Computing Neural Network Gradients

1. Jacobian Matrix

1.1 Definition of Jacobian Matrix

Suppose we have a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ that takes input of a vector of length n and outputs a vector of length m :

$$\boldsymbol{f(x)} = [f_1(x_1; \dots; x_)); f_2(x_1; \dots; x_n); \dots; f_m(x_1; \dots; x_n)]$$

Then its Jacobian is the following $m \times n$ matrix:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

This can be represented as

$$(\frac{\partial f}{\partial x})_{ij} = \frac{\partial f_i}{\partial x_j}$$

1.2 Something we need to know about Jacobian Matrix

Suppose we have a function $\boldsymbol{f(x)} = [f_1(x), f_2(x)]$ that takes input of a scalar x and outputs a vector of size 2. We also have a function $\boldsymbol{g(x)} = [g_1(y_1, y_2), g_2(y_1, y_2)]$ that takes input of a vector of size 2 and outputs another vector of size 2.

We can compose $\boldsymbol{f(x)}$ and $\boldsymbol{g(x)}$ and get $\boldsymbol{g(x)} = [g_1(f_1(x), f_2(x)), g_2(f_1(x), f_2(x))]$.

Then using the regular chain rule, we can compute the derivative of $\boldsymbol{g(x)}$ as the Jacobian

$$\frac{\partial g}{\partial x} = \begin{bmatrix} \frac{\partial}{\partial x} g_1\left(f_1(x), f_2(x)\right) \\ \frac{\partial}{\partial x} g_2\left(f_1(x), f_2(x)\right) \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial f_1}\frac{\partial f_1}{\partial x} + \frac{\partial g_1}{\partial f_2}\frac{\partial f_2}{\partial x} \\ \frac{\partial g_2}{\partial f_1}\frac{\partial f_1}{\partial x} + \frac{\partial g_2}{\partial f_2}\frac{\partial f_2}{\partial x} \end{bmatrix}$$

The above is equivalent to

$$\frac{\partial \boldsymbol{g}}{\partial x} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{f}}\frac{\partial \boldsymbol{f}}{\partial x} = \begin{bmatrix} \frac{\partial g_1}{\partial f_1} & \frac{\partial g_1}{\partial f_2} \\ \frac{\partial g_2}{\partial f_1} & \frac{\partial g_2}{\partial f_2} \end{bmatrix} \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \frac{\partial f_2}{\partial x} \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial f_1}\frac{\partial f_1}{\partial x} + \frac{\partial g_1}{\partial f_2}\frac{\partial f_2}{\partial x} \\ \frac{\partial g_2}{\partial f_1}\frac{\partial f_1}{\partial x} + \frac{\partial g_2}{\partial f_2}\frac{\partial f_2}{\partial x} \end{bmatrix}$$

2. Some useful identities

2.1 Matrix times column vector with respect to the column vector

We have matrix W times column vector x represented as $z = Wx$. We want to compute the Jacobian of z with respect to the column vector x, that is, $\frac{\partial z}{\partial x}$.

$$W = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1m} \\ W_{21} & W_{22} & \cdots & W_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ W_{n1} & W_{n2} & \cdots & W_{nm} \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

Suppose matrix $W \in \mathbb{R}^{n \times m}$. Then we can think of z as a function of x that takes input of an m-dimensional vector and outputs an n-dimensional vector. So its Jacobian will be $n \times m$.

$$z_i = \sum_{k=1}^{m} W_{ik} x_k$$

When $i = 1,\ z_1 = W_{11}x_1 + W_{12}x_2 + \cdots + W_{1m}x_m$.

So an entry $(\frac{\partial z}{\partial x})_{ij}$ of the Jacobian will be

$(\frac{\partial z}{\partial x})_{ij} = \frac{\partial z_i}{\partial x_j} = \frac{\partial}{\partial x_j} \sum_{k=1}^{m} W_{ik} x_k = \sum_{k=1}^{m} W_{ik} (\frac{\partial}{\partial x_j} x_k) = W_{ij}$

where

$$\frac{\partial}{\partial x_j} x_k = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{if otherwise} \end{cases}$$

In short, we have

$$\frac{\partial z}{\partial x} = W$$

2.2 Row vector times matrix with respect to the row vector

We have row vector x times matrix W represented as $z = xW$. We want to compute the Jacobian of z with respect to the row vector x, that is, $\frac{\partial z}{\partial x}$.

$$x = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}$$

$$W = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W_{m1} & W_{m2} & \cdots & W_{mn} \end{bmatrix}$$

$$z = \begin{bmatrix} z_1 & z_2 & \cdots & z_n \end{bmatrix}$$

Similar to section 2.1, we have

$$\frac{\partial z}{\partial x} = W^T$$

2.3 A vector with itself

We have a vector $z = x$ and want to compute the Jacobian of z with respect to vector x, that is, $\frac{\partial z}{\partial x}$.

$$\left(\frac{\partial z}{\partial \boldsymbol{x}}\right)_{ij} = \frac{\partial z_i}{\partial x_j} = \frac{\partial}{\partial x_j} x_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases}$$

Essentially, the Jacobian $\frac{\partial z}{\partial x}$ is a diagonal matrix where the entry at (i, i) is 1. This is just the identity matrix $\frac{\partial z}{\partial x} = \boldsymbol{I}$.

When applying the chain rule, this term will disappear because a matrix or vector multiplied by the identity matrix does not change.

2.4 An elementwise function applied a vector

We have an elementwise function $z = f(x)$ and want to compute the Jacobian of z with respect to vector x, that is, $\frac{\partial z}{\partial x}$.

Since f is being applied elementwise, we have $z_i = f(x_i)$. So

$$\left(\frac{\partial z}{\partial \boldsymbol{x}}\right)_{ij} = \frac{\partial z_i}{\partial x_j} = \frac{\partial}{\partial x_j} f(x_i) = \begin{cases} f'(x_i) & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases}$$

Essentially, the Jacobian $\frac{\partial z}{\partial x}$ is a diagonal matrix where the entry at (i, i) is the derivative of f applied to x_i. This can be written as $\frac{\partial z}{\partial x} = \text{diag}(f'(x))$.

Since multiplication by a diagonal matrix is the same as doing elementwise multiplication by the diagonal, we could also write $\circ f'(x)$ when applying the chain rule.

2.5 Matrix times column vector with respect to the matrix

We have matrix W times column vector x represented as $z = Wx$. We also have a loss function J that has derivative $\frac{\partial J}{\partial z} = \delta$. We want to compute the Jacobian of loss function J with respect to the matrix W, that is, $\frac{\partial J}{\partial W} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial W} = \delta \frac{\partial z}{\partial W}$.

First of all, the matrix W satisfies $W \in \mathbb{R}^{n \times m}$. We can then think of loss function J as a function that takes $n \times m$ inputs (the elements in matrix W), and outputs a single scalar J. This means the Jacobian $\frac{\partial J}{\partial W}$ would be a $1 \times nm$ vector. However, in practice we want to rearrange its dimensions so that it has the same shape as W:

$$\frac{\partial J}{\partial W} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}} & \cdots & \frac{\partial J}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial W_{n1}} & \cdots & \frac{\partial J}{\partial W_{nm}} \end{bmatrix}$$

Therefore, in gradient descent (updating values of weight matrix W), we can just subtract it (times the learning rate) from W.

Let's see how we can resolve it below.

Firstly, from section 2.1 we have

$$z_k = \sum_{l=1}^{m} W_{kl} x_l$$

where z_k is the k^{th} element of vector z.

Then we can compute the derivative of one element of column vector z, that is z_k, with respective to any element of weight matrix W, that is W_{ij}:

$$\frac{\partial z_k}{\partial W_{ij}} = \sum_{l=1}^{m} x_l \frac{\partial}{\partial W_{ij}} W_{kl}$$

where

$$\frac{\partial}{\partial W_{ij}} W_{kl} = \begin{cases} 1 & \text{if } k = i \text{ and } l = j \\ 0 & \text{if otherwise} \end{cases}$$

and

- i is the row index of weight matrix $W \in \mathbb{R}^{n \times m}$ and is from 1 to n
- j is the column index of weight matrix $W \in \mathbb{R}^{n \times m}$ and is from 1 to m
- k is the index of elements in column vector $z \in \mathbb{R}^n$ and is from 1 to n
- l is the index of elements in column vector $x \in \mathbb{R}^m$ and is from 1 to m

Because we are computing the derivative of one particular element in column vector z with respective to any element in the weight matrix W. Therefore, in the result matrix, many elements will be zero. For instance, z_1 only relates to the first row of W which is $\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1n} \end{bmatrix}$ (see section 2.1).

Therefore, z's index k must match W's row index i. if $k \neq i$ the result of $\frac{\partial z_k}{\partial W_{ij}}$ is zero.

Then we can get

$$\frac{\partial z_k}{\partial W_{ij}} = \sum_{l=1}^{m} x_l \frac{\partial}{\partial W_{ij}} W_{kl} = \sum_{l=1}^{m} x_l \frac{\partial}{\partial W_{kj}} W_{kl}$$

Similarly, refer to section 2.1, x' index l must match W's column index j. In the above equation, specifically for x_l, we are summing from $l = 1$ to $l = m$. Since only the element $x_j \neq 0$, we can remove the sum $\sum_{l=1}^{m}$ and make $\sum_{l=1}^{m} x_l = x_j$. Furthermore, in the above equation, $\frac{\partial}{\partial W_{kj}} W_{kl}$ also has index l in W_{kl}. We can simply replace l with j to get $\frac{\partial}{\partial W_{kj}} W_{kj} = 1$

So

$$\frac{\partial z_k}{\partial W_{ij}} = \sum_{l=1}^{m} x_l \frac{\partial}{\partial W_{ij}} W_{kl} = \sum_{l=1}^{m} x_l \frac{\partial}{\partial W_{kj}} W_{kl} = x_j$$

Another way of writing this is

$$\frac{\partial z}{\partial W_{ij}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ x_j \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where x_j is in i^{th} element of $\frac{\partial z}{\partial W_{ij}}$.

Now let's compute $\frac{\partial J}{\partial W_{ij}}$:

$$\frac{\partial J}{\partial W_{ij}} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial W_{ij}} = \delta\frac{\partial z}{\partial W_{ij}} = \sum_{k=1}^{m}\delta_k\frac{\partial z_k}{\partial W_{ij}} = \delta_i x_j$$

Note that the only nonzero term in the sum is $\delta_i\frac{\partial z_i}{\partial W_{ij}}$.

In practice, to get $\frac{\partial J}{\partial W}$ we want a matrix where entry (i, j) is $\delta_i x_j$. This matrix is equal to the outer product

$$\frac{\partial J}{\partial W} = \delta^T x^T$$

2.6 Row vector time matrix with respect to the matrix

We have a row vector x times matrix W represented as $z = xW$. We also have a loss function J that has derivative $\frac{\partial J}{\partial z} = \delta$. We want to compute the Jacobian of loss function J with respect to the matrix W, that is, $\frac{\partial J}{\partial W} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial W} = \delta\frac{\partial z}{\partial W}$.

Similar to section 2.5, we can get $\frac{\partial J}{\partial W} = x^T\delta$

2.7 Cross-entropy loss with respect to logits

We have model prediction $\hat{y} = \mathrm{softmax}(x)$ and cross-entropy loss $J = CE(y, \hat{y})$. We want to compute $\frac{\partial J}{\partial \theta}$.

$$\begin{aligned} J &= -\sum_i y_i log(\hat{y}_i) \\ \frac{\partial J}{\partial \theta} &= -\sum_k y_k \frac{\partial log(\hat{y}_k)}{\partial \theta_i} \\ &= -\sum_k y_k \frac{\partial log(\hat{y}_k)}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\theta_i} \\ &= -\sum_k y_k \frac{1}{\hat{y}_k}\frac{\partial \hat{y}_k}{\theta_i} \end{aligned}$$

where $\frac{\partial \hat{y}_k}{\theta_i}$ is the derivative of a softmax regression function.

It can be proven that the derivative of a softmax regression $\hat{y} = \mathrm{softmax}(x)$ satisfies

$$\frac{\partial \hat{y}_k}{\partial \theta_i} = \begin{cases} \hat{y}_k(1-\hat{y}_i) & \text{if } k = i \\ -\hat{y}_i \cdot \hat{y}_k & \text{if otherwise} \end{cases}$$

Therefore, we can rewrite $\frac{\partial J}{\partial \theta}$ as

$$\begin{aligned} \frac{\partial J}{\partial \theta} &= -\sum_k y_k \frac{1}{\hat{y}_k}\frac{\partial \hat{y}_k}{\theta_i} \\ &= -y_i(1-\hat{y}_i) - \sum_{k \neq i} y_k \frac{1}{\hat{y}_k}(-\hat{y}_k\hat{y}_i) \\ &= -y_i(1-\hat{y}_i) - \sum_{k \neq i} y_k\hat{y}_i \\ &= -y_i + y_i\hat{y}_i + \sum_{k \neq i} y_k\hat{y}_i \\ &= \hat{y}_i(y_i + \sum_{k \neq i} y_k) - y_i \end{aligned}$$

y is a one hot encoded vector for the labels, so $\sum_k y_k = 1$ and $y_i + \sum_{k \neq i} y_k = 1$. So we have

$$\frac{\partial J}{\partial \theta} = \hat{y}_i - y_i$$
$$\text{or}$$
$$\frac{\partial J}{\partial \theta} = (\hat{y}_i - y_i)^T \quad \text{if y is a column vector}$$

Additional reading: Derivative of softmax function

Softmax function takes an N-dimensional vector of real numbers and transforms it into a vector of real number in range $(0, 1)$ which add up to 1

$$y_i = \frac{e^{\theta_i}}{\sum\limits_{k=1}^{N} e^{\theta_k}}$$

As the name suggests, softmax function is a “soft” version of max function. Instead of selecting one maximum value, it breaks the whole (1) with maximal element getting the largest portion of the distribution, but other smaller elements getting some of it as well.

This property of softmax function that it outputs a probability distribution makes it suitable for probabilistic interpretation in classification tasks.

The derivative of the softmax function is

$$\frac{\partial y_i}{\partial \theta_j} = \frac{\partial \frac{e^{\theta_i}}{\sum\limits_{k=1}^{N} e^{\theta_k}}}{\partial \theta_j}$$

From quotient rule we know that for $f(x) = \frac{g(x)}{h(x)}$, we have $f'(x) = \frac{g'(x)h(x) - h'(x)g(x)}{h(x)^2}$.

In our case

- $g(x) = e^{\theta_i}$
- $h(x) = \sum\limits_{k=1}^{N} e^{\theta_k}$
- $\frac{\partial h(x)}{\partial e^{\theta_j}} = \frac{\partial \sum\limits_{k=1}^{N} e^{\theta_k}}{\partial e^{\theta_j}}$ will always be e^{θ_j} because $\frac{\partial e^{\theta_j}}{\partial e^{\theta_j}} = e^{\theta_j}$ and $\frac{\partial \sum\limits_{k \neq j}^{N} e^{\theta_k}}{\partial e^{\theta_j}} = 0$
- $\frac{\partial g(x)}{\partial e^{\theta_j}} = \frac{\partial e^{\theta_i}}{\partial e^{\theta_j}} = \begin{cases} e^{\theta_j} & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases}$

Therefore, if $i = j$

$$\begin{aligned}
\frac{\partial y_i}{\partial \theta_j} = \frac{\partial \frac{e^{\theta_i}}{\sum\limits_{k=1}^{N} e^{\theta_k}}}{\partial \theta_j} &= \frac{e^{\theta_i} \sum\limits_{k=1}^{N} e^{\theta_k} - e^{\theta_j} e^{\theta_i}}{(\sum\limits_{k=1}^{N} e^{\theta_k})^2} \\
&= \frac{e^{\theta_i} (\sum\limits_{k=1}^{N} e^{\theta_k} - e^{\theta_j})}{(\sum\limits_{k=1}^{N} e^{\theta_k})^2} \\
&= \frac{e^{\theta_i}}{\sum\limits_{k=1}^{N} e^{\theta_k}} \frac{(\sum\limits_{k=1}^{N} e^{\theta_k} - e^{\theta_j})}{\sum\limits_{k=1}^{N} e^{\theta_k}} \\
&= y_i (1 - y_j)
\end{aligned}$$

If $i \neq j$

$$\begin{aligned}\frac{\partial y_i}{\partial \theta_j} = \frac{\partial \frac{e^{\theta_i}}{\sum\limits_{k=1}^{N} e^{\theta_k}}}{\partial \theta_j} &= \frac{0 - e^{\theta_j} e^{\theta_i}}{(\sum\limits_{k=1}^{N} e^{\theta_k})^2} \\ &= \frac{-e^{\theta_j}}{\sum\limits_{k=1}^{N} e^{\theta_k}} \frac{e^{\theta_i}}{\sum\limits_{k=1}^{N} e^{\theta_k}} \\ &= -y_j y_i\end{aligned}$$

In sum

$$\frac{\partial y_i}{\partial \theta_j} = \begin{cases} y_i(1 - y_j) & \text{if } i = j \\ -y_j y_i & \text{if otherwise} \end{cases}$$

3. Example: 1-Layer Neural Network

This section provides an example of computing the gradients of a full neural network. In particular we are going to compute the gradients of a one-layer neural network trained with cross-entropy loss. The forward pass of the model is as follows:

$$\begin{aligned} x &= \text{input} \\ z &= Wx + b_1 \\ h &= ReLU(z) \\ \theta &= Uh + b_2 \\ \hat{y} &= \text{softmax}(\theta) \\ J &= CE(y, \hat{y}) \end{aligned}$$

The dimensions of the model's parameters are

$$\begin{aligned} x &\in \mathbb{R}^{D_x \times 1} \\ b_1 &\in \mathbb{R}^{D_h \times 1} \\ W &\in \mathbb{R}^{D_h \times D_x} \\ b_2 &\in \mathbb{R}^{N_c \times 1} \\ U &\in \mathbb{R}^{N_c \times D_h} \end{aligned}$$

where

- D_x is the size of our input
- D_h is the size of our hidden layer
- N_c is the number of classes

In this example, we will compute all of the network's gradients:

$$\frac{\partial J}{\partial U} \quad \frac{\partial J}{\partial b_2} \quad \frac{\partial J}{\partial W} \quad \frac{\partial J}{\partial b_1} \quad \frac{\partial J}{\partial x}$$

To start with, recall that $\text{ReLU}(x) = max(x, 0)$. This means

$$\text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if otherwise} \end{cases} = \text{sng}(\text{ReLU}(x))$$

where sng is the signum function. Note that we are able to write the derivative of the activation ReLU in terms of the activation itself.

Now let's write out the chain rule for $\frac{\partial J}{\partial U}$ and $\frac{\partial J}{\partial b_2}$:

$$\frac{\partial J}{\partial U} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial \theta}\frac{\partial \theta}{\partial U}$$

$$\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial \theta}\frac{\partial \theta}{\partial b_2}$$

Notice that $\frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial \theta} = \frac{\partial J}{\partial \theta}$ is present in both gradients. This makes the math a bit cumbersome. Even worse, if we’re implementing the model without automatic differentiation, computing $\frac{\partial J}{\partial \theta}$ twice will be inefficient. So it will help use to define some variables to represent the intermediate derivatives:

$$\delta_1 = \frac{\partial J}{\partial \theta} \;\; \delta_2 = \frac{\partial J}{\partial z}$$

These can be thought as the error signals passed down to θ and z when doing backpropagation. We can compute them as follows and use them to compute other derivatives later:

$$\begin{aligned}
\delta_1 &= \tfrac{\partial J}{\partial \theta} = (\hat{y} - y)^T && \text{this is just identity (2.7)} \\
\delta_2 &= \tfrac{\partial J}{\partial z} = \tfrac{\partial J}{\partial \theta}\tfrac{\partial \theta}{\partial h}\tfrac{\partial h}{\partial z} && \text{using the chain rule} \\
&= \delta_1 \tfrac{\partial \theta}{\partial h}\tfrac{\partial h}{\partial z} && \text{substituting in } \delta_1 \\
&= \delta_1 U \tfrac{\partial h}{\partial z} && \text{using identity (2.1)} \\
&= \delta_1 U \circ \mathrm{ReLU}'(z) && \text{using identity (2.4)} \\
&= \delta_1 U \circ \mathrm{sng}(h) && \text{we compute this earlier}
\end{aligned}$$

A good way for checking our work above is by looking at the dimensions of the Jacobians:

$$\begin{array}{llllll}
\frac{\partial J}{\partial z} & = & \delta_1 & U & \circ & \mathrm{sng}(h) \\
(1 \times D_h) & & (1 \times N_c) & (N_c \times D_h) & & D_h
\end{array}$$

Now we can use the error terms to compute our gradients. Note that we transpose out answers when computing the gradients for column vectors terms to follow the shape convention.

$$\begin{aligned}
\tfrac{\partial J}{\partial U} &= \tfrac{\partial J}{\partial \theta}\tfrac{\partial \theta}{\partial U} = \delta_1 \tfrac{\partial \theta}{\partial U} = \delta_1^T h^T && \text{using identity (2.5)} \\
\tfrac{\partial J}{\partial b_2} &= \tfrac{\partial J}{\partial \theta}\tfrac{\partial \theta}{\partial b_2} = \delta_1 \tfrac{\partial \theta}{\partial b_2} = \delta_1^T && \text{using identity (2.3) and transposing} \\
\tfrac{\partial J}{\partial W} &= \tfrac{\partial J}{\partial z}\tfrac{\partial z}{\partial W} = \delta_2 \tfrac{\partial z}{\partial W} = \delta_2^T x^T && \text{using identity (2.5)} \\
\tfrac{\partial J}{\partial W} &= \tfrac{\partial J}{\partial z}\tfrac{\partial z}{\partial b_1} = \delta_2 \tfrac{\partial z}{\partial b_1} = \delta_2^T && \text{using identity (2.3) and transposing} \\
\tfrac{\partial J}{\partial x} &= \tfrac{\partial J}{\partial z}\tfrac{\partial z}{\partial x} = (\delta_2 W)^T && \text{using identity (2.1) and transposing}
\end{aligned}$$