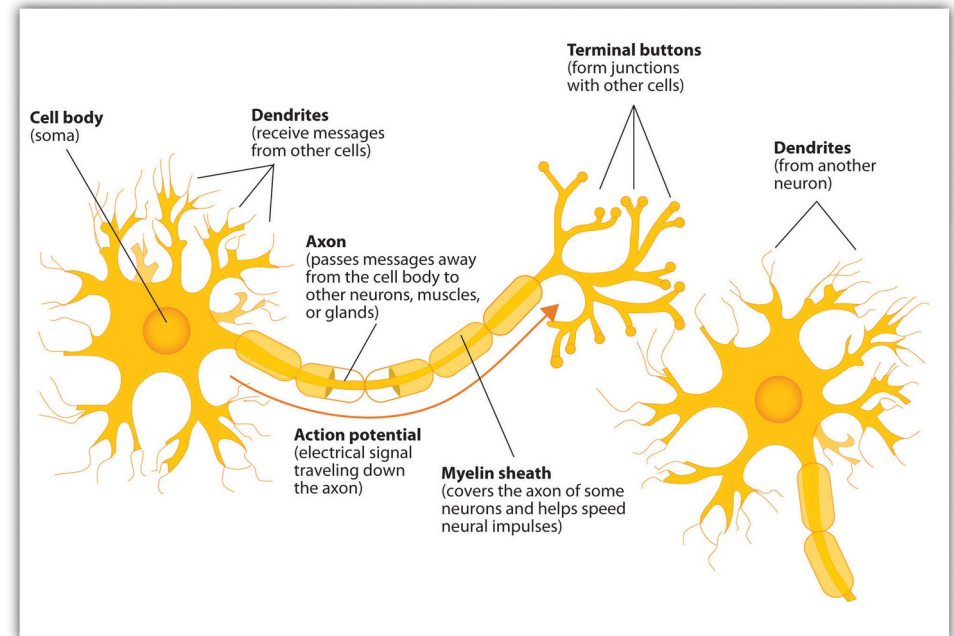


Linear Regression, Neural Networks, etc.

CS 445 Machine Learning
Nathan Sprague
James Madison University

Neurons

- Neurons communicate using discrete electrical signals called “spikes” (or action potentials).
 - Spikes travel along axons.
 - Reach axon terminals.
 - Terminals release neurotransmitters.
 - Postsynaptic neurons respond by allowing current to flow in (or out).
 - If voltage crosses a threshold a spike is created



[Beginning Psychology](#) (v. 1.0).

<http://2012books.lardbucket.org/books/beginning-psychology/>

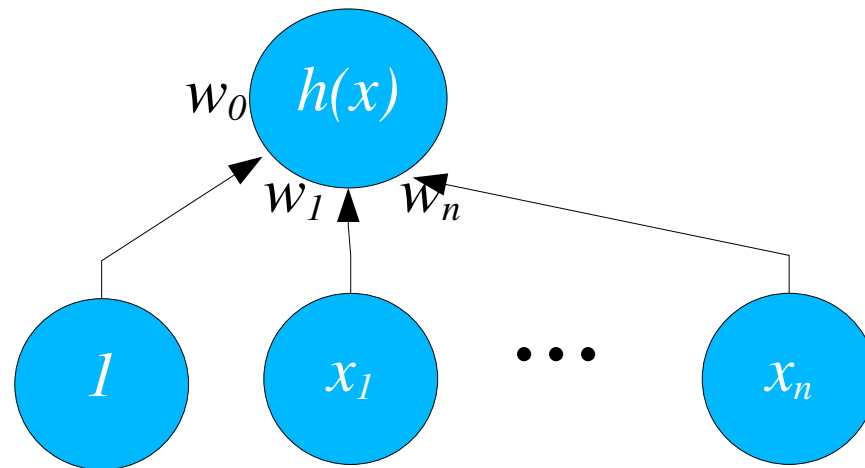
[Creative Commons by-nc-sa 3.0](#)

Multivariate Linear Regression

- Multi-dimensional input vectors:

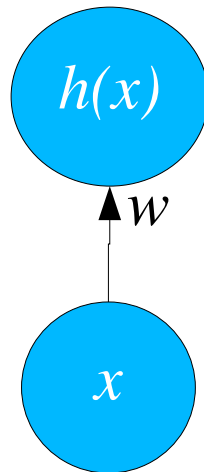
$$h(x_1, x_2, \dots, x_n) = w_0 + w_1 x_1 + \dots + w_n x_n$$

- Or: $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$



Linear Regression – The Neural View

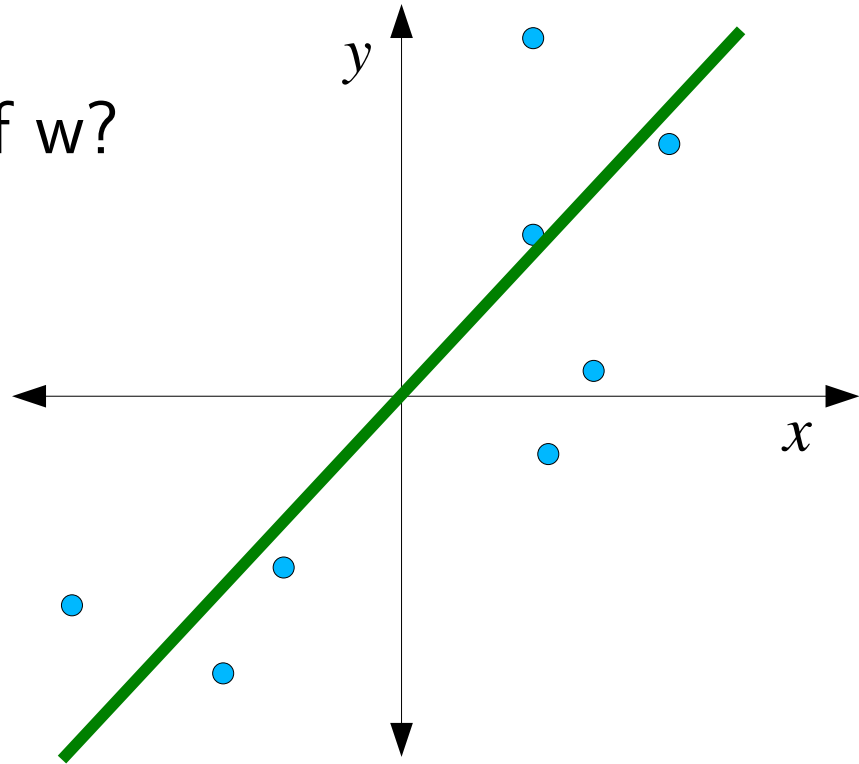
- input = x , desired output = y , weight = w .
- $h(x) = wx$



- We are given a set of inputs, and a corresponding set of outputs, and we need to choose w .
- What's going on geometrically?

Lines

- $h(x) = wx$ is the equation of a line with a y intercept of 0.
- What is the best value of w ?
- How do we find it?

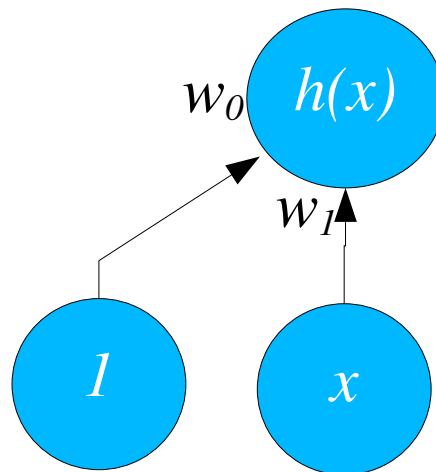


Bias Weights

- We need to use the general equation for a line:

$$h(x) = w_1x + w_0$$

- This corresponds to a new neural network with one additional weight, and an input fixed at 1.



Error Metric

- Sum squared error (y is the desired output):

$$Error_E = \sum_{e \in E} \frac{1}{2} (y_e - h(\mathbf{x}_e))^2$$

- The goal is to find a w that minimizes E . How?

Gradient Descent



http://en.wikipedia.org/wiki/File:Glacier_park1.jpg

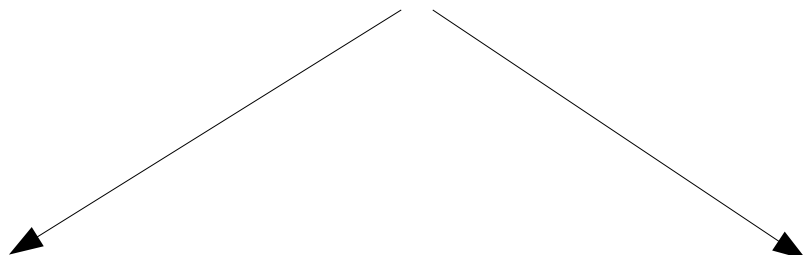
Attribution-Share Alike 3.0 Unported

Gradient Descent

- One possible approach (maximization):
 - 1) take the derivative of the function: $f'(w)$
 - 2) guess a value of w : \hat{w}
 - 3) move \hat{w} a little bit according to the derivative:
$$\hat{w} \leftarrow \hat{w} - \eta f'(\hat{w})$$
 - 4) goto 3, repeat.

Partial Derivatives

- Derivative of a function of multiple variables, with all but the variable of interest held constant.

$$f(x, y) = x^2 + xy^2$$


$$f_x(x, y) = 2x + y^2$$

OR

$$\frac{\partial f(x, y)}{\partial x} = 2x + y^2$$

$$f_y(x, y) = 2xy$$

OR

$$\frac{\partial f(x, y)}{\partial y} = 2xy$$

Gradient

- The gradient is just the generalization of the derivative to multiple dimensions.

$$\nabla f(\mathbf{w}) = \begin{bmatrix} \frac{\partial f(\mathbf{w})}{\partial w_1} \\ \frac{\partial f(\mathbf{w})}{\partial w_2} \\ \vdots \\ \frac{\partial f(\mathbf{w})}{\partial w_n} \end{bmatrix}$$

- Gradient descent update:

$$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} - \eta \nabla f(\hat{\mathbf{w}})$$

Gradient Descent for MVLR

- Error for the multi-dimensional case:

$$Error_E(\mathbf{w}) = \sum_{e \in E} \frac{1}{2} (y_e - \mathbf{w}^T \mathbf{x}_e)^2$$

$$\begin{aligned} \frac{\partial Error_E(\mathbf{w})}{\partial w_i} &= \sum_{e \in E} (y_e - \mathbf{w}^T \mathbf{x}_e) (-x_{e,i}) \\ &= - \sum_{e \in E} (y_e - \mathbf{w}^T \mathbf{x}_e) x_{e,i} \end{aligned}$$

- The new update rule: $w_i \leftarrow w_i + \eta \sum_{e \in E} (y_e - \mathbf{w}^T \mathbf{x}_e) x_{e,i}$

- Vector version: $\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{e \in E} (y_e - \mathbf{w}^T \mathbf{x}_e) \mathbf{x}_e$

Analytical Solution

$$w = (X^T X)^{-1} X^T y$$

- Where X is a matrix with one input per row, y the vector of target values.

Notice that we get Polynomial Regression for Free

$$y = w_1 x^2 + w_2 x + w_0$$

Batch Gradient Descent

- **Batch gradient descent** involves updating based on the full summed gradient value:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{e \in E} (y_e - \mathbf{w}^T \mathbf{x}_e) \mathbf{x}_e$$

Or more generally:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \sum_{e \in E} \nabla_w f(\mathbf{x}_e)$$

- With a huge data set this may involve a large amount computation before any updates can be performed.
- If we are trying to do the calculation in parallel (say on a GPU) it may take a huge amount of memory.

Stochastic Gradient Descent (SGD)

- Update the weights immediately after the gradient is calculated for each data point:

For K epochs:

- Shuffle the data in E
- For e in E:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} f(\mathbf{x}_e)$$

- In practice this often converges in fewer iterations.

Mini-Batch Gradient Descent

- Select a fixed-sized subset of data points and perform an update based on the summed gradient for that subset:

For K epochs:

- Repeat $|E| / |B|$ times:
 - Randomly draw B from E

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \sum_{e \in B} \nabla_w f(\mathbf{x}_e)$$