FMAN45 - Assignment 3

Jingmo Bai

June 7, 2023

1 Exercise 1

In this part, we set the number of x as m, and the number of y as n. To derive the expressions for $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial W}$, $\frac{\partial L}{\partial b}$, we use the chain rule:

$$\frac{\partial L}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} \frac{\partial y_l}{\partial x_i} \tag{1}$$

$$= \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} \frac{\partial}{\partial x_i} (\sum_{i=1}^{m} W_{li} x_i + b_l) \tag{2}$$

$$= \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} W_{li} \tag{3}$$

$$\begin{bmatrix} \frac{\partial L}{\partial x_1} \\ \vdots \\ \frac{\partial L}{\partial x_m} \end{bmatrix} = \begin{bmatrix} W_{11} & \cdots & W_{n1} \\ \vdots & \ddots & \vdots \\ W_{1m} & \cdots & W_{nm} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1} \\ \vdots \\ \frac{\partial L}{\partial y_n} \end{bmatrix}$$

$$\frac{\partial L}{\partial W_{li}} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} \frac{\partial y_l}{\partial W_{li}} \tag{4}$$

$$= \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} \frac{\partial}{\partial W_{li}} (\sum_{i=1}^{m} W_{li} x_i + b_l) \tag{5}$$

$$= \frac{\partial L}{\partial y_l} xi \tag{6}$$

$$\begin{bmatrix} \frac{\partial L}{\partial W_{11}} & \cdots & \frac{\partial L}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial W_{n1}} & \cdots & \frac{\partial L}{\partial W_{nm}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial y_1} \\ \vdots \\ \frac{\partial L}{\partial y_n} \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_m \end{bmatrix}$$

$$\frac{\partial L}{\partial b_l} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} \frac{\partial y_l}{\partial b_l} \tag{7}$$

$$= \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} \frac{\partial}{\partial b_l} (\sum_{i=1}^{m} W_{li} x_i + b_l) \tag{8}$$

$$= \frac{\partial L}{\partial y_l} \tag{9}$$

$$\begin{bmatrix} \frac{\partial L}{\partial b_1} \\ \vdots \\ \frac{\partial L}{\partial b_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial y_1} \\ \vdots \\ \frac{\partial L}{\partial y_n} \end{bmatrix}$$

2 Exercise 2

$$Y = \begin{bmatrix} y_1^1 & \cdots & y_1^N \\ \vdots & \ddots & \vdots \\ y_n^1 & \cdots & y_n^N \end{bmatrix} = \begin{bmatrix} W_{11} & \cdots & W_{1m} \\ \vdots & \ddots & \vdots \\ W_{n1} & \cdots & W_{nm} \end{bmatrix} \begin{bmatrix} x_1^1 & \cdots & x_1^N \\ \vdots & \ddots & \vdots \\ x_m^1 & \cdots & x_m^N \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

To get the expressions we want, we just need to sum the expression we obtained in the previous exercise over all elements in the batch N.

$$\frac{\partial L}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l} W_{ij}$$

$$\frac{\partial L}{\partial X} = W^T \frac{\partial L}{\partial Y}$$

$$\begin{bmatrix} \frac{\partial L}{\partial x_1^{(1)}} & \cdots & \frac{\partial L}{\partial x_1^{(N)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial x_m^{(1)}} & \cdots & \frac{\partial L}{\partial x_m^{(N)}} \end{bmatrix} = \begin{bmatrix} W_{11} & \cdots & W_{n1} \\ \vdots & \ddots & \vdots \\ W_{1m} & \cdots & W_{nm} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1^{(1)}} & \cdots & \frac{\partial L}{\partial y_1^{(N)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial y_n^{(1)}} & \cdots & \frac{\partial L}{\partial y_n^{(N)}} \end{bmatrix}$$

$$\frac{\partial L}{\partial W_{ij}} = \sum_{l=1}^{N} \frac{\partial L}{\partial y_k^{(l)}} x_j^{(l)}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y} X^T$$

$$\begin{bmatrix} \frac{\partial L}{\partial W_{11}} & \cdots & \frac{\partial L}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial W_{n1}} & \cdots & \frac{\partial L}{\partial W_{nm}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial y_1^{(1)}} & \cdots & \frac{\partial L}{\partial y_1^{(N)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial y_n^{(1)}} & \cdots & \frac{\partial L}{\partial y_n^{(N)}} \end{bmatrix} \begin{bmatrix} x_1^{(1)} & \cdots & x_m^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \cdots & x_m^{(N)} \end{bmatrix}$$

$$\frac{\partial L}{\partial b_i} = \sum_{l=1}^{N} \frac{\partial L}{\partial y_i^{(l)}}$$

$$\frac{\partial L}{\partial b} = \begin{bmatrix} \frac{\partial L}{\partial b_1} \\ \vdots \\ \frac{\partial L}{\partial b_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial y_1^{(1)}} & \cdots & \frac{\partial L}{\partial y_1^{(N)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial y_n^{(1)}} & \cdots & \frac{\partial L}{\partial y_n^{(N)}} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N*1}$$

Code:

```
b = repmat(b, 1, batch);
Y = W * X + b;
dldX = W' * dldY;
dldX = reshape(dldX, sz);
dldW = dldY * X';
dldb = sum(dldY, 2);
```

3 Exercise 3

$$\frac{\partial L}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial y_l}{\partial x_i} = \sum_{l=1}^{n} \frac{\partial L}{\partial y_l}\frac{\partial}{\partial x_i}(\max(xi, 0))$$

$$= \begin{cases} \frac{\partial L}{\partial y_l} & \text{for} \quad x_i > 0 \\ 0 & \text{for} \quad x_i \leq 0 \end{cases}$$

Code:
Y = max(X, 0);
dldX = (X>0).* dldY;

4 Exercise 4

Given that

$$L(x, c) = -log(y_c) = -x_c + log(\sum_{j+1}^{m} e^{x_j})$$

We can find the expression for

$$\frac{\partial L}{\partial x_i} = \frac{\partial}{\partial x_i}(-x_c + log(\sum_{j+1}^{m} e^{x_j}))$$

$$= \begin{cases} \frac{e^{x_i}}{\sum_{j=1}^{m} e^{x_j}} & \text{for} \quad i \neq c \\ \frac{e^{x_i}}{\sum_{j=1}^{m} e^{x_j}} - 1 & \text{for} \quad i = c \end{cases}$$

$$= \begin{cases} y_i & \text{for} \quad i \neq c \\ y_i - 1 & \text{for} \quad i = c \end{cases}$$

Code:
index = sub2ind(sz, labels', 1:batch);
y = exp(x) ./ sum(exp(x));
L = -log(y);
L = mean(L(index));

index = sub2ind(sz, labels', 1:batch);
dldx = exp(x) ./ sum(exp(x));
dldx(index) = dldx(index) - 1;
dldx = dldx/batch;

5 Exercise 5

Code:
momentum{i}.(s) = opts.momentum * momentum{i}.(s) + (1-opts.momentum) * grads{i}.(s);
net.layers{i}.params.(s) = net.layers{i}.params.(s) - opts.learning_rate * (momentum{i}.(s)+ opts.weight_decay * net.layers{i}.params.(s));

6 Exercise 6

The results from training:
Iteration 3000:
Classification loss: 0.076386
Weight decay loss: 0.120861
Total loss: 0.197247
Training accuracy: 0.977146
Accuracy on the test set: 0.978200

Figure 1 is the plot of the kernels in the first convolution layer, there are 16 filters in total. Those filters extract features from different areas in the original images.

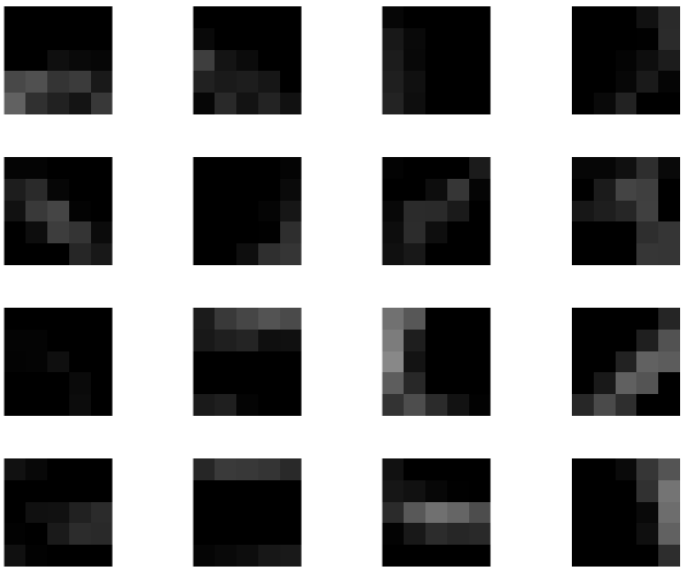

Figure 1: Kernels in the first convolutional layer

Figure 2 is the plot of a few digits that are misclassified. Some of them are indeed written in a strange way, and maybe even difficult for me to recognize. For example, the "5" in row 3 column 1, and the "5" in row 2 column 4, don't look like "5" for me. So it is not so surprising that the network has some problems classifying "5" and "3" correctly, which will be shown further from the confusion matrix later.

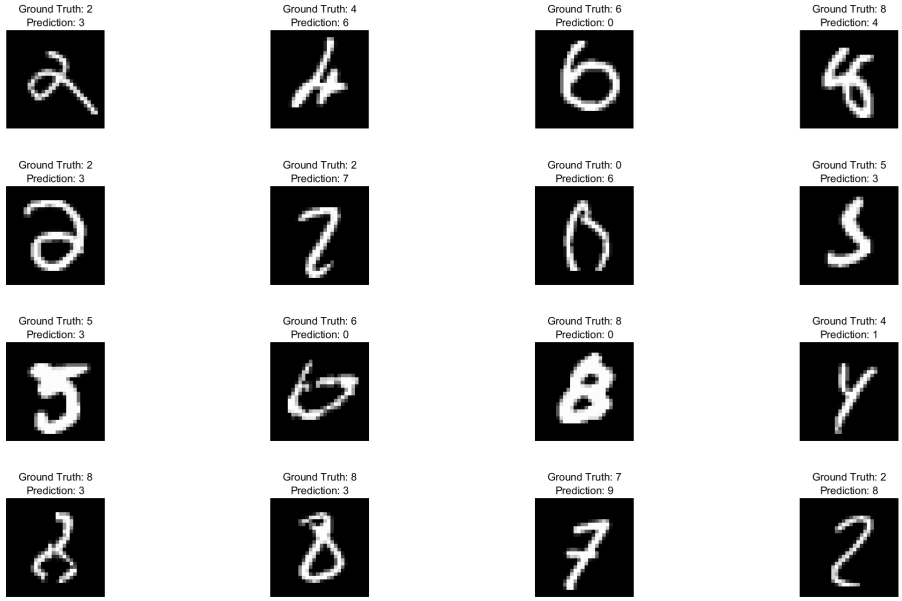

Figure 2: A few misclassified handwritten digits

Figure 3 is the plot of the confusion matrix, the precision and the recall.

On the confusion matrix chart, the columns correspond to the predicted class and the rows correspond to the true class. The diagonal blue cells correspond to observations that are correctly classified. The off-diagonal cells correspond to incorrectly classified observations.

The row at the bottom of the plot shows the percentages of all the examples predicted to belong to each class that are correctly and incorrectly classified. These metrics are often called the precision (or positive predictive value) and false discovery rate, respectively.

The column on the far right of the plot shows the percentages of all the examples belonging to each class that are correctly and incorrectly classified. These metrics are often called the recall (or true positive rate) and false negative rate, respectively.

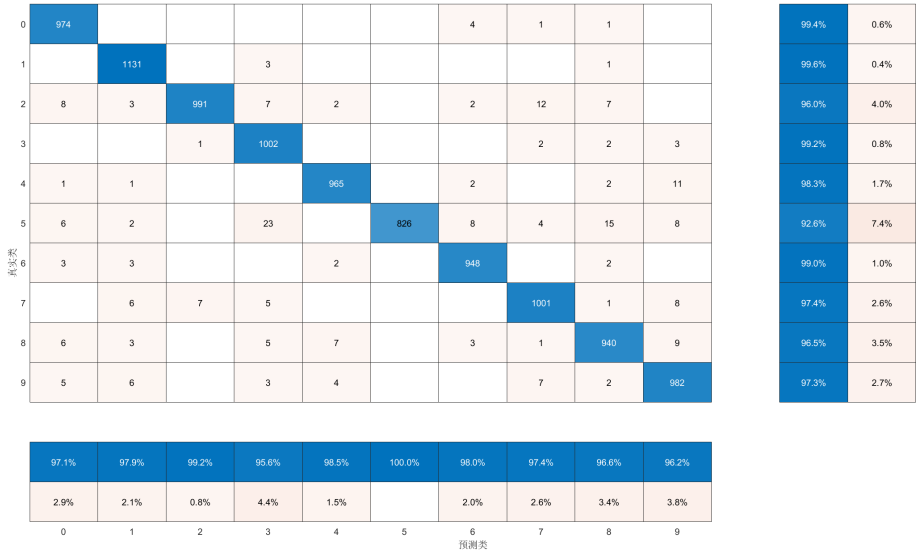

Figure 3: Confusion matrix, the precision and the recall

From the confusion matrix chart, we can see that the network has some problems classifying "5". There are 23 handwritten "5" that have been misclassified as "3", which is the most frequent error in the test, and another 15 "5" have been misclassified as "8". So the recall rate for "5" is low.

However, all the digits that have been classified as "5" are true "5", which means the precision rate for the prediction "5" is 100%.

In general, the network has a good performance when classifying "0" and "1". And it has some problems when classifying "2" and "5".

For the number of parameters for all layers in the network:

The 1st convolution layer: the number of trainable parameters is $5 * 5 * 1 * 16 + 16 = 416$

The 2nd convolution layer: the number of trainable parameters is $5 * 5 * 16 * 16 + 16 = 6416$

The fully connected layer: the number of trainable parameters is $10 * 784 + 10 = 7850$

There are no trainable parameters in Relu, Maxpooling and Softmax layers. So the total number of parameters is 14682.

7 Exercise 7

First, I test the default baseline network with 50000 images loaded. I notice that the training accuracy and the validation accuracy stop improving after about 1500 iterations when they reach approximately 0.44 of accuracy. So I tried to decrease the learning rate to 1e-4. This time, the training accuracy improves slowly, but it keeps improving until 5000 iterations when it reaches approximately 0.48. But it was too slow, so I change it back to 1e-3.

Then I tried to increase the network complexity, to add a third convolution

layer after the second one as well as a Relu activation layer. The structure of the convolution layer is the same as the second one. It results in good improvement in the accuracy, which reaches 0.541 after 5000 iterations.

The results from the training:
Iteration 5000:
Classification loss: 1.257372
Weight decay loss: 0.055485
Total loss: 1.312857
Training accuracy: 0.556241
Validation accuracy: 0.487005
Accuracy on the test set: 0.541000

Figure 4 is the plot of the kernels in the first convolution layer, there are 16 filters in total. Those filters extract features from different areas in the original images.

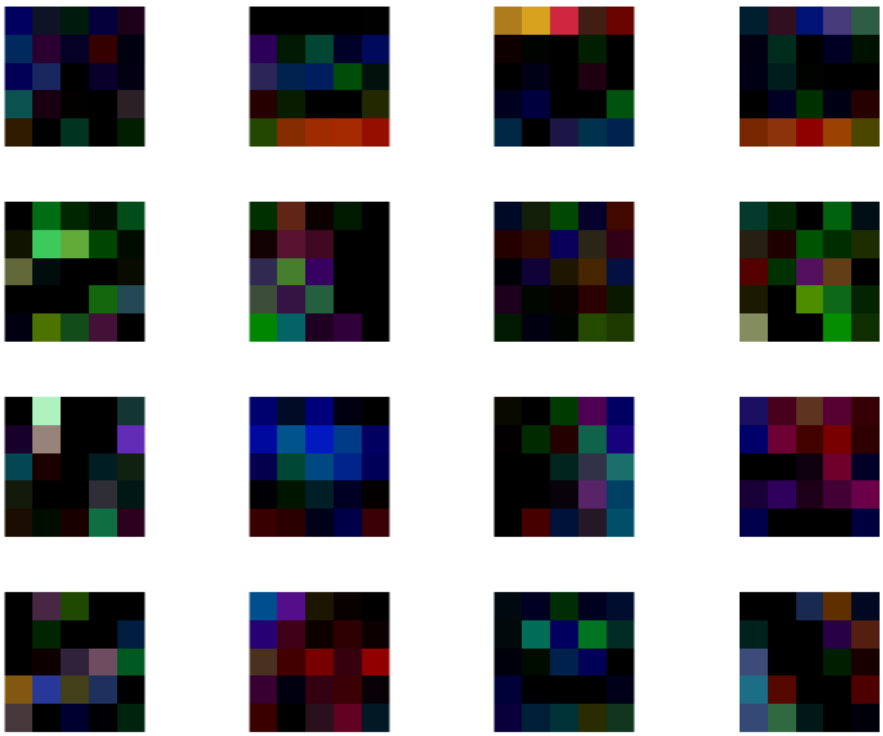

Figure 4: Kernels in the first convolutional layer

Figure 5 is the plot of a few digits that are misclassified. We can see that the network does not work so well. Sometimes it still fails to classify the objects even when the features of the targets are quite clear. This is probably due to we train on limited data and time, and an abundance of variations of the same class.

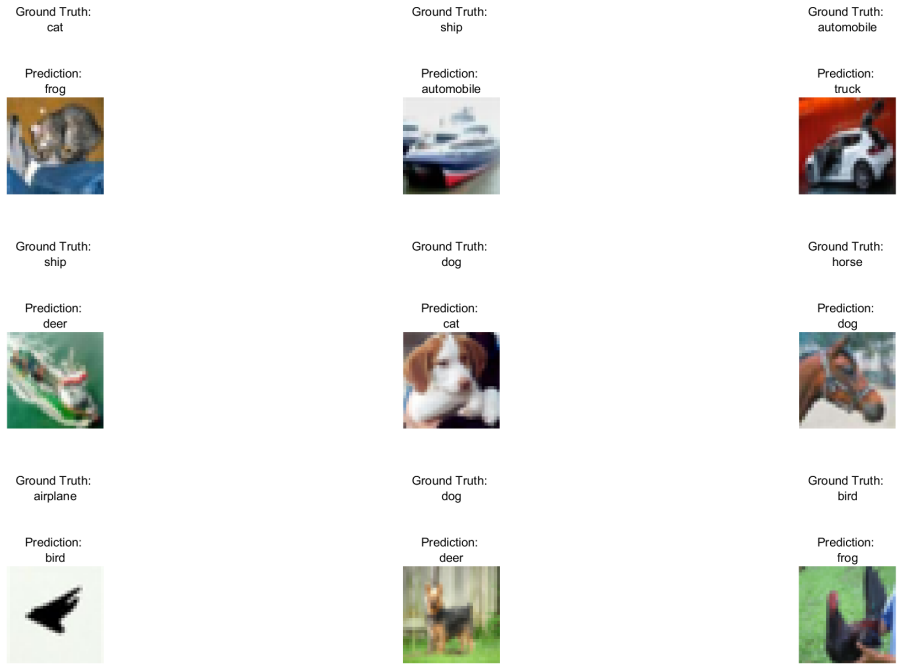

Figure 5: A few misclassified handwritten digits

Figure 6 is the plot of the confusion matrix, the precision and the recall. Class 1 to class 10, represent '1-airplane' '2-automobile' '3-bird' '4-cat' '5-deer' '6-dog' '7-frog' '8-horse' '9-ship' '10-truck'.

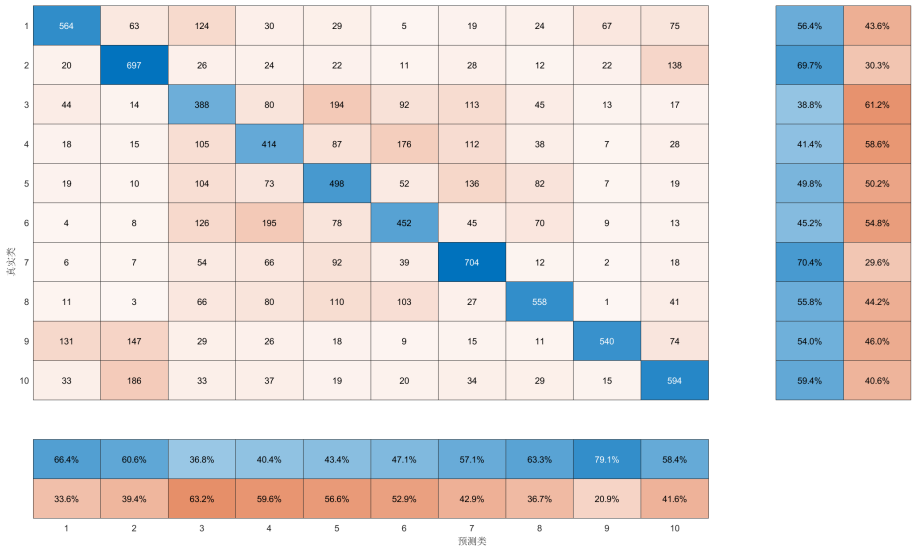

Figure 6: Confusion matrix, the precision and the recall

From the confusion matrix, we can see that the model has the best per-

formance when classifying 'automobile' and 'frog'. It has the most terrible performance when classifying 'bird'.

"Frog" has the highest precision rate, while "ship" has the highest recall rate. However, "bird" seems to be the most difficult class to classify, both the precision and recall rate for "bird" is very low.

The number of parameters for all layers in the network:
5*5*3*16+16=1216
5*5*16*16+16=6416
5*5*16*16+16=6416
10*4096+10=40970
So there are 55018 trainable parameters in total.

The number of parameters is much more than the previous task, but I think it is reasonable because the task is more difficult than classifying handwritten digits.