Backpropagation in Neural Networks

Have a look at the simple neural network presented in the lecture slides.

Using a Single Training Example Per Step

Throughout this example, D is the dimension of the input space, H the dimension of the hidden layer and C the number of different classes. The seemingly unnecessary notational detail (e.g. the φ activation functions) will come in handy in the back propagation.

Let $x \in \mathbb{R}^D$, $y \in \{0,1\}^C$ where $\sum_i y_i = 1$ (one-hot coding).

Forward Pass:

Let $W^{(1)} \in \mathbb{R}^{H \times D}$, $b^{(1)} \in \mathbb{R}^H$.

By matrix concatenation $\widetilde{x} = \begin{bmatrix} 1 \\ x \end{bmatrix} \in \mathbb{R}^{D+1}$ and $\tilde{W}^{(1)} = \begin{bmatrix} b^{(1)} \, W^{(1)} \end{bmatrix} \in \mathbb{R}^{H \times (D+1)}$

we can simplify the network input of the hidden layer:

$$Z^{(2)} = W^{(1)} x + b^{(1)}$$

to an equivalent representation:

$$Z^{(2)} = \widetilde{W}^{(1)} \widetilde{x}$$

The activation is defined as

$$A^{(2)} = \varphi^{(2)}\big(Z^{(2)}\big) \qquad \Big[= \max(0, Z^{(2)}) \text{ in the example} \Big]$$

Again, to simplify the netork input calculation for the output layer, concatenate

$$\widetilde{A}^{(2)} = \begin{bmatrix} 1 \\ A^{(2)} \end{bmatrix} \qquad \widetilde{W}^{(2)} = \begin{bmatrix} b^{(2)} \; W^{(2)} \end{bmatrix}$$

now the network input of the output layer can be written as

$$Z^{(3)} = \widetilde{W}^{(2)} \widetilde{A}^{(2)} \;\in \mathbb{R}^C$$

and the activation of the output layer is:

$$A^{(3)} = \varphi^{(3)}\big(Z^{(3)}\big) \qquad \Big[= id \text{ in the example} \Big]$$

This is the prediction of the model given families of weight matrices W and biases b:

$$h_{W,b}(x) = \frac{1}{\sum_{i=1}^{C} e^{A_i^3}} \, e^{A^{(3)}} = \frac{1}{\sum_{i=1}^{C} e^{Z_i^3}} \, e^{Z^{(3)}} \;\in \mathbb{R}^C$$

Backward Pass:

The general crossentropy loss

$$J(W,b) = -\tfrac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{C}\mathbb{1}\{y^{(i)}=k\} log\, h_{W,b}(x^{(i)}) + \frac{\lambda}{2}\sum_{l=1}^{L}\|W^{(l)}\|^2$$

can be simplified as we have only one training example and y is a one-hot vector:

$$J(W,b) = -\sum_{k=1}^{C} y_k\; log\, h_{W,b}(x) + \frac{\lambda}{2}\sum_{l=1}^{L}\|W^{(l)}\|^2$$

We now want to change the weights (which include the bias) in order to minimize J. Notice that for our fully connected network $\widetilde{W}_{i,j}^{(l)}$ is the weight connecting the j-th output component of layer l (which is $A_j^{(l)}$) to the i-th neuron of layer $l+1$.
By the chain rule, we get:

$$\frac{\partial J}{\partial \widetilde{W}_{i,j}^{(l)}} = \frac{\partial J}{\partial A_j^{(l+1)}}\frac{\partial A_j^{(l+1)}}{\partial Z_j^{(l+1)}}\frac{\partial Z_j^{(l+1)}}{\partial \widetilde{W}_{i,j}^{(l)}}$$

For further clarification, let's have a look at the factors in this product:

$$\frac{\partial A_j^{(l+1)}}{\partial Z_j^{(l+1)}} = \varphi'^{(l+1)}(Z_j^{(l+1)}) \quad \text{is the derivative of the activation function}$$

$$\frac{\partial Z_j^{(l+1)}}{\partial \widetilde{W}_{i,j}^{(l)}} = A_i^{(l)} \quad \text{is the output of the ``previous'' neuron (linearity of matrix mult.)}$$

So far this formulation is independent of l. The gradient of the error function can only be evaluated directly at the output layer:

$$\frac{\partial J}{\partial Z_j^{(l+1)}} = Z_j^{(l+1)} - y_j \qquad \Big[\text{where } l=3 \text{ in the example}\Big]$$

For the lower layers, this error signal propagate backwards (thus the name) as follows, where η is the learning rate:

$$\widetilde{W}_{i,j}^{(l)} \leftarrow -\eta\Delta_j^{(l+1)}A_i^{(l)} \quad \big[+regularization\big]$$

where

$$\Delta_j^{(l)} = \begin{cases}\varphi'\big(Z_j^{(l)}\big)(A_j - y_j) & \text{for the output layer}\\ \varphi'\big(Z_j^{(l)}\big)\sum_k \Delta_k^{(l+1)}\,\widetilde{W}_{j,k}^{(l)} & \text{else}\end{cases}$$