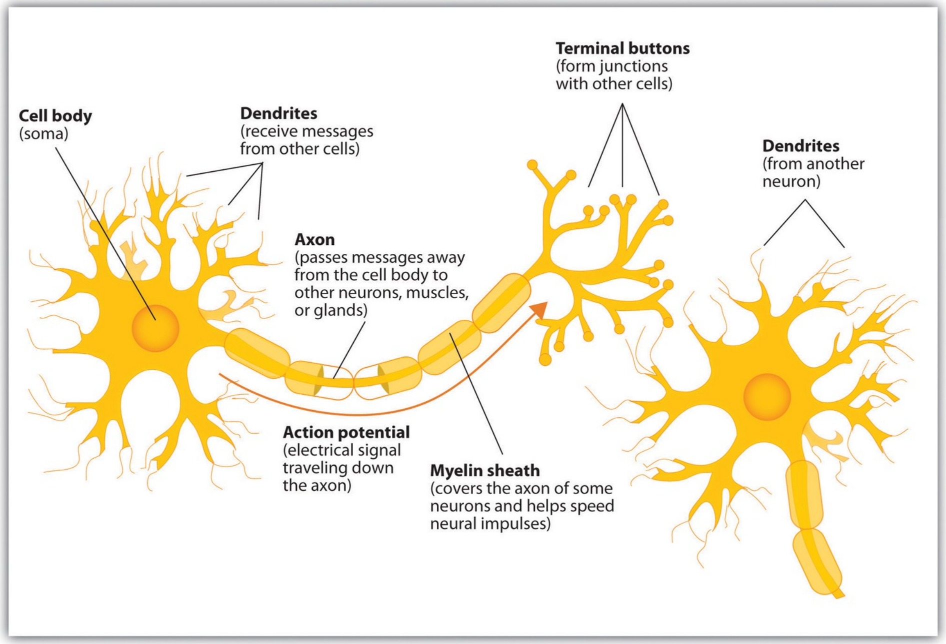
Linear Regression, Neural Networks, etc.

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

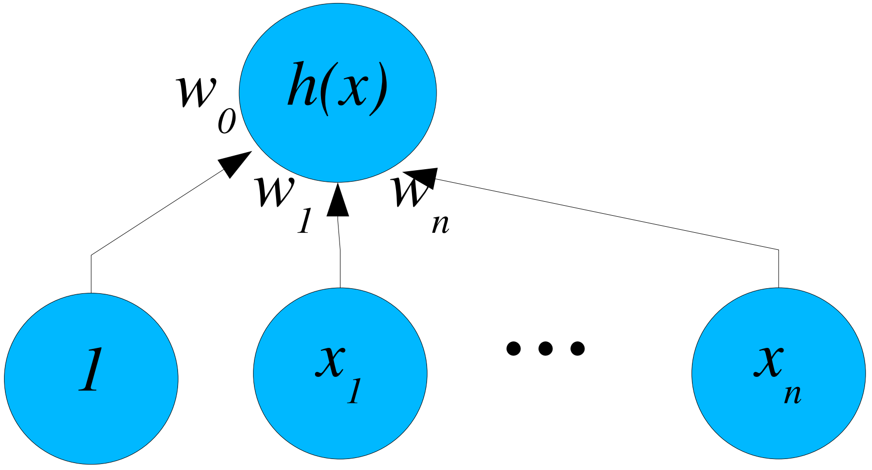


Multivariate Linear Regression

- Multi-dimensional input vectors:

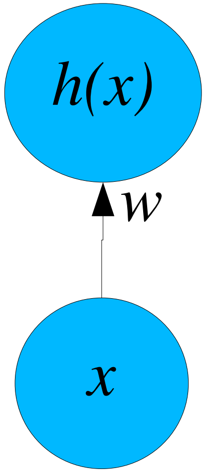
$$h(x_1, x_2, \ldots, x_n) = w_0 + w_1 x_1 + \ldots + w_n x_n$$

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



Linear Regression – The Neural View

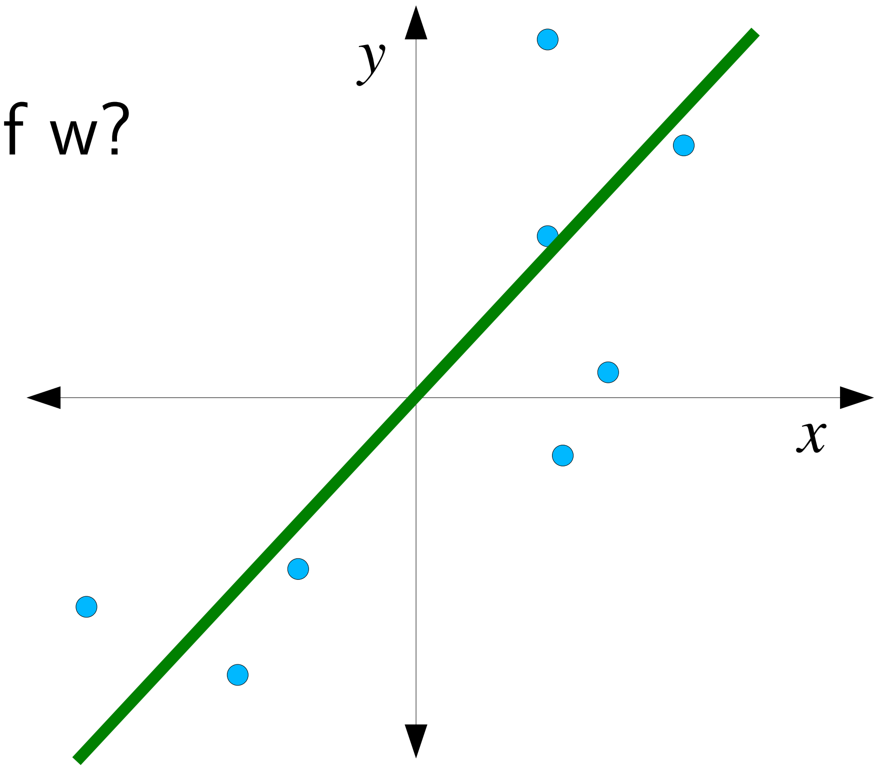
- input $= x$, desired output $= y$, weight $= w$.
- $h(x) = wx$



- 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?

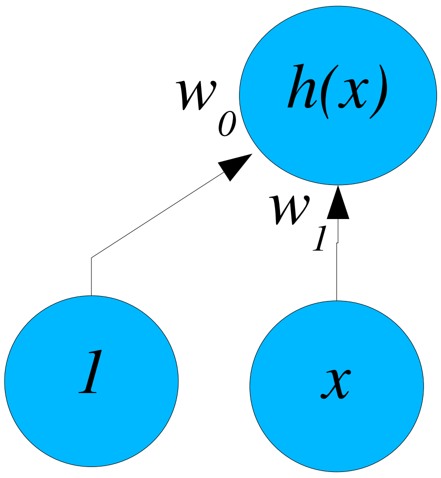


Bias Weights

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

$$h(x) = w_1 x + w_0$$

- 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):

$$Error_E = \sum_{e \in E} \frac{1}{2} \left(y_e - h(\boldsymbol{x_e}) \right)^2$$

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

Gradient Descent

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

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.

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

$$f_x(x,y)=2\mathrm{x}+y^2$$

OR

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

$$f_y(x,y)=2\mathrm{xy}$$

OR

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

Gradient

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

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

- Gradient descent update:

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

Gradient Descent for MVLR

- Error for the multi-dimensional case:

$$Error_E(\boldsymbol{w}) = \sum_{e \in E} \frac{1}{2}\left(y_e - \boldsymbol{w}^T \boldsymbol{x_e}\right)^2$$

$$\frac{\partial\, Error_E(\boldsymbol{w})}{\partial w_i} = \sum_{e \in E} \left(y_e - \boldsymbol{w}^T \boldsymbol{x_e}\right)\left(-x_{e,i}\right)$$

$$= -\sum_{e \in E} \left(y_e - \boldsymbol{w}^T \boldsymbol{x_e}\right) x_{e,i}$$

- The new update rule: $w_i \leftarrow w_i + \eta \sum_{e \in E} \left(y_e - \boldsymbol{w}^T \boldsymbol{x_e}\right) x_{e,i}$

- Vector version: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \sum_{e \in E} \left(y_e - \boldsymbol{w}^T \boldsymbol{x_e}\right) \boldsymbol{x_e}$

Analytical Solution

$$\boldsymbol{w} = \left(X^T X\right)^{-1} X^T y$$

- 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:

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

  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:

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

- In practice this often converges in fewer iterations.

Batch Gradient Descent (SGD)

- 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

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