Backpropagation

Hikmat Farhat

November 18, 2020

1 Supervised Learning of Classification Problems

We derive the formulas needed to perform backpropagation in a fully connected neural network when the problem is supervised earning of k classes. Since we have multiple *distinct* classes the labels are represented in one_hot encoding. So $y = [10 \dots 0]$ represents class 1, $y = [01 \dots 0]$ represents class 2, etc.

Let $\mathbf{x}$ be the input, our goal is to compute $p(y \mid \mathbf{x})$. Following Bishop we write

$$p(y \mid \mathbf{x}) = \prod_{i=1}^{k} p_i^{y_i} \tag{1}$$

Where p_i is the probability that $y = c_i$ given that the input is x, y is the one_hot representation of the class. This means that only one of the y_i is 1 and the rest are zeros with $\sum_i p_i = 1$. This implies

$$p(y = y_i \mid \mathbf{x}) = p_i$$

A special case is when we have only two categories then $p_1 = 1 - p_0$ and we can write

$$p(y \mid \mathbf{x}) = p_0^y \cdot (1 - p_0)^{1-y}$$

2 Kullback-Leibler

In classification problem we have a neural network that approximates $p(y \mid \mathbf{x})$. Let z be the output of the neural network then we model the probabilities by

$$\hat{p}_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

The loss is defined by the Kullback-Leibler divergence

$$
\begin{aligned}
KL(p \parallel \hat{p}) &= \sum p \log \frac{\hat{p}}{p} \\
&= \sum p \log \hat{p} - p \log p \geq 0
\end{aligned} \tag{2}
$$

Since we can't influence the unknown distribution p we minimize the first term, or alternatively we maximize its negative (likelihood). The summation in equation (2) is an average over the true (unknown) distribution p. We approximate by averaging over the samples. Replacing the expression in (1) we define the *loss* function as:

$$
\mathcal{L} = -\sum_s \sum_k y_{sk} \log \hat{p}_{sk}
$$

Where s is the index of the sample and y_{sk} is the one_hot vector encoding of the label of sample s. Note that for each s there is only one $y_{sk} = 1$ and the rest are zeros.

3 Gradient

The loss function $\mathcal{L}$ depends on the all the parameters of the neural network. We will see that computing the derivatives with respect to the parameters of one layer depend on the the derivatives of the **next** layer.

The output of layer l, is a vector a^l given by $a^l_{sj} = f(z^l_{sj})$ where f is some nonlinear function, called the activation function, s is the index of the sample, j is the component of a and z, and

$$
z^{l+1}_{sj} = \sum_k a^l_{sk} w^l_{kj} + b^l_j \tag{3}
$$

In a neural network with $L + 1$ layers, used for classification problems, we usually use the *softmax* function as the activation of the last layer a^L. Furthermore, we interpret the output of the last layer as the probability of a given class, i.e. $a^L \equiv \hat{p}$

3.1 Derivation with respect to the last parameters

Let L be the index of the last layer so $a^L = \sigma(z^L)$ where σ is the softmax function (or sigmoid for 2 classes).

Since the samples are independent we will drop the sample index in what follows for simplicity. To compute the gradient we need an expression for the following quantity:

$$
\frac{\partial \mathcal{L}}{\partial z^l_j}
$$

3.2 Derivative of the last layer

To simplify the notation we use $z \equiv z^L$ and $a \equiv a^L$ and we drop the index of the samples (i.e. we consider one sample. All the others are the same)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_j} &= -\frac{\partial}{\partial z_j} \sum_k y_k \log a_k \\ &= -\sum_k \frac{y_k}{a_k} \frac{\partial a_k}{\partial z_j}\end{aligned}$$

Since

$$a_k = \frac{e^{z_k}}{\sum_t e^{z_t}}$$

then

$$\frac{\partial a_k}{\partial z_j} = \frac{\delta_{ik} \sum_t e^{z_t} - e^{z_k} e^{z_j}}{(\sum_t e^{z_t})^2}$$

Where δ_{ik} is the Kronecker delta. Multiplying by $\frac{1}{a_k}$ we get

$$\frac{1}{a_k} \frac{\partial a_k}{\partial z_j} = \delta_{jk} - a_j$$

Therefore

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_j} &= \sum_k (a_j y_k - \delta_{jk} y_k) \\ &= a_j - y_j\end{aligned}$$

Using Δ_j^L for $\frac{\partial \mathcal{L}}{\partial z_j}$ and reinserting the sample index we get

$$\Delta_{sj}^L = a_{sj} - y_{sj}$$

3.3 Backpropagation

In this section we derive an expression for Δ^l in terms of Δ^{l+1}.

$$\begin{aligned}\Delta_j^l = \frac{\partial \mathcal{L}}{\partial z_j^l} &= \sum_k \frac{\partial \mathcal{L}}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \Delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \\ &= \sum_k \Delta_k^{l+1} w_{jk}^l \theta_j^l\end{aligned}$$

Where $\theta^l_j \equiv \frac{\partial a^l_j}{\partial z^l_j}$ is the derivative of the activation function of layer l with respect to its parameter. Reinserting the sample index we get

$$\Delta^l_{sj} = \sum_k \Delta^{l+1}_{sk} w^l_{jk} \theta^l_{sj}$$

3.3.1 Gradient wrt parameters

Since the neural network is "optimized" by varying the parameters w^l, b^l we need an expression for the gradient of the loss wrt to the parameters.

$$\frac{\partial \mathcal{L}}{\partial w^l_{ij}} = \frac{\partial \mathcal{L}}{\partial z^{l+1}_j} \frac{\partial z^{l+1}_j}{\partial w^l_{ij}}$$

Using eq. (**??**) we get

$$\frac{\partial \mathcal{L}}{\partial w^l_{ij}} = \Delta^{l+1}_j a^l_i$$

Reinserting the sample index and averaging over the samples (total m samples):

$$\frac{\partial \mathcal{L}}{\partial w^l_{ij}} = \frac{1}{m} \sum_s a^l_{si} \Delta^{l+1}_{sj}$$

Similarly for the bias we get

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b^l_j} &= \frac{\partial \mathcal{L}}{\partial z^{l+1}_j} \frac{\partial z^{l+1}_j}{\partial b^l_j} \\ &= \Delta^{l+1}_j\end{aligned}$$

Averaging over the m samples

$$\frac{\partial \mathcal{L}}{\partial b^l_j} = \frac{1}{m} \sum_s \Delta^{l+1}_{sj}$$