Deriving Backpropagation

Rahul Menon

June 2021

1 Introduction

This document is a list of derivations of equations related to backpropagation in a multi-layer perceptron. Generalized equations for backpropagation are covered, as well as specific cases for ReLU, sigmoid, and softmax activated layers.

1.1 Notation

Within this document, we have a particular notation to refer to various parts of the network. L represents the loss function. J represents the total number of layers in the network. The symbols a, z, w, and b represent a specific layer's activations, pre-activations, weights, and biases, respectively. The index of the layer is provided by a superscript (e.g. a^j represents the activations of layer j). We also use σ to represent a general activation function. We use ϕ^j as a short hand for $\nabla_{z^j} L$. Any of the vector values can be indexed by a subscript (e.g. a_i^j is the i'th element of the activations of the j'th layer). Any of the matrix values can be indexed by multiple comma-separated subscripts (e.g. $w_{i,k}^j$ is the element in the i'th row and k'th column of the weights matrix of the j'th layer).

1.2 A Quick Review

This is a brief explanation of how a multi-layer perceptron (MLP) works. Every layer in an MLP is composed of a linear function followed by a non-linear activation function. The result of the linear function is denoted z and is called a pre-activation. The result of the nonlinear function is denoted a and is called an activation; the activation function itself is denoted σ. a and z have the following definitions.

$$z^j = w^j a^{j-1} + b^j$$
$$a^j = \sigma(z^j)$$

Note that for the first layer ($j = 1$), we treat the network inputs as a^0.

Once the activations for the last layer (a^J) have been calculated, we can then quantify the performance of the MLP with a loss function. In this document, we

denote the loss function as L. There are several kinds of loss functions but they all take the produced output of the network and calculate some form of difference from the ground truth. Some examples that we will see in this document include mean squared-error (MSE), binary cross-entropy, and categorical cross-entropy.

Recall that the end goal is to produce network outputs that are as similar as possible to the ground truths. We can achieve this by minimizing the loss function. We will slightly nudge the values of the weights and biases of every layer in the direction which minimizes the loss the most. By repeating this process many times, we can arrive at a minimum of the loss function. This process is known as gradient descent, and the process of calculating the nudges for each weight and bias value is known as backpropagation.

2 General Backpropagation

In this section, we derive the following four equations that define the majority of backpropagation.

$$\phi^J = \nabla_{a^J} L \circ \sigma'(z^J) \tag{1}$$

$$\phi^j = (w^{j+1})^T \phi^{j+1} \circ \sigma'(z^j) \tag{2}$$

$$\nabla_{w^j} L = \phi^j (a^{j-1})^T \tag{3}$$

$$\nabla_{b^j} L = \phi^j \tag{4}$$

Equation 1 describes the gradient of the loss function with respect to the pre-activations of the last layer. Equation 2 describes a recursive formula for calculating the gradient of the loss function with respect to the pre-activations of any layer given the same for the next layer. Equation 3 defines the gradient of the loss function with respect to the weights of any layer, a scaled version of which is used as the value to update those weights by. Equation 4 defines a similar gradient but for biases instead of weights.

2.1 Last Layer Gradient (Equation 1)

We wish to derive an equation for $\nabla_{z^J} L$ which we denote with ϕ^J. Recall that a loss function is defined in terms of a ground truth y and network output a^J, often denoted $\hat{y}$. $\hat{y}$ in turn is a function of z^J. We can then use the chain rule to get the gradient we want.

$$\begin{aligned}
\phi^J &= \nabla_{z^J} L \\
&= \nabla_{\hat{y}} L \circ \nabla_{z^J} \hat{y} \\
&= \nabla_{a^J} L \circ \nabla_{z^J} a^J
\end{aligned}$$

Since we define $a^J = \sigma(z^J)$, we can also define $\nabla_{z^J} a^J = \sigma'(z^J)$. The value of $\nabla_{a^J} L$ will depend on the specific loss function used. A few examples are shown in section 3. This gives us Equation 1.

$$\phi^J = \nabla_{a^J} L \circ \sigma'(z^J)$$

2.2 Recursive Layer Gradient (Equation 2)

We can continue using the chain rule to find equations for any ϕ^j. However, in this section it is simpler if we also use the component form of our equations.

$$\begin{aligned}\phi_i^j &= \frac{\partial L}{\partial z_i^j} \\ &= \sum_k \frac{\partial L}{\partial z_i^{j+1}} \frac{\partial z_i^{j+1}}{\partial z_i^j} \\ &= \sum_k \phi_i^{j+1} \frac{\partial z_i^{j+1}}{\partial z_i^j}\end{aligned}$$

To evaluate $\frac{\partial z_i^{j+1}}{\partial z_i^j}$, observe that

$$z_k^{j+1} = \sum_i w_{k,i}^{j+1} a_i^j + b_i^{j+1} = \sum_i w_{k,i}^{j+1} \sigma(z_i^j) + b_i^{j+1}$$

Differentiating this, we have

$$\frac{\partial z_i^{j+1}}{\partial z_i^j} = w_{k,i}^{j+1} \sigma'(z_i^j)$$

We can substitute this into our earlier equation to obtain

$$\phi_i^j = \sum_k w_{k,i}^{j+1} \phi_i^{j+1} \sigma(z_i^j)$$

Converting this from component form to matrix form, we obtain Equation 2.

$$\phi^j = (w^{j+1})^T \phi^{j+1} \circ \sigma(z_i^j)$$

Observe that by using both Equations 1 and 2, we can calculate ϕ^j for any $1 \leq j \leq J$.

2.3 Weights Gradient (Equation 3)

Given ϕ^j we can calculate any $\nabla_{w^j} L$ by using the chain rule. We can use component form to simplify our expressions.

$$\begin{aligned}\frac{\partial L}{\partial w_{i,k}^j} &= \frac{\partial L}{\partial z_i^j} \frac{\partial z_i^j}{\partial w_{i,k}^j} \\ &= \phi_i^j \frac{\partial z_i^j}{\partial w_{i,k}^j}\end{aligned}$$

Recall that we have the following definition of z_i^j.

$$z_i^j = b_i^j + \sum_k w_{i,k}^j a_k^{j-1}$$

Differentiating, we obtain

$$\frac{\partial z_i^j}{\partial w_{i,k}^j} = a_k^{j-1}$$

We can then substitute this into our earlier expression and convert back to matrix form to obtain Equation 3.

$$\frac{\partial L}{\partial w_{i,k}^j} = \phi_i^j a_k^{j-1}$$
$$\nabla_{w^j} L = \phi^j (a^{j-1})^T$$

2.4 Biases Gradient (Equation 4)

We can calculate any $\nabla_{b^j} L$ given ϕ^j. We will once again be using the chain rule.

$$\begin{aligned}\nabla_{b^j} L &= \nabla_{z^j} L \nabla_{b^j} z^j \\ &= \phi^j \nabla_{b^j} z^j\end{aligned}$$

Recall our definition of z^j.

$$z^j = w^j a^j + b^j$$

Differentiating, we have that

$$\nabla_{b^j} z^j = \mathbb{1}$$

where $\mathbb{1}$ is a matrix composed of ones. Then, substituting back into our earlier equation, we obtain Equation 4.

$$\nabla_{b^j} L = \phi^j \mathbb{1}$$
$$\nabla_{b^j} = \phi^j$$

3 Specific cases

In the following sections, we evaluate Equations 1 and 2 for specific common activation functions. Equation 2 applies for hidden layers, while Equation 1 applies for output (last) layers.

3.1 ReLU

The Rectified Linear Unit (ReLU) activation function operates on a scalar and is applied elementwise on a vector input. It is defined as follows.

$$\sigma(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

The ReLU function is technically not differentiable at $x = 0$, but we need to be able to evaluate the derivative at that point. For the purposes of most machine learning, therefore, we often choose to define the derivative as 0 at 0. Thus, the derivative of ReLU is defined as follows.

$$\sigma'(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

3.1.1 ReLU in the Last Layer

A variety of loss functions are often used with ReLU activations in the last layer. We will only consider mean squared-error (MSE) in this document. MSE is defined as follows.

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Recall that $\hat{y} = a^J$ so $\nabla_{\hat{y}} L = \nabla_{a^J} L$. Then we have the following.

$$\begin{aligned} \nabla_{a^J} L &= \nabla_{\hat{y}} L \\ &= \nabla_{\hat{y}} \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right) \\ &= -\frac{2}{n} (y - \hat{y}) \\ &= -\frac{2}{n} (y - a^J) \end{aligned}$$

Substituting this and our defined derivative for ReLU into Equation 1, we get the ReLU form of Equation 1.

$$\begin{aligned} \phi^J &= \nabla_{a^J} L \circ \sigma'(z^J) \\ &= -\frac{2}{n} (y - a^J) \circ \begin{cases} 1 & z^J > 0 \\ 0 & z^J \leq 0 \end{cases} \\ \phi^J &= \begin{cases} -\frac{2}{n} (y - a^J) & z^J > 0 \\ 0 & z^J \leq 0 \end{cases} \end{aligned}$$

Note that the comparisons performed above are performed elementwise and so produce individual elements of a vector.

3.1.2 ReLU in a Hidden Layer

Evaluating the ReLU form of Equation 2 is just a matter of plugging in our ReLU derivative.

$$\begin{aligned}\phi^j &= (w^{j+1})^T \phi^{j+1} \circ \sigma'(z^j) \\ &= (w^{j+1})^T \phi^{j+1} \circ \begin{cases} 1 & z^j > 0 \\ 0 & z^j \leq 0 \end{cases} \\ \phi^j &= \begin{cases} (w^{j+1})^T \phi^{j+1} & z^j > 0 \\ 0 & z^j \leq 0 \end{cases}\end{aligned}$$

3.2 Sigmoid

The sigmoid activation function also operates on scalar values and is applied elementwise on a vector input. It is defined as follows.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Finding the derivative of sigmoid is not covered in this document (consider it an exercise for the reader). This derivative can be expressed as below, which is convenient for code implementation purposes.

$$\sigma'(x) = \sigma(x) \circ (1 - \sigma(x))$$

3.2.1 Sigmoid in the Last Layer

In the last layer, sigmoid is often paired with the binary cross-entropy loss function, which is defined as follows.

$$L = \sum_{i=1}^{n} -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

We keep the summation in the equation for use with multi-label classifications problems, but for binary classification problems $n = 1$ so the summation is not necessary. However, for the sake of generality, we will consider the summation

form. Now we can again use the fact that $\hat{y} = a^J$ so $\nabla_{\hat{y}} L = \nabla_{a^J} L$.

$$\begin{aligned}
\nabla_{a^J} L &= \nabla_{\hat{y}} L \\
&= \nabla_{\hat{y}} \left(\sum_{i=1}^{n} -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \right) \\
&= -\frac{y}{\hat{y}} \log(e) - \frac{1 - y}{1 - \hat{y}} \log(e) \\
&= -\frac{y}{\hat{y}} - \frac{1 - y}{1 - \hat{y}} \\
&= -\frac{y}{a^J} - \frac{1 - y}{1 - a^J}
\end{aligned}$$

Notice that we drop the constant $\log(e)$. Dropping such constants are common practice in machine learning math because they only serve to linearly scale and not to provide any extra information. We are now ready to substitute this with our sigmoid derivative into Equation 1 to obtain the sigmoid form of Equation 1.

$$\begin{aligned}
\phi^J &= \nabla_{a^J} L \circ \sigma'(z^J) \\
\phi^J &= \left(-\frac{y}{a^J} - \frac{1 - y}{1 - a^J} \right) \circ \sigma(x) \circ (1 - \sigma(x))
\end{aligned}$$

3.2.2 Sigmoid in a Hidden Layer

As with ReLU, evaluating the sigmoid form of Equation 2 is simply a matter of substituting in our sigmoid derivative.

$$\begin{aligned}
\phi^j &= (w^{j+1})^T \phi^{j+1} \circ \sigma'(z^j) \\
\phi^j &= (w^{j+1})^T \phi^{j+1} \circ \sigma(x) \circ (1 - \sigma(x))
\end{aligned}$$

3.3 Softmax

Unlike ReLU or sigmoid, the softmax activation function operates on vector values. It converts the input vector into a sort of probability density function such that the sum of elements is equal to 1. It is defined as follows.

$$\sigma(x) = \frac{e^x}{\sum_i e^{x_i}}$$

It is important to note that softmax is exclusively used as an activation function for the output layers of networks, and as such we do not evaluate Equation 2 for it. We also will not evaluate the derivative of softmax, nor will it be necessary for evaluating Equation 1.

3.3.1 Softmax in the Last Layer

Softmax is almost always paired with a categorical cross-entropy loss function, which is defined as follows.

$$L = -\sum_j y_j \log \hat{y}_j$$

With softmax, trying to substitute into Equation 1 as we did with ReLU and sigmoid is rather difficult. Instead, we attempt to directly differentiate L with respect to z^J using the fact that $\hat{y} = a^J = \sigma(z^J)$.

We can first take advantage of the fact that y is a one-hot vector to simplify our definition of L. Observe that by definition, a one-hot vector has one entry with value 1 and the rest with value 0. This means that only a single term in the sum in our definition of L is nonzero, so the sum itself is equivalent to that term. Let the index of that term be γ such that $y_\gamma = 1$.

$$\begin{aligned}
L &= -y_\gamma \log \hat{y}_\gamma \\
&= -\log \hat{y}_\gamma \\
&= -\log \left(\frac{e^{z_\gamma^J}}{\sum_k e^{z_k^J}} \right) \\
&= -\log e^{z_\gamma^J} + \log \sum_k e^{z_k^J} \\
&= -z_\gamma^J + \log \sum_k e^{z_k^J}
\end{aligned}$$

With this new definition of L, finding $\nabla_{z^J} L$ is much simpler.

$$\begin{aligned}
\nabla_{z^J} L &= \nabla_{z^J} \left(-z_\gamma^J + \log \sum_k e^{z_k^J} \right) \\
&= \nabla_{z^J} \log \sum_k e^{z_k^J} - \nabla_{z^J} z_\gamma^J
\end{aligned}$$

Here we can make use of a property of logarithms. Specifically, we can use the fact that $\frac{d}{dx}\ln(f(x)) = \frac{1}{f(x)}\frac{d}{dx}f(x)$. Notice that this is simply an application of the chain rule.

$$\begin{aligned}
\nabla_{z^J} L &= \frac{1}{\sum_k e^{z_k^J}} \nabla_{z^J} \sum_k e^{z_k^J} - \nabla_{z^J} z_\gamma^J \\
&= \frac{e^{z^J}}{\sum_k e^{z_k^J}} - \nabla_{z^J} z_\gamma^J \\
&= \frac{e^{z^J}}{\sum_k e^{z_k^J}} - \begin{cases} 1 & i = \gamma \\ 0 & i \neq \gamma \end{cases} \\
&= \hat{y} - \begin{cases} 1 & i = \gamma \\ 0 & i \neq \gamma \end{cases} \\
&= \hat{y} - y \\
&= a^J - y
\end{aligned}$$

The piecewise portion of the above equations refers to the creation of a one-hot matrix where the γ'th element is 1. The result of the above is the softmax form of Equation 1. Thus we have

$$\phi^J = a^J - y$$