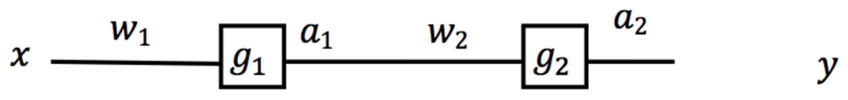
CS188 Fall 2016: Discussion 11 Solutions

Note: The original worksheet was buggy. Here's the fixed version.

Neural Nets and Computation Graphs

Consider the following two-neuron network for binary classification:



Here x is a single real-valued input (not a vector) with an associated class y (0 or 1). There are two neurons, with input weights w_1 and w_2, and activation functions g_1 and g_2. The output

$h_w(x) = a_2$

is a value between 0 and 1, representing the probability of being in class 1. We will be using a real-valued loss function $Loss_w(x, y)$.

Q1**:**

Let z_1 and z_2 refer to the pre-activation values at neuron 1 and neuron 2, repsecitvely. Write z_1, a_1, z_2, and a_2 in terms of the previous values of the neural network**.**

$$\begin{aligned} z_1 &= xw_1 \\ a_1 &= g_1(z_1) \\ z_2 &= a_1 w_2 \\ a_2 &= g_2(z_2) \end{aligned}$$

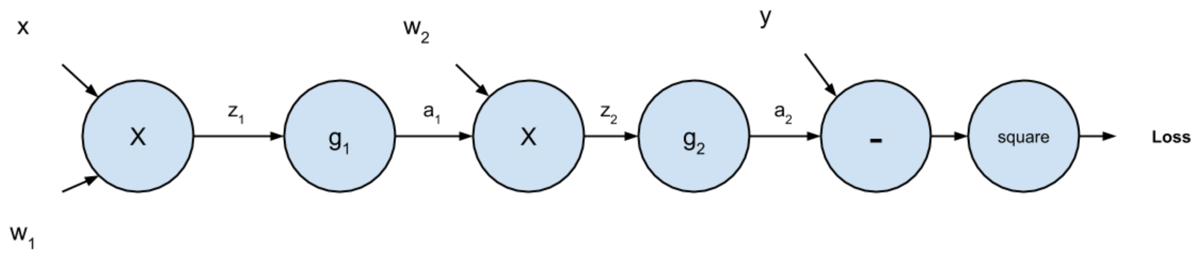
Q2:

Write the output a_2 in terms of the input x, weights w_i , and activation functions g_i.

$$h_{\mathbf{w}}(x) = g_2(w_2 g_1(w_1 x))$$

Q3:

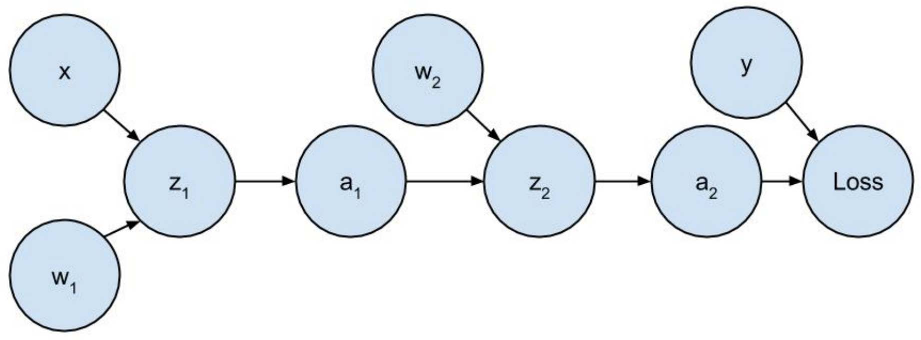
Suppose the loss function is quadratic: ($Loss_w(x, y) = (y - a_2)^2$). Draw the computational graph for the loss function in terms of w_1, w_2, x, y, z_1, a_1, z_2, and a_2.



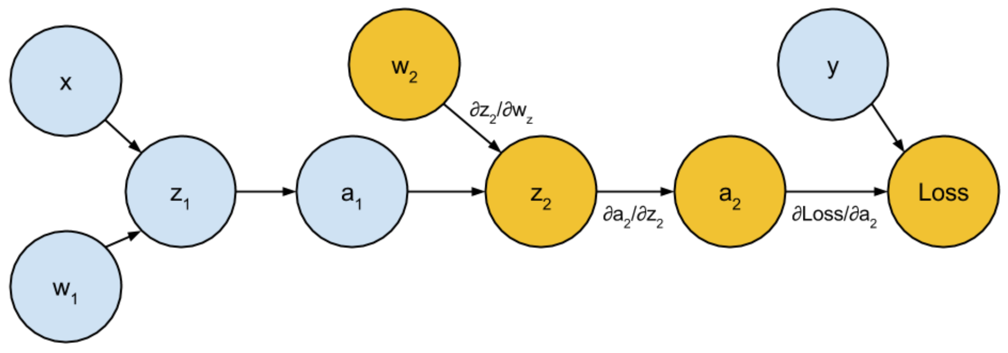
Q4:

Use the chain rule to derive $\partial Loss/\partial w_2$. Write your expression as a product of partial derivatives that can be directly computed – you don't have to directly compute them. (Hint: the series of expressions you wrote in part 1 will be very useful; you may use any of those variables. Also use the graph from Q3).

It might help to redraw the computation graph above with the variables as nodes. This simplifies things and more clearly shows the relationships between variables:



We can put partials on the edges in the above graph, since these partials relate one variable to another that it is directly in terms of. Applying the chain rule, we follow all edges on the path from w_2 to $Loss$, multiplying the partials as we go.



This gives us:

$$\frac{\delta Loss}{\delta w_2} = \frac{\delta Loss}{\delta a_2}\frac{da_2}{dz_2}\frac{\delta z_2}{\delta w_2}$$

(If you don't understand this process of using the Chain Rule, please take a look at the Calculus Review Worksheet on Piazza.)

Q5:

Now use the chain rule to derive $\partial Loss/\partial w_1$ in terms of the same quantities as Q4.

Using the same process as before:

$$\frac{\delta Loss}{\delta w_1} = \frac{\delta Loss}{\delta a_2}\frac{da_2}{dz_2}\frac{\delta z_2}{\delta a_1}\frac{da_1}{dz_1}\frac{\delta z_1}{\delta w_1}$$

Q6:

Suppose the loss function is quadratic ($Loss_w(x, y) = (y - a_2)^2$) and g_1 and g_2 were both sigmoid functions $1/(1 + e^{-z})$. Using the fact that $\partial g_i/\partial z_i = g_i(z_i)(1 - g_i(z_i))$, write $\partial Loss/\partial w_2$ and $\partial Loss/\partial w_1$ in terms of x, y, w_i , a_i , and z_i.

$$\begin{aligned}
\frac{\delta Loss}{\delta w_2} &= -2(y-a_2)g_2(z_2)(1-g_2(z_2))a_1 \\
&= -2(y-a_2)a_1\frac{1}{1+e^{-z_2}}\left(1-\frac{1}{1+e^{-z_2}}\right) \\
\frac{\delta Loss}{\delta w_1} &= 2(y-a_2)g_2(z_2)(1-g_2(z_2))w_2 g_1(z_1)(1-g_2(z_2))x \\
&= -2(y-a_2)a_1a_2w_2x\frac{1}{1+e^{-z_1}}\left(1-\frac{1}{1+e^{-z_1}}\right)\frac{1}{1+e^{-z_2}}\left(1-\frac{1}{1+e^{-z_2}}\right)
\end{aligned}$$

Q7:

Write the stochastic gradient descent update for w_1 in terms of the step size α and the values computed above.

$$w_1 \leftarrow w_1 + \alpha(y-a_2)a_1a_2w_2x\frac{1}{1+e^{-z_1}}\left(1-\frac{1}{1+e^{-z_1}}\right)\frac{1}{1+e^{-z_2}}\left(1-\frac{1}{1+e^{-z_2}}\right)$$

Q8:

True or False: For this classifier and all possible weights w_1 and w_2, there exists some value S for which $x < S$ is classified as belonging to class 0, and $x > S$ is classified as belonging to class 1.

False!

This statement implies that the network is always a linear classifier (draw out the supposed decision boundary if you don't believe this). This neural network actually has no decision boundary: all the points are always in one class. If w_2 is positive, then all inputs would be classified as class 1, and if w_2 is negative, then all inputs would be classified as class 0. Gotcha!

In general, the point of using a neural net is to compose a bunch of non-linear functions together and get a super powerful, non-linear function at the end. As a general rule for neural nets, our weights are such that the classifier is non-linear.