Practical 1. MLPs, CNNs, Backpropagation

University of Amsterdam – Deep Learning Course

Huishi Qiu
12955582
`huishiqiu@gmail.com`

November 15, 2020

1 MLP backprop and NumPy Implementation

1.1 Evaluating the Gradients

1.1.1 Linear Module

1. $\frac{\partial L}{\partial W}$ Firstly rewrite the derivatives by chain rule,

$$\frac{\partial L}{\partial W} = \left[\frac{\partial L}{\partial W}\right]_{ij} = \frac{\partial L}{\partial W_{ij}} = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial W_{ij}} \tag{1}$$

looking at the second part of equation 1,

$$\frac{\partial Y_{mn}}{\partial W_{ij}} = \frac{\partial \sum_s X_{ms} W^T_{ns} + B_{mn}}{\partial W_{ij}} = \frac{\partial X_{mi} W^T_{ni}}{\partial W_{ij}} + 0 = \frac{\partial X_{mi} W_{in}}{\partial W_{ij}} = X_{mi}\delta_{nj} \tag{2}$$

Combining equation 1 and 2 we have:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial W_{ij}} = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial W_{ij}} = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} X_{mi}\delta_{nj} = \sum_{m} \frac{\partial L}{\partial Y_{mj}} X_{mi} = \sum_{m} X^T_{im} \frac{\partial L}{\partial Y_{mj}} \tag{3}$$

which is the dot product of matrix X^T's i row and $\frac{\partial L}{\partial Y}$'s j column, thus

$$\frac{\partial L}{\partial W} = X^T \cdot (\frac{\partial L}{\partial Y}) \tag{4}$$

2. $\frac{\partial L}{\partial b}$ Secondly for $\frac{\partial L}{\partial b}$, which should return a row vector same as b, Again, first we rewrite the derivatives by chain rule ,

$$\frac{\partial L}{\partial b} = \left[\frac{\partial L}{\partial b}\right]_{i} = \frac{\partial L}{\partial b_i} = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial b_i} \tag{5}$$

Looking at second part of equation 5,

$$\frac{\partial Y_{mn}}{\partial b_i} = \frac{\partial \sum_s X_{ms} W^T_{sn} + B_{mn}}{\partial b_i} = 0 + \frac{\partial B_{mn}}{\partial b_i} = \frac{\partial B_{mn}}{\partial B_{mi}} = \delta_{ni} \tag{6}$$

Combining equation 5 and 6 we have:

$$\frac{\partial L}{\partial b} = \left[\frac{\partial L}{\partial b}\right]_i = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \delta_{ni} = \sum_m \frac{\partial L}{\partial Y_{mi}} \tag{7}$$

Which is sum over according to the y axis of the output.

3. $\frac{\partial L}{\partial X}$ Lastly for $\frac{\partial L}{\partial X}$, changing the derivetive by chain rule

$$\frac{\partial L}{\partial X} = \left[\frac{\partial L}{\partial X}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial X_{ij}} \tag{8}$$

looking at the second part of equation 8, we have:

$$\frac{\partial Y_{mn}}{\partial X_{ij}} = \frac{\partial \sum_s X_{ms} W^T_{ns} + B_{mn}}{\partial X_{ij}} = \frac{\partial X_{mj} W^T_{nj}}{\partial X_{ij}} + 0 = \frac{\partial X_{mj} W_{jn}}{\partial X_{ij}} = \delta_{mi} W_{jn} \tag{9}$$

Combining equation 8 and 9 we have:

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial W_{ij}} = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} \frac{\partial Y_{mn}}{\partial X_{ij}} = \sum_{m,n} \frac{\partial L}{\partial Y_{mn}} W_{jn} \delta_{mi} = \sum_n \frac{\partial L}{\partial Y_{in}} W_{jn} = \sum_n \frac{\partial L}{\partial Y_{in}} W^T_{nj} \tag{10}$$

which is the dot product of matrix $\frac{\partial L}{\partial Y}$'s i row and W^T's j column, therefore

$$\frac{\partial L}{\partial X} = (\frac{\partial L}{\partial Y}) \cdot W^T \tag{11}$$

1.1.2 Activation Module

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \frac{\partial Y}{\partial X} \tag{12}$$

Where second part of equation 12 can be written as

$$\frac{\partial Y}{\partial X} = \left[\frac{\partial Y}{\partial X}\right]_{ij} = \frac{\partial Y_{ij}}{\partial X_{ij}} = \frac{\partial h(X_{ij})}{\partial X_{ij}} = h^{'}(X_{ij})\delta_{ij} = \begin{bmatrix} h^{'}(X_{11}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & h^{'}(X_{nn}) \end{bmatrix} \tag{13}$$

In the equation 13, only when $i = j$ the derivative of $h^{'}(X_{ij})$ is not equal to 0, while all others are, in other words $h^{'}(X_{ij})\delta_{ij}$ turn out to be diagonal matrix, combining equation 12 and 13 we have

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \begin{bmatrix} h^{'}(X_{11}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & h^{'}(X_{nn}) \end{bmatrix} = \frac{\partial L}{\partial Y} \circ h^{'}(X)(14)$$

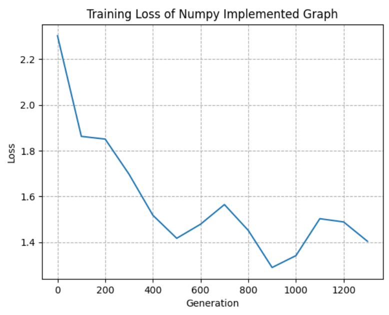
1.1.3 Softmax and Loss Module

1. Softmax Module
   By the definition of softmax, we look at derivatives over the rows, which is a vector.
   $$Y_{ij} = Softmax(\boldsymbol{X}_{ij}) = \frac{e^{X_{ij}}}{\sum_k e^{X_{ik}}} \tag{15}$$
   $$\frac{\partial L}{\partial X} = \sum_m \frac{\partial L}{\partial Y_m} \frac{\partial Y_m}{\partial X_m} \tag{16}$$
   Focusing on the second part of equation 16, the derivatives between vector and vector is a Jacobian matrix, therefore we discuss whether the element is on the diagonal line, e.g., whether i=j. Given $\frac{g(X)}{h(X)} = \frac{g'(x)h(x)-h'(x)g(x)}{[h(x)]^2}$, when i=j, we have
   $$\frac{\partial Y_{mi}}{\partial X_{mi}} = \frac{\partial Softmax(\boldsymbol{X}_{mi})}{\partial X_{mi}} = \frac{\partial \frac{e^{X_{mi}} \sum_k e^{X_{mk}} - e^{X_{mi}} e^{X_{mi}}}{(\sum_k e^{X_{mk}})^2}}{\partial X_{mi}} = \frac{e^{X_{mi}}}{\sum_k e^{X_{mk}}} - (\frac{e^{X_{mi}}}{\sum_k e^{X_{mk}}})^2$$
   $$= Y_{mi}(1 - Y_{mi})(17)$$
   when $i \neq j$:
   $$\frac{\partial Y_{mj}}{\partial X_{mi}} = \frac{\partial Softmax(\boldsymbol{X}_{mj})}{\partial X_{mi}} = \frac{\partial \frac{e^{X_{mj}} \sum_k e^{X_{mk}} - e^{X_{mj}} e^{X_{mi}}}{(\sum_k e^{X_{mk}})^2}}{\partial X_{mi}} = 0 - \frac{e^{X_{mi}} e^{X_{mj}}}{(\sum_k e^{X_{mk}})^2}$$
   $$= -Y_{mi}Y_{mj}(18)$$
   Merging equation 17 and equation 18 above two equation by Kronecker delta we have
   $$\frac{\partial Y_{mj}}{\partial X_{mi}} = Y_{mi}(\delta_{mi} - Y_{mj}) \tag{19}$$
   Merging equation 16 and 19, the result turn out to be
   $$\frac{\partial L}{\partial X} = \sum_m \frac{\partial L}{\partial Y_m} \frac{\partial Y_m}{\partial X_m} = \sum_m \frac{\partial L}{\partial Y_m} Y_{mi}(\delta_{mi} - Y_{mj}) = \frac{\partial L}{\partial Y} Y_i(\delta_{ij} - Y_j) \tag{20}$$

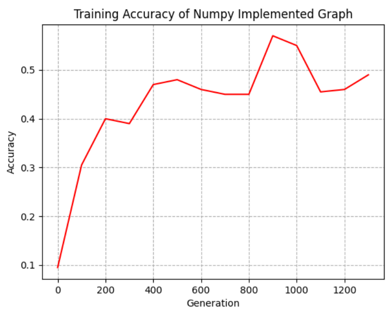
2. Loss Module
   $$\frac{\partial L}{\partial X} = \left[\frac{\partial L}{\partial X}\right]_{ij} = \frac{\partial L}{\partial X_{ij}} = -\frac{\partial \frac{1}{S} \sum_{ik} T_{ik} log_(X_{ik})}{\partial X_{ij}} = -\frac{1}{S}\frac{T}{X} \tag{21}$$

1.2 NumPy Implementation

Figure 1 showed the loss and accuracy on training set corresponding to the generations. From both plots we see the model met bottleneck after around 400 generations and the performance hesitated. The model achieved accuracy of 48.66% on CIFAR10 testing set, which is slightly beyond the threshold of 48%.



(a) Training Loss



(b) Training Accuracy

Figure 1: Result of NumPy Implemented Model

2 PyTorch MLP

2.1 Result

With original parameter setting, PyTorch implemented model achieved the accuracy of 49.28% on testing set, compared with the accuracy of 48.66% numpy implemented network. To get familiar with parameter setting and improve the performance, following changes had been made.

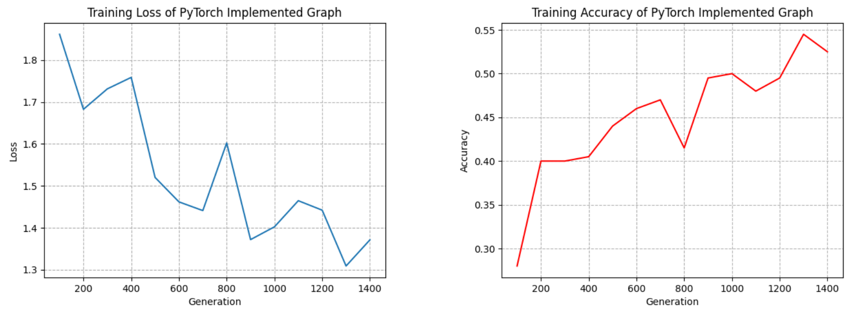
1. Network structure
   The number of hidden units has been increased from 100 to 256. This is in consider of the model input and output numbers. After flatten the input dimension space, the input side has more than 3000 features. To classify 10 categories, here I adopt the formula of $H_{hidden} = \sqrt{M_{input} * N_{output}}$ and round up to the nearest index of 2, which is basically based on the experience[1]. Besides hidden neuron numbers, more hidden layers has also been tested. Overall more layers would obtain higher accuracy on training set but also lead to overfitting and longer training time. To balance, here I adopt the 2 hidden layer setting of [256,256], which increased performance over 2%.

2. Optimizer
   Various kind of optimizer has been tested. Optimizer directly work on the model parameter update. Besides default SGD in the experiment I have tried Adam, SGD with momentum and Adagrad additionally. By the end Adagrad yield the best performance. Moreover, I set weight decay has been introduced, e.g. L2 regularization. After testing the values of [0.1,0.01,0.001], weight decay=0.001 output the best result.

3. Training Generation
   Training iteration has lifted from 1400 to 5000. This is based on the finding

	Default	After
Hidden Layer Setting	100	256,256
Optimizer	SGD	Adagrad
Generations	1400	5000

Table 1: Parameter modification



(a) Training Loss (b) Training Accuracy

Figure 2: Result of PyTorch Implemented Model

that after previous two modifications, the model will not converge in the given training iterations, which limited the performance.

4. Others
   Other modification includes initialize parameters with kaiming set up, changing batch size to lower or higher values etc, which didn't benefit the model, thus has not been introduced to the final model.

Finally, here we conclude the parameter setting after modification, which is showed in table 1.

After above modifications, the accuracy on testing set improved from 49.45%to 54.74%, which is fairly over the threshold of 52 %. Figure 2 showed the curve of loss and accuracy curve on training set. Compared with figure 1 the model could overcome the bottleneck and could keep improving in the later generations, with higher accuracy of 65% on the training set, though on the test set performance barely improved 4%. Loss value also decreased from 1.4 to the level of 0.8.

2.2 Activation module

The main problem of Tanh function is vanishing gradient. For instance, when input value x is beyond the space of [-5,+5], the gradient almost do not change no matter how big or how small the input value is. On the other hand, it's also a advantage since the output value could be limited. Compared with ELU, given the positive input, the output could be as big as infinity. Therefore to use which activation function should take input values' characteristics into consideration.

3 Custom Module: Layer Normalization

3.1 Automatic differentiation

3.2 Manual implementation of backward pass

3.2.1 a)

$$\frac{\partial L}{\partial \gamma} = \left[\frac{\partial L}{\partial \gamma}\right]_i = \frac{\partial L}{\partial \gamma_i} = \sum_{s,i} \frac{\partial L}{\partial Y_{si}} \frac{\partial Y_{si}}{\partial \gamma_i} \tag{22}$$

Looking at the second part of equation 22,

$$\frac{\partial Y_{si}}{\partial \gamma_i} = \frac{\gamma_i \hat{X} + \beta_i}{\partial \gamma_i} = \hat{X} + 0 = \frac{X_{si} - \mu_s}{\sqrt{\sigma_s^2 + \epsilon}} \tag{23}$$

Merging equation 22 with 21 we have following,

$$\frac{\partial L}{\partial \gamma} = \sum_{s,i} \frac{\partial L}{\partial Y_{si}} \frac{\partial Y_{si}}{\partial \gamma_i} = \sum_{s} \frac{\partial L}{\partial Y_{si}} \frac{X_{si} - \mu}{\sqrt{\sigma_s^2 + \epsilon}} \tag{23}$$

b)

$$\frac{\partial L}{\partial \beta} = \left[\frac{\partial L}{\partial \beta}\right]_i = \frac{\partial L}{\partial \beta_i} = \sum_{s,i} \frac{\partial L}{\partial Y_{si}} \frac{\partial Y_{si}}{\partial \beta_i} \tag{24}$$

Looking at the second part of equation 24,

$$\frac{\partial Y_{si}}{\partial \beta_i} = \frac{\gamma_i \hat{X} + \beta_i}{\partial \beta_i} = 0 + 1 = 1 \tag{25}$$

Merging equation 25 into 24 we have

$$\frac{\partial L}{\partial \beta} = \sum_{s,i} \frac{\partial L}{\partial Y_{si}} \frac{\partial Y_{si}}{\partial \beta_i} = \frac{\partial L}{\partial Y} \tag{26}$$

c)

$$\frac{\partial L}{\partial \boldsymbol{X}} = \left[\frac{\partial L}{\partial \boldsymbol{X}}\right]_{ri} = \frac{\partial L}{\partial X_{ri}} = \sum_{s,j} \frac{\partial L}{\partial Y_{sj}} \frac{\partial Y_{sj}}{\partial X_{ri}} \tag{27}$$

Looking at the second part of equation 27, since its partial derivative we have to sum up all elements where X appeared, thus we have

$$\frac{\partial Y_{sj}}{\partial X_{ri}} = \frac{\partial \gamma_s \hat{X} + \beta_s}{\partial X_{ri}} = \frac{\partial L}{\partial \hat{X}} \frac{\partial \hat{X}}{\partial X_{ri}} + \frac{\partial L}{\partial \sigma^2} \frac{\partial \sigma^2}{\partial X_{ri}} + \frac{\partial L}{\partial \mu} \frac{\partial \mu}{\partial X_{ri}} \tag{28}$$

Next, we have to look each derivative of equation 28 independently,

$$\frac{\partial L}{\partial \hat{X}} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial \hat{X}} = \frac{\partial L}{\partial Y}\gamma \tag{29}$$

$$\frac{\partial \hat{X}}{\partial X_{ri}} = \frac{\partial \frac{X_{si}-\mu_s}{\sqrt{\sigma_s^2+\epsilon}}}{\partial X_{ri}} = \frac{1}{\sqrt{\sigma_s^2+\epsilon}} \tag{30}$$

$$\begin{aligned}\frac{\partial L}{\partial \sigma^2} &= \frac{\partial L}{\partial \hat{X}}\frac{\partial \hat{X}}{\partial \sigma^2} = \frac{\partial L}{\partial Y}\gamma\sum_s (X_s-\mu)(\sigma^2+\epsilon)^{-0.5-1}(-0.5) \\ &= (-0.5)*\frac{\partial L}{\partial Y}\gamma\sum_s (X_s-\mu)(\sigma^2+\epsilon)^{-1.5}\end{aligned} \tag{31}$$

$$\frac{\partial \sigma^2}{\partial X_{ri}} = \frac{\partial \frac{1}{M}\sum_r (X_r-\mu)^2}{\partial X_{ri}} = \frac{2}{M}(X_{ri}-\mu) \tag{32}$$

$$\frac{\partial L}{\partial \mu} = \frac{\partial L}{\partial \hat{X}}\frac{\partial \hat{X}}{\partial \mu} + \frac{\partial L}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial \mu} = \frac{\partial L}{\partial Y}\gamma\sum_s^M \frac{-1}{\sqrt{\sigma^2+\epsilon}} + 0 \tag{33}$$

$$\frac{\partial \mu}{\partial X_{ri}} = \frac{1}{M} \tag{34}$$

Merging equation 29-34 into equation 28 we have,

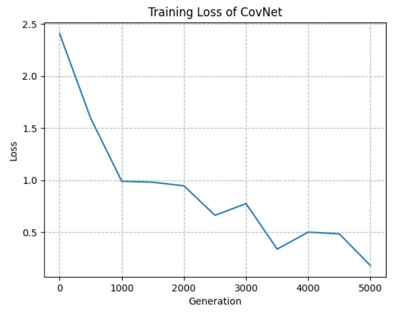
$$\begin{aligned}\frac{\partial L}{\partial X_{ri}} &= \frac{\partial L}{\partial \hat{X}}\frac{\partial \hat{X}}{\partial X_{ri}} + \frac{\partial L}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial X_{ri}} + \frac{\partial L}{\partial \mu}\frac{\partial \mu}{\partial X_{ri}} \\ &= \left(\frac{\partial L}{\partial Y}\gamma\frac{1}{\sqrt{\sigma_s^2+\epsilon}}\right) + \\ &\left((-0.5)*\frac{\partial L}{\partial Y}\gamma\sum_s^M (X_s-\mu)(\sigma^2+\epsilon)^{-1.5}\frac{2}{M}(X_{ri}-\mu)\right) + \\ &\left(\frac{1}{M}\sum_s^M \frac{\partial L}{\partial Y}\gamma\frac{-1}{\sqrt{\sigma^2+\epsilon}}\right)\end{aligned} \tag{35}$$

To simplify we extract $\frac{1}{\sqrt{\sigma_s^2+\epsilon}}$ from each component thus we have,

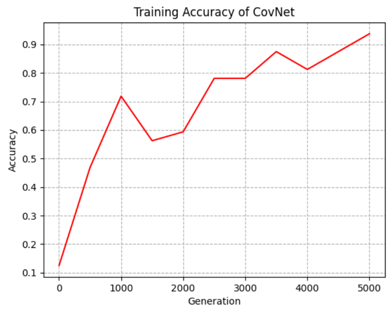
$$\frac{\partial L}{\partial \boldsymbol{X}} = \frac{1}{\sqrt{\sigma_s^2+\epsilon}}\left(\frac{\partial L}{\partial Y}\gamma - \frac{1}{M}\left(\sum_s^M \frac{\partial L}{\partial Y}\gamma + \frac{X_{si}-\mu_s}{\sqrt{\sigma_s^2+\epsilon}}\sum_s^M \frac{\partial L}{\partial Y}\gamma\frac{X_{si}-\mu_s}{\sqrt{\sigma_s^2+\epsilon}}\right)\right) \tag{36}$$

3.2.2 d) Batch Normalization and Layer Normalization Comparison

Batch Normalization trying to normalize each input feature to reduce its deviation, to do so we have to store each feature's mean and average for calculation, which occupied the memory. Layer Normalization balance entire layer of features despite the batch size. However, when features do not belong to same category, e.g. colors and figure size for pictures, the layer normalization may reduce the model's performance. Regarding to batch size, it directly affect batch normalization. The higher is the batch size, the input features are better normalized, while the output of Layer Normalization is not affected, only increased more samples in each batch.



(a) Training Loss



(b) Training Accuracy

Figure 3: Result of ConvNet

4 PyTorch CNN

4.1 VGG net

Figure 3 gave the loss and accuracy on training set produced by CovNet. From both plots we clearly see CNN based CovNet perform significantly better than the previous two MLP based model. The loss value arrived at the level of 0.5 compared with 0.8 of PyTorch model with modification, and training accuracy over 80% compared with 65% previously. The model finally obtained accuracy of 79.28% on CIFAR10 test set, improved more than 20% from MLP based model. At the same time training such a complex network also lead to much longer training time, which is a trade off of the increased performance.

References

[1] Medium. 2020. Beginners Ask "How Many Hidden Layers/Neurons To Use In Artificial Neural Networks?". [online] Available at:

<https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e>