Tashfeen, Ahmad
Personal Project
July 4, 2020

Tashfeen's Notes on Backpropagation

1. Chain Rule

At the heart of Backpropogation is a simple rule of calculus that helps us take the derivatives of composite functions known as *the chain rule*.

$$\zeta(x) = f(g(x)) \quad \text{then according to the chain rule:} \quad \frac{\mathrm{d}\,\zeta}{\mathrm{d}\,x} = \frac{\mathrm{d}\,f}{\mathrm{d}\,g} \times \frac{\mathrm{d}\,g}{\mathrm{d}\,x}$$

It is important to note there that $\frac{\mathrm{d}\,f}{\mathrm{d}\,x}$ denotes the total derivative of f with respect to x and not the partial derivative which is denoted as $\frac{\partial f}{\partial x}$. The general idea behind applying chain rule is that you start with the outer most function and take it's derivative while treating all inner functions as variables and recursively repeat this for the inner functions. Here is an intuitive way to think about it [**?**],

$$\begin{array}{c}\text{The change in } \zeta \text{ caused by} \\ \text{a small unit change in } x\end{array} = \begin{array}{c}\text{The change in } f \text{ caused by} \\ \text{a small unit change in } g\end{array} \times \begin{array}{c}\text{The change in } g \text{ caused by} \\ \text{a small unit change in } x.\end{array}$$

Let's look at an example where $\zeta(x) = \ln^2(x)$, we can write this as $\zeta(x) = f(x) = g(x)^2$ where $g(x) = \ln(x)$. Here f is the outer most function and then we have g as the inner function.

$$\frac{\mathrm{d}\,\zeta}{\mathrm{d}\,x} = \frac{\mathrm{d}\,f}{\mathrm{d}\,g} \times \frac{\mathrm{d}\,g}{\mathrm{d}\,x} = 2g(x) \times \frac{1}{x} = \frac{2\ln(x)}{x}$$

For a slightly more involved example let $\zeta(x) = (z(x))^2$ where $z(x) = x + f(x), f(x) = \ln(g(x))$ and $g(x) = \frac{1}{2}x^2$ then ζ' or $\frac{d\zeta}{dx}$ is defined as,

$$\begin{aligned}
\zeta'(x) &= \frac{\mathrm{d}\,\zeta}{\mathrm{d}\,z} \times \frac{\mathrm{d}\,z}{\mathrm{d}\,x} = 2z(x) \times z'(x) \\
&= \frac{\mathrm{d}\,\zeta}{\mathrm{d}\,z}\left(\frac{\mathrm{d}\,x}{\mathrm{d}\,x} + \frac{\mathrm{d}\,f}{\mathrm{d}\,x}\right) = 2z(x) \times (1 + f'(x)) && \text{Substituting in place of } \frac{\mathrm{d}\,z}{\mathrm{d}\,x} \\
&= \frac{\mathrm{d}\,\zeta}{\mathrm{d}\,z}\left(\frac{\mathrm{d}\,x}{\mathrm{d}\,x} + \left(\frac{\mathrm{d}\,f}{\mathrm{d}\,g}\frac{\mathrm{d}\,g}{\mathrm{d}\,x}\right)\right) = 2z(x)\left(1 + \left(\frac{1}{g(x)}g'(x)\right)\right) && \text{Substituting in place of } \frac{\mathrm{d}\,f}{\mathrm{d}\,x} \\
&= \frac{\mathrm{d}\,\zeta}{\mathrm{d}\,z}\left(\frac{\mathrm{d}\,x}{\mathrm{d}\,x} + \left(\frac{\mathrm{d}\,f}{\mathrm{d}\,g}\frac{\mathrm{d}\,g}{\mathrm{d}\,x}\right)\right) = 2z(x)\left(1 + \left(\frac{1}{g(x)}x\right)\right) && \text{Substituting in place of } \frac{\mathrm{d}\,g}{\mathrm{d}\,x}
\end{aligned}$$

2. Gradient & Gradient Decent

Most people are familiar with the slope of a linear function, e. g., for $y = f(x) = mx + b$ we know that the rate of change or in other words the slope is m. Let $f(x) = \frac{3}{2}x - 3$ and suppose the problem is to find such a value of x such that $f(x) = 0$. Obviously in this case we can see that for $x = 2$ we have $f(x) = 0$. For the sake of illustration though pretend this was not such a simple problem that we could just observe the minimising x value. The next step would be to set f equal to zero and solve for x but what if we did not have an easy way to write down f? What if we could only compute the numerical value of $f(x)$ for a given x using some algorithm. How could we find a value of x for which $f(x) = 0$?

The first question we need to answer is that in how many directions can we nudge the value of x and observe the corresponding value of $f(x)$. Since the notion of direction becomes inconceivable in higher

dimensions, let's pose this question as, how many ways can x be changed? Fortunately, for a linear function there are only two ways or two directions we can nudge x to, i. e., positive and negative.

Can we be smart about picking if we want to nudge x in the positive of negative direction? Yes, we can if we know how $f(x)$ changes with respect to a change in x. If the slop is positive, we should nudge x in the negative direction to reduce $f(x)$ to 0. Similarly, if the slope is negative, then we should nudge x in positive direction. There seems to be an informal generalisation here. To reduce a function $f(x)$ from some starting point x_0, we should move in the opposite direction of the positive rate of change.

When we move from a linear to a polynomial functions say $f(x) = x^2$, the idea of moving in the opposite direction of the positive rate of change remains the same. However, now instead of slops, we have derivatives. We know that $\frac{\mathrm{d}f}{\mathrm{d}x} = 2x$. Therefore, if we start at $x_0 = 10$ or any $x_0 > 0$, we should nudge x_0 in the negative direction since $f'(10) > 0$, is positive. Similarly if we start at any $x_0 < 0$, we should nudge x_0 in the positive direction.

At this point we can sketch out an algorithm to minimise a function $f(x)$ by adjusting x according to it's rate of change.

1) Pick a starting point, call it x_0.
2) Pick the next point x_1 according to the rate of change at $f(x_0)$.
3) If the difference between $f(x_1)$ and $f(x_0)$ is under some desired value ε stop. Else, repeat step 2 with the new x_1.

What if the function in question was defined in terms of multiple variables, as in $f : \mathbb{R}^n \to \mathbb{R}^n$ where $n > 1$. Now we figure out the positive rate of change (after slopes and derivatives) using the gradient of the multi-variable function. A gradient of a function $f(p_0, p_1, p_2, \ldots, p_n)$ is denoted as $\nabla f(p_0, p_1, p_2, \ldots, p_n)$ and is defined in terms of the partial derivatives,

$$\nabla f(p_0, p_1, p_2, \ldots, p_n) = \nabla f(\mathbf{p}) = \left[\frac{\partial f(\mathbf{p})}{\partial x_0}, \frac{\partial f(\mathbf{p})}{\partial x_1}, \frac{\partial f(\mathbf{p})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{p})}{\partial x_n}\right]$$

The vector $\nabla f(\mathbf{p})$ gives us the nudges we can make in the direction of the steepest ascent. Similar to our idea from before, we need to step in the opposite direction of the gradient to find the $\mathbf{p} = [p_0, p_1, p_2, \ldots, p_n]$ that minimises f. Assume for $f : \mathbb{R}^2 \to \mathbb{R}^2$, we have $f(x, y) = x^2 + y^2$. Calculating the gradient we get,

$$\nabla f(x, y) = \left[\frac{\partial f(x, y)}{\partial x}, \frac{\partial f(x, y)}{\partial y}\right] = [2x, 2y]$$

As seen in the gradient above, for $(x, y) = (0, 0)$ the function f is minimised. In other words, there is a global minima at $(0, 0)$. But in order to keep our method general for even the more complicated functions, we repeat the same algorithm we sketched with the polynomial functions. We can now rewrite the algorithm in terms of the gradient; this final algorithm is called the *gradient decent*.

1) Pick a random $\mathbf{p}_{n=0}$.
2) Let $\mathbf{p}_{n+1} = \mathbf{p}_n - \eta \nabla f(\mathbf{p}_n)$.
3) If $||f(\mathbf{p}_{n+1}) - f(\mathbf{p}_n)|| < \varepsilon$ then stop.
4) Repeat step 2 with $\mathbf{p}_{n+1}$.

Here ε is the error we are okay with and η is the size of the nudge we make. η is also known as the learning rate since the bigger the nudge, the faster we learn.

Following is the graph for gradient decent on $f(x, y) = x^2 + y^2$ and $\nabla f(x, y) = [2x, 2y]$.

3. Backpropagation with Gradient Descent & Chain Rule

The aim of learning in a Multi-layer Perceptron a. k. a. the simplest Neural Network is to find a set of weight matrices $\mathcal{W}$ and bias vectors $\mathbf{b}$ which minimise the cost/loss function. We use the quadratic cost function defined as,

$$C(\mathcal{W}, \mathbf{b}) = \frac{1}{n} \sum_x (y - a^L)^2$$

Here $a^{(L)}$ is the squishified output of the network calculated using $(\mathcal{W}, \mathbf{b})$ when the example x is fed and y is the expected output. We are summing this expression over all training examples which gives us the cost of the network.

Here are other forward feeding equations (might add more about these later),

$$\begin{aligned} a^{(l)} &= \sigma(z^{(l)}) \quad \text{or} \quad \text{relu}(z^{(l)}) \\ z^{(l)} &= w^{(l)}a^{(l-1)} + b^{(l)} \qquad\qquad \text{where } w^{(l)} \in \mathcal{W}, b^{(l)} \in \mathbf{b} \end{aligned}$$

Here is where we need to be able to calculate the gradient of cost function since we want to find some $(\mathcal{W}, \mathbf{b})$ which minimise the cost of the network. With the gradient, we'll be able to employ the (stochastic) gradient decent.

$$\nabla C(\mathcal{W}, \mathbf{b}) = \left[\frac{\partial C}{\partial w^{(L)}}, \frac{\partial C}{\partial b^{(L)}}, \frac{\partial C}{\partial w^{(L-1)}}, \frac{\partial C}{\partial b^{(L-1)}} \cdots, \frac{\partial C}{\partial w^{(1)}}, \frac{\partial C}{\partial b^{(1)}}\right]$$

To figure out $\frac{\partial C}{\partial w^{(L)}}$ we shall employ the chain rule,

$$\begin{aligned} \frac{\partial C}{\partial w^{(L)}} &= \frac{1}{n}\sum_x \left(\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\frac{\partial (y-a^{(L)})}{\partial w^{(L)}}\right) && \text{Chain Rule } \frac{\partial f(g(x))}{\partial (x)} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} \\ &= \frac{1}{n}\sum_x \left(\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\Big(\frac{\partial (y)}{\partial w^{(L)}} - \frac{\partial (a^{(L)})}{\partial w^{(L)}}\Big)\right) && \text{Derivative Property } \frac{\partial (f \pm g)}{\partial x} = \frac{\partial (f)}{\partial x} \pm \frac{\partial (g)}{\partial x} \\ &= \frac{1}{n}\sum_x \left(\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\Big(0 - \frac{\partial (a^{(L)})}{\partial w^{(L)}}\Big)\right) && \frac{\partial (y)}{\partial w^{(L)}} = 0, \text{ since } y \text{ does not change with respect to} \\ &= \frac{1}{n}\sum_x \left(\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\Big(-\frac{\partial (a^{(L)})}{\partial w^{(L)}}\Big)\right) \\ &= \frac{1}{n}\sum_x \left(-\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\frac{\partial (a^{(L)})}{\partial w^{(L)}}\right) && \text{Brought the negative sign to the front of differential} \\ &= \frac{1}{n}\sum_x \left(-\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\Big(\frac{\partial (\sigma(z^{(L)}))}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}}\Big)\right) && \text{Chain Rule} \\ &= \frac{1}{n}\sum_x \left(-\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\frac{\partial (\sigma(z^{(L)}))}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}}\right) \end{aligned}$$

$$\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})} = 2(y-a^{(L)}) \qquad \text{Evaluating differentials}$$

$$\frac{\partial (\sigma(z^{(L)}))}{\partial z^{(L)}} = \sigma'(z^{(L)}) \qquad \text{Derivative of the sigmoid } \sigma'(x) = \sigma(x)(1-\sigma(x))$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = \frac{\partial (w^{(L)}a^{l-1} + b^{(L)})}{\partial w^{(L)}} = a^{l-1}$$

Therefore,

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n}\sum_x \left(-\frac{\partial (y-a^{(L)})^2}{\partial (y-a^{(L)})}\frac{\partial (\sigma(z^{(L)}))}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}}\right) = \frac{1}{n}\sum_x \Big(-2(y-a^{(L)})\sigma'(z^{(L)})a^{L-1}\Big)$$

Similarly,

$$\frac{\partial(w^{(L)}a^{l-1}+b^{(L)})}{\partial b^{(L)}}=1\Rightarrow\frac{\partial C}{\partial b^{(L)}}=\frac{1}{n}\sum_x\left(-\frac{\partial(y-a^{(L)})^2}{\partial(y-a^{(L)})}\frac{\partial(\sigma(z^{(L)}))}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial b^{(L)}}\right)=\frac{1}{n}\sum_x\left(-2(y-a^{(L)})\sigma'(z^{(L)})(1)\right)$$

If you are reading other literature on Neural Networks you may also come by $\delta^{(L)}$ known as the *error in layer l*. In our calculations above these can be identified like this,

$$\frac{1}{n}\sum_x\left(\underbrace{-\frac{\partial(y-a^{(L)})^2}{\partial(y-a^{(L)})}\frac{\partial(\sigma(z^{(L)}))}{\partial z^{(L)}}}_{\delta^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}}\right)=\frac{1}{n}\sum_x\left(\underbrace{-2(y-a^{(L)})\sigma'(z^{(L)})}_{\delta^{(L)}}a^{l-1}\right)$$

But what about $\frac{\partial C}{\partial w^{(l)}}$, where $l<L$, e. g., $l=L-1$. We just keep applying the chain rule since,

$$a^{(L)}=\sigma(w^{(L)}\underbrace{a^{(L-1)}}_{\sigma(z^{(L-1)})}+b^{(L)})\iff a^{(L)}=\sigma(w^{(L)}\sigma(z^{L-1})+b^{(L)})=\sigma(w^{(L)}\sigma(w^{(L-1)}a^{(L-2)}+b^{(L-1)})+b^{(L)})$$

$$\begin{aligned}
\frac{\partial C}{\partial w^{(L-1)}}&=\frac{1}{n}\sum_x\left(-\frac{\partial(y-a^{(L)})^2}{\partial(y-a^{(L)})}\frac{\partial(\sigma(z^{(L)}))}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}}\right)\\
&=\frac{1}{n}\sum_x\left(-\frac{\partial(y-a^{(L)})^2}{\partial(y-a^{(L)})}\frac{\partial(\sigma(z^{(L)}))}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial a^{(L-1)}}\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}}\frac{\partial z^{(L-1)}}{\partial w^{(L-1)}}\right)\\
&=\frac{1}{n}\sum_x\left(\underbrace{\delta^{(L)}\frac{\partial z^{(L)}}{\partial a^{(L-1)}}\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}}}_{\delta^{(L-1)}}\frac{\partial z^{(L-1)}}{\partial w^{(L-1)}}\right)\\
&=\frac{1}{n}\sum_x\left(\underbrace{\delta^{(L)}w^{(L)}\sigma'(z^{(L-1)})}_{\delta^{(L-1)}}a^{(L-2)}\right)\\
&=\frac{1}{n}\sum_x\left(\delta^{(L-1)}\frac{\partial z^{(L-1)}}{\partial w^{(L-1)}}\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial C}{\partial w^{(L-i)}}&=\frac{1}{n}\sum_x\Bigg(\underbrace{\underbrace{\underbrace{-\frac{\partial(y-a^{(L)})^2}{\partial(y-a^{(L)})}\frac{\partial a^{(L)}}{\partial z^{(L)}}}_{\delta^{(L)}}\frac{\partial z^{(L)}}{\partial a^{(L-1)}}\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}}}_{\delta^{(L-1)}}\frac{\partial z^{(L-1)}}{\partial a^{(L-2)}}\frac{\partial a^{(L-2)}}{\partial z^{(L-2)}}\cdots\frac{\partial z^{(L-i+1)}}{\partial a^{(L-i)}}}_{\delta^{(L-2)}}\frac{\partial a^{(L-i)}}{\partial z^{(L-i)}}\frac{\partial z^{(L-i)}}{\partial w^{(L-i)}}\Bigg)\\
&=\frac{1}{n}\sum_x\left(\delta^{(L-i)}\frac{\partial z^{(L-i)}}{\partial w^{(L-i)}}\right)
\end{aligned}$$

Where we write $\delta^{(L-i)}$ succinctly as,

$$\delta^{(L-i)}=\delta^{(L-i+1)}\frac{\partial z^{(L-i+1)}}{\partial a^{(L-i)}}\frac{\partial a^{(L-i)}}{\partial z^{(L-i)}}=\delta^{(L-i+1)}w^{(L-i+1)}\sigma'(z^{(L-i)})$$

Let $l=L-i, i\in\mathbb{N}$

$$\delta^{(l)} = \delta^{(l+1)} w^{(l+1)} \sigma'(z^{(l)})$$

For the biases,

$$\frac{\partial C}{\partial b^{(L-i)}} = \frac{1}{n} \sum_x \left(\delta^{(L-i)} \frac{\partial z^{(L-i)}}{\partial b^{(L-i)}} \right) = \frac{1}{n} \sum_x \delta^{(L-i)}$$

Since $\frac{\partial z^{(L-i)}}{\partial b^{(L-i)}} = 1$.

North Warren Avenue