CHAPTER 8: APPLICATIONS AND TIPS OF MACHINE LEARNING ALGORITHM

Theory

In this chapter, we will discuss the tips needed in the use of machine learning algorithm, such as to regulate the learning rate, preprocess the training data, and prevent the overfitting which is the biggest challenge in the machine learning.

Recap

Let's remember the contents of the previous chapter.

The most important in multinomial classification is to have a good understanding of the softmax classification and its cost function.

Softmax Classification

Softmax classification is the most used classification in the multinomial classification when there are several classes.

And we call this function as **Softmax**.

$$S(y_i) = \frac{e^{y_i}}{\sum e^{y_i}}$$

Cost Function

In the softmax classification, the cross-entropy function is used for cost function.

$$L = \frac{1}{N}\sum_i D\left(S\left(wx_i + b\right), L_i\right)$$

$$D(S, L) = -\sum_i L_i \log\left(S_i\right)$$

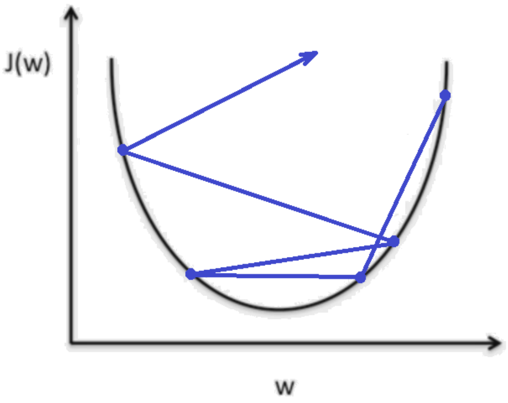
Learning Rate

In the gradient descent algorithm, the weights of the model can be updated as below equation.

$$W := W - \alpha \frac{\partial}{\partial W} \cos t\left(W\right)$$

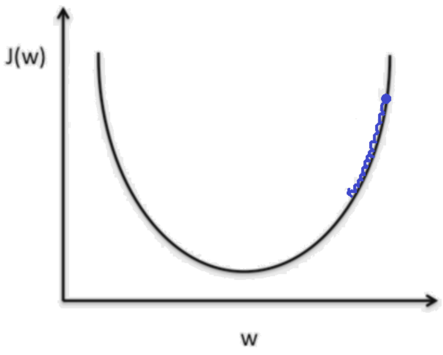
In here α is the learning rate and very important to decide the value of learning rate in the training.

Let's assume we set the learning rate as big value. The learning rate is big means that the value of the descent step from the original position is big.



If the learning rate is too big, the cost value couldn't converge to the minimal value and even divergent. This is called as **Overshooting**.

In contrast, let's assume that the learning rate is too small.



If the learning rate is too small then the cost function will decrease very small, so it will take a long time and steps to optimize and will be stopped before the minimal value.

Then how can we set the learning rate?

There isn't a unique and correct answer, but you can follow these commands.

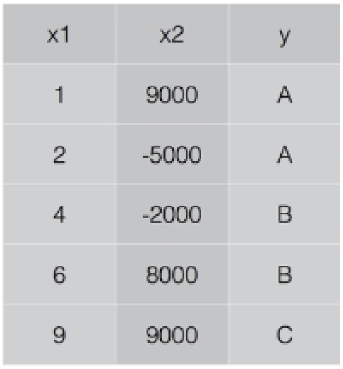
- Observe the cost function
- Check it goes down to a reasonable rate

You should check the cost value during training. If the cost reduces very little then you can increase the learning rate. And if the cost changes significantly and does not stabilize, you should decrease the learning rate.

Data Preprocessing

Sometimes we need to preprocess the training data which called feature data.

Let's consider this two dimension training data.



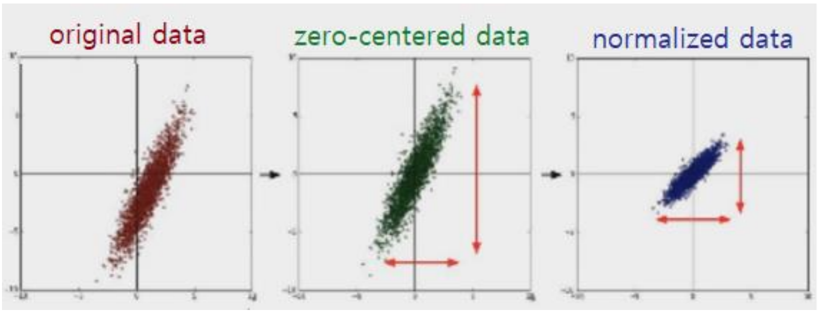
x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C

As we can know the table, the range of x1 is [1, 9] and x2 is [-5000, 9000] and the training will optimize the cost function which can be expressed as x1*w1+x2*w2. So w1 and w2 will be squeeze.

So, in this case, we need data preprocessing.

There are several methods for preprocessing, one of them is move the original data so as to situate on the zero position.

The other method is called **Normalize**, which map the original data to the proper range of the value.



The mathematical tool for normalize is simple.

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

In here, μ_j is the mean value, and σ_j is distribution. And we can implement this with Python as following command.

```
X_std[:, 0] = (x[:, 0] - x[:, 0].mean()) / x[:, 0].std()
```

We call this kind of normalization as **Standardization**.

Overfitting

Sometimes we can see the following case.

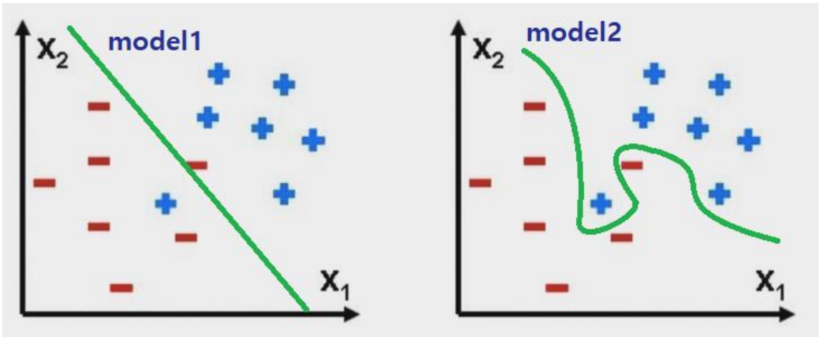
- Our model is very good with the training data set
- Not good at test data set or real use

We call this **Overfitting**.

For example, let’s consider the machine learning problem which can classify the “+” and “-“ in the following graph.

First, we can simply make the linear model as shown on the left graph.

Next, we can make a model so as to classify the training data with 100% accuracy such as the right graph.



Then which model is better? Of course, model1 is better.

Although there is some incorrect classification result, it is a generalized model, so when new data comes, the model can classify perfectly.

Model2 is an overfitting model, so it is good for training data, but when new data comes, the accuracy will be low.

Here are the methods of solutions of overfitting.

- Use more training data
- Reduce the number of features
- Regularization: don’t use too big numbers in the weight.

AIM

The aim of the following lab exercise is to confirm the impact of the learning rate on training and have understoodd how to evaluate the training is going well or not using PyCharm and Tensorflow.

Following steps are required.

Task 1: Preparation of the development environment

Task 2: Confirm the impact of the learning rate

Task 3: Evaluate the training status

LAB EXERCISE 8: EVALUATION OF LEARNING RATE AND TRAINING

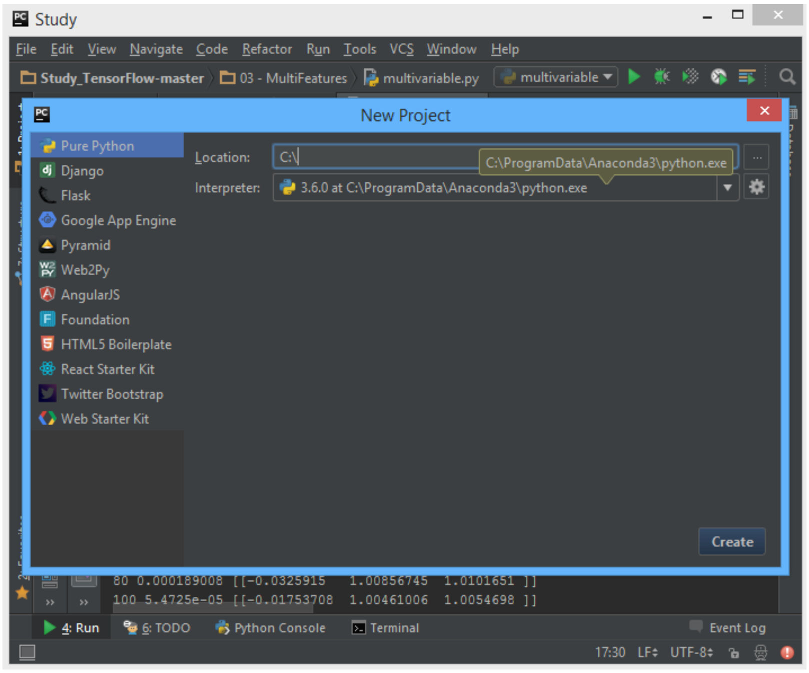
Preparation
Learning Rate
Evaluate training

1. Preparation of the development environment
2. Confirm the impact of the learning rate
3. Evaluate the training status

Task 1: Preparation of the development environment

STEP 1: Run PyCharm and create project

First, run PyCharm and create a new project using File/New Project menu.


Study
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Study_TensorFlow-master 03 - MultiFeatures multivariable.py multivariable
New Project
Pure Python
Django
Flask
Google App Engine
Pyramid
Web2Py
AngularJS
Foundation
HTML5 Boilerplate
React Starter Kit
Twitter Bootstrap
Web Starter Kit
Create
4: Run 6: TODO Python Console Terminal Event Log
17:30 LF UTF-8

Location: C:\

Interpreter: 3.6.0 at C:\ProgramData\Anaconda3\python.exe

C:\ProgramData\Anaconda3\python.exe

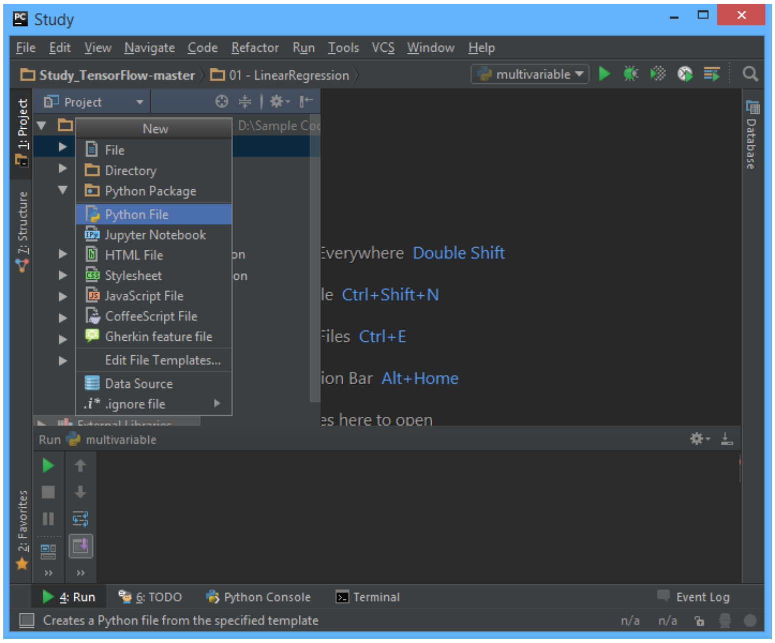
```
80 0.000189008 [[-0.0325915   1.00856745  1.0101651 ]]
100 5.4725e-05 [[-0.01753708  1.00461006  1.0054698 ]]
```

Select the project location and interpreter there and click Create Button. Then the empty project will be created.

STEP 2: Create a python script file

Create a new python file using File/New/Python File.

And give the file name as “Softmax Classification.py”



Task1 is completed.

Task 2: Confirm the impact of learning rate

STEP 3: Prepare the project

Let’s use the multinomial project considered in previous chapter.

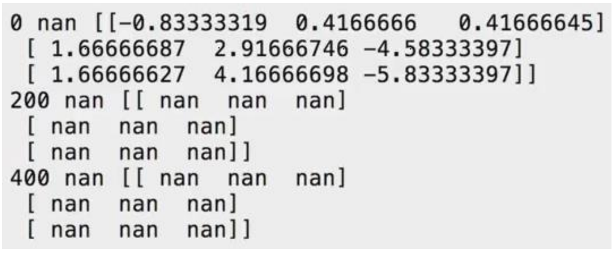
STEP 4: Change the learning rate as big value

Let’s change the learning rate as big value such as 10.

```
learning_rate = 10.
```

STEP 5: Run the code and check the result

When running the code after change the learning rate as big value, then we can see this result. As we can know from the result, the training failed because the learning rate isn’t appropriate.



```
0 nan [[-0.83333319  0.4166666   0.41666645]
 [ 1.66666687  2.91666746 -4.58333397]
 [ 1.66666627  4.16666698 -5.83333397]]
200 nan [[ nan  nan  nan]
 [ nan  nan  nan]
 [ nan  nan  nan]]
400 nan [[ nan  nan  nan]
 [ nan  nan  nan]
 [ nan  nan  nan]]
```

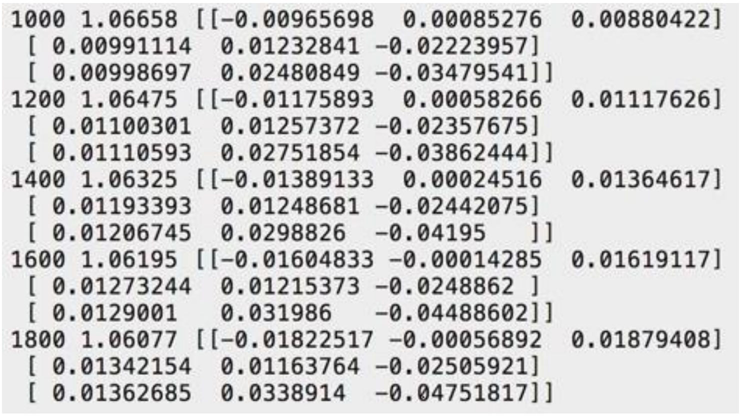
STEP 6: Change the learning rate as small value

Let’s change the learning rate as small value such as 0.0001

```
learning_rate = 0.0001
```

STEP 7: Run the code and check the result

When run the code after change the learning rate as small value, the cost value isn’t decreased even though the step is 2000.



```
1000 1.06658 [[-0.00965698  0.00085276  0.00880422]
 [ 0.00991114  0.01232841 -0.02223957]
 [ 0.00998697  0.02480849 -0.03479541]]
1200 1.06475 [[-0.01175893  0.00058266  0.01117626]
 [ 0.01100301  0.01257372 -0.02357675]
 [ 0.01110593  0.02751854 -0.03862444]]
1400 1.06325 [[-0.01389133  0.00024516  0.01364617]
 [ 0.01193393  0.01248681 -0.02442075]
 [ 0.01206745  0.0298826  -0.04195    ]]
1600 1.06195 [[-0.01604833 -0.00014285  0.01619117]
 [ 0.01273244  0.01215373 -0.0248862 ]
 [ 0.0129001   0.031986   -0.04488602]]
1800 1.06077 [[-0.01822517 -0.00056892  0.01879408]
 [ 0.01342154  0.01163764 -0.02505921]
 [ 0.01362685  0.0338914  -0.04751817]]
```

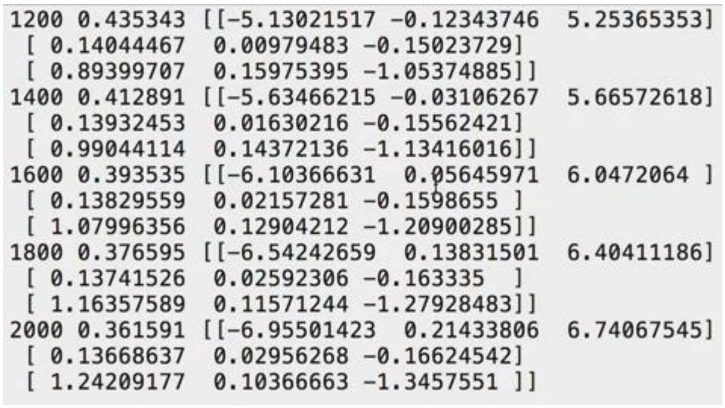
STEP 8: Change the learning rate as medium value

Let’s change the learning rate as medium value such as 0.1

```
learning_rate = 0.1
```

STEP 9: Run the code and check the result

When running the code after change the learning rate as medium value, we can see the cost is reduced successfully.



```
1200 0.435343 [[-5.13021517 -0.12343746  5.25365353]
 [ 0.14044467  0.00979483 -0.15023729]
 [ 0.89399707  0.15975395 -1.05374885]]
1400 0.412891 [[-5.63466215 -0.03106267  5.66572618]
 [ 0.13932453  0.01630216 -0.15562421]
 [ 0.99044114  0.14372136 -1.13416016]]
1600 0.393535 [[-6.10366631  0.05645971  6.0472064 ]
 [ 0.13829559  0.02157281 -0.1598655 ]
 [ 1.07996356  0.12904212 -1.20900285]]
1800 0.376595 [[-6.54242659  0.13831501  6.40411186]
 [ 0.13741526  0.02592306 -0.163335  ]
 [ 1.16357589  0.11571244 -1.27928483]]
2000 0.361591 [[-6.95501423  0.21433806  6.74067545]
 [ 0.13668637  0.02956268 -0.16624542]
 [ 1.24209177  0.10366663 -1.3457551 ]]
```

Task2 is completed.

Task 3: Evaluate the training statue

STEP 10: Prepare the MNIST datasets for training

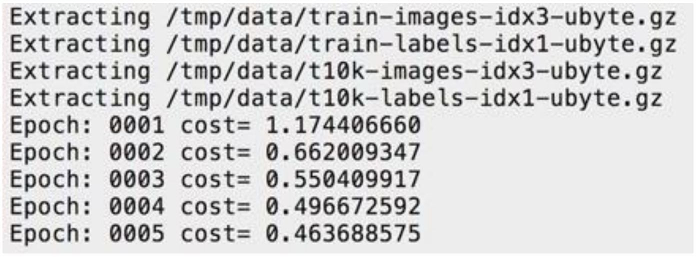
train-images-idx3-ubyte.gz:	training set images
train-labels-idx1-ubyte.gz:	training set labels
t10k-images-idx3-ubyte.gz:	test set images
t10k-labels-idx1-ubyte.gz:	test set labels

STEP 11: Prepare the code

Please use the training.py and input_data.py file in the lab directory.

STEP 12: Run the code

Run the training.py



```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
Epoch: 0001 cost= 1.174406660
Epoch: 0002 cost= 0.662009347
Epoch: 0003 cost= 0.550409917
Epoch: 0004 cost= 0.496672592
Epoch: 0005 cost= 0.463688575
```

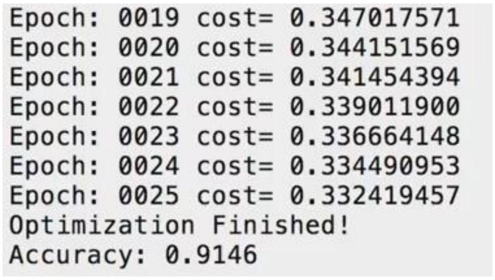
The code extracts the training and test data first and then start training. We can know from the displayed result, the cost is reduced and training is going well.

STEP 13: Evaluate the trained model

After training the trained model can be evaluated with test data set which aren’t attend in the training using these command lines.

```
accuracy =
tf.reduce_mean(tf.cast(correct_prediction,
"float"))
print ("Accuracy:", accuracy.eval({x:
mnist.test.images, y: mnist.test.labels}))
```

The result is displayed as following image.



```
Epoch: 0019 cost= 0.347017571
Epoch: 0020 cost= 0.344151569
Epoch: 0021 cost= 0.341454394
Epoch: 0022 cost= 0.339011900
Epoch: 0023 cost= 0.336664148
Epoch: 0024 cost= 0.334490953
Epoch: 0025 cost= 0.332419457
Optimization Finished!
Accuracy: 0.9146
```

The accuracy of model with test data set is 0.9146.

Task3 is completed

LAB CHALLENGE

Challenge

In the task2, we have changed learning rate with several values and run the softmax project and then check the training status depend on learning rate, and it the task3, we have learned the method to evaluate the trained model with test data set.

Copy the project of chapter6, logistic classification project and try to change the learning rate with several values from 0.0001 to 10, and then run the code and check the training status again. And check the MNIST data set structure, it will be used in the next chapter.

SUMMARY

We should pay attention to the following rules when training the model.

- Proper learning rate
    - Observe the cost function
    - Check it goes down in a reasonable rate
- Date Preprocessing
    - Zero-centered data
    - Normalization
- Overfitting
    - Use more training data
    - Reduce the number of features
    - Regularization: don’t use too big numbers in the weight.

REFERENCES

- https://en.wikipedia.org/wiki/learning rate
- https://en.wikipedia.org/wiki/TensorFlow
- https://en.wikipedia.org/wiki/overfitting

INDEX