CS 189: Introduction to Machine Learning - Discussion 11

1. **Initialization of weights for backpropagation**

Recall that backpropagation is simply a clever method to solve for the gradient of the loss function so we can use it in a numerical optimization method such as gradient descent. Backpropagation uses the chain rule to pass the gradient backwards through the network. Let $\mathcal{L}$ be the final loss. For layer l, let $x^{(l-1)}$ be the input to the layer, and $\delta^{(l)}$ be the gradient with respect to the input. Then we have:

$$\delta^{(l)} = \frac{\partial \mathcal{L}}{\partial x^{(l-1)}} = \left(\frac{\partial x^{(l)}}{\partial x^{(l-1)}} \right) \delta^{(l+1)}$$

Assume a fully connected 1-hidden layer network with K output nodes and a nonlinearity g. Let $d^{(l)}$ be the number of nodes at layer l. We have:

$$x_j^{(l)} = g\left(\sum_{i=1}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} \right)$$

(a) Imagine that we initialize the values of our weights to be some constant w. After performing the forward pass, what is the relation between the members of the set $\{x_j^{(1)} : j = 1, \ldots, d^{(1)}\}$ and $\{x_i^{(0)} : i = 1, \ldots, d^{(0)}\}$?

Solution: Since all of the weights are equal, we have

$$x_j^{(1)} = g\left(\sum_{i}^{d^{(0)}} w x_i^{(0)} \right) = g\left(w \sum_{i}^{d^{(0)}} x_i^{(0)} \right)$$

Note that this equation does not depend on j, thus, all the $x_j^{(1)}$ are equal. Note that the $x_i^{(0)}$ are just the inputs, so they may be different.

(b) After the backwards pass of backpropagation, what is the relation between the members of the set $\{\delta_j^{(2)} : j = 1, \ldots, d^{(1)}\}$ and $\{\delta_k^{(3)} : k = 1, \ldots, d^{(2)}\}$?

Solution:

$$x_j^{(2)} = g\left(\sum_{i}^{d^{(1)}} w x_i^{(1)} \right)$$

Again, since all of the weights are equal,

$$\delta_i^{(2)} = \frac{\partial \mathcal{L}}{\partial x_i^{(1)}} = \sum_{j=1}^{d^{(2)}} \delta_j^{(3)} \frac{\partial x_j^{(2)}}{\partial x_i^{(1)}} = \sum_{j=1}^{d^{(2)}} \delta_j^{(3)} g' \left(\sum_k^{d^{(1)}} w x_k^{(1)} \right) w$$

Again, there is no dependency on i here, thus, all the $\delta_i^{(2)}$ are equal. All of the $\delta_i^{(3)}$ must also be equal, because it is easy to show that all the $x_j^{(2)}$ are equal.

(c) After the weights are updated and one iteration of gradient descent has been completed, what can we say about the weights?

Solution: Our gradient descent update looks like this:

$$\begin{aligned} w_{ij}^{(l)} &= w_{ij}^{(l)} - \eta \left(g' \left(\sum_k^{d^{(l-1)}} w x_k^{(l-1)} \right) x_i^{(l-1)} \right) \delta_j^{(l+1)} \\ &= w - \eta \left(g' \left(\sum_k^{d^{(l-1)}} w x_k^{(l-1)} \right) x_i^{(l-1)} \right) \delta_j^{(l+1)} \end{aligned}$$

One can see that all $w_{ij}^{(1)}$ may be different across i, but will be the same for all values of j. Basically, for each dimension of the input, the "outgoing" weights of this dimension will have the same weight, but the weights may be different across different dimensions. $w_{ij}^{(2)}$ will be the same for all values of i and all values of j. This pattern will continue and will not break for as many iterations you do.

(d) To solve this problem, we randomly initialize our weights. This is called symmetry breaking. Why are we able to set our weights to 0 for logistic regression?

Solution: Logistic regression can be viewed as a neural network with one output node and zero hidden layers. Because there is only one output, there is no issue with the outputs being the same, and our weights will be updated differently because the gradient depends on the input and the input is different across dimensions.

2. Modifying neural networks for fun and profit

   (a) How could we modify a neural network to perform regression instead of classification?

   Solution: Change the output function of the final layer to be a linear function rather than the normal non-linear function.

   Consider a neural network with the addition that the input layer is also fully connected to the output layer. This type of neural network is also called "skip-layer".

   (b) How many weights would this model require? (Let d_0 be the dimensionality of the input vector, and $d_1 \dots d_L$ be the number of nodes in the L following layers. Don't worry about the bias term. Also, you may want to try drawing out the NN.)

   Solution:

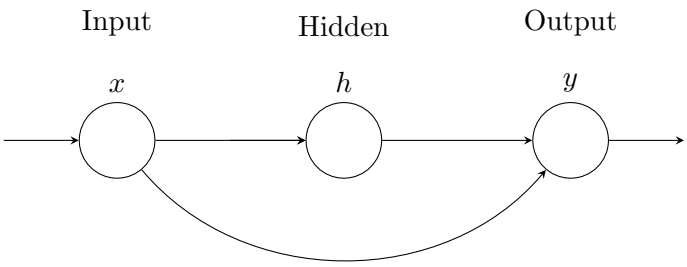
   $$\sum_{i=0}^{L-1} d_i d_{i+1} + d_0 d_L$$

   (c) What sort of problems may this sort of neural network introduce? How do we compensate for these problems?

   Solution: If skip layer has a lot of parameters, then we might have increased the number of weights (parameters). So now there may be a higher chance of overfitting the data. We could fix this by reducing the number of nodes at each layer and/or reducing the number of layers to accomodate for the increased connectivity.

   One potential advantage of the skip layers in the cases when network contains lot of layers. In this setting, higher layers tend to learn higher level concepts and normally output would connect just to these higher layers. The presence of skip layers ensures that output gets direct access to both low-level concepts (initial layers) and high level concepts (later layers). This can be a better modeling choice in many settings.

   (d) Consider the simplest skip-layer neural network pictured below. The weights are $w = [w_{xh}, w_{hy}, w_{xy}]^T$.



Given some non-linear function g, calculate $\nabla_w y$.

Solution: The output y is given by the function:

$$y = g(s_y) = g(w_{hy}h + w_{xy}x) = g(w_{hy}g(s_h) + w_{xy}x) = g(w_{hy}g(w_{xh}x) + w_{xy}x)$$

To calculate ∇y we need all the partial derivatives $\frac{\partial y}{\partial w_h}$, $\frac{\partial y}{\partial w_{hy}}$, $\frac{\partial y}{\partial w_{xy}}$. We'll start with the ones closest to the output.

$$\frac{\partial y}{\partial w_{hy}} = \frac{\partial y}{\partial s_y} \cdot \frac{\partial s_y}{\partial w_{hy}} = g'(s_y)h = \delta_y h$$

$$\frac{\partial y}{\partial w_{xy}} = \frac{\partial y}{\partial s_y} \cdot \frac{\partial s_y}{\partial w_{xy}} = g'(s_y)x = \delta_y x$$

$$\frac{\partial y}{\partial w_{xh}} = \frac{\partial y}{\partial s_y} \cdot \frac{\partial s_y}{\partial w_{xh}} = g'(s_y)\frac{\partial}{\partial w_{xh}}(w_{hy}h + w_{xy}x)$$

$$= g'(s_y)(\frac{\partial}{\partial s_h}(w_{hy}g(s_h))\frac{\partial s_h}{\partial w_{xh}} + \frac{\partial}{\partial w_{xh}}w_{xy}x)$$

$$= w_{hy}g'(s_y)g'(s_h)\frac{\partial s_h}{\partial w_{xh}} + 0$$

$$= w_{hy}g'(s_y)g'(s_h)\frac{\partial(w_{xh}x)}{\partial w_{xh}}$$

$$= w_{hy}g'(s_y)g'(s_h)x = w_{hy}\delta_y g'(s_h)x$$