Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program

605-447/625-438

Dr. Mark Fleischer

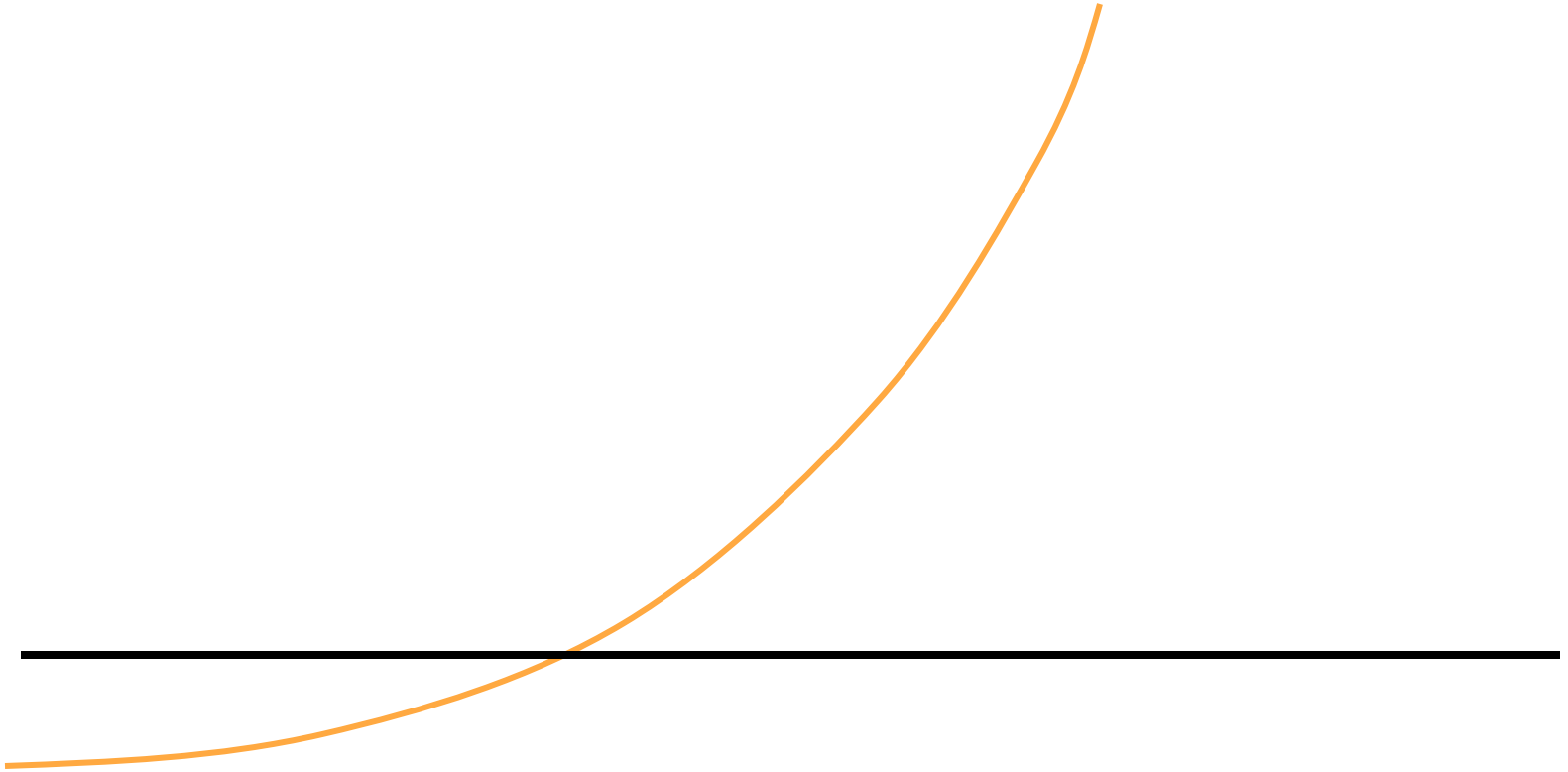
Module 4.1: Training Perceptrons

This Sub-Module Covers …

- A dynamical systems approach and iterative method for solving root finding problems.
- The Method of Steepest Descent.
- A similar dynamical systems approach for training a Perceptron.

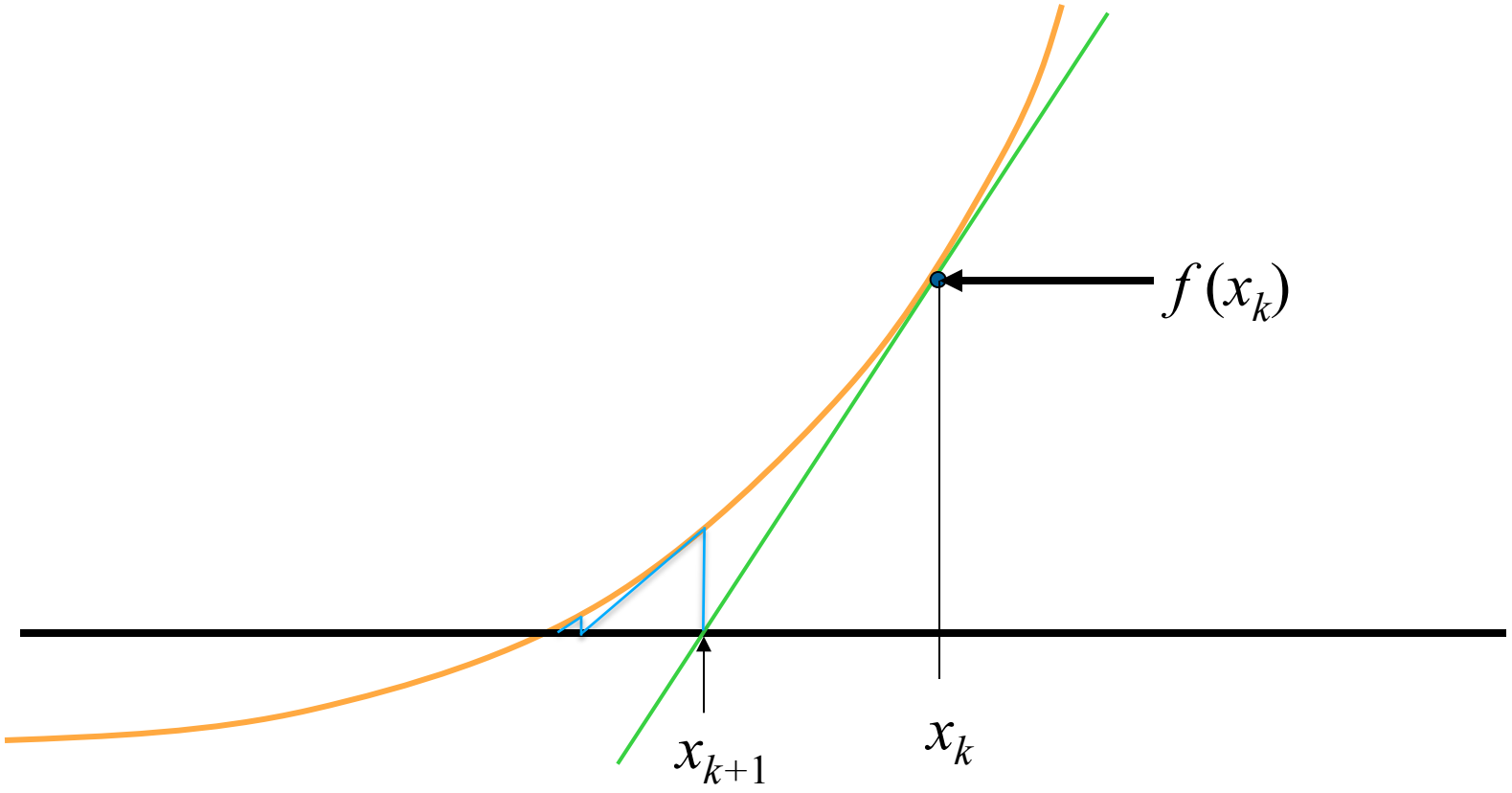
Look at Taylor Series

$$f(x) = f(x_0) + f'(x_0)(x - x_0)$$



Look at Taylor Series

$$0 = f(x_k) + f'(x_k)(x_{k+1} - x_k)$$



A Little Geometry and Algebra

$$\frac{\left(f(x_k)-0\right)}{\left(x_k-x_{k+1}\right)}=f'(x_k)$$

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$

Newton's Method

An Example of Using an Iterative Method

$$f(x) = x^n - C$$

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

$$0 = x^2 - 2$$

$$x^2 = 2$$

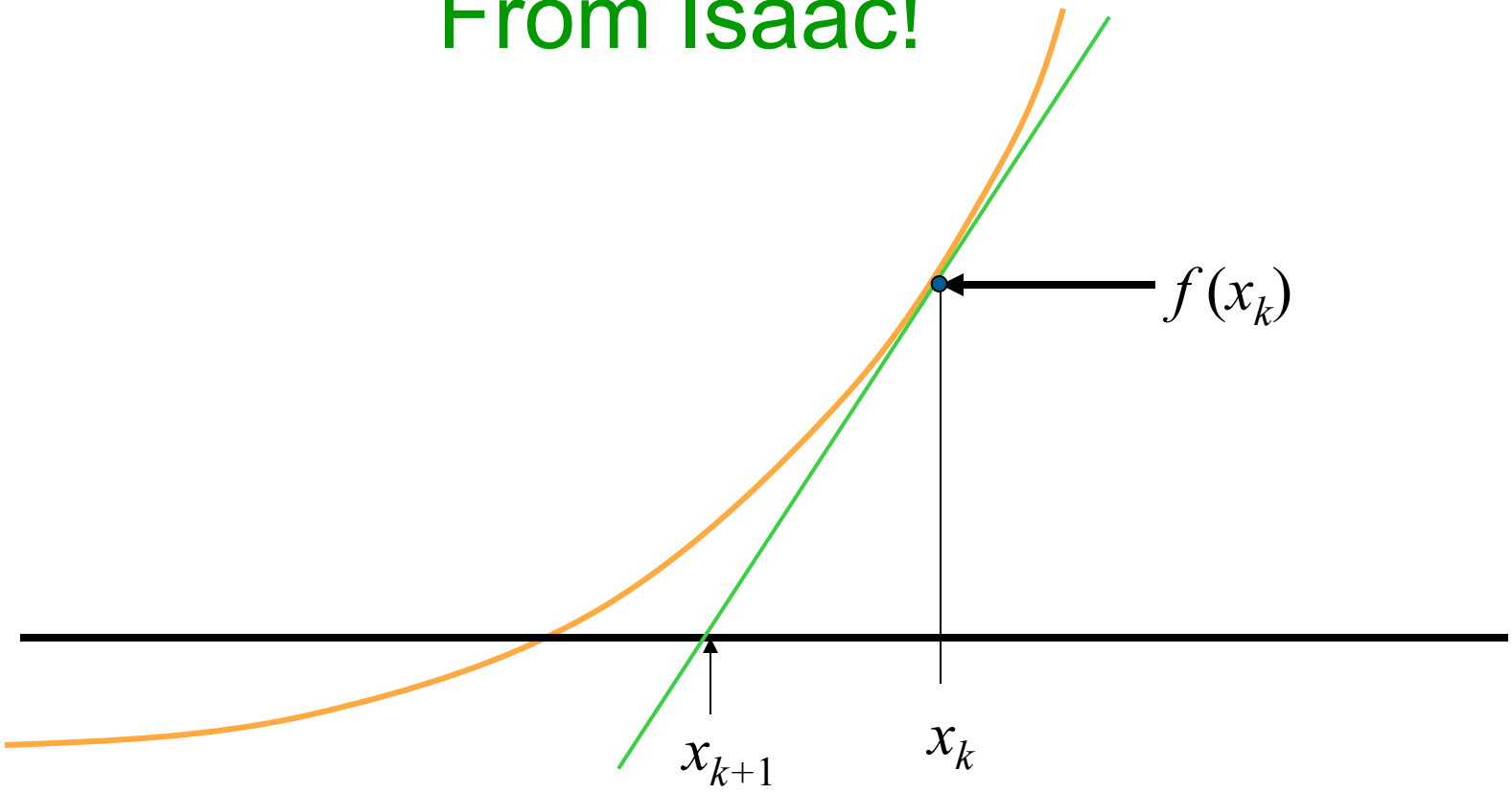
$$x = \pm\sqrt{2}$$

$$x_{k+1} = x_k - \left(\frac{x_k^2 - 2}{2x_k} \right)$$

A Numerical Example

		100	50.01
5	2.7	50.01	25.024996
2.7	1.72037037	25.024996	12.55245805
1.72037037	1.441455368	12.55245805	6.355894695
1.441455368	1.414470981	6.355894695	3.335281609
1.414470981	1.414213586	3.335281609	1.967465562
1.414213586	1.414213562	1.967465562	1.49200089
1.414213562	1.414213562	1.49200089	1.416241332
1.414213562	1.414213562	1.416241332	1.414215014
		1.414215014	1.414213562
		1.414213562	1.414213562
		1.414213562	1.414213562

Let Takes Some Inspiration From Isaac!

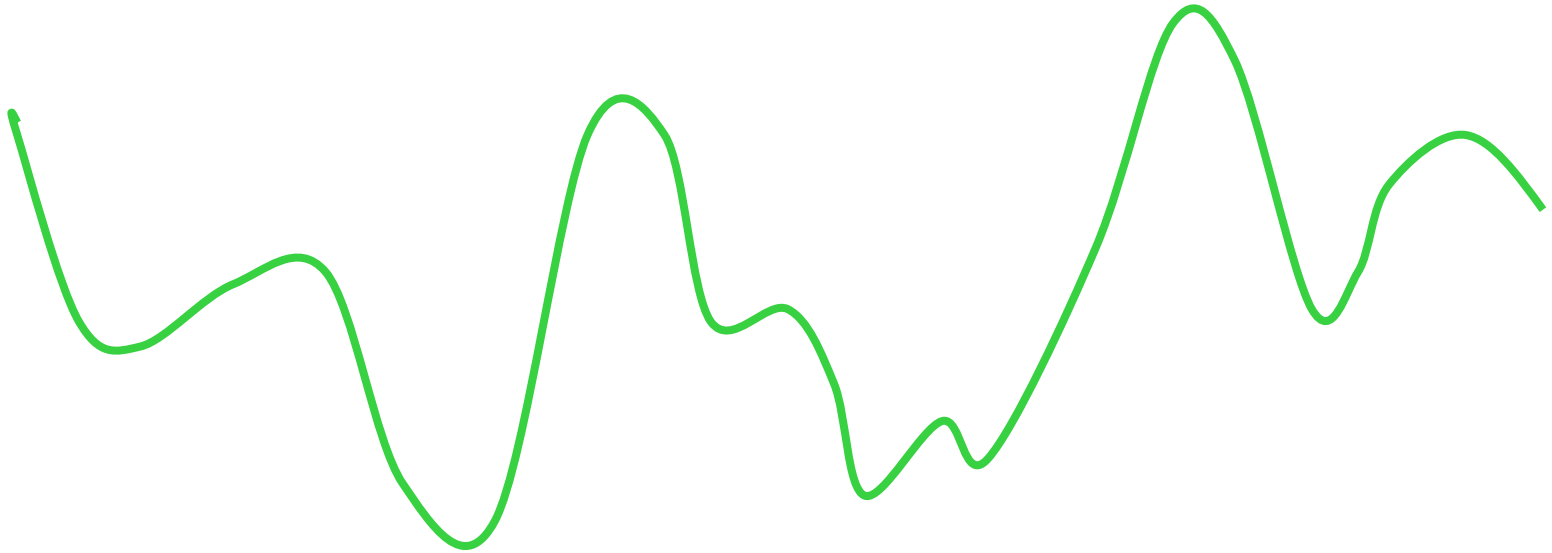


$$x_{k+1} = x_k - \eta f'(x_k)$$

Finding a Minimum

- In Neural Networks, we want to set the weights so the Activity has the correct slope/orientation so we can take advantage of linear separability.
- We can define some 'function' that is affected by the weights.
- Such a function can relate what we want the output to be with what the output is … an error function.
- But assuming such a function is possible, we cannot simply set derivatives to zero.
- Why?

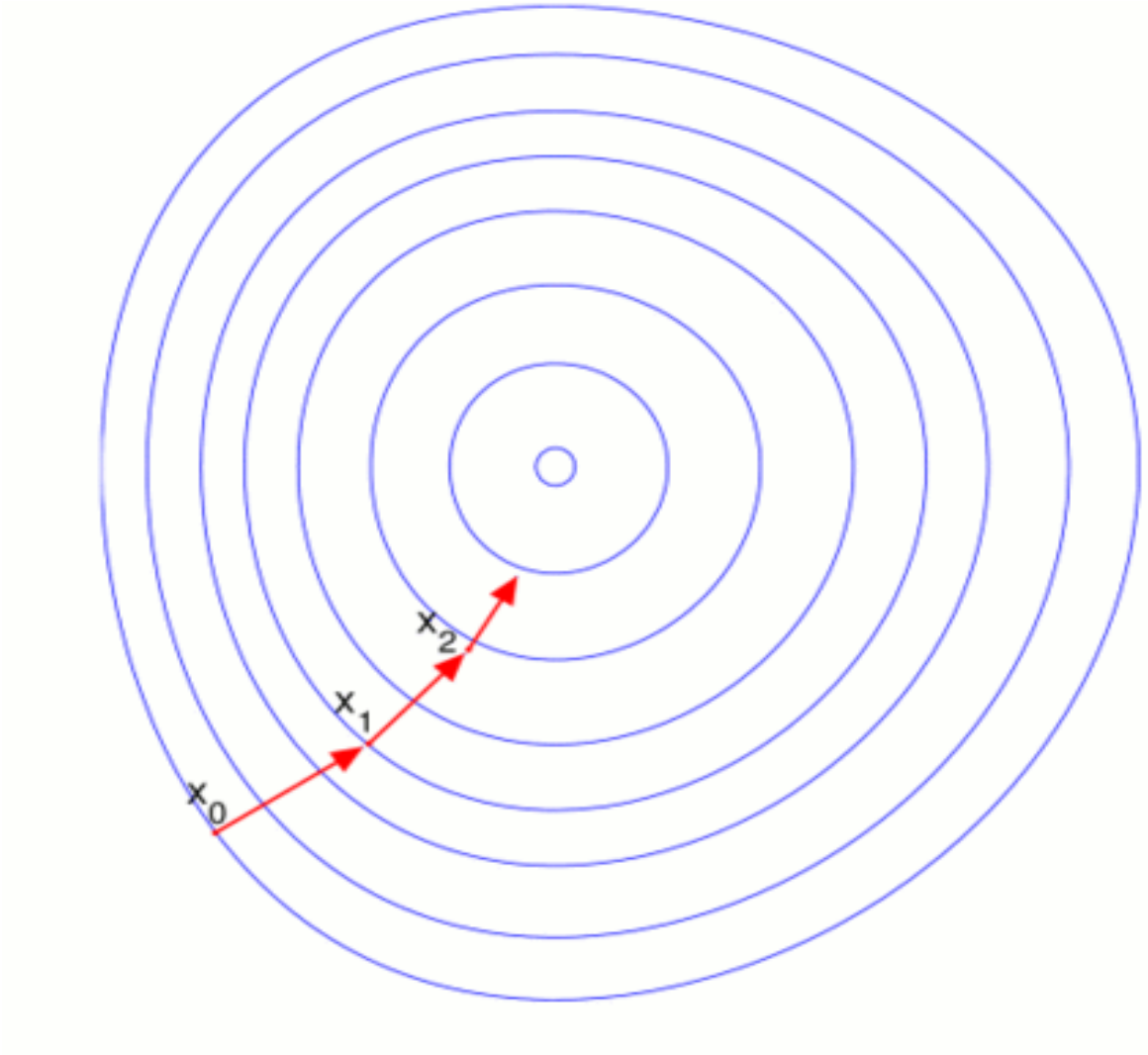
Minima, where for art thou?



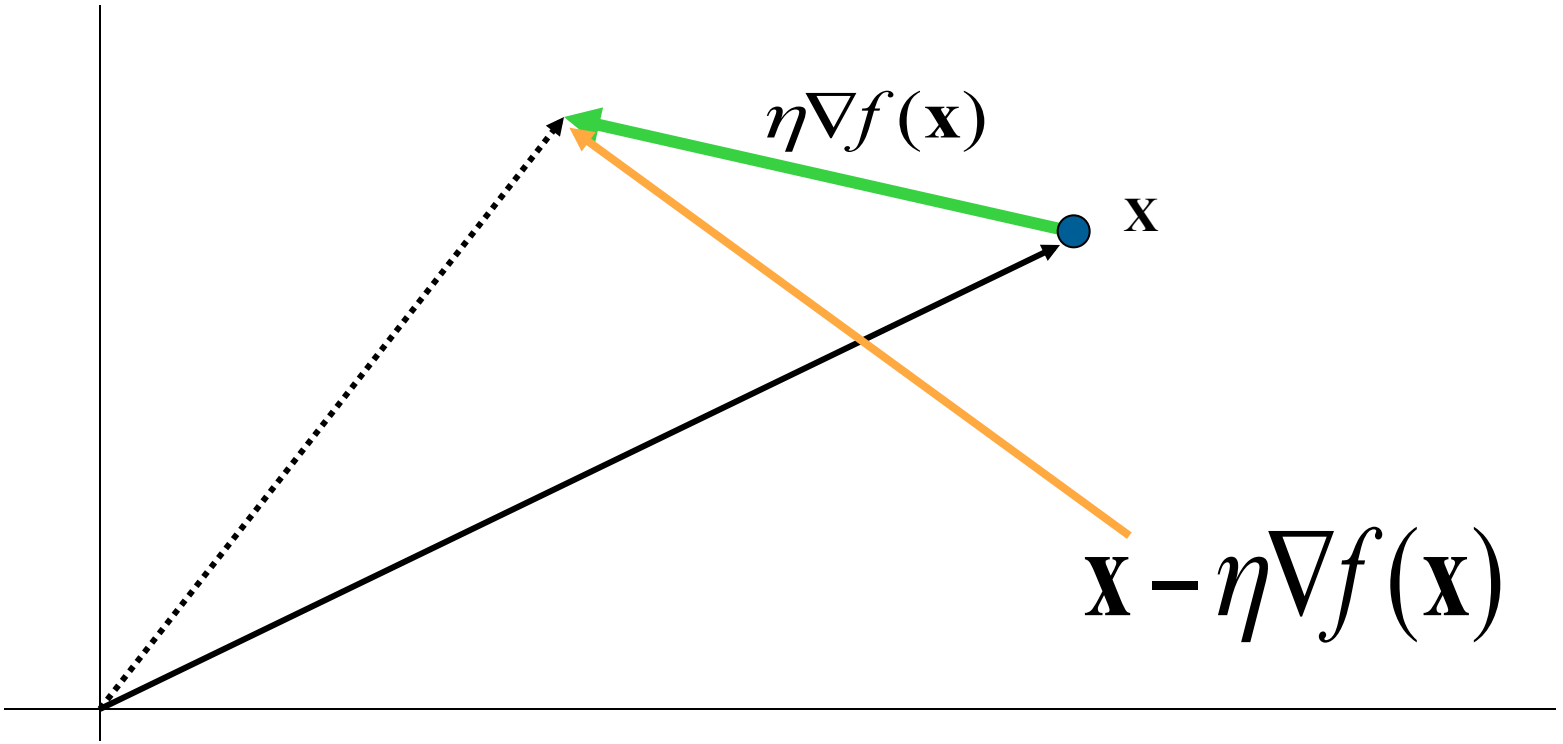
The number of minima is large.

The equations for this landscape is quite complicated.

Finding the arguments that correspond to the derivatives equaling zero is not realistic.


x_0
x_1
x_2

Can Only Go From Where We Are



Method of Steepest Descent

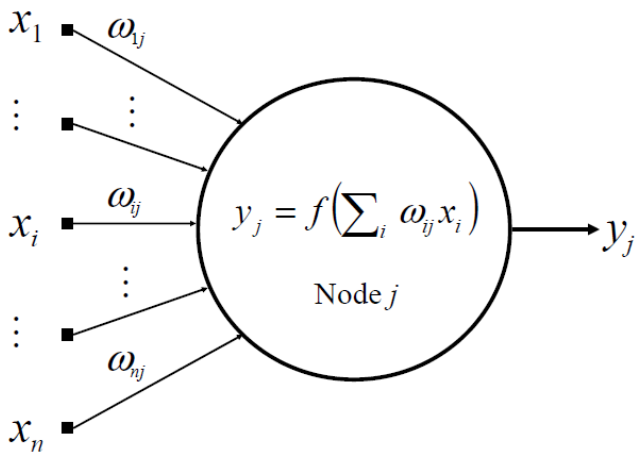
Single Variable Case:

$$x_{k+1} = x_k - \eta f'(x_k)$$

Multi-Variate Case:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k)$$

Getting Back to The Perceptron



Every node computes an output signal where $v_j = \sum_{i=1}^{n} x_i \omega_{ij} - \theta$ and is referred to as the *local field* impinging on node j. The output of node j is

$$y_j = f(v_j)$$

where

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

Want to do something useful!

- Let's assume we can train the perceptron to produce a desired output value.
- Define:

$$e_j = d_j - y_j$$

Let's do something even better!

- Define:

$$E_j = \frac{1}{2}\left(d_j - y_j\right)^2$$

If we have *m* output nodes, define:

$$E = \frac{1}{2}\sum_{j=1}^{m} e_j^2 = \frac{1}{2}\sum_{j=1}^{m}(d_j - y_j)^2$$

The Perceptron Delta Function

$$w_j(k+1) = w_j(k) - \eta \nabla E_j$$

$$\Delta\omega_{ij}(t+1) = \omega_{ij}(t+1) - \omega_{ij}(t) = \eta \frac{\partial E}{\partial \omega_{ij}(t)}$$

or in vector form

$$(\Delta\omega_{i1}, \Delta\omega_{i2}, \ldots, \Delta\omega_{in}) = \eta \nabla(E)$$

The Perceptron Delta Function

Using the Chain Rule, we get:

$$\frac{\partial E_j}{\partial w_i} = \frac{\partial E_j}{\partial e_j} \times \frac{\partial e_j}{\partial y_j} \times \frac{\partial y_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_i}$$

Factors: 1, 2, 3, 4

Factor 1

Recall that

$$E_j = \tfrac{1}{2} e_j^2, \text{ so } \frac{\partial E_j}{\partial e_j} = e_j$$

Factor 2

Recall that $e_j = d_j - y_j$

$$\text{so } \frac{\partial e_j}{\partial y_j} = -1$$

Factor 3

Recall that $y_j = \dfrac{1}{1+e^{-A_j}}$

So,

$$\frac{\partial y_j}{\partial A_j} = \frac{\partial}{\partial A_j}\left(\frac{1}{1+e^{-A_j}}\right)$$

$$= \frac{f'g - fg'}{g^2} = \frac{-e^{-A_j}(-1)}{\left(1+e^{-A_j}\right)^2}$$

$$= \frac{e^{-A_j}}{\left(1+e^{-A_j}\right)} \times \left(\frac{1}{1+e^{-A_j}}\right)$$

$$= \left[1 - y_j\right] y_j$$

Factor 4

Recall that $$A_j = \sum_{i=1}^{n} w_i x_i$$

So,
$$\frac{\partial A_j}{\partial w_k} = \frac{\partial}{\partial w_k} \sum_{i=1}^{n} w_i x_i$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial w_k} \left(w_i x_i \right)$$

$$= x_k \quad \text{for } i = k, \; 0 \quad \text{otherwise}$$

The Perceptron Delta Function

Putting it all together,

$$\frac{\partial E_j}{\partial w_i} = \frac{\partial E_j}{\partial e_j} \times \frac{\partial e_j}{\partial y_j} \times \frac{\partial y_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_i}$$

$$\frac{\partial E}{\partial \omega_{ij}} = -e_j[1 - y_j]y_j x_i$$

The Perceptron Delta Function

Putting it all together,

$$\frac{\partial E}{\partial \omega_{ij}} = -e_j[1 - y_j]y_j x_i$$

and letting $\delta_j = -e_j[1 - y_j]y_j$ then

$$\Delta\omega_{ij} = \eta\frac{\partial E}{\partial \omega_{ij}} = \eta\delta_j x_i$$