Homework 5

Colin Murphy Nate Holland Greg Dehmlow

7 April 2014

Single Layer Neural Network

1. First let's start by deriving the stochastic gradient descent for the Mean squared error. At a high level the stochastic gradient descent is basically taking the weight and then adding or subtracting a small epsilon times the gradient at that point. So we start with:

$J = \frac{1}{2}\sum_{k=1}^{n_{out}}(t_k - y_k)^2$

Then at a high level we get:

$w_{new} = w_{old} - \alpha\frac{\partial J}{\partial w}$

$$w_{new} = w_{old} - \alpha \begin{bmatrix} \frac{\partial J}{\partial w_{11}} & \cdots & \frac{\partial J}{\partial w_{1n}} \\ \cdots & \frac{\partial J}{\partial w_{ij}} & \cdots \\ \frac{\partial J}{\partial w_{m1}} & \cdots & \frac{\partial J}{\partial w_{mn}} \end{bmatrix}$$

Then we know that:

$y_k = \partial(\sum w_{jk}x_j + b_k) = \frac{1}{1+exp(-(s_k+b_k))^2}$

For notations purposes let $S_k = \sum w_{jk}x_j$. Then:

$\frac{\partial y_k}{\partial S_k} = \frac{-exp(-(S_k+b_k))}{(1+exp(-(S_k+b_k)))^2}$

Now since we are only dealing with one layer we get:

$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial S_j}\cdot\frac{\partial S_j}{\partial w_{ij}} = \delta_i X_i$

$\frac{\partial J}{\partial S_j} = \frac{\partial J}{\partial y_j}\cdot\frac{\partial y_j}{\partial S_j} = -(y_j - t_j)\cdot\frac{-exp(-(S_k+b_k))}{(1+exp(-(S_k+b_k)))^2}$

$\frac{\partial J}{\partial w_{ij}} = \frac{-exp(-(S_k+b_k))}{(1+exp(-(S_k+b_k)))^2}\cdot(y_j - t_j)\cdot x_i$

Now we select a random w to start computing on and we compute $\frac{\partial J}{\partial w}$ for a given (x_i, y_i) or set of data points. Then we use $\frac{\partial J}{\partial w}$ to perform the gradient descent update. Then if we want to include a bias term b_j we get:

$\frac{\partial J}{\partial b_j} = \frac{\partial J}{\partial y_j}\cdot\frac{\partial y_j}{\partial b_j} = \frac{exp(-(S_k+b_k))}{(1+exp(-(S_k+b_k)))^2}\cdot(y_j - t_j)$

2. Now let us derive the stochastic gradient descent of the cross entropy error. The equation we start out with is:

$J = -\sum_{k=1}^{n_{out}} [t_k ln(y_k) + (1 - t_k) ln(1 - y_k)]$

Again like in the previous derivation, since we are dealing with only a single layer we get:

$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial S_j} \cdot \frac{\partial S_j}{\partial w_{ij}}$

We also know:

$y_k = \partial(s_k, b_k) = \frac{1}{1+exp(-(s_k+b_k))}$

$\frac{\partial J}{\partial w_{ij}} = -(\frac{t_k}{y_k} - \frac{1-t_k}{1-y_k}) \cdot \frac{exp(-(s_k+b_k))}{(1+exp(-(s_k+b_k)))^2} \cdot x_i$

$\frac{\partial J}{\partial w_{ij}} = (\frac{y_k-t_k}{y_k(1-y_k)}) \cdot \frac{exp(-(s_k+b_k))}{(1+exp(-(s_k+b_k)))^2} \cdot x_i$

Then we use this to compute the gradient and perform the update step like on the previous page. Then for the bias we get:

$\frac{\partial J}{\partial b_j} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial b_j} = (\frac{y_k-t_k}{y_k(1-y_k)}) \cdot \frac{exp(-(s_k+b_k))}{(1+exp(-(s_k+b_k)))^2}$

Again you compute $\frac{\partial J}{\partial b_j}$ for a point and then do the gradient step. For mini batches you simply average $\frac{\partial J}{\partial w}$ for each data point in the minibatch.

Multilayer Neural Network

1. Now let's derive the parameter update equation for the mean squared error with the multilayer neural network. Again we are starting with the equation:

$J = \frac{1}{2}\sum_{k=1}^{n_{out}}(t_k - y_k)^2$

Now note that: $C_j^{(l)} = (\sum_i w_{ij}x_i) + b_j^{(l)}$.

And for notation purposes let $S_j^{(l)} = (\sum_i w_{ij}x_i)$.

Now we know the derivative of $C_j^{(l)}$ is given by:

$\partial C_j^{(l)} = tanh(y_i^{(l)})$

$\frac{\partial \sigma C_j^{(l)}}{\partial b_j^{(l)}} = \frac{\partial \sigma C_j^{(l)}}{\partial S_j^{(l)}} = 1 - tanh^2(C_j^{(l)})$

Then we get:

$\frac{\partial J}{\partial b_j^{(l)}} = \frac{\partial J}{\partial S_j^{(}l)} \cdot \frac{\partial S_{ij}^{(l)}}{\partial b_j^{(l)}} = \delta_j^{(l)} \cdot 1$

Then we can use this equation to compute the bias gradient.Now for updating the weights we start with the basic equation below:

$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial C_j^{(l)}} \cdot \frac{\partial C_j^{(l)}}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} \cdot X_i^{(l-1)}$

Now let's derive what $\delta_j^{(l)}$ is.

$\delta_j^{(l-1)} = \frac{\partial J}{\partial S_i^{(l)}} = \sum_j \frac{\partial J}{\partial S_j^{(l)}} \cdot \frac{\partial S_j^{(l)}}{\partial X_i^{(l-l)}} \cdot \frac{\partial X_i^{(l-l)}}{\partial S_i^{(l-l)}}$

$= \sum_j \delta_j^{(l)} \cdot w_{ij}^{(l)} \cdot \frac{\partial \sigma(C_j^{(l-1)})}{\partial S_i^{(l-1)}}$

$\delta_j^{(l-1)} = (1 - tanh^2(C_i^{(l-1)}))\sum_j \delta_j^{(l)} w_{ij}^{(l)}$

$\delta_j^L = \frac{exp(-C_j^L)}{(1+exp(-C_j^L))^2}(y_j - t_j)$

Now we can use the last two formulas to continue to propagate the δ values until you get the the input layer. At this point you've made one whole round through the neural network.

$\frac{\partial J}{\partial b_i^{(l-1)}} = \sum \delta_l^{(l)} w_{ij}^{(l)}(1 - tanh^2(C_j^{(l-1)}))$

It is worth noting that the weights and biases affect the error in almost identical ways. The difference between them is that the linear combination plus bias is affected by the bias at a constant rate, while its rate of change with respect to the weights is dependent on the inputs. So the computation

for the bias terms are identical except after computing the δ_j'sof the above layer, you do not multiply by the value of node i's output.

2. Now let's derive the parameter update equation using the cross entropy error.

$J = -\sum_{k=1}^{n_{out}} [t_k ln(y_k) + (1 - t_k) ln(1 - y_k)]$

$\delta_i^{(l-1)} = (1 - tanh(C_i^{(l-1)})) \sum_j \delta_j^{(l)} w_{ij}$

$\delta_j^L = \frac{exp(-C_j^{(l)})}{(1+exp(-C_j^{(l)}))^2} \cdot \frac{y_j - t_j}{y_j(1-y_j)}$

Again like the previous derivation we can use these to propagate down to the input layer and therefore compute and entire epoch of the neural network.