Machine Learning Project1

Yunke Zhu

1330327

Part One: Logistic Regression for Digit Classification

1.1 Compare the score and log loss with different iterations

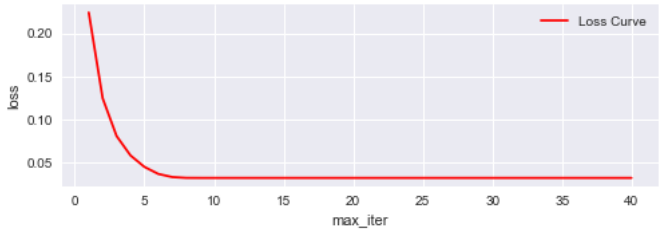

Figure1: Log Loss

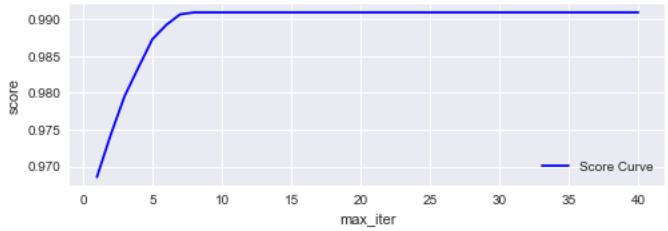

Figure2: Accuracy

From the two figures above we can see that, the accuracy line(figure 2) increases as max_iter increase; the log loss line(figure1) decreases as max_iter increase. But both become flat after about 7-8 times of iterations. We can conclude that, the amount of iterations can help us improve the accuracy of our model and decrease the loss. But after several times of iterations, we don't have to do more iterations since our model already becomes stable. And for some large models, the number of iterations affect a lot to the time complexity. We can find an optimal iterations which can build the model accurately and not that time consuming.

1.2 Plot the trends of the weight on pixel 000

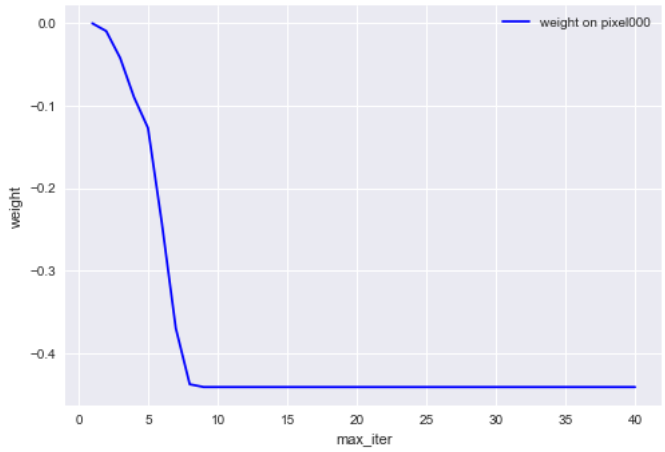

Figure 3: Weight (pixel000)

We already collect the weight of piexl000 when building the model. Basically, pixel000 can be treated as the first feature from our dataset, and the plot above shows how its weight changes as max_iter increases. We can see that quite similar to loss and accuracy plots, after 7-8 iterations, the line becomes flat, which means this feature adjust itself to a relatively stable value after 7-8 iterations. And the final weight of pixel000 is around -0.45, which shows how these features dominate the model.

1.3 Find the Best Penalty Strength C

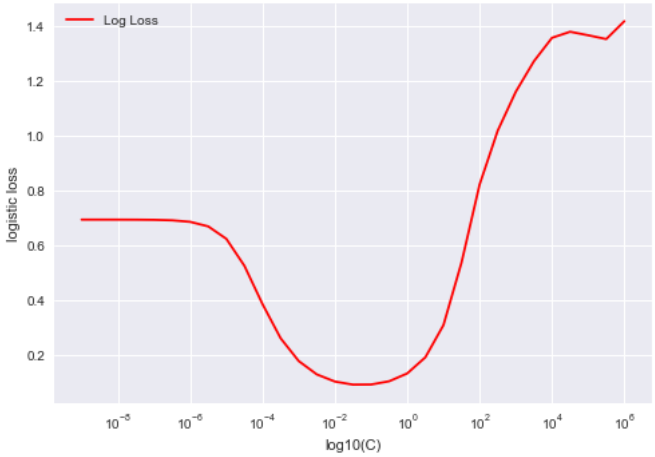

Figure 4: log loss under best C value

Predicted	0	1
True		
0	942	32
1	33	976

Figure 5: Confusion Matrix

Best C-value for LR data: 0.0316227
Validation set log-loss at best C-value: 0.0897
The accuracy under the best C value: 0.8174

From the confusion matrix, we can know that the amount of

True Positive is: 976;
True Negative: 942;
False Positive: 32;
False Negative: 33.

1.4 Plot False Cases

Figure 6: False Positive

Figure 7: False Negative

False negative cases stands for those are 9 but mis-treat them as 8 while false positive cases stands for those are 8 but mis-treat them as 9. From these mis-treated cases, we can conclude that they are surely hard to recognize. This is due to the occurrence that the lower half of 9 are treated as a part of 8 lower half. The main difference between number 8 and number 9 is basically the lower half. If there is a ring, then it is 8 otherwise 9. But for these wrong cases, the similarities is the confusing lower half.

1.5

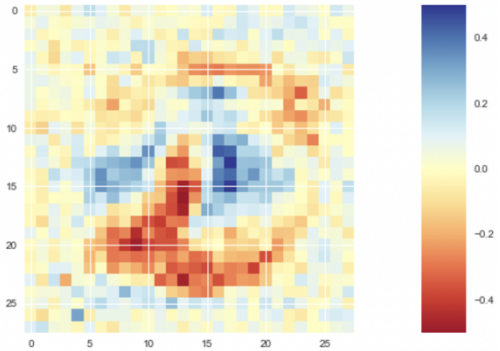

Figure 8: Weight Coefficients

Firstly, we can identify that the yellow color stands for those pixels are irrelevant to the results since their weight are close to 0. The main part that we need take into consideration is the blue and red parts where has most absolute weight which have a greater impact on the final results.

As we discussed in 1.4, the most important way to distinguish 8 and 9 is the lower half of the digit. From Figure 8 above we can identify this, the red part, especially left bottom corner are composed of red pixels, which means the 1 value in this area will have negative impact when judging if it is 9. This makes sense since digit 9 doesn't need the left corner. From this figure we can say that our model does quite good job on distinguishing digit 8 and 9.

Part Two: Sneakers versus Sandals

1.1 Step One: Upload Data

Read data from csv file eliminate the first header line. Then try to plot the figure in the data. Since the image is composed of 28 * 28, we transform the 784 pixels into a 28 * 28 matrix then use imshow() to show the image. List two image of sneakers and sandals respectively. And from the output we know that 0 represents for sneakers and 1 represents for sandals. Then we show two images which represent sneaker and sandal respectively.

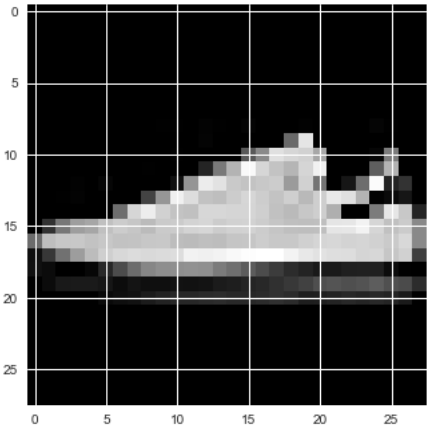

Figure 9: sneaker

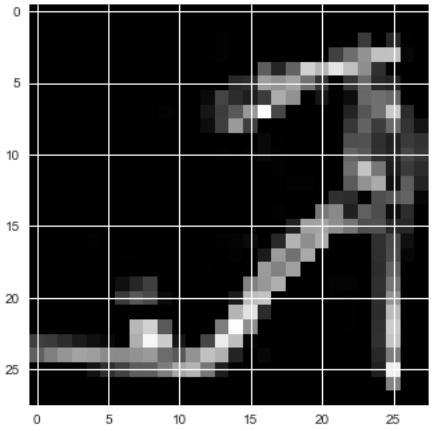

Figure 10: sandal

1.2 Step Two: Build the Basic Model

First of all, we use the training data to build the basic model without adding any other parameters. The basic model we used is logistic regression model from sklearn library. Notice that this is the most normal model, and let's see how the result performs.

After submitting the first edition result from the basic model, we get the error rate as **4.25%**, which is an acceptable error compared to part one's best accuracy. But since we have so many data unlike the first part, we need to adjust the model by applying more features and parameters. Here come three ideas:

1. Using more features in higher degree just like what we did in linear regression and find the best degree model.

2. Find the optimal penalty strength value: C and use the logistic regression model with the best C value.

3. Using more features but not in a polynomial way, we can add more features for example, the number of black pixels/ white pixels...

But before we implement each method, we need a way to check the feasibility of our model instead of submitting the result on gradescope. Among all the method we've learned, we can take the advantage of cross-validation to check the accuracy, log loss... So, let's separate our data firstly. For this project, I will use the number of folds as 10.

1.3 Tryout of adding Polynomial Features

For each pixel, I created the new feature in higher dimension. Firstly, I use degree 2 for a tryout. The total feature changes from 784 to 1568. But after I apply these new features, my model performs worse. And when I hesitate if it was the problem of overfitting problem in degree 2, I found that this approach is exactly useless.

The reason is that all the pixels are composed of either 0 or 1, which means they are useless in higher dimension. Since $0^k = 0$ and $1^k = 1$ this new adding feature won't help us improve the model.

1.4 Best Penalty Strength Value C

Previously, we take the advantage of finding the best penalty strength value C and then build our model. Then how about this case?

After applying the C grid ranging from 10^{-9} to 10^6, the best given C value with least log loss we found is 10^5, then I apply C trying to improve the performance of the new model. But after I submitting the results, the error rate = **0.056** is extremely higher than the basic model with C value as default 1.0 with error rate = **0.0425.** Then I use the cross validation to check how the roc curves look like between the original penalty and best penalty, which shows as below:

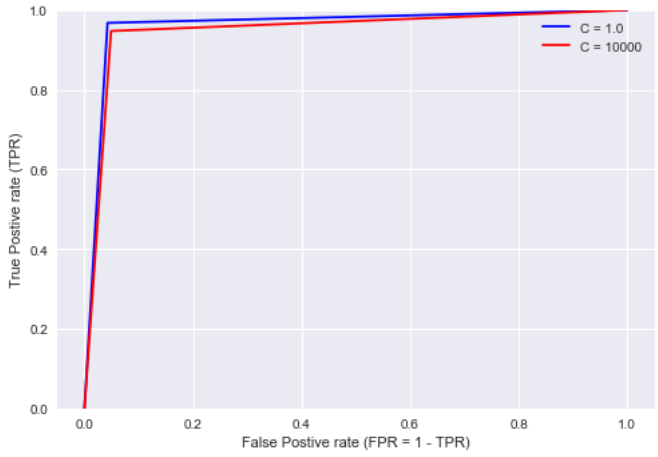

Figure 11: ROC Curve

From the roc curve we notice that the original curve has a greater area which means the performance of the original penalty is better. The possible reason that makes this failure may probably due to an overfitting problem since we only detect the results on the training dataset. So, an improve approach is to check the

best C value in each cross-validation dataset from 10 folds. This step is time consuming, but after I generate a the new 10 folds with different best C-value and accuracy, I found that the best penalty for each folds differs from 1.0 to 0.31622777. Since I have already tested the default penalty with 1.0, I apply with the new penalty as 0.3162.

The model with the new penalty value performs very well than before with a smaller error rate: **0.03850** compared to the original error rate. And let's check how the roc curve looks like this time.

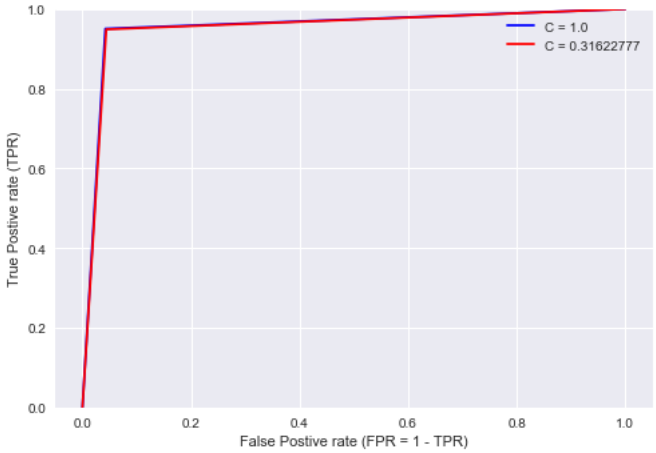

Figure 12: ROC Curve

Predicted	0	1
True		
0	582	30
1	31	557

Figure 13: Confusion Matrix with C = 1.0

Predicted	0	1
True		
0	586	26
1	19	569

Figure 14: Confusion Matrix with C = 10000

Although the two line with different penalty looks similar, but this also proves that new penalty value C = 0.3162277 works better then C = 10000. This approach, however, can be improved with a better penalty. Notice that our original range of penalty is quite huge, we can narrow down the range from $[10^{-9}, 10^{6}]$ to [0, 2*C] which means we can find a more accurate optimal penalty in a small range.

The best C value is 0.57 after adjustment, and the error rate turns out to be **0.03847** which is a small progress.

1.5 How about Adding New Features

1.5.1 Adding the total number of black/white pixels as a new feature.

As we discussed we discussed before, our dataset contains 12,000 data but only with 784 pixels. The amount of features is too small, not to say all of them are only pixels. So, an intuition is to build the model with more effective features. One of the possible features we can add is the total amount of all the black/white pixels. This feature can help us improve the model for sure since the amount of black/white pixels can stands for how much sandal/sneaker occupy the total image. With the help of this new feature, we can decrease the error rate from 0.0425 to **0.0420**. And by applying the best penalty value, the error rate becomes 0.418766. So, adding this feature is a small improvement compared to original 784 pixel features.

1.5.2 Adding the adjacent pixel's feature.

Another approach to adding more features is to observe the pixels around one pixel and check if they are the same. Broadly speaking, this helps us check how the pixels change from one to the other. With the help of adding this feature, we can know how each pixel gradually changed. Take the figure below as an example, the pixels composed of a sandal is more condensed while a sneaker is sparse. I build two models where the first model adds 784 * 4 features detecting if the 4 adjacent pixels have the same value as the middle pixel. And the second model adds 784 * 8 features for 8 adjacent pixels.

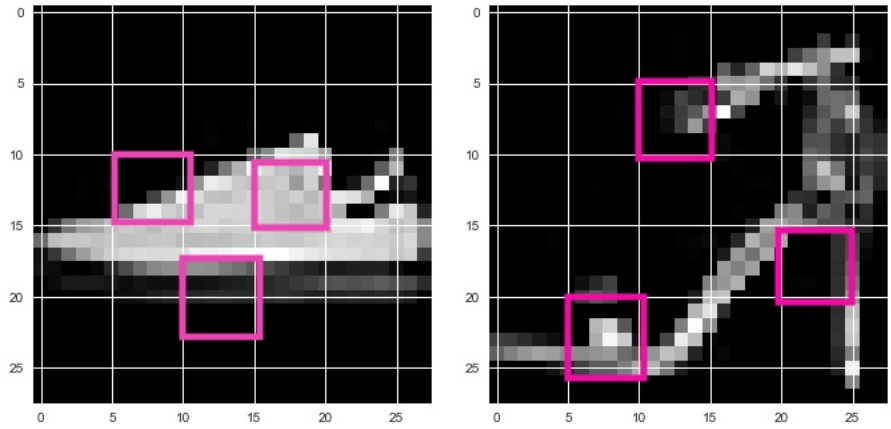

Figure 15: Adding More Features

The performance of the two models works surprisingly well. The first model gives the error rate: **0.0220**, and with the help of the optimal penalty C, the error rate decreases to **0.0200**. And the second model with the most optimal penalty C gives the error rate: **0.01850.**

Then we plot the ROC curve between the origin logistic regression model and the adjusted model by adding more useful features.

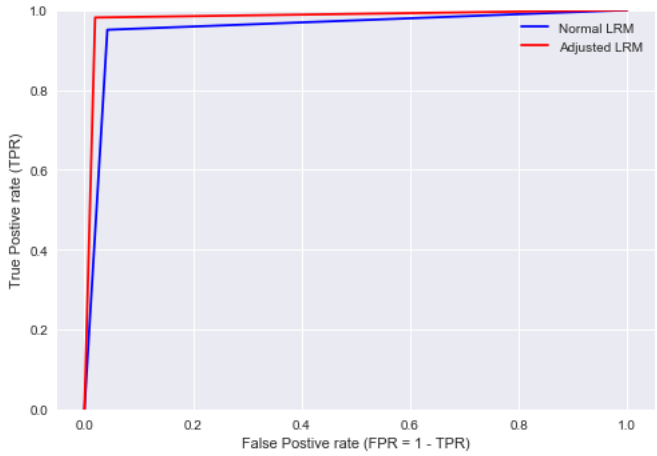

Figure 16: ROC Curve Between Normal Model and Adjusted Model

Predicted	0	1
True		
0	600	12
1	11	577

Figure 17: Confusion Matrix Given from my Best Model

From the plot, we can see that the red curve which stands for our best model performs much better than the original model, since the area of the red line is bigger than the blue line. And from the confusion matrix, we can observe 12 False Positive and 11 False Negative cases. The accuracy is acceptable.

The final Error Rate and AUROC is shown as below:

7	Pokemon Master	0.01849999999999996	0.9815

Figure 15: Final Error Rate and Auroc

1.6 What's More

Besides the models I made above, I also use the support vector machines and decision tree method building the model. The process is super super super time consuming, which takes more than 10 hours for each with 784 original features. But even more frustrating is that the result is worse than all the model before. After I did some research from Google, I've learned that both approaches are not suitable to be applied in image classification. They normally take small features and rise to higher dimension doing the computation, and 784 features is super big for either models which will lead to an overfitting problem in a sense.

1.7 Conclusion and Further Research

Conclusion:

1. Among all the changes I've made for this logistic regression model, adding more meaningful features greatly helps improve the accuracy of the prediction and reduced the error by half from the original model. Choosing the optimal penalty helps as well, but the progress is not that big compared to adding features. And more importantly, some given penalty not performs well if take the whole dataset as the training model, we need separate the data to find the optimal penalty strength value, for example using cross-validation.
2. The second point I've learned from the project is that we can separate the training data, partly for validation to check the accuracy/log loss of our model.

Further Research:

1. In my model, I just checked the adjacent 8 pixels to see how they gradually changed, in future research if I want a higher accuracy, I can observe more pixels, and this is also known as **Convolutional neural network**
2. Although my error rate is acceptable compared to the average, but my AUROC is abnormal shown above. And I still don't know why this happens. So, improve my AUROC can be counted as a new research in the future.
3. For my final model, I can list all the wrong cases and check what and why my model goes wrong, and then adjust the corresponding weight to generate a more well-trained model.