DS 5220, Lecture 15: The Backpropagation Algorithm

October 25, 2024

In this lecture, we will provide an in-depth study of the backpropagation algorithm. First, let us consider a multi-layer, linear neural network example. I believe this is the simplest possible example, for which we could illustrate the key steps behind the backpropagation algorithm.

Example 1. *In a linear neural network, we assume that each layer uses a linear activation function. That is, $\sigma(x) = x$. Suppose there are L layers in total. In each layer, we have one variable w_i associated with that layer, for $i = 1, 2, \ldots, L$. No bias is involved.*

The output of this network would be equal to

$$f(x) = w_L w_{L-1} \cdots w_1 x.$$

Suppose we consider the squared loss. Then, the loss function would be

$$\ell(f(x), y) = (f(x) - y)^2.$$

Here, let us work out the gradient of the above loss function, with respect to the weight variables $W = [w_1, w_2, \ldots, w_L]$. In other words, we want to compute $\nabla_W \ell(f(x), y)$. We shall work backward. In other words, we will first find out $\frac{\partial \ell}{\partial w_L}$, then $\frac{\partial \ell}{\partial w_{L-1}}, \ldots$, finally $\frac{\partial \ell}{\partial w_1}$.

To facilitate this calculation, let us set up the input and output to a layer i as follows, for any $i = 1, 2, \ldots, L$:

- Input to layer i: This is $z_i = w_{i-1} \cdots w_1 x$.
- Output of layer i: This is $o_i = w_{i-1} \cdots w_1 x$ (since activation function is linear).

First, let us work on $\frac{\partial \ell}{\partial w_L}$. We can rewrite the loss as $\ell(f(x), y) = (w_L z_L - y)^2$. Thus,

$$\frac{\partial \ell}{\partial w_L} = 2(w_L z_L - y) z_L$$

Next, we look at

$$\frac{\partial \ell}{\partial w_{L-1}} = \frac{\partial \ell}{\partial z_L} \cdot \frac{\partial z_L}{\partial w_{L-1}} = \frac{\partial \ell}{\partial z_L} \cdot o_{L-1}. \tag{1}$$

Above, we notice that $z_L = w_{L-1} o_{L-1}$. We can generalize equation (1) to any intermediate layer as

$$\frac{\partial \ell}{\partial w_i} = \frac{\partial \ell}{\partial z_{i+1}} \cdot \frac{\partial z_{i+1}}{\partial w_i} = \frac{\partial \ell}{\partial z_{i+1}} \cdot o_i, \tag{2}$$

because $z_{i+1} = w_i o_i$.

Based on equation (2), we will then need to work out the expression for $\frac{\partial \ell}{\partial z_{i+1}}$, for any $i = L-1, L-2, \ldots, 1$. Here, we will again set up a recursion to derive this:

$$\begin{aligned}\frac{\partial \ell}{\partial z_i} &= \frac{\partial \ell}{\partial z_{i+1}} \cdot \frac{\partial z_{i+1}}{\partial z_i} \\ &= \frac{\partial \ell}{\partial z_{i+1}} \cdot w_i, \end{aligned} \tag{3}$$

using the fact that $z_{i+1} = w_i z_i$, since the activation is linear.

To summarize this discussion, we may write the backpropagation algorithm corresponding to Example 1 as follows:

1. First, compute $z_1, z_2, \ldots, z_L$ according to the forward pass.
2. Then, compute
$$\frac{\partial \ell}{\partial z_L} = \frac{\partial (w_L z_L - y)^2}{\partial z_L} = 2(w_L z_L - y) w_L$$
3. For any $i = L-1, \ldots, 1$, use equation (3) to get $\frac{\partial \ell}{\partial z_i}$ from $\frac{\partial \ell}{\partial z_{i+1}}$.
4. Finally, use equation (2) to get $\frac{\partial \ell}{\partial w_i}$ for any $i = L-1, L-2, \ldots, 1$.

We can streamline steps 2-4 in a more compact way, leading to the backward pass as follows:

- At layer L, calculate $\frac{\partial \ell}{\partial w_L}$ and $\frac{\partial \ell}{\partial z_L}$.
- For any $i = L-1, L-2, \ldots, 1$, calculate $\frac{\partial \ell}{\partial w_i}$ and $\frac{\partial \ell}{\partial z_i}$ based on equations (2) and (3).

Example 2 (Two-layer, multi-dimensional ReLU network)**.** *In the above example, we considered a one-dimensional setting, thus ignoring the complexity introduced by matrix multiplications. Now, let's take that into account and do the calculation again. Suppose we have an input $x \in \mathbb{R}^p$. We pass through a two-layer ReLU neural network with W_1, b_1 in the first layer and W_2, b_2 in the second layer. We shall consider a regression problem first and we'll discuss the case of classification problems after we finish this.*

For this regression problem, we can set $W_1 \in \mathbb{R}^{p \times d_1}$, $b_1 \in \mathbb{R}^{d_1}$, and $W_2 \in \mathbb{R}^{d_1}$. For simplicity, let's not worry about b_2 (incorporating that should be simple in principle). We can write the network output as

$$f(x) = \sigma(W_1^\top x + b_1)^\top W_2,$$

where $\sigma(\cdot)$ is the ReLU activation function applied to every coordinate of the input.

Now, let's work out the gradient of the squared loss with respect to the three variables in Example 2. We need to calculate the following: $\nabla_{W_1} \ell$, $\nabla_{b_1} \ell$, and $\nabla_{W_2} \ell$. The last one is simple (note: illustrate vector calculus on the board):

$$\nabla_{W_2} \ell = (2(f(x) - y)) \cdot \sigma(W_1^\top x + b_1).$$

As for W_1 and b_1, let us focus on $\nabla_{W_1} \ell$ (the case for $\nabla_{W_1} \ell$ should be similar). Since W_1 is a p by d_1 matrix/array, we shall look into an individual coordinate of W_1, then we can extrapolate the patterns from that and summarize it in matrix calculus. Let $W_1[i,j]$ denote the i,j-th entry of the matrix W_1. We shall look at

$$\begin{aligned}\frac{\partial \ell}{\partial W_1[i,j]} &= (2(f(x)-y))\frac{\partial f(x)}{\partial W_1[i,j]} && (4)\\ &= (2(f(x)-y))\frac{\partial(\sigma(W_1^\top x+b_1)^\top)W_2}{\partial W_1[i,j]} && (5)\end{aligned}$$

Notice that for $\sigma(W_1^\top x+b_1)^\top$, except the for the j-th entry, the rest of the entries are independent of $W_1[i,j]$. As for the j-th entry, by chain rule, the derivative is equal to

$$\frac{\partial \ell}{\partial W_1[i,j]} = (2f(x)-y)\sigma'(W_1^\top x+b_1)[j]\cdot x_i\cdot W_2[j]. \tag{6}$$

As a result, we may write the gradient as

$$\nabla_{W_1}\ell = (2f(x)-y)x\cdot(\sigma'(W_1^\top x+b_1)\odot W_2)^\top. \tag{7}$$

The above two examples illustrate the difficulty in terms of deriving the backpropagation algorithm. We now discuss the most general case to finish this discussion. Suppose we have L layers of feedforward neurons. From layer i to layer $i+1$, the transformation goes as follow:

$$\begin{aligned} z_{i+1} &= W_{i+1}^\top o_i + b_{i+1}, && (8)\\ o_{i+1} &= \sigma(z_{i+1}), && (9)\end{aligned}$$

for $i=0,1,\ldots,L-1$, where $W_{i+1}\in\mathbb{R}^{d_i\times d_{i+1}}$, and $b_{i+1}\in\mathbb{R}^{d_{i+1}}$. Until at the last layer, o_L is used in the loss function along with the final label of y.

Suppose we already knew what is $\frac{\partial\ell}{\partial z_{i+1}}$ and $\frac{\partial\ell}{\partial W_{i+1}}$. Based on these results, we are going to use them to infer $\frac{\partial\ell}{\partial z_i}$ and $\frac{\partial\ell}{\partial W_i}$.

Since z_i is a vector of dimension d_i, we'll use the multivariate chain rule[1], which requires us to look into every coordinate of z_i. Let j be any value between 1 and d_i. Then,

$$\frac{\partial\ell}{\partial z_i[j]} = \langle\frac{\partial\ell}{\partial z_{i+1}},\frac{\partial z_{i+1}}{\partial z_i[j]}\rangle \tag{10}$$

Here, notice that

$$z_{i+1} = W_{i+1}^\top\sigma(z_i)+b_{i+1}$$

Therefore,

$$\frac{\partial z_{i+1}}{\partial z_i[j]} = W_{i+1}^\top[:,j]\cdot\sigma'(z_i)[j]$$

Hence, we can write the above into equation (10), leading to

$$\sigma'(z_i)[j]\frac{\partial\ell}{\partial z_{i+1}}^\top W_{i+1}^\top[:,j] = \sigma'(z_i)[j]W_{i+1}[j,:]\frac{\partial\ell}{\partial z_{i+1}}$$

[1] https://math.libretexts.org/Bookshelves/Calculus/Calculus_(OpenStax)/14%3A_Differentiation_of_Functions_of_Several_Variables/14.05%3A_The_Chain_Rule_for_Multivariable_Functions

Thus, we find that

$$\frac{\partial \ell}{\partial z_i} = \mathrm{diag}\left(\sigma'(z_i)\right) W_{i+1} \frac{\partial \ell}{\partial z_{i+1}} \tag{11}$$

With this result, we could apply recursion to get $\frac{\partial \ell}{\partial z_i}$, for all $i = L, L-1, \ldots, 1$.

We could also follow the above procedure to derive $\frac{\partial \ell}{\partial W_{i+1}}$. This is by taking the chain rule on equation (8). In particular, we could verify that $\frac{\partial \ell}{\partial W_{i+1}} = o_i (\frac{\partial \ell}{\partial z_{i+1}})^\top$ (the details are omitted; please try to verify it by yourself after class!).

Consequences: Vanishing & exploding gradients.

Exercise: Can you come up with an example to explain why vanishing gradients can happen within backpropagation?