Gradients for FF NN

March 2025

1 Definitions

- B - Batch size
- D - Input Feature Size
- H - Hidden Size
- C - Number of Classes
- y_i - True class for a input image
- x_i - Input image 28x28

Now defining the dimensions of each matrix

- $W_1 \in D \times H$
- $b_1 \in 1 \times H$
- $W_2 \in H \times H$
- $b_2 \in 1 \times H$
- $W_3 \in H \times H$
- $b_3 \in 1 \times H$
- $W_4 \in H \times C$
- $b_4 \in 1 \times C$

We can now define the loss function for a multiclass classification task as cross entropy loss.

$$L(f(X,\theta),Y) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{c=1}^{C}\log\left(\frac{\exp(f(X)_{i,c})}{\sum_{j=1}^{C}\exp(f(X)_{i,j})}\right)y_{i,c}$$

2 Finding Gradients

To take the gradient with respect to the logits (the output vector $f(x)$) we need to first explore the derivative with respect to the softmax of a vector. Define a vector $z = \begin{bmatrix} z_1 & z_2 & \dots & z_C \end{bmatrix}$ to be vector of logits for a discrete probability distribution. We can then define the softmax as,

$$S_i(z) = \frac{\exp(z_i)}{\sum_{j=1}^{C}\exp(z_j)}$$

Which has a vector valued output. Taking the partial with respect to an arbitrary z_k,

$$\begin{aligned}
\frac{\partial S_i}{\partial z_k} &= \frac{\partial}{\partial z_k}\frac{\exp(z_i)}{\sum_{j=1}^{C}\exp(z_j)} \\
&= \frac{\partial}{\partial z_k}\frac{\exp(z_i)}{\exp(z_k)+\sum_{j\neq k}^{C}\exp(z_j)} \\
&= \frac{\partial}{\partial z_k}\exp(z_i)(\exp(z_k)+\sum_{j\neq k}^{C}\exp(z_j))^{-1}
\end{aligned}$$

Now there are two cases, starting with the first of $i \neq k$,

$$\begin{aligned}
\frac{\partial S_i}{\partial z_k} &= \frac{\partial}{\partial z_k} \exp(z_i) \left(\exp(z_k) + \sum_{j \neq k}^{C} \exp(z_j) \right)^{-1} \\
&= 0 \left(\exp(z_k) + \sum_{j \neq k}^{C} \exp(z_j) \right)^{-1} - \exp(z_i) \exp(z_k) \left(\exp(z_k) + \sum_{j \neq k}^{C} \exp(z_j) \right)^{-2} \\
&= -\exp(z_i) \exp(z_k) \left(\sum_{j=1}^{C} \exp(z_j) \right)^{-2} \\
&= -\frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} \frac{\exp(z_k)}{\sum_{j=1}^{C} \exp(z_j)} \\
&= -S_i(z) S_k(z)
\end{aligned}$$

Now for $i = k$,

$$\begin{aligned}
\frac{\partial S_i}{\partial z_i} &= \frac{\partial}{\partial z_i} \exp(z_i) \left(\exp(z_i) + \sum_{j \neq i}^{C} \exp(z_j) \right)^{-1} \\
&= \exp(z_i) \left(\exp(z_i) + \sum_{j \neq i}^{C} \exp(z_j) \right)^{-1} - \exp(z_i) \exp(z_i) \left(\exp(z_i) + \sum_{j \neq i}^{C} \exp(z_j) \right)^{-2} \\
&= \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} - \frac{\exp(z_i)^2}{\left(\sum_{j=1}^{C} \exp(z_j) \right)^2} \\
&= S_i - S_i^2 \\
&= S_i(1 - S_i)
\end{aligned}$$

We can now use the multivariate chain rule for our row vector gradient case $\frac{\partial l}{\partial z} = \frac{\partial l}{\partial S(z)} \frac{\partial S(z)}{\partial z}$. We can now find the components of this,

$$\begin{aligned}
\frac{\partial l}{\partial S(z)} &= \begin{bmatrix} \frac{\partial}{\partial S_1(z)} - \sum_{i=1}^{C} y_i \log(S_i(z)) & \dots & \frac{\partial}{\partial S_C(z)} - \sum_{i=1}^{C} y_i \log(S_i(z)) \end{bmatrix} \\
&= -\begin{bmatrix} \frac{\partial}{\partial S_1(z)} \left(y_1 \log(S_1(z)) + \sum_{i \neq 1}^{C} y_i \log(S_i(z)) \right) & \dots & \frac{\partial}{\partial S_C(z)} \left(y_C \log(S_C(z)) + \sum_{i \neq C}^{C} y_i \log(S_i(z)) \right) \end{bmatrix} \\
&= -\begin{bmatrix} \frac{y_1}{S_1(z)} & \dots & \frac{y_C}{S_C(z)} \end{bmatrix}
\end{aligned}$$

Now for the Jacobian of $S(z)$ we know from the previous parts that the diagonals are $S_i(z)(1 - S_i(z))$ and off diagonals are $-S_i(z)S_j(z)$

$$\frac{\partial S(z)}{\partial z} = \begin{bmatrix} S_1(z)(1 - S_1(z) & \dots & -S_1(z)S_C(z) \\ \vdots & \ddots & \vdots \\ -S_1(z)S_C(z) & \dots & S_C(z)(1 - S_C(z)) \end{bmatrix}$$

We can now put this together to find the $\frac{\partial l}{\partial z}$,

$$
\begin{aligned}
\frac{\partial l}{\partial z} &= \frac{\partial l}{\partial S(z)} \frac{\partial S(z)}{\partial z} \\
&= -\begin{bmatrix} \frac{y_1}{S_1(z)} & \cdots & \frac{y_C}{S_C(z)} \end{bmatrix} \begin{bmatrix} S_1(z)(1-S_1(z) & \ldots & -S_1(z)S_C(z) \\ \vdots & \ddots & \vdots \\ -S_1(z)S_C(z) & \ldots & S_C(z)(1-S_C(z)) \end{bmatrix} \\
&= \begin{bmatrix} -y_1(1-S_1(z)) + \sum_{i\neq 1}^{C} y_i S_1(z) & \ldots & -y_C(1-S_C(z)) + \sum_{i\neq C}^{C} y_i S_C(z) \end{bmatrix} \\
&= \begin{bmatrix} -y_1 + y_1 S_1(z) + \sum_{i\neq 1}^{C} y_i S_1(z) & \ldots & -y_C + y_C S_C(z) + \sum_{i\neq C}^{C} y_i S_C(z) \end{bmatrix} \\
&= \begin{bmatrix} -y_1 + S_1(z) \sum_{i=1}^{C} y_i & \ldots & -y_C + S_C(z) \sum_{i=1}^{C} y_i \end{bmatrix} \\
&= \begin{bmatrix} S_1(z) - y_1 & \ldots & S_C(z) - y_C \end{bmatrix} \\
&= S(z) - y
\end{aligned}
$$

We can now use this as a starting point to calculate our gradients with respect to the weights and biases. Using a 4 layer NN with ReLU activations the forward prediction for a batch of inputs $x \in 1 \times D$ is,

$$
f(x) = z = \sigma\left(\sigma\left(\sigma\left(XW_1 + b_1\right)W_2 + b_2\right)W_3 + b_3\right)W_4 + b_4
$$

We will define some useful variables,

$$
\begin{aligned}
z_1 &= XW_1 + b_1 \\
z_2 &= sigma\left(XW_1 + b_1\right)W_2 + b_2 \\
z_3 &= \sigma\left(\sigma\left(XW_1 + b_1\right)W_2 + b_2\right)W_3 + b_3 \\
\delta^4 &= \frac{\partial l}{\partial z} = \frac{\partial l}{\partial f(x)} = S(z) - y
\end{aligned}
$$

Finding the gradient with respect to b_4 is trivial because $\frac{\partial z}{\partial b_4} = I$, so this yields a gradient of

$$
\begin{aligned}
\nabla_{b_4} &= \frac{\partial l}{\partial f(x)} \frac{\partial f(x)}{\partial b_4} \\
&= \delta^4 I \\
&= \delta^4
\end{aligned}
$$

Finding the gradient with respect to the weights is a bit more difficult because it involves finding the Jacobian tensor or each element of f(x) has a gradient with respect to a single weight. this means the Jacobian $\frac{\partial f(x)}{\partial W_4}$ is a $C \times C \times H$ tensor. Examining one of these sub-Jacobians,

$$
\begin{aligned}
\nabla_{W_4} l &= \begin{bmatrix} \frac{\partial l}{\partial w_{1,1}} & \cdots & \frac{\partial l}{\partial w_{1,C}} \\ \vdots & \ddots & \vdots \\ \frac{\partial l}{\partial w_{H,1}} & \cdots & \frac{\partial l}{\partial w_{H,C}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial l}{\partial f(x)} \frac{\partial f(x)}{\partial w_{1,1}} & \cdots & \frac{\partial l}{\partial f(x)} \frac{\partial f(x)}{\partial w_{1,C}} \\ \vdots & \ddots & \vdots \\ \frac{\partial l}{\partial f(x)} \frac{\partial f(x)}{\partial w_{H,1}} & \cdots & \frac{\partial l}{\partial f(x)} \frac{\partial f(x)}{\partial w_{H,C}} \end{bmatrix}
\end{aligned}
$$

With a sub-jacobian,

$$
\begin{aligned}
\frac{\partial l}{f(x)}\frac{\partial f(x)}{\partial w_{1,1}} &= \delta^L \begin{bmatrix} \frac{\partial}{\partial w_{1,1}}\sigma(z_3)W^4_{:,1} \\ \vdots \\ \frac{\partial}{\partial w_{1,1}}\sigma(z_3)W^4_{:,C} \end{bmatrix} \\
&= \delta^4 \begin{bmatrix} \frac{\partial}{\partial w_{1,1}}\sum_{i=1}^{H}\sigma(z_i^3)w_{i,1} \\ \vdots \\ \frac{\partial}{\partial w_{1,1}}\sum_{i=1}^{H}\sigma(z_i^3)w_{i,C} \end{bmatrix} \\
&= \delta^4 \begin{bmatrix} \frac{\partial}{\partial w_{1,1}}w_{1,1}\sigma(z_1^3) + \sum_{i\neq 1}^{H}\sigma(z_i^3)w_{i,1} \\ \vdots \\ 0 \end{bmatrix} \\
&= \delta^4 \begin{bmatrix} \sigma(z_1^3) \\ \vdots \\ 0 \end{bmatrix} \\
&= \delta_1^4\sigma(z_1^3)
\end{aligned}
$$

Similarly for other weights in the gradient matrix,

$$
\begin{aligned}
\frac{\partial l}{f(x)}\frac{\partial f(x)}{\partial w_{1,2}} &= \delta_2^4\sigma(z_1^3) \\
\frac{\partial l}{f(x)}\frac{\partial f(x)}{\partial w_{2,1}} &= \delta_1^4\sigma(z_2^3)
\end{aligned}
$$

Giving a full matrix gradient of,

$$
\begin{aligned}
\nabla_{W_4} l &= \begin{bmatrix} \delta_1^4\sigma(z_1^3) & \delta_2^4\sigma(z_1^3) & \dots & \delta_C^4\sigma(z_1^3) \\ \delta_1^4\sigma(z_2^3) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \delta_1^4\sigma(z_H^3) & \dots & \dots & \delta_C^4\sigma(z_H^3) \end{bmatrix} \\
&= \sigma(z_3)^T\delta^L
\end{aligned}
$$

Which just so happens to be an outer product between the input to the final layer and δ^L (A moderately efficient result). We can now do the same for the parameters of the third layer,

$$
\begin{aligned}
\nabla_{b_3} l &= \frac{\partial l}{\partial f(x)}\frac{\partial f(x)}{\partial \sigma(z_3)}\frac{\partial \sigma(z_3)}{\partial b_3} \\
&= \delta^4\frac{\partial f(x)}{\partial \sigma(z_3)}\frac{\partial \sigma(z_3)}{\partial z_3}\frac{\partial z_3}{\partial b_3}
\end{aligned}
$$

Solving for some intermediate values,

$$
\begin{aligned}
\frac{\partial f(x)}{\partial \sigma(z_3)} &= \frac{\partial}{\partial \sigma(z_3)}\sigma(z_3)W_4 + b_4 \\
&= \frac{\partial}{\partial \sigma(z_3)}\sigma(z_3)W_4 + b_4 \\
&= W_4^T \\
\frac{\partial \sigma(z_3)}{\partial z_3} &= \frac{\partial}{\partial z_3}\sigma(z_3) \\
&= diag(\sigma'(z_3)) \\
\frac{\partial z_3}{\partial b_3} &= I
\end{aligned}
$$

So we get,

$$\begin{aligned}\nabla_{b_3} l &= \delta^4 \frac{\partial f(x)}{\partial \sigma(z_3)} \frac{\partial \sigma(z_3)}{\partial z_3} \frac{\partial z_3}{\partial b_3} \\ &= \delta^4 W_4^T diag(\sigma'(z_3)) I \\ &= (\delta^4 W_4^T) \odot \sigma'(z_3)\end{aligned}$$

Where a the diagonal matrix with $\sigma'(z_3)$ can be turned into a hadamar product (element-wise multiplication) for efficiency. We can now define a useful intermediate value δ^l

$$\delta^l = (\delta^{l+1} W_{l+1}^T) \odot \sigma'(z_l)$$

Now finding the gradient for the W_3,

$$\begin{aligned}\nabla_{W_3} l &= \delta^4 \frac{\partial f(x)}{\partial z_3} \frac{\partial z_3}{\partial W_3} \\ &= \delta^3 \frac{\partial z_3}{\partial W_3}\end{aligned}$$

We also see that $\frac{\partial z_3}{\partial W_3}$ has an eerily similar form to the previous gradient with respect to the weights. So the full gradient will be,

$$\nabla_{W_3} l = \sigma(z_2)^T \delta^3$$

This process can be repeated for an arbitrary number of layers but I will define each gradient generally then specifically for our NN. Please remember this is for row vector outputs specifically, column vector outputs have a slightly different form.

2.1 Gradients for Arbritrary FF NN

$$\begin{aligned}\delta^L &= \frac{\partial \mathcal{L}}{\partial f(x)} \frac{\partial f(x)}{\partial z_L} \\ \delta^l &= (\delta^{l+1} W_{l+1}^T) \odot \sigma'(z_l) \\ \nabla_{b_l} \mathcal{L} &= \delta^l \\ \nabla_{W_l} \mathcal{L} &= \sigma(z_{l-1})^T \delta^l\end{aligned}$$

2.2 Gradients for 4 Layer FF

For our specific 4-layer network,

$$
\begin{aligned}
\delta^4 &= S(z_4) - y \\
\nabla_{b_4}\mathcal{L} &= \delta^4 \\
\nabla_{W_4}\mathcal{L} &= \sigma(z_3)^T\delta^4 \\
\delta^3 &= ((S(z_4) - y)W_4^T) \odot \sigma'(z_3) \\
\nabla_{b_3}\mathcal{L} &= \delta^3 \\
\nabla_{W_3}\mathcal{L} &= \sigma(z_2)^T\delta^3 \\
\delta^2 &= (((S(z_4) - y)W_4^T) \odot \sigma'(z_3))W_3^T) \odot \sigma'(z_2) \\
\nabla_{b_2}\mathcal{L} &= \delta^2 \\
\nabla_{W_2}\mathcal{L} &= \sigma(z_1)^T\delta^2 \\
\delta^1 &= ((((S(z_4) - y)W_4^T) \odot \sigma'(z_3))W_3^T) \odot \sigma'(z_2))W_2^t) \odot \sigma'(z_1) \\
\nabla_{b_1}\mathcal{L} &= \delta^1 \\
\nabla_{W_1}\mathcal{L} &= x^T\delta^1
\end{aligned}
$$

3 Mini-Batches

The gradients above are great when evaluated at a single point, however this is a very noisy estimate of the gradient. We can see this by taking the expectation of the gradient and using a monte-carlo estimator

$$
\begin{aligned}
E[\nabla\mathcal{L}] &\approx \frac{1}{|B|}\sum_{(x_i,y_i)\in B}\nabla_\theta\mathcal{L}(f(x_i,\theta),y_i) \\
&= \lim_{|B|\to\infty}\frac{1}{|B|}\sum_{(x_i,y_i)\in B}\nabla_\theta\mathcal{L}(f(x_i,\theta),y_i)
\end{aligned}
$$

This means our SGD estimate of the gradient is simply the average gradient over the mini-batch (which is nice for us). However vectorizing this runs into the question, how do we implement the $\frac{1}{|B|}$ for the weights and the bias?

For a mini-batch the gradient of the bias will have a batch dimension or $\delta^l \in B \times K$ where K is the dimension of the bias. In this case, the batch dimension indicates the gradient for each sample in the mini-batch, so the estimated gradient will be,

$$
\nabla_{b_l}\mathcal{L} = \frac{1}{|B|}\mathbf{1}\delta^l \quad (1 \in 1 \times B)
$$

So we effectively perform a summation of each component of the gradient along the batch dimension and divide by B. For the gradient of the weights we can examine the formula,

$$
\begin{aligned}
\nabla_{W_l}\mathcal{L} &= \sigma(z_{l-1})^T\delta^l \qquad (z_{l-1}) \in B \times P, \delta^l \in B \times K) \\
&= \begin{bmatrix}\sigma(z_{l-1}^1)^T & \sigma(z_{l-1}^2)^T & \dots & \sigma(z_{l-1}^B)^T\end{bmatrix}\begin{bmatrix}\delta_1^l \\ \delta_2^l \\ \vdots \\ \delta_B^l\end{bmatrix} \\
&= \begin{bmatrix}\sum_{i=1}^B \sigma(z_{1,i}^{l-1})\delta_{i,1}^l & \dots & \sum_{i=1}^B \sigma(z_{1,i}^{l-1})\delta_{i,K}^l \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^B \sigma(z_{P,i}^{l-1})\delta_{i,1}^l & \dots & \sum_{i=1}^B \sigma(z_{P,i}^{l-1})\delta_{i,K}^l\end{bmatrix}
\end{aligned}
$$

One can see with the form of the gradient of the weights that the summation over the batch dimension is already occuring in the outer product. As a result, for the weights, we simply have to divide the result by B to get an estimate for the gradient.