CSE 446: Machine Learning **Lecture**

One hidden layer networks

Instructor: Sham Kakade

Please email the staff mailing list should you find typos/errors in the notes. Thank you.

1 Terminology

- non-linear decision boundaries and the XOR function (see CIML)
- multi-layer neural networks & multi-layer perceptrons
- # of layers (definitions sometimes not consistent)
- input layer is x. output layer is y. hidden layers.
- activation function or transfer function or link function.
- forward propagation
- back propagation

Issues related to training are:

- non-convexity
- initialization
- weight symmetries and "symmetry breaking"
- saddle points & local optima & global optima
- vanishing gradients

2 One hidden layer network

Our *data set* is: $\{(x_1, y_1), \ldots (x_N, y_N)\}$, the x_n's are real vectors and where $y_n \in \mathbb{R}$.

A one *hidden layer* network will give some prediction $\widehat{y}(x)$ on input x which is specified as follows. Let $x = (x[1], x[2], \ldots x[d])$. Suppose there are $d^{(1)}$ *hidden nodes*. The input or *activation* to node j at *hidden* layer 1 is denoted by:

$$a_j^{(1)} = \sum_{i=1}^{d} w_{ji}^{(1)} x[i]$$

One could include a bias term, by adding w_{j0} to the above (not shown here). The input is transformed with a *transfer function* (or an *activation* function or *link* function):

$$z_j^{(1)} = h(a_j^{(1)})$$

Common choices for transfer functions $h(\cdot)$ are often the sigmoid function:

$$h(a) = \frac{1}{1+\exp(-a)},$$

the tanh function,

$$h(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)},$$

or the "ReLu" (the rectified linear) function

$$h(a) = \max\{a, 0\},$$

which is a if a is positive and 0 otherwise.

The target function, our prediction with input x, is:

$$\widehat{y}(x) = \sum_{j=1}^{d^{(1)}} w_j^{(2)} z_j^{(1)}$$

The parameters of the model are the second layer weights, $w^{(2)}$ which is a vector of dimension $d^{(1)}$, and the first layer weights $w^{(1)} = \{w_{ji}^{(1)}\}$.

2.1 The gradient in a one hidden layer network

The backprop algorithm contains this derivative as a special case.

In general, the loss function is:

$$L(w^{(1)}, w^{(2)}) = \frac{1}{N}\sum_n \ell(y, \widehat{y}(x))$$

and for the special case of the square loss we have:

$$\frac{1}{N}\sum_n \ell(y_n, \widehat{y}(x_n)) = \frac{1}{N}\frac{1}{2}\sum_n (y_n - \widehat{y}(x_n))^2$$

Note that:

$$\nabla L(w^{(1)}, w^{(2)}) = \frac{1}{N}\sum_n \nabla \ell(y_n, \widehat{y}(x_n))$$

where the gradient here is with respect to all the parameters.

The gradient is specified as follows:

$$\frac{\partial \ell(y, \widehat{y}(x))}{\partial w_j^{(2)}} = -(y - \widehat{y}(x)) z_j^{(1)}, \qquad \frac{\partial \ell(y, \widehat{y})}{\partial w_{ji}^{(1)}} = -(y - \widehat{y}(x)) w_j^{(2)} h'(a_j^{(1)}) x[i]\,.$$

Note that the above equations specifies the gradient entirely (with respect to all the parameters).

We now derive both of these equations. Let us now compute this gradient on each term, $\nabla \ell(y_n, \widehat{y}(x_n))$. We have:

$$\frac{\partial \ell(y, \widehat{y}(x))}{\partial w_j^{(2)}} = -(y - \widehat{y}(x))\,\frac{\partial \widehat{y}(x)}{\partial w_j^{(2)}} = -(y - \widehat{y}(x)) z_j^{(1)}$$

where z_j is the output of hidden node j, where the network input is x.

Now let us take the derivatives with respect to $w_{ji}^{(1)}$. By the chain rule,

$$\frac{\partial \ell(y,\widehat{y})}{\partial w_{ji}^{(1)}} = \frac{\partial \ell(y,\widehat{y})}{\partial a_j^{(1)}} \frac{\partial a_j^{(1)}}{\partial w_{ji}^{(1)}} =$$

We have:

$$\frac{\partial a_j^{(1)}}{\partial w_{ji}^{(1)}} = x[i],$$

and

$$\frac{\partial \ell(y,\widehat{y})}{\partial a_j^{(1)}} = \frac{\partial \ell(y,\widehat{y})}{\partial z_j^{(1)}} \frac{\partial z_j^{(1)}}{\partial a_j^{(1)}} = -(y-\widehat{y}(x))w_j^{(2)} \frac{\partial z_j^{(1)}}{\partial a_j^{(1)}} = -(y-\widehat{y}(x))w_j^{(2)} h'(a_j^{(1)})$$

Putting this together:

$$\frac{\partial \ell(y,\widehat{y})}{\partial w_{ji}^{(1)}} = -(y-\widehat{y}(x))w_j^{(2)} h'(a_j^{(1)}) x[i]$$