



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING



Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438

Dr. Mark Fleischer

Copyright 2014 by Mark Fleischer

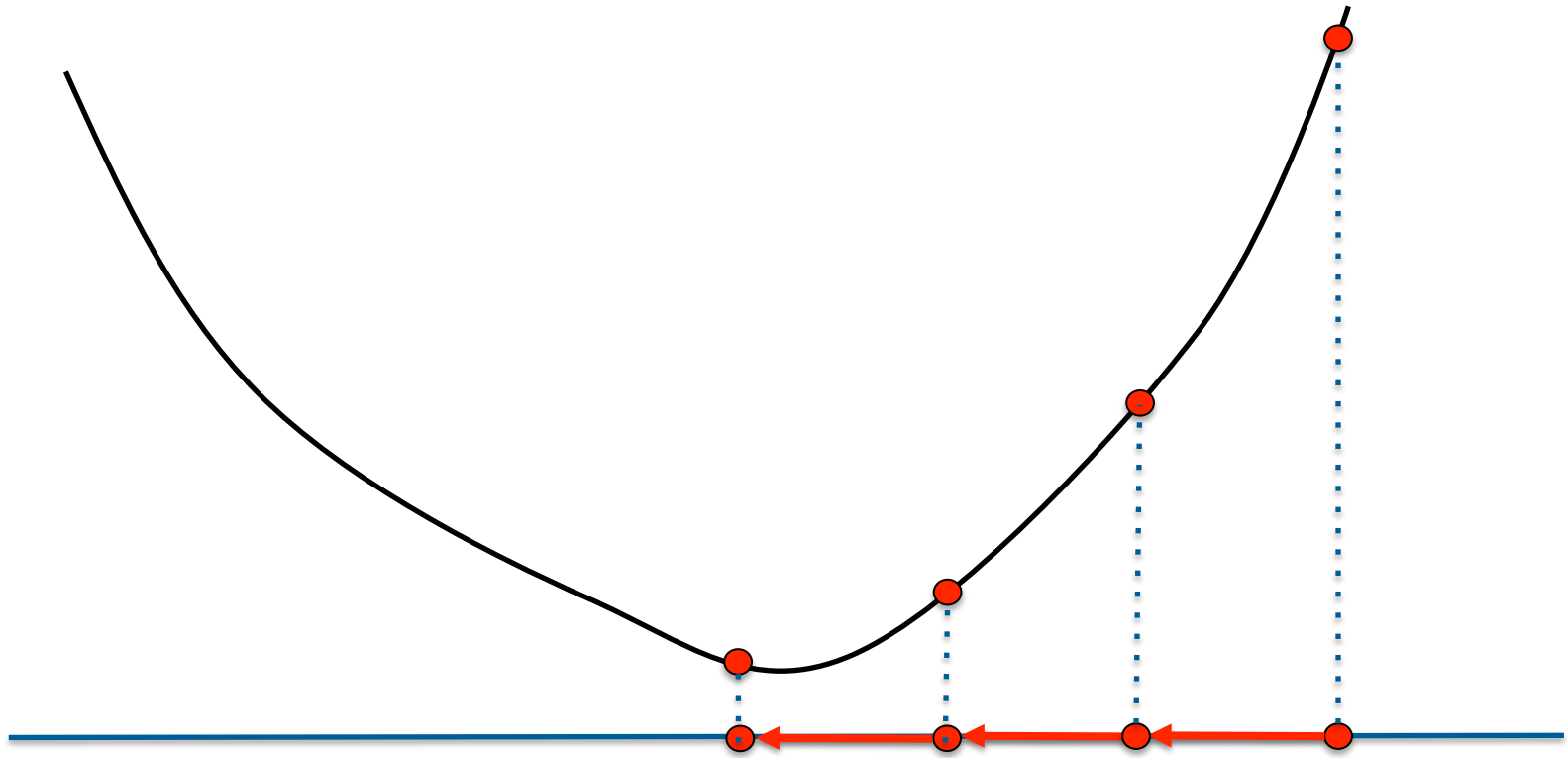
Module 7.2: Training Efficiency

This Sub-Module Covers ...

- We know that in training and testing, there can be a high computational burden.
- Want to make training more efficient.
 - Explore ways for speeding up the FFBP algorithm with a momentum term;
 - Explore some gardening techniques (pruning techniques) to decrease the network size.

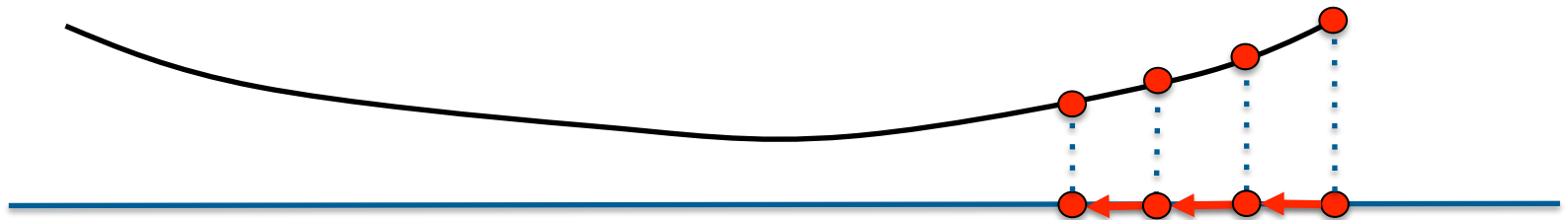
Training: Method of Steepest Descent

The problem with 'fixed' step sizes: $w_{k+1} = w_k - \eta \nabla f(w_k)$



Training: Method of Steepest Descent

The problem with 'fixed' step sizes: $w_{k+1} = w_k - \eta \nabla f(w_k)$



Issues with Steepest Descent Methods

- Can take a long time to find a local minimum when the step size η is too small.
- Can miss or overstep where the local minimum is when the step size η is too large and even lead to oscillations or meandering around the minima.

Getting to Minima Faster

- Some approaches can be proven to converge to the minima:
 - Simulated annealing, but there are issues in continuous variable problems.
 - Stochastic approximation methods.
 - Nelder-Meade ...
- The No-Free Lunch Theorem applies!

The Momentum Term

- Strikes a good balance between ease of implementation and computational overhead and speeding up the process.
- Based on using historical information in the step size.

$$\Delta w_i(k) = -\eta \frac{\partial E}{\partial w_i} + \alpha \Delta w_i(k-1)$$

The Momentum Term

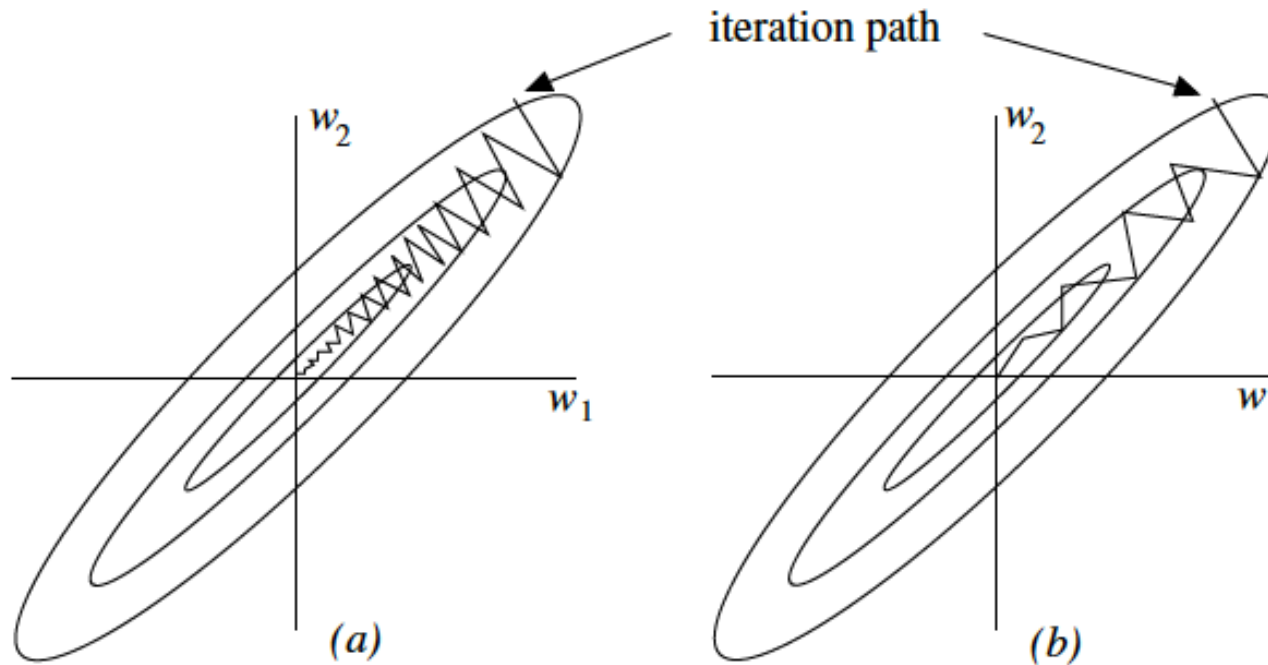


Fig. 8.1. Backpropagation without (a) or with (b) momentum term

From Rojas, p.186

Other Improvement Approaches

- Training and evaluating a network can take a very long time to get it right (we'll see what 'right' means later).
- We have seen how to improve the training algorithm with momentum terms.
- Let's improve training by modifying the network.

Can A Neural Network Model a Polynomial?

- From the manner of training a Neural Network, they can
 - Interpolate data given the training data.
 - Can they model a polynomial?
- How can we mathematically define a polynomial given specific data?
- What does this have to do with training efficiency?

Collocation Method of Polynomials

To define an n -degree polynomial function $P_n(x)$ that goes “through” a specified set of points (x,y) , define

$$P_n(x) = \sum_{i=0}^n f_i L_i(x)$$

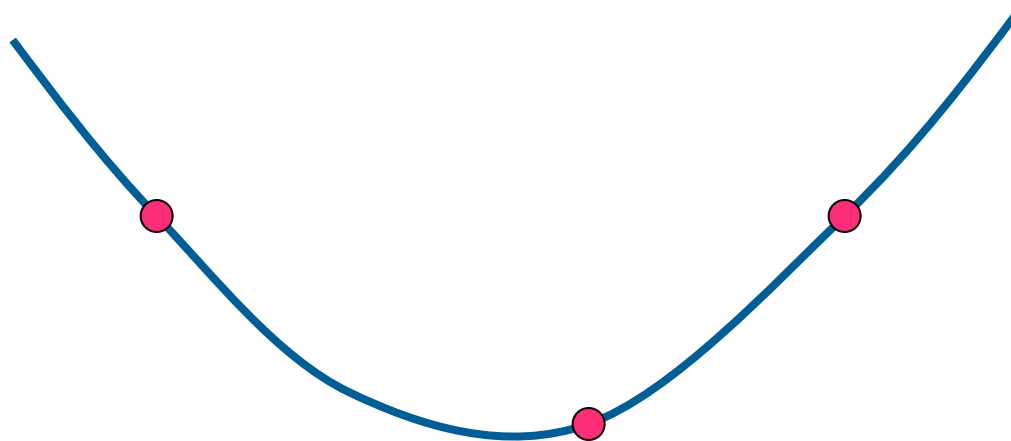
where

$$L_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^n \frac{(x - x_k)}{(x_i - x_k)}$$

Example

Say we want a polynomial to go through specific points in some space. For three points . . .

$$(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))$$



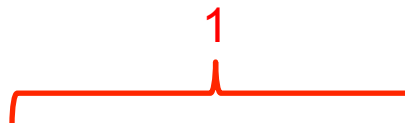
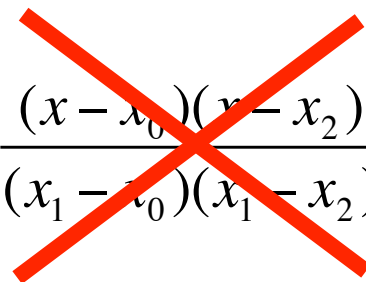
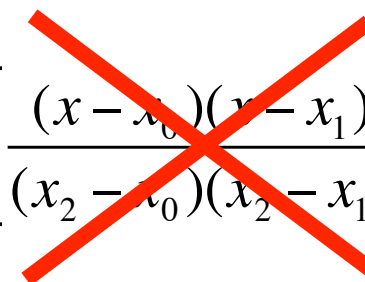
Remembering our function definitions:

$$P_n(x) = \sum_{i=0}^n f_i L_i(x)$$

$$L_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^n \frac{(x - x_k)}{(x_i - x_k)}$$

then

$$P_n(x) = f_0 \left[\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \right] + f_1 \left[\frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} \right] + f_2 \left[\frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \right]$$

$$(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))$$

Remembering our function definitions:

$$P_n(x) = \sum_{i=0}^n f_i L_i(x)$$

$$L_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^n \frac{(x - x_k)}{(x_i - x_k)}$$

then

$$P_n(x) = f_0 \left[\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \right] + f_1 \left[\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \right] + f_2 \left[\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \right]$$

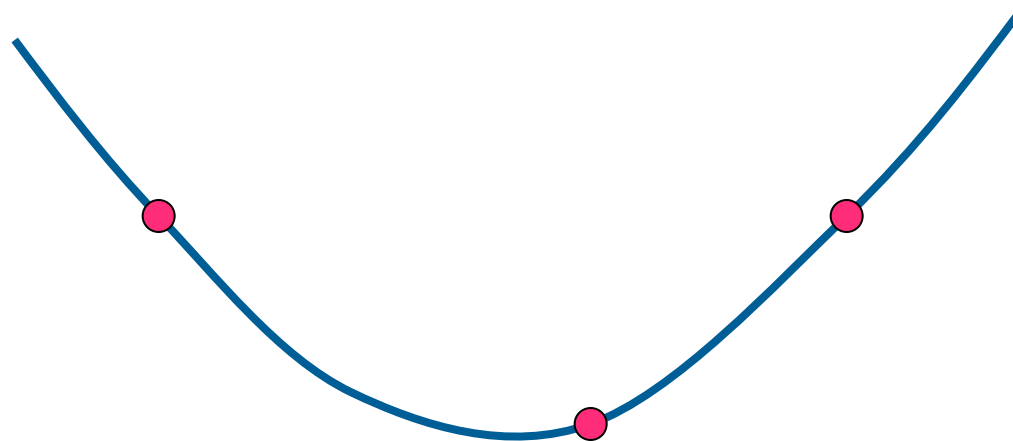
1

$$(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))$$

Example

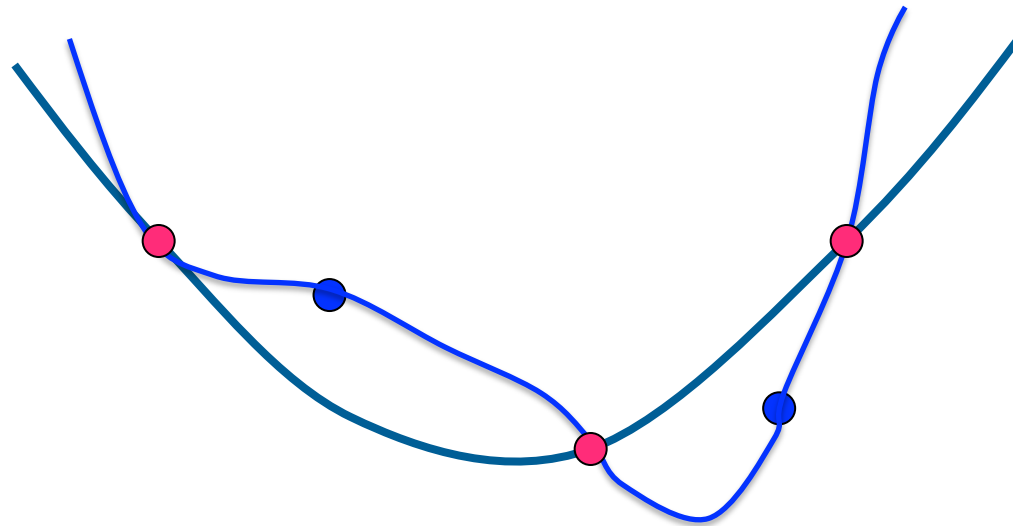
Say we want a polynomial to go through specific points in some space. For three points . . .

$$(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))$$



Example

Now let's add some “dummy” points.



Remembering our function definitions:

$$P_n(x) = \sum_{i=0}^n f_i L_i(x)$$

$$L_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^n \frac{(x - x_k)}{(x_i - x_k)}$$

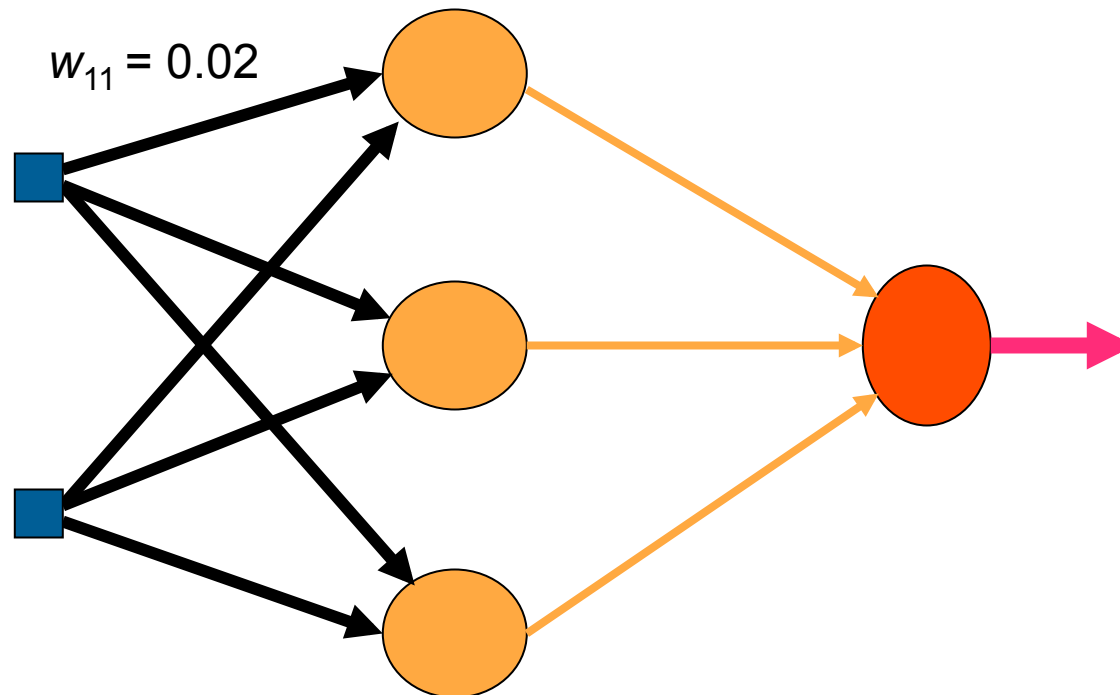
then

$$P_n(x) = f_0 \left[\frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)} \right] + f_1 \left[\frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)} \right] + f_2 \left[\frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)} \right] + f_3 \left[\frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)} \right] + f_4 \left[\frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_3)} \right]$$

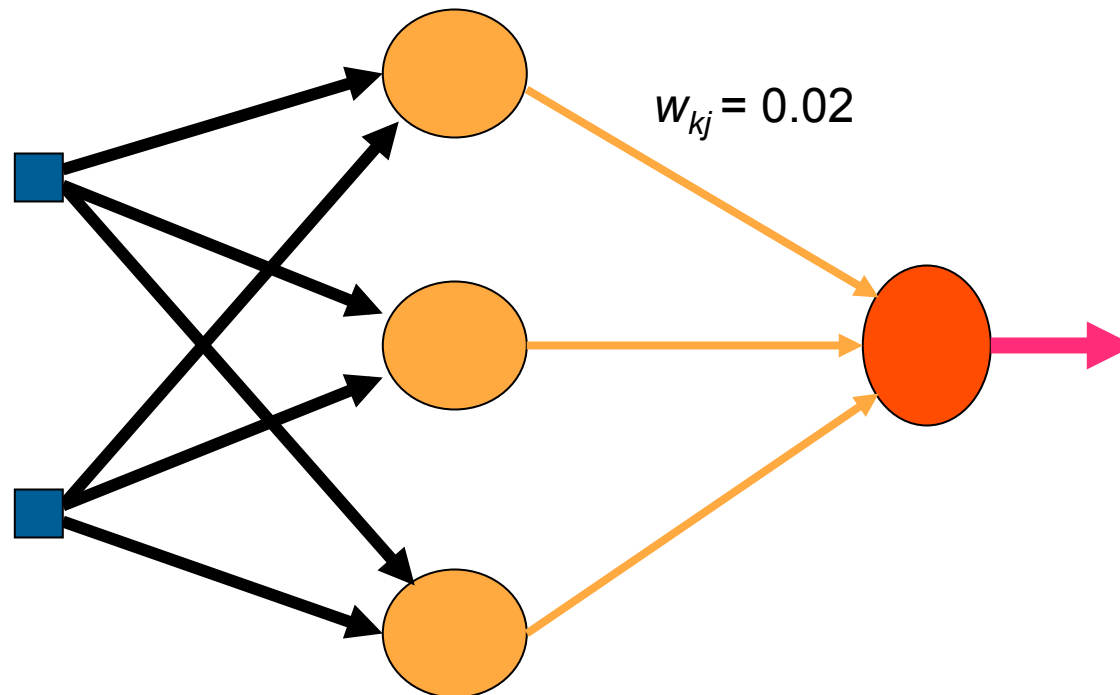
Well, you get the idea... complications, complexity, etc.



Pruning The Network



Pruning The Network



Summary – 7.2

- Described using the momentum term to speed-up gradient descent optimization (i.e., FFBP).
- Collocation Polynomial and how to define a polynomial function that passes through a specified set of points.
- Pruning the network to eliminate ‘weak’ links and/or nodes that do not carry much weight.