Homework 1: Pen and Paper Exercises

Dana Kianfar
11391014
University of Amsterdam

January 6, 2018

Exercise 1

We derive the batch update expressions without resorting to tensor operations. For more concise notation we refer to all variables indexed by $_{out}$ as $_3$.

$$X : d_0 \times N$$
$$W_1 : d_1 \times d_0, \quad S_1, Z_1 : d_1 \times N$$
$$W_2 : d_2 \times d_1, \quad S_2, Z_2 : d_2 \times N$$
$$W_3 : d_3 \times d_2, \quad S_3, Z_3, Y_3 : d_3 \times N$$

The loss L can be written as, where $||\cdot||_F$ is the Forbenius norm.

$$L = \frac{1}{2}||Y_{out} - Y_{gt}||_F^2$$

We use the chain rule to expand the derivative of W_3, and derive its update.

$$\frac{\partial L}{\partial W_3} = \underbrace{\frac{\partial L}{\partial Y_3}}_{d_3 \times N} \cdot \underbrace{\frac{\partial Y_3}{\partial S_3}}_{d_3 \times N} \cdot \underbrace{\frac{\partial S_3}{\partial W_3}}_{d_3 \times N \times d_3 \times d_2}$$

The derivative for each term is

$$\frac{\partial L}{\partial Y_3} = \frac{1}{2}\frac{\partial \text{Tr}\left[(Y_3 - Y_{gt})^T(Y_3 - Y_{gt})\right]}{\partial Y_3} = \frac{1}{2}\frac{\partial \text{Tr}\left[Y_3^t Y_3 + Y_{gt}^T Y_{gt} - Y_3^T Y_{gt} - Y_{gt}^T Y_3\right]}{\partial Y_3}$$
$$= \frac{1}{2}2(Y_3 - Y_{gt}) = Y_3 - Y_{gt}$$

As $Y_3 = Z_3 = f_3(S_3)$ and $f_3(\cdot)$ is an element-wise operation, we have

$$\frac{\partial Y_3}{\partial S_3} = f_3'(S_3)$$

The derivative of S_3 with respect to any element W_{3ij} is an $d_3 \times N$ matrix whose ith row is $Z_{j:}$ as shown below.

$$\frac{\partial S_3}{\partial W_{3ij}} = \left[\frac{\partial S_{3kn}}{\partial W_{3ij}}\right]_{\substack{i:1\to d_3 \\ j:1\to d_2 \\ k:1\to d_3 \\ n:1\to N}} = \left[\frac{\partial W_{3k:} Z_{2:n}}{\partial W_{3ij}}\right] = \left[Z_{2jn} \cdot \delta_i^k\right]_{kn}$$

Putting all the results together, we have the following. Here $\odot$ is the hadamard operator between two matrices.

$$\frac{\partial L}{\partial W_3} = \underbrace{\left[\frac{\partial L}{\partial Y_3} \odot \frac{\partial Y_3}{\partial S_3}\right]}_{\Delta_3} \cdot Z_2^T = \left[(Y_3 - Y_{gt}) \odot f_3'(S_3)\right] Z_2^T$$

In a similar fashion we can write the gradients $\frac{\partial L}{\partial Z_2}$

$$\frac{\partial L}{\partial Z_2} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial Z_2} = W_3^T \cdot \underbrace{\left[\frac{\partial L}{\partial Y_3} \odot \frac{\partial Y_3}{\partial S_3}\right]}_{\Delta_3} = W_3^T \cdot \left[(Y_3 - Y_{gt}) \odot f_3'(S_3)\right]$$

We can then continue to write the gradients for other model parameters.

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial S_2} \cdot \frac{\partial S_2}{\partial W_2}$$
$$= \underbrace{\left[(W_3^T \cdot \Delta_3) \odot f_2'(S_2)\right]}_{\Delta_2} \cdot Z_1^T$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial S_2} \cdot \frac{\partial S_2}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial S_1} \cdot \frac{\partial S_1}{\partial W_1}$$
$$= \underbrace{\left[(W_2^T \cdot \Delta_2) \odot f_1'(S_1)\right]}_{\Delta_1} \cdot X^T$$

Error Propagation

$$\partial W_k = \frac{\partial L}{\partial W_k} = \Delta_k \cdot Z_{k-1}^T$$

Exercise 2

$$X = \begin{bmatrix} 0.75 & 0.2 & -0.75 & 0.2 \\ 0.8 & 0.05 & 0.8 & -0.05 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} 0.6 & 0.01 \\ 0.7 & 0.43 \\ 0.0 & 0.88 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.02 & 0.03 & 0.09 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}$$

Forward-Propagation

$$S_1 = W_1 X = \begin{bmatrix} 0.4580 & 0.1205 & -0.4420 & 0.1195 \\ 0.8690 & 0.1615 & -0.1810 & 0.1185 \\ 0.7040 & 0.0440 & 0.7040 & -0.0440 \end{bmatrix}$$

$$Z_1 = f_1(S_1) = \begin{bmatrix} 0.4580 & 0.1205 & 0.0000 & 0.1195 \\ 0.8690 & 0.1615 & 0.0000 & 0.1185 \\ 0.7040 & 0.0440 & 0.7040 & 0.0000 \end{bmatrix}$$

$$S_2 = W_2 Z_1 = Z_2 = Y_{out} = \begin{bmatrix} 0.0986 & 0.0112 & 0.0634 & 0.0059 \end{bmatrix}$$

$$\mathcal{L} = 0.4916$$

Error Signals

$$\Delta_{out} = \frac{\partial \mathcal{L}}{\partial Y_{out}} = \begin{bmatrix} -0.2254 & -0.2472 & 0.2658 & 0.2515 \end{bmatrix}$$

$$\Delta_1 = \frac{\partial \mathcal{L}}{\partial S_1} = \begin{bmatrix} -0.0045 & -0.0049 & 0.0000 & 0.0050 \\ -0.0068 & -0.0074 & 0.0000 & 0.0075 \\ -0.0203 & -0.0222 & 0.0239 & 0.0000 \end{bmatrix}$$

Parameter Derivatives

$$\frac{\partial \mathcal{L}}{\partial W_2} = \begin{bmatrix} -0.1029 & -0.2060 & 0.0176 \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = \begin{bmatrix} -0.0034 & -0.0041 \\ -0.0050 & -0.0062 \\ -0.0376 & 0.0018 \end{bmatrix}$$

Parameter Updates

$$W_1 = \begin{bmatrix} 0.6017 & 0.0121 \\ 0.7025 & 0.4331 \\ 0.0188 & 0.8791 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.0715 & 0.1330 & 0.0812 \end{bmatrix}$$

$$\mathcal{L} = 0.4657$$

Exercise 3

Multi-class Hinge Loss

This objective function penalizes all predicted *false* classes that are within *margin* units of predicted *true* class probability. The aim is to construct an error signal that reduces the predicted class probability for false classes. As we use a softmax function to obtain probability estimates, reducing false class probabilities also means increasing the true class probability. While traditional classification objectives such as cross-entropy aim to increase the true class probability to one (one-hot encoding targets), hinge-loss is not computed over the true class predicted probability, but instead attempts to maximize the separation between the true class decision boundary and false class predictions that fall within a close proximity (i.e. the supports).

Derivatives

We refer to the $margin$ as C for a more concise notation.

$$\mathcal{L} = \sum_{j \neq y_i} \begin{cases} 0 & p_{yi} - p_j > C \\ p_j - p_{y_i} + C & \text{else} \end{cases}$$

$$\frac{\partial \mathcal{L}}{\partial o_j} = \sum_{j \neq y_i} \begin{cases} 0 & p_{yi} - p_j > C \\ \frac{\partial p_j - p_{y_i}}{\partial o_j} & \text{else} \end{cases}$$

$$\frac{\partial p_k - p_{y_i}}{\partial o_j} = \frac{\partial \left(\frac{e^{o_k} - e^o_{y_i}}{\sum_l e^o_l} \right)}{\partial o_j}$$

To apply the quotient rule to the above, we compute the derivative for the numerator and denominator.

$$\frac{\partial (e^{o_k} - e^{o_{y_i}})}{\partial o_j} = e^{o_j} \delta^k_j - e^{o_j} \delta^{y_i}_j = e^{o_j} (\delta^k_j - \delta^{y_i}_j)$$

$$\frac{\partial \sum_l e^{o_l}}{\partial o_j} = e^{o_j}$$

For brevity, we introduce the abbreviation $\Sigma = \sum_l e^{o_l}$.

$$\begin{aligned} \frac{\partial \left(\frac{e^{o_k} - e^{o_{y_i}}}{\sum_l e^{o_l}} \right)}{\partial e^{o_j}} &= \frac{e^{o_j} (\delta^k_j - \delta^{y_i}_j) \cdot \Sigma - e^{o_j} (e^{o_k} - e^{o_{y_i}})}{\Sigma^2} \\ &= \frac{e^{o_j} (\delta^k_j - \delta^{y_i}_j)}{\Sigma} - \frac{e^{o_j} (e^{o_k} - e^{o_{y_i}})}{\Sigma^2} \\ &= p_j (\delta^k_j - \delta^{y_i}_j) - p_j p_k + p_j p_{y_i} = p_j (\delta^k_j - \delta^{y_i}_j - p_k + p_{y_i}) \end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial o_j} = \sum_{j \neq y_i} \begin{cases} 0 & p_{yi} - p_j > C \\ p_j (\delta^k_j - \delta^{y_i}_j - p_k + p_{y_i}) & \text{else} \end{cases}$$

Exercise 4

As we've observed in exercise 1, the gradients for deep networks can be very length equations and expensive to compute individually. If we use the recursive properties in backprob,

namely that the gradient updates for each layer can be computed based on the gradient of its proceeding layer, we can save a lot of computation. Furthermore, by caching intermediate values such as pre-activations and activations during the forward pass, we eliminate the need to recompute these values in the backward pass. Finally, once we have computed the error signal δ_{out} for the final layer, we can use it to propagate the error signals for the previous layers as shown in exercise 1.