CS 189: Introduction to Machine Learning - Discussion 11

This discussion assumes the knowledge of backpropagation, and builds upon the fundamental concepts we discussed in last section. Notations are similar to notations used last time.

The network is of 1, . . . ,L layers, that is, the input data passes through L groups of neurons to produce an output. Since we want to allow varying number of neurons in each group, we denote the number of neurons in each layer by $d^{(l)}$. Consider the l^{th} layer. It's input $x^{(l-1)}$ comes from the previous layer therefore it can be represented by a vector of length equal to the number of neurons in the layer $l-1$ i.e. $d^{(l-1)}$. As we discussed, back propagation uses chain rule to pass the gradient backwards. Let $\mathcal{L}$ be the final loss. For layer l, let $\delta^{(l)}$ be the gradient that this layer passes backwards, thus it is

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

1. **Initialization of weights for back propagation**

   Recall that back propagation 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. Methods like these require the parameters to be initialized to some value. In the case of logistic regression, we were able to set the weights to be 0. Assume a fully connected 1-hidden layer network with K output nodes. For notation, let us assume that $x_j^{(l)} = g(S_j^{(l)})$, and $S_j^{(l)} = \sum_i^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}$, where g is the nonlinearity.

   (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 $\{x_i^{(0)} : i = 1, \ldots, d^{(0)}\}$?

   (b) After the backwards pass of back propagation, what is the relation between the members of the set $\{\delta_j^{(1)} : j = 1, \ldots, d^{(0)}\}$ and $\{\delta_k^{(2)} : k = 1, \ldots, d^{(1)}\}$?

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

   (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?

2. Modifying neural networks for fun and profit

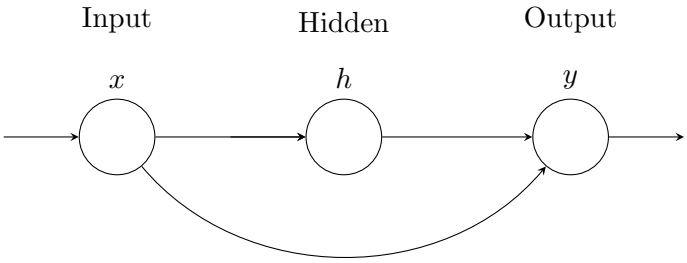
   (a) How could we modify a neural network to perform regression instead of classification?

   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.)

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

   (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 $\nabla_w y$.