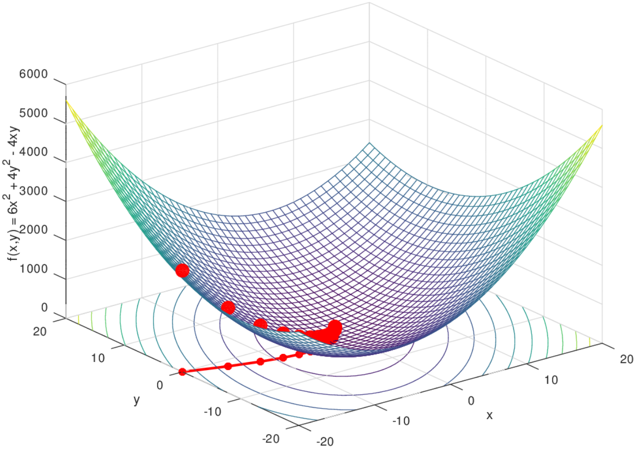
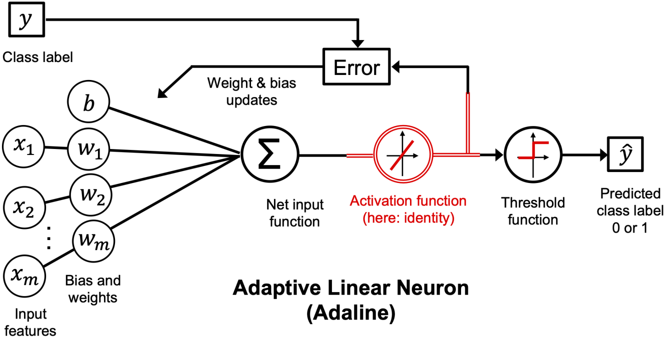
Gradient Descent

Recall: Gradient Descent . . .



Logistic Regression

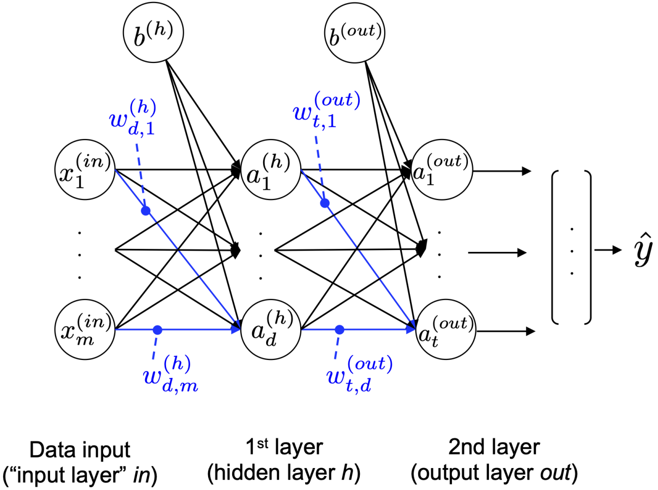
Recall: Logistic Regression . . .



**Adaptive Linear Neuron
(Adaline)**

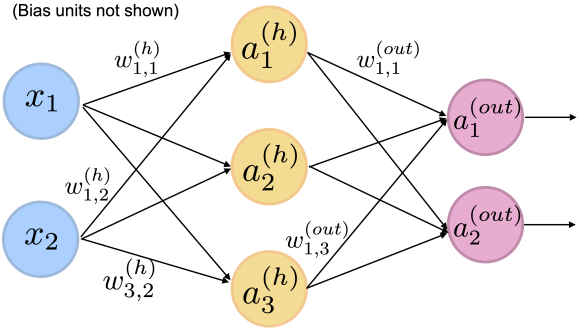
Neural Network Architecture

Idea: Layer this architecture ...



Neural Network Architecture

Idea: Meaning?



- There are other activation functions
- What awaits at the end?

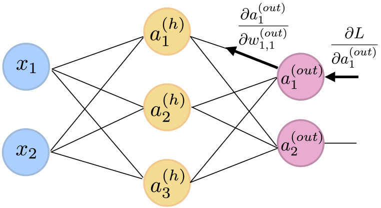
Back Propagation

A loss function . . .

$$L(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t}\sum_{j=1}^{t}\left(y_j^{[i]} - a_j^{(out)[i]}\right)^2$$

We need:

$$\frac{\partial}{\partial w_{j,l}^{(l)}} = L(\boldsymbol{W}, \boldsymbol{b})$$



Gradient for output layer weight:

$$\frac{\partial L}{\partial w_{1,1}^{(out)}} = \frac{\partial L}{\partial a_1^{(out)}} \cdot \frac{\partial a_1^{(out)}}{\partial w_{1,1}^{(out)}}$$

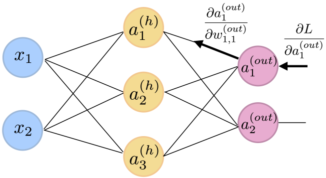
Chain Rule

Given a *composition* of functions, the derivative is:

$$\frac{dF}{dx} = \frac{d}{dx}F(x) = \frac{d}{dx}f(g(h(u(v(x))))) = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

In our case, this means:

$$\frac{\partial L}{\partial w_{1,1}^{(out)}} = \frac{\partial L}{\partial a_1^{(out)}} \cdot \frac{\partial a_1^{(out)}}{\partial z_1^{(out)}} \cdot \frac{\partial z_1^{(out)}}{\partial w_{1,1}^{(out)}}$$



Gradient for output layer weight:

$$\frac{\partial L}{\partial w_{1,1}^{(out)}} = \frac{\partial L}{\partial a_1^{(out)}} \cdot \frac{\partial a_1^{(out)}}{\partial w_{1,1}^{(out)}}$$

Breaking apart the Chain Rule

$$\frac{\partial L}{\partial a_1^{(out)}} = \frac{\partial}{\partial a_1^{(out)}}\left(y_1 - a_1^{(out)}\right)^2 = 2\left(a_1^{(out)} - y\right)$$

The next term is the derivative of the logistic sigmoid activation function that we used in the output layer:

$$\frac{\partial a_1^{(out)}}{\partial z_1^{(out)}} = \frac{\partial}{\partial z_1^{(out)}} \frac{1}{1+e^{z_1^{(out)}}} = \quad \dots \quad = \left(\frac{1}{1+e^{z_1^{(out)}}}\right)\left(1 - \frac{1}{1+e^{z_1^{(out)}}}\right)$$

$$= a_1^{(out)}\left(1 - a_1^{(out)}\right)$$

Lastly, we compute the derivative of the net input with respect to the weight:

$$\frac{\partial z_1^{(out)}}{\partial w_{1,1}^{(out)}} = \frac{\partial}{\partial w_{1,1}^{(out)}} a_1^{(h)} w_{1,1}^{(out)} + b_1^{(out)} = a_1^{(h)}$$

Putting all of it together, we get the following:

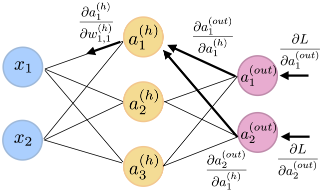
$$\frac{\partial L}{\partial w_{1,1}^{(out)}} = \frac{\partial L}{\partial a_1^{(out)}} \cdot \frac{\partial a_1^{(out)}}{\partial z_1^{(out)}} \cdot \frac{\partial z_1^{(out)}}{\partial w_{1,1}^{(out)}} = 2\left(a_1^{(out)} - y\right) \cdot a_1^{(out)}\left(1 - a_1^{(out)}\right) \cdot a_1^{(h)}$$

We then use this value to update the weight via the familiar stochastic gradient descent update with a learning rate of η:

$$w_{1,1}^{(out)} := w_{1,1}^{(out)} - \eta \frac{\partial L}{\partial w_{1,1}^{(out)}}$$

It gets more complicated . . .

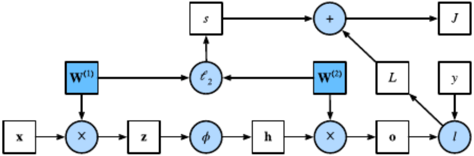
Deeper into Backprop



Gradient for hidden layer weight:

$$\frac{\partial L}{\partial w_{1,1}^{(h)}} = \frac{\partial L}{\partial a_1^{(out)}} \cdot \frac{\partial a_1^{(out)}}{\partial a_1^{(h)}} \cdot \frac{\partial a_1^{(h)}}{\partial w_{1,1}^{(h)}} + \frac{\partial L}{\partial a_2^{(out)}} \cdot \frac{\partial a_2^{(out)}}{\partial a_1^{(h)}} \cdot \frac{\partial a_1^{(h)}}{\partial w_{1,1}^{(h)}}$$

Computational graphs are key . . .



Idea: You backprop along the computational graph . . .