Backprop: A Simple Case

phunc20

January 21, 2021

1 NN Model

As I said in `README.md`, we shall adopt the simple NN model from one of Professor Ng's deep learning lecture. But we are going to make it slightly different. Not only are we going to make slight modification to the notations, (e.g. we don't use the bracketed superscript as in $W^{[1]}$; instead, we use ordinary W_1.) but we would also like to adopt the more commonly used convention for arranging the data X in the shape (m, n), where

- $m = \#$(instances)
- $n = \#$(input features)

instead of using shape (n, m) as in the lecture. But do not worry: Modifications are minimal and easy to understand/spot.

Ok, now we are on the same page. Let's first recall how we defined the two-layered NN: (*Oh, BTW, in this article we will use all the time the abbreviation* NN *to mean* Neural Network, *in case any one of you who are reading this article don't know what I want to mean by that.*) There were two standard layers, each followed by an activation function, and after the second activation, we take the cross-entropy with the true probability as our loss function. That was it.

The same thing expressed in mathematical language would be like: Let X be the data at hand (of shape (m, n)) and L denote the loss function. We abbreviate $M_{p,q} = M_{p,q}(\mathbb{R})$, i.e. the set of all p by q matrices with entries in $\mathbb{R}$. Note that L is not to be viewed as depending on X; instead, L depends on the weights and biases and X is more like constants.

$$\begin{array}{rlcl} L: & M_{n,n_1} \times M_{1,n_1} \times M_{n_1,n_2} \times M_{1,n_2} & \to & \mathbb{R} \\ & (W_1, b_1, W_2, b_2) & \mapsto & \Omega\,(\, g_1(XW_1 + b_1)\, W_2 + b_2\,) \end{array}$$

In the above mapping, g_1 represents the activation function of the first layer, W_j, b_j the weight and bias of the j-th layer, resp., Ω the final cross-entropy loss.

We have actually omitted several details which might cause difficulties for those who possess rigorous spirit and who have not seen NN before. First, in the expression $XW_1 + b_1$, XW_1 and b_1 have differentl shapes. By $XW_1 + b_1$, we mean that we shall first broadcast b_1 to the same shape as XW_1 before

doing the addition. Second, in the expression $g_1(XW_1+b_1)$, g_1 as in the lecture was tanh. And by $g_1(Z_1)$, where Z_1 is a matrix, $g1 : \mathbb{R} \to \mathbb{R}$ a function, we mean that the outcome should be a matrix of the same shape as Z_1 with entries $(g_1(Z_1))_{i,j} = g_1((Z_1)_{i,j})$.

Finally, the last function also deserves some explanation. Ω is actually composed of two functions: $\Omega = J \circ f$, where

$$\begin{aligned} f: \quad & \mathbb{R} \to \mathbb{R} \\ & x \mapsto \text{sigmoid}(x) = \frac{e^x}{1+e^x} \end{aligned}$$

and

$$\begin{aligned} J: \quad & M_{m,1} \to \mathbb{R} \\ & A_2 \quad \mapsto -\frac{1}{m}\left(y^T \log(A_2) + (1-y)^T \log(1-A_2)\right) . \end{aligned}$$

Again, we had better explain J a little bit. First of all, we use A_2 to mean an arbitrary m by 1 matrix. Next, y is also an m by 1 matrix, representing the labels (aka ground truth). So J is the (negative) average of a sum of two inner products.

2 Differential Calculus

This simple, two-layer NN results in a loss function which already looks somewhat complicated. Now we want to find its gradient w.r.t. each one of its variables in order to minimize the loss.

Let's recall the definition of differentiability of a function $s : U \subset \mathbb{R}^p \to \mathbb{R}^q$, where U is an open set in $\mathbb{R}^p$. We say that s is differentiable at a point $a \in U$ if in the neighborhood of a, s can be approximated by a linear function $l : \mathbb{R}^p \to \mathbb{R}^q$. That is to say, $s(a+h) = s(a) + l(h) + o(\|h\|)$. Conventionally, people also use the notation $s'(a)$ instead of l. (For readers who feel less clear about what we just said, please refer to, for example, [1,2]).

In deep learning, more often we are dealing with matrices than vectors. For example, if we consider a single layer, we have

$$\begin{aligned} M_{v,w} &\to M_{u,w} \\ W &\mapsto AW + b\,. \end{aligned}$$

Sure enough, there is no difficulty to reshape matrices into vectors by putting columns (or rows) of W one after another to form a big column (or row) vector. I have tried this line of thought and found it **non-trivial** to do the computations. The readers could try themselves; maybe they'll find a new path, or appreciate how hard that might be.

Besides, staying inside the wonderland of matrices has an extra benefit. *Multiplying a matrix by another is a linear operation for both.*

Indeed, the space $M_{v,w}(\mathbb{R}) \cong \mathbb{R}^{vw}$ with its usual operations (scalar multiplication, addition) is no different from the vector space $\mathbb{R}^{vw}$. Furthermore, if we write $\xi(W) = AW + b$ as a function, then $\xi(W + \Delta W) = A(W + \Delta W) + b = \xi(W) + A\Delta W$. The function $\Delta W \mapsto A\Delta W$ being easily verified to be linear, we see that ξ is differentiable and that

$$\begin{aligned} \xi'(W): \quad & M_{v,w} \to M_{u,w} \\ & \Delta W \mapsto A\Delta W \end{aligned}$$

As one can see, things are straight-forward and neat if we choose to stay with the matrices instead of turning them into vectors.

Ok, enough theory for the moment. Let's go back to our original problem and not lose our focus.

3 Try to Find $\frac{\partial L}{\partial W_1}$

Let's try to find a typical gradient, say $\frac{\partial L}{\partial W_1}$, and hope that the other gradients could be computed in a similar manner. Recall that

$$\begin{aligned} L: \quad & M_{n,n_1} \times M_{1,n_1} \times M_{n_1,n_2} \times M_{1,n_2} && \to && \mathbb{R} \\ & (W_1, b_1, W_2, b_2) && \mapsto && \Omega\,(\, g_1(XW_1 + b_1)\, W_2 + b_2\,)\,. \end{aligned}$$

When we compute $\frac{\partial L}{\partial W_1}$, the other variables (i.e. $b1, W_2, b_2$) can be regarded as constants. Thus, we are really dealing with the function

$$\begin{aligned} & M_{n,n_1} \to \mathbb{R} \\ & W_1 \mapsto \Omega\,(\, g_1(XW_1 + b_1)\, W_2 + b_2\,)\,. \end{aligned}$$

This is the composite of several functions; we can make use of the property of differential of composite functions: *The differential of the composite equals the composite of the differentials.* Introducing intermediate variables allows us to write

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Z_2} \circ \frac{\partial Z_2}{\partial A_1} \circ \frac{\partial A_1}{\partial Z_1} \circ \frac{\partial Z_1}{\partial W_1}\,, \quad \text{where}$$

(Note that both sides of the equation are functions $M_{n,n_1} \to \mathbb{R}$)

$$\begin{aligned} Z_1 &= XW_1 + b_1 \\ A_1 &= g_1(Z_1) \\ Z_2 &= A_1W_2 + b_2 \\ L &= \Omega(Z_2)\,. \end{aligned}$$

Note that we already know how to compute one of them.

$$\frac{\partial Z_1}{\partial W_1}: \ \Delta W \mapsto X\Delta W\,.$$

Actually, $\frac{\partial Z_2}{\partial A_1}$ can be proved similarly to be

$$\frac{\partial Z_2}{\partial A_1} : \ \Delta A \mapsto \Delta A W_2 \, .$$

There remain two differentials for us to find out: $\frac{\partial L}{\partial Z_2}$ and $\frac{\partial A_1}{\partial Z_1}$.

Let's first look at $\frac{\partial A_1}{\partial Z_1}$, because it will shed some light later on when we do $\frac{\partial L}{\partial Z_2}$. Consider a more general setting – Let $g : U \subset \mathbb{R} \to \mathbb{R}$ be a differentiable function, U an open set. We use the same notation to denote another related function

$$\begin{aligned} g : \quad & M_{u,v}(U) \to M_{u,v}(\mathbb{R}) \\ & Z \mapsto A = g(Z) \ , where \\ \forall \, 1 \leq i \leq u, 1 \leq j \leq v, \ & A_{i,j} = g(Z_{i,j}) \end{aligned}$$

We want to find the differential of this last function defined on the set of matrices.

Rmk. In many places of this article, we will only give a sketch of the proofs. For detailed, down-to-earth proofs, either we will edit them in an appendix if we find time, or they will be left to the readers.

Since the g defined on U, we have

$$g(z + \Delta z) = g(z) + g'(z)\Delta z + o(|\Delta z|) \, .$$

In the matrix case, we have

$$\begin{aligned} g(Z + \Delta Z) &= \left[g(Z_{i,j} + \Delta Z_{i,j}) \right] \\ &= \left[g(Z_{i,j}) + g'(Z_{i,j}) \Delta Z_{i,j} + o\left(|\Delta Z_{i,j}| \right) \right] \\ &= g(Z) + g'(Z) \odot \Delta Z + \left[o\left(|\Delta Z_{i,j}| \right) \right] \ . \end{aligned}$$

Some explanations might be needed here.

- By the square brackets like $\left[g(Z_{i,j} + \Delta Z_{i,j}) \right]$, we mean a matrix whose (i, j) entry equals the one inside the brackets and whose shape can be sensibly deduced from the context.
- The binary operator $\odot$ is called *Hadamard product* (aka *element-wise product*). Its left and right operands should be matrices of the same shape, and its output is also of that same shape. What it does, like the name *element-wise product* suggests, each entry of the output matrix is the product of the two corresponding entries of its left and right operands. Hadamard product shows up quite often in deep learning; for users of Python's `numpy` package, $A \odot B$ is nothing but `A*B`, where `A`, `B` are `ndarray`s.
- For all matrix A fixed, the transformation $\Delta Z \mapsto A \odot \Delta Z$ is easily verified as a linear transformation. Besides, it is also easily seen that $\odot$ is commutative, i.e. $A \odot B = B \odot A$ for all A, B.

So far, most readers have probably already seen where we are going. We are going to say that

$$g'(Z) = \frac{\partial A}{\partial Z} : \quad \begin{aligned} M_{u,v}(U) &\to M_{u,v}(\mathbb{R}) \\ \Delta Z &\mapsto g'(Z) \odot \Delta Z \end{aligned}$$

We have used the same notation $g'(Z)$ for two very different things, but please bear with that – The first $g'(Z)$, written as "equals to $\frac{\partial A}{\partial Z}$" means the differential, while the second one in $g'(Z) \odot \Delta Z$ means element-wise application of the function $g' : \mathbb{R} \to \mathbb{R}$ to the matrix Z.

To prove the differential $\frac{\partial A}{\partial Z}$ really is as we described above, it suffices to show that

$$\left[o\left(|\Delta Z_{i,j}|\right)\right] = o\left(\|\Delta Z\|_1\right) .$$

We choose to omit the detailed proof here. However, a proof of this shouldn't be too difficult, much like what a usual sophomore math major would often do in their analysis classes.

Ok, one last differential to go: $\frac{\partial L}{\partial Z_2}$. First of all, let's introduce some more convenience notations: Since we are really dealing with $\Omega(Z_2)$ and $\Omega = J \circ f$, we denote

$$\begin{aligned} A_2 &= f(Z_2) \\ L &= -\frac{1}{m}\left(y^T \log(A_2) + (1-y)^T \log(1-A_2)\right) . \end{aligned}$$

We will once again use the formula of differentiation of composite functions

$$\frac{\partial L}{\partial Z_2} = \frac{\partial L}{\partial A_2} \circ \frac{\partial A_2}{\partial Z_2} .$$

From our previous discussion, we know how to compute $\frac{\partial A_2}{\partial Z_2}$.

$$\frac{\partial A_2}{\partial Z_2} : \quad \Delta Z \mapsto f'(Z) \odot \Delta Z .$$

Since f is the sigmoid function $f(z) = \frac{e^z}{e^z+1}$ for all $z \in \mathbb{R}$, we have

$$f'(z) = \frac{-e^z}{(e^z+1)^2} = -\frac{e^z}{e^z+1}\left(1 - \frac{e^z}{e^z+1}\right) = f(z)\left(f(z)-1\right) .$$

This implies

$$\begin{aligned} f'(Z_2) &= f(Z_2) \odot \left(f(Z_2) - 1\right) \\ &= A_2 \odot \left(A_2 - 1\right) , \end{aligned}$$

where $A_2 - 1$ means A_2 minus a matrix of equal shape whose entries are all 1's. Thus we have

$$\frac{\partial A_2}{\partial Z_2} : \quad \Delta Z \mapsto A_2 \odot \left(A_2 - 1\right) \odot \Delta Z .$$

There is no ambiguity of associativity because Hadamard product is associative.

Move on to $\frac{\partial L}{\partial A_2}$. Recall again that

$$L = J(A_2) = -\frac{1}{m}\left(y^T \log(A_2) + (1-y)^T \log(1-A_2)\right) .$$

We see that J is the sum of two functions J_1 and J_2, where

$$J_1(A_2) = -\frac{1}{m}\, y^T \log(A_2)$$
$$J_2(A_2) = -\frac{1}{m}\, (1-y)^T \log(1-A_2)\,.$$

We state without proof the result that *the differential of a sum equals the sum of the differentials*. So we have

$$\frac{\partial L}{\partial A_2}: \quad \Delta A \mapsto -\frac{1}{m}\, y^T \left(\log'(A_2) \odot \Delta A\right) - \frac{1}{m}\, (1-y)^T \left(\log'(1-A_2) \odot (-\Delta A)\right)$$

As suggested by Professor Ng, considering $\frac{\partial L}{\partial A_2}$ and $\frac{\partial A_2}{\partial Z_2}$ separately is no better than considering them as a whole $\frac{\partial L}{\partial A_2}$. Indeed, on the one hand, we have $\log'(x) = \frac{1}{x}$ for all $x > 0$, whence

$$\log'(A_2) = \frac{1}{A_2}$$
$$\log'(1-A_2) = \frac{1}{1-A_2}\,,$$

where by $\frac{1}{A_2}$ we mean the matrix $\left[\frac{1}{A_{2\,i,j}}\right]$. On the other hand, composing $\frac{\partial L}{\partial A_2} \circ \frac{\partial A_2}{\partial Z_2}$, we obtain

$$\frac{\partial L}{\partial Z_2}: \quad \Delta Z \quad \mapsto$$
$$-\frac{1}{m}\, y^T \left(\frac{1}{A_2} \odot (A_2 \odot (A_2 - 1) \odot \Delta Z)\right) + \frac{1}{m}\, (1-y)^T \left(\frac{1}{1-A_2} \odot (A_2 \odot (A_2 - 1) \odot \Delta Z)\right)$$
$$= -\frac{1}{m}\, y^T \left(\,(A_2 - 1) \odot \Delta Z\,\right) - \frac{1}{m}\, (1-y)^T \,(A_2 \odot \Delta Z)$$
$$= -\frac{1}{m}\, y^T \,(A_2 \odot \Delta Z - \Delta Z) - \frac{1}{m}\, {A_2}^T \Delta Z + \frac{1}{m}\, y^T \,(A_2 \odot \Delta Z)$$
$$= \frac{1}{m}\, y^T \Delta Z - \frac{1}{m}\, {A_2}^T \Delta Z$$
$$= \frac{1}{m}\, (y - A_2)^T \Delta Z$$

Having found all the intermediate differentials, we can now write

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Z_2} \circ \frac{\partial Z_2}{\partial A_1} \circ \frac{\partial A_1}{\partial Z_1} \circ \frac{\partial Z_1}{\partial W_1}: \quad \Delta W \quad \mapsto \quad \frac{1}{m}\, (y - A_2)^T \left(\left(g_1'\,(Z_1) \odot (X \Delta W)\right) W_2\right)$$

4 Gradient Descent

When we talk about the differential of a real-valued function $s : U \subset \mathbb{R}^p \to \mathbb{R}$, $s'(a)$ will be a linear function from $\mathbb{R}^p$ to $\mathbb{R}$, pretty much like s itself. When this linear function is written as a matrix, its shape is $(1, p)$, which is different from the shape of a column vector, say a of shape $(p, 1)$. This is because we want to use the notation to do linear transformation, or matrix multiplication.

On the other hand, when we talk about gradient descent, we often think of $\nabla s(a)$ as having the same shape as a, and we often speak of descending in the direction of $-\nabla s(a)$ in the space of $\mathbb{R}^p$.

The above derivation of the formula for the differential $\frac{\partial L}{\partial W_1}$ is rigorous and indeed a linear transformation. However, it does not serve very well the pratical purpose of gradient descent. We would like to have something of the same shape as W_1, i.e. of shape (n, n_1), instead of a bunch of matrix operations, sometimes multiplying from the left, sometimes from the right.

And indeed, if we switch back to the perspective of n-dimensional vector space instead of $M_{u,v}$,

$$\frac{\partial L}{\partial W_1} : \quad M_{n,n_1} \cong \mathbb{R}^{nn_1} \to \mathbb{R} \,,$$

the differential should be an **inner product**.

Definition. *For arbitrary matrices A, B of the same shape, say (m, n), we define (quite naturally) their **inner product** as $A \cdot B = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{i,j} B_{i,j}$*

Instead of

$$\frac{\partial L}{\partial W_1} : \quad \Delta W \quad \mapsto \quad \frac{1}{m} (y - A_2)^T \Big(\Big(g_1' (Z_1) \odot (X \Delta W) \Big) W_2 \Big) \,,$$

We seek

$$\frac{\partial L}{\partial W_1} : \quad \Delta W \quad \mapsto \quad \boxed{?} \cdot \Delta W \,.$$

I have to admit that this is one of the major steps that help me continue to advance in this line of thought; before thinking in this way, I have been trapped in several sheets of paper of calculations for quite a while. Subsequent

computations just follows naturally:

$$\begin{aligned}
&\frac{1}{m}\,(y-A_2)^T\Big(\Big(g_1'\,(Z_1)\odot(X\Delta W)\Big)W_2\Big)\\
&=\frac{1}{m}\,(y-A_2)\bullet\Big(\Big(g_1'\,(Z_1)\odot(X\Delta W)\Big)W_2\Big)\\
&=\frac{1}{m}\,\Big((y-A_2){W_2}^T\Big)\bullet\Big(g_1'\,(Z_1)\odot(X\Delta W)\Big)\\
&=\frac{1}{m}\,\left(X^T\Big(g_1'\,(Z_1)\odot\Big((y-A_2){W_2}^T\Big)\Big)\right)\bullet\Delta W
\end{aligned}$$

Therefore, the gradient equals

$$\boxed{?}\;=\;\frac{1}{m}\,\left(X^T\Big(g_1'\,(Z_1)\odot\Big((y-A_2){W_2}^T\Big)\Big)\right)$$

The readers can verify that the shapes match perfectly. During the above computation, we did not explain much; much of the explanation will be done in the next section, except a simple one: It is easy to show that, for all matrices A, B, C of the same shape, we have

$$A\bullet(B\odot C)=(B\odot A)\bullet C\,.$$

5 Some Properties of Inner Product for Matrices

We need to prove the properties about matrix inner product we claimed before.

Property. *Let A, B, C be matrices of shape $(m,n),(m,q),(q,n)$, respectively. Then we have*

$$\begin{aligned}
A\bullet(BC)&=(AC^T)\bullet B\\
A\bullet(BC)&=(B^TA)\bullet C
\end{aligned}$$

Proof. A quick check for the correctness of shapes of both sides of the equations shows that at least there are perfect matches of the shapes. A good sign for the validity.

$$\begin{aligned}
A \cdot (BC) &= \sum_{i=1}^{m}\sum_{j=1}^{n} A_{i,j}(BC)_{i,j} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} A_{i,j}(\sum_{k=1}^{q} B_{i,k}C_{k,j}) \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{q} A_{i,j}B_{i,k}C_{k,j} \\
&= \sum_{i=1}^{m}\sum_{k=1}^{q} B_{i,k}\sum_{j=1}^{n}(A_{i,j}C^T_{j,k}) \\
&= (AC^T) \cdot B
\end{aligned}$$

$A \cdot (BC) = (B^T A) \cdot C$ can also be shown in the same fashion. □

Rmk. It's easy to convince oneself that $A \cdot B = B \cdot A$. Combined with the properties above, we have

$$\begin{aligned}
(BC) \cdot A = A \cdot (BC) = (AC^T) \cdot B = B \cdot (AC^T) \\
(BC) \cdot A = A \cdot (BC) = (B^T A) \cdot C = C \cdot (B^T A)\,.
\end{aligned}$$

That is,

$$\begin{aligned}
(BC) \cdot A = B \cdot (AC^T) \\
(BC) \cdot A = C \cdot (B^T A)\,.
\end{aligned}$$

In words, these laws can be summarized into something like *"every time we try to move a matrix from inside a pair of parentheses to the opposite side of the inner product symbol ($\cdot$), we put that matrix in the same relative position but transposed."*