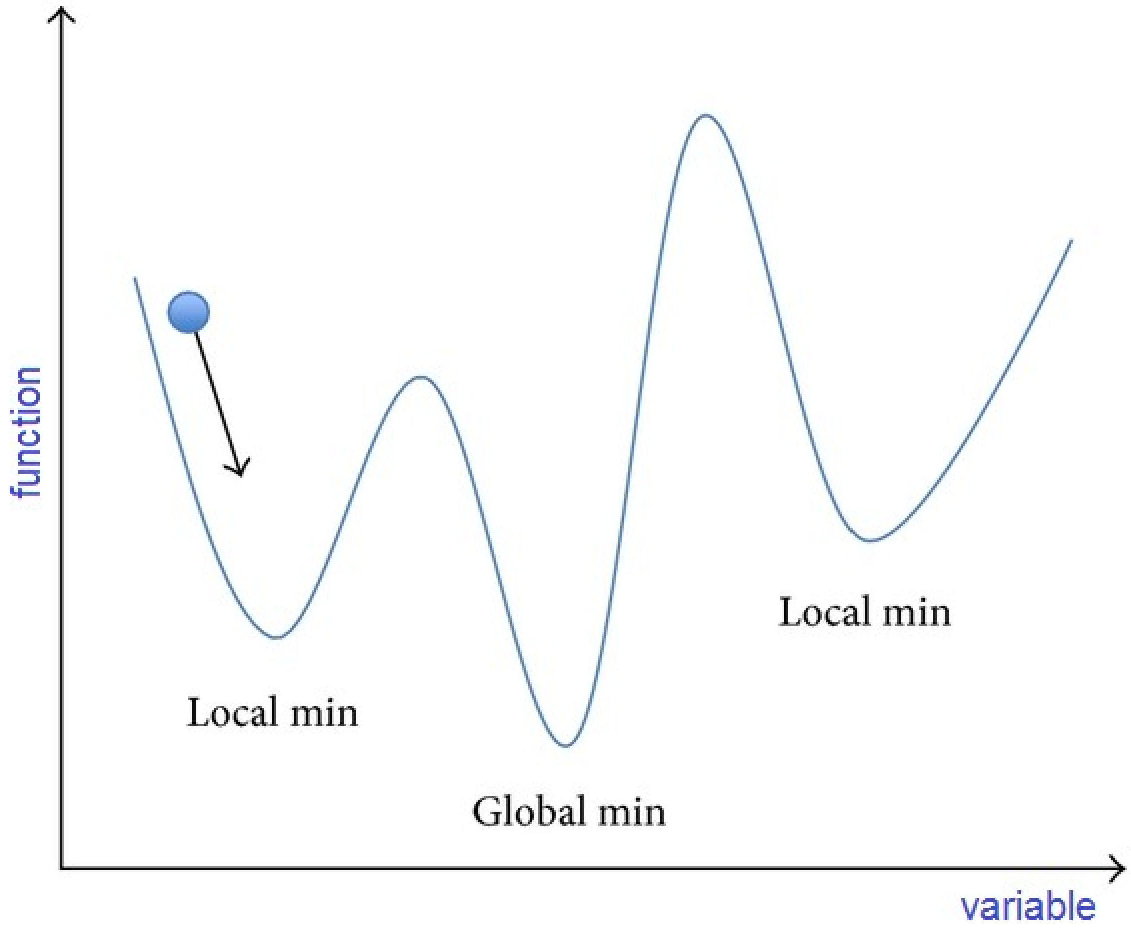
Calculus review and Widrow-Hoff learning

Dr. Ahmad Al-Mahasneh

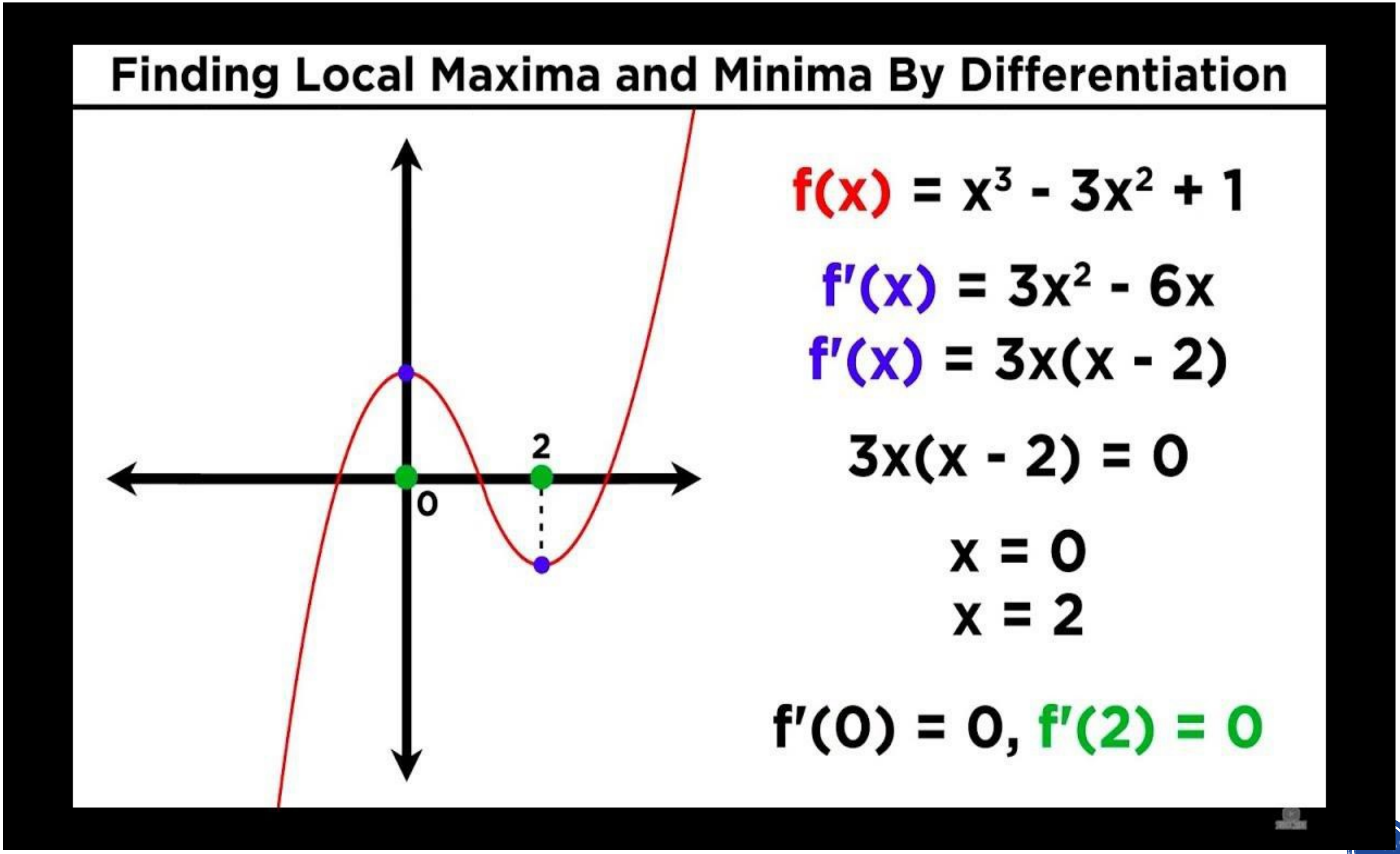
OUTLINE

- Global and local minima
- Steepest Descent Algorithm
- Newton-Raphson Algorithm
- Widrow-Hoff Algorithm
- Perceptron example

GLOBAL AND LOCAL MINIMA



DERIVATIVES



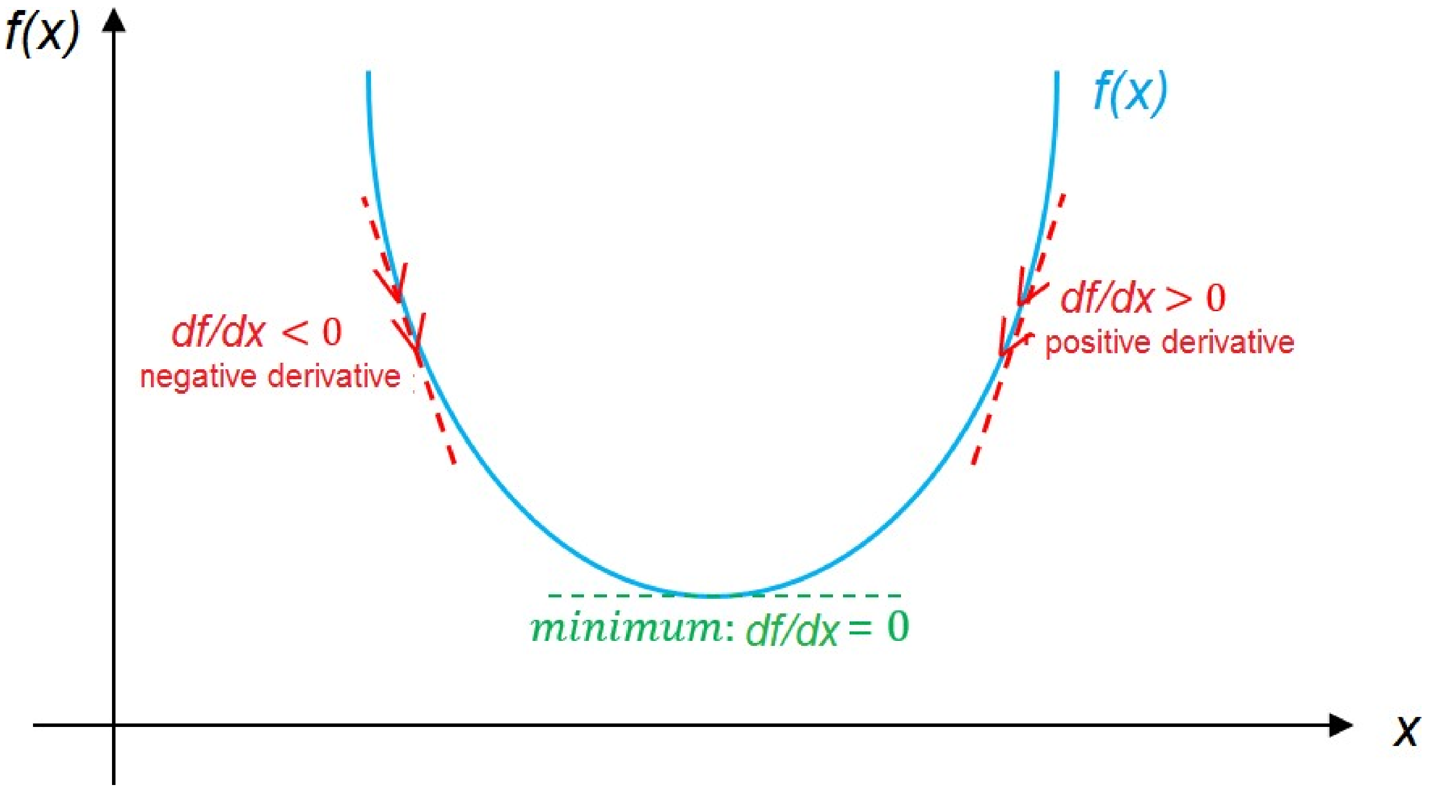
GRADIENT DESCENT

- Gradient Descent (GD) is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function.

- To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.

Steepest Descent

Steepest descent is one form of gradient descent algorithms



STEEPEST DESCENT ALGORITHM: SINGLE VARIABLE

- Start with initial value x^0
- Select step length 0 < $\alpha < 1$
- k=0;
- Repeat
  - k=k+1
  - Calculate the **derivative** $f'(x^k) = \frac{df(x=x^k)}{d}$
  - **Update** $x^{k+1} = x^k - \alpha f'(x^k)$
  - **Calculate** $f(x^{k+1})$
- Until $f(x^{k+1})$ < ε

STEEPEST DESCENT: EXAMPLE

function: $f(x) = (x-2)^2$

$x^0 = 0$ and $\alpha = 0.1$

$f'(x) = 2(x-2)$

Iteration 1

$f'(x=0) = 2(0-2) = -4$

$x^{k+1} = x^k - \alpha f'(x^k)$

$x = 0 - 0.1\ (-4)$ ➔ x = 0.4

Iteration 2

$f'(x=0.4) = 2(x-2) = 2(0.4-2) = -3.2$

$x = 0.4 - 0.1\ (-3.2)$ ➔ x = 0.72

Converges to x = 2 after 40 iterations

STEEPEST DESCENT ALGORITHM: MULTI-VARIABLE

- Start with **initial vector** x^0
- Select step length $0 < \alpha < 1$
- k=0;
- Repeat
  - k=k+1
  - Calculate the **gradient** $\nabla f\left(x^k\right)$
  - Update $x^{k+1} = x^k - \alpha \nabla f\left(x^k\right)$
  - Calculate $f(x^{k+1})$
- Until $f(x^{k+1}) < \varepsilon$

STEEPEST DESCENT: EXAMPLE

- Example One: min $f(x_1, x_2) = 8{x_1}^2 + 4x_1x_2 + 5{x_2}^2$
- Initial vector x = [10 10], use α = 0.056

$$g^k = \nabla f(x^k) = \begin{bmatrix} 16x_1 + 4x_2 \\ 4x_1 + 10x_2 \end{bmatrix}$$

$$x^1 = x^0 - \alpha^0 g^0$$

$$= \begin{bmatrix} 10 \\ 10 \end{bmatrix} - \alpha^0 \begin{bmatrix} 200 \\ 140 \end{bmatrix} \quad \Rightarrow x^1 = \begin{bmatrix} -1.20 \\ 2.16 \end{bmatrix} \qquad f(x^1) = 24.33$$

$$x^4 = \begin{bmatrix} 0.0021 \\ 0.0081 \end{bmatrix} \qquad f(x^4) \approx 0$$

NEWTON-RAPHSON ALGORITHM

- The goal is to estimate a solution of the equation f'(x)=0 by producing a sequence of approximations that approach the solution.

- **Step 1**. Start with k = 0 and initial point x^0
- **Step 2**. Update the variable: $x^{k+1} = x^k - \frac{f'(x)}{f''(x)}$
- **Step 3**. If f'(x) = 0 stop. Critical point has been found
  Else
    - k = k+1 and go back to Step 2.

NEWTON-RAPHSON: EXAMPLE ONE

- Find the minimum $f(x) = \frac{x^3}{3} - 3x^2 + 8x$ using Netwon-Raphson Method. Try initial point $x^0 = 1$ and $x^1 = 5$

- Solution: $f'(x) = x^2 - 6x + 8$ and $f''(x) = 2x - 6$
- Using initial point $x^0 = 1, f'(x=1) = 3, f''(x=1) = -4$
- Iteration 1: $x^1 = x^0 - \frac{f'(x^0)}{f''(x^0)} \Rightarrow x^1 = 1 - \frac{3}{-4} = 1.75$
- Iteration 2: $x^2 = x^1 - \frac{f'(x^1)}{f''(x^1)} \Rightarrow x^2 = 1.75 - \frac{0.5625}{-2.5} = 1.975$
- The solution will converge to x = 2
- Using initial point $x^0 = 5$, solution will converge to x = 4

NEWTON-RAPHSON WITH SEVERAL VARIABLES

$$x^{k+1} = x^k - H(x^k)^{-1}\nabla f(x^k)$$

$$f(x) = 2x_1^2 + 4x_1x_2^3 - 10x_1x_2 + x_2^2$$

$$x^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad f(x) = -3, \quad g(x) = \begin{bmatrix} -2 \\ 4 \end{bmatrix}, \quad H = \begin{bmatrix} 4 & 2 \\ 2 & 26 \end{bmatrix}$$

$$x^1 = \begin{bmatrix} 1.6 \\ 0.8 \end{bmatrix}, \quad f(x) = -3.7632, \quad g(x) = \begin{bmatrix} 0.448 \\ -2.112 \end{bmatrix}, \quad H = \begin{bmatrix} 4 & -2.32 \\ -2.32 & 32.72 \end{bmatrix}$$

$$x^2 = \begin{bmatrix} 1.522 \\ 0.859 \end{bmatrix}, \quad f(x) = -3.8443, \quad g(x) = \begin{bmatrix} 0.0343 \\ -0.0245 \end{bmatrix}, \quad H = \begin{bmatrix} 4 & -1.1447 \\ -1.1447 & 33.382 \end{bmatrix}$$

$$x^3 = \begin{bmatrix} 1.5138 \\ 0.8595 \end{bmatrix}, \quad f(x) = -3.8445, \quad g(x) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} 4 & -1.135 \\ -1.135 & 33.22 \end{bmatrix}$$

MATLAB CODE

$$f(x) = 2x_1^2 + 4x_1x_2^3 - 10x_1x_2 + x_2^2$$

```
x = [ 1 ; 1];
for i=1:10
    f = 2*x(1)^2 + 4*x(1)*x(2)^3 - 10*x(1)*x(2) + x(2)^2;
    g = [ 4*x(1) + 4*x(2)^3 - 10*x(2) ; 12*x(1)*x(2)^2 - 10*x(1) + 2*x(2)];
    H = [ 4  12*x(2)^2-10 ; 12*x(2)^2-10  24*x(1)*x(2)+2];
    x = x - inv(H)*g;
end;
```

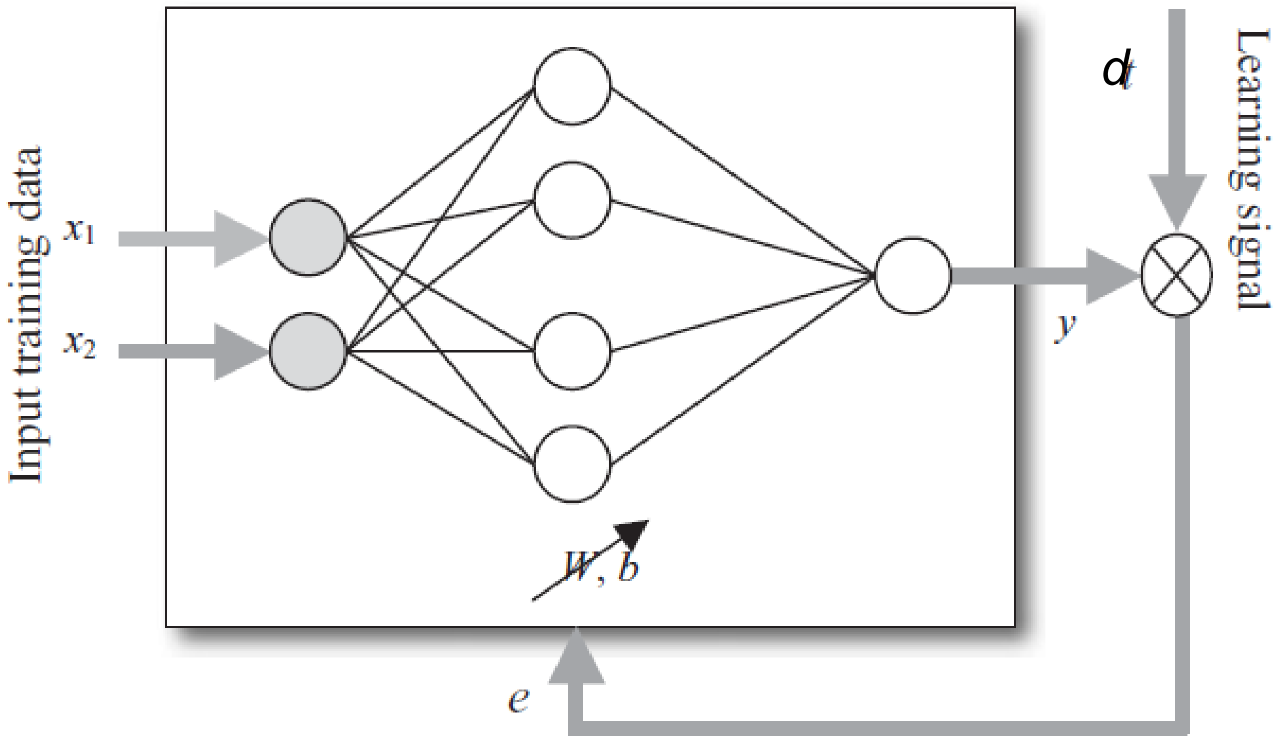
CONVERGENCE VS. DIVERGENCE

- These algorithms can **converge** to the desired solution as shown in previous slides.

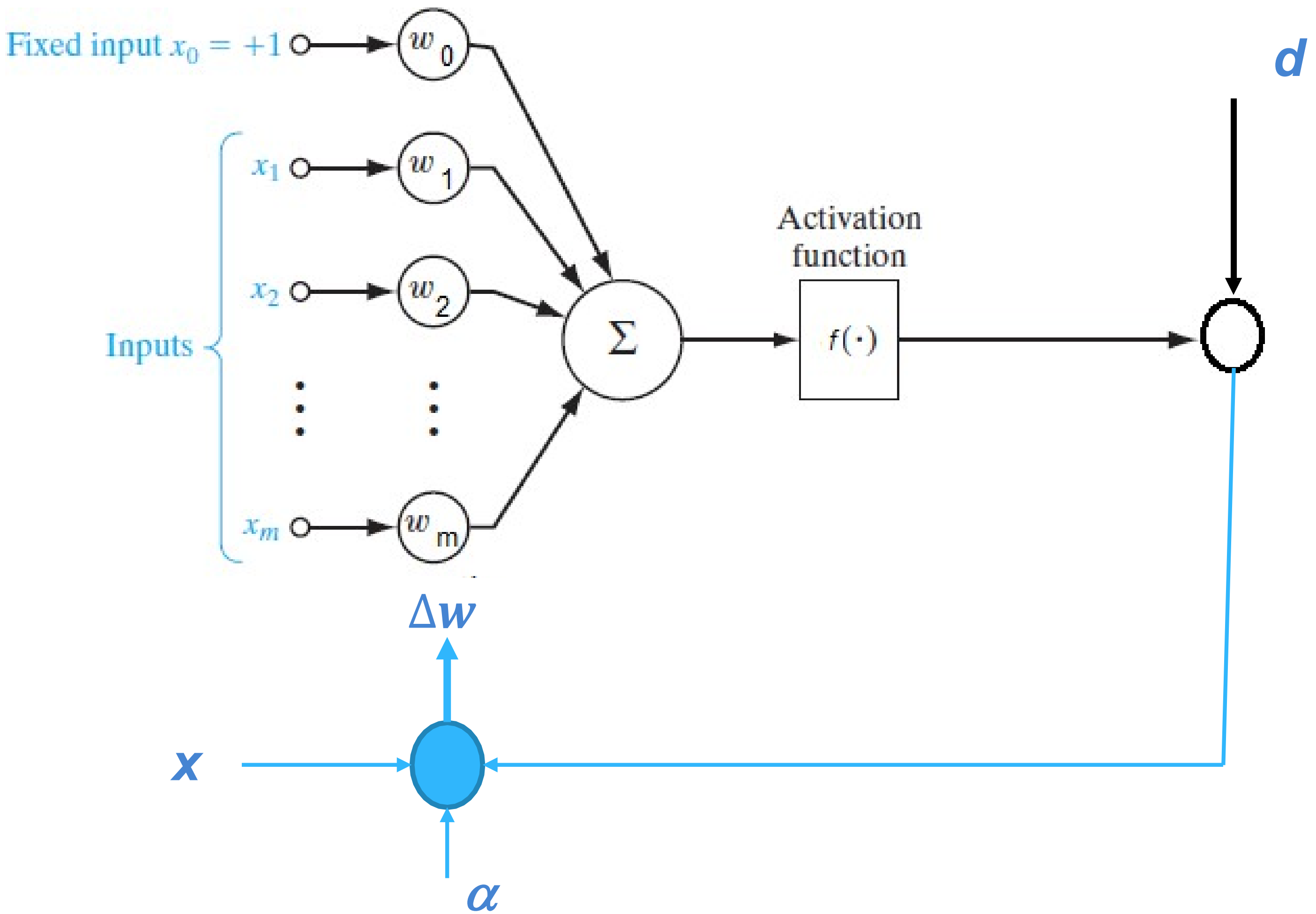
- However, they can also fail to converge or **diverge** to an undesired solution.

- The result depends on:
  - **The initial point**
  - **The step length value**

Supervised Learning



WIDROW-HOFF LEARNING ALGORITHM



WIDROW-HOFF LEARNING ALGORITHM

1. $net = \sum_{i=1}^{m} w_i x_i$

2. Using a linear activation function $y = f(net) = net$

3. The error between the network output and the desired output is:

$$e = \frac{1}{2}(d - y)^2$$

4. Using the chain rule, the derivative w.r.t. weights is:

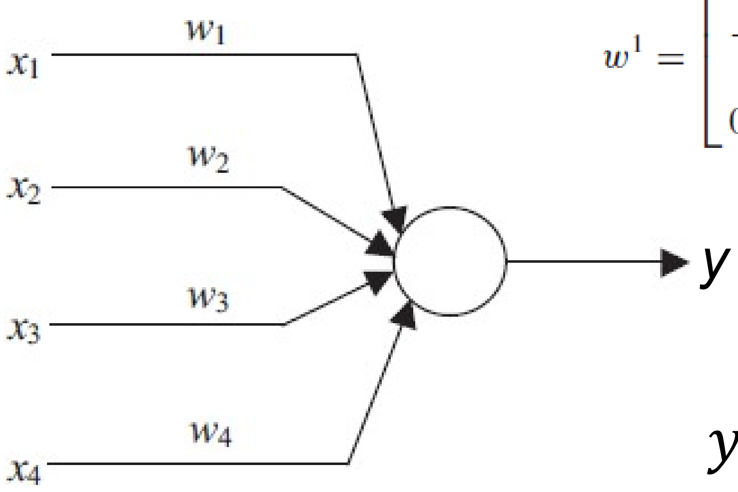
$$\frac{de}{dw} = \frac{de}{dy}\frac{dy}{dw} = -(d - y)x$$

5. Update the weighs using steepest descent:

$$w = w - \alpha \frac{\partial e}{\partial w_i}$$

Define $\Delta w = -\frac{\alpha \partial e}{\partial w_i}$ then $w = w + \Delta w$

EXAMPLE



$$w^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix},\ x^1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix},\ x^2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} \text{ and } x^3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$

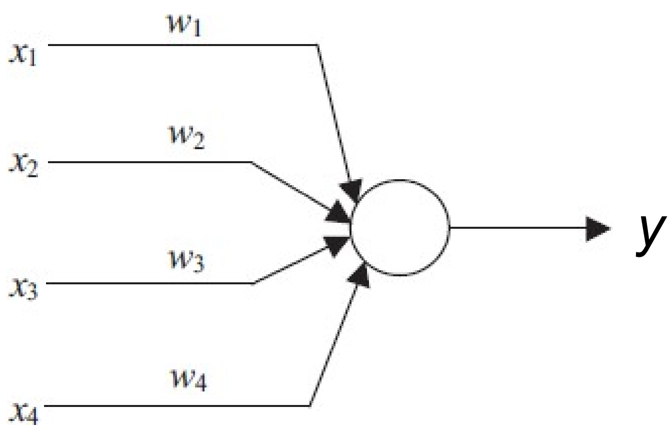
Desired output, d = [1 -1 0]

$$y = f(net) = net$$

Use ***Widrow-Hoff*** learning to update the weights

Let $\alpha = 1$

EXAMPLE



$$w^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix},\ x^1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix},\ x^2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} \text{ and } x^3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$

Desired output, d = [*1* -1 0]

Iteration One

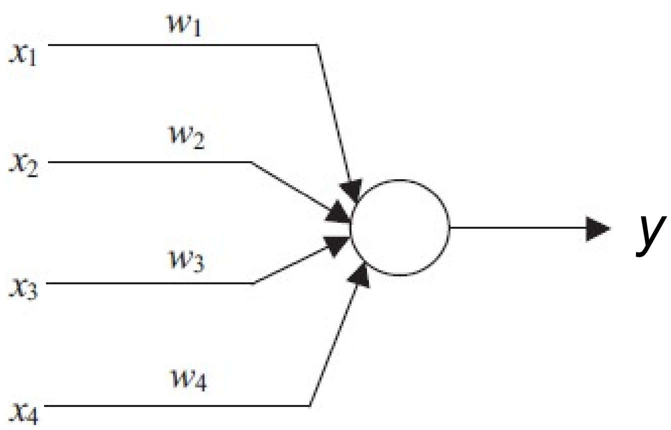
$$net^1 = w^1 x^1 = \begin{bmatrix} 1 & -1 & 0 & .5 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = 3$$ ➔ y = 3

$$\Delta w^1 = \alpha\,(d^1 - y^1)\,x^1 = 1 * (1 - 3) \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \\ -3 \\ 0 \end{bmatrix}$$

$$w^2 = w^1 + \Delta w^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} + \begin{bmatrix} -2 \\ 4 \\ -3 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ -3 \\ 0.5 \end{bmatrix}$$

EXAMPLE

$$w^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix},\ x^1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix},\ x^2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} \text{ and } x^3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$



Desired output, d = [1 ***-1*** 0]

Iteration Two

$$net^2 = [-1 \quad 3 \ -3 \quad 0.5] \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} = 2.75 \Rightarrow y^2 = 2.75$$

$$\Delta w^2 = \alpha(d^2 - y^2)x^2 = 1\,(-1 - 2.75\,) \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} = \begin{bmatrix} -3.75 \\ 1.875 \\ 7.5 \\ 5.62 \end{bmatrix}$$

$$w^3 = w^2 + \Delta w^2 = \begin{bmatrix} -1 \\ 3 \\ -3 \\ 0.5 \end{bmatrix} + \begin{bmatrix} -3.75 \\ 1.875 \\ 7.5 \\ 5.62 \end{bmatrix}$$

CONCLUSIONS

- Widrow-Hoff learning algorithm is a simple supervised learning algorithm used for Perceptron

- The weight change at each iteration by minimizing an error function between the perceptron output and the desired output

- Steepest descent algorithms are iterative procedures that are used to find the minimum of functions.

- These algorithms update the variables in the direction of the negative derivative (for single variable) or gradient (for multi-variables)