Project 01 Jiawei Wang 3/8/2020

Part One: Logistic Regression for Digit Classification

1.1 Iterations vs Accuracy and Log Loss

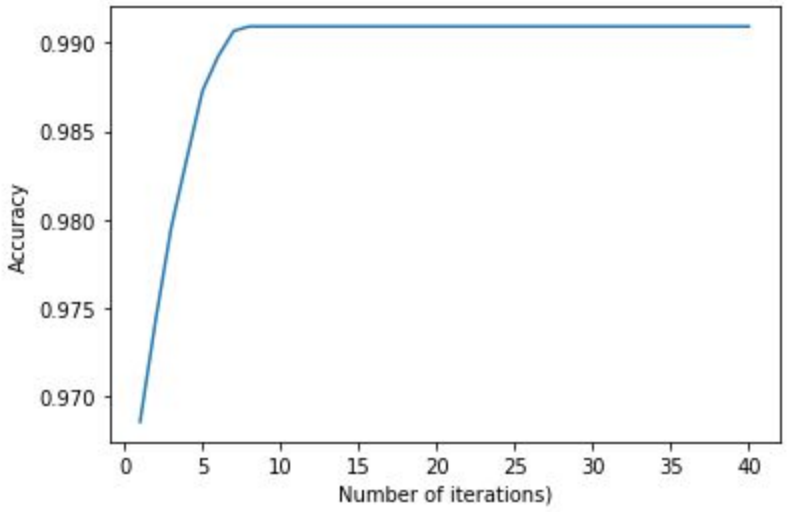


Figure 1: Number of Iterations vs Accuracy

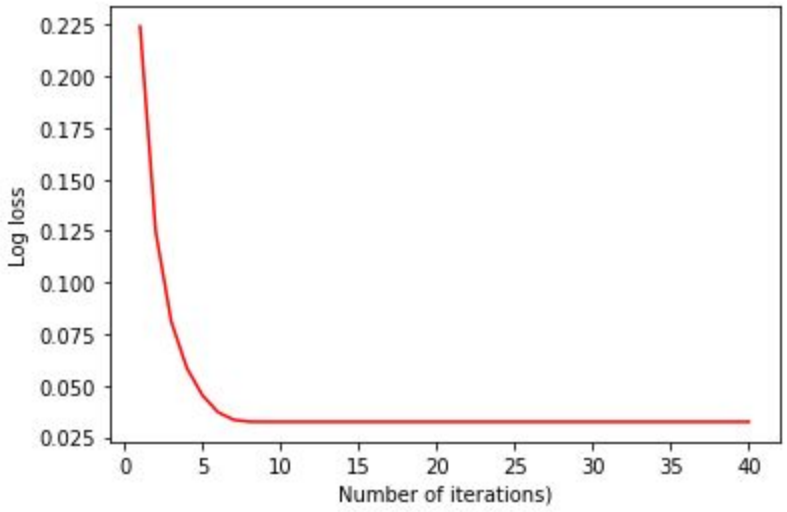


Figure 2: Number of iterations vs Logistic Loss

In the first plot, we can see that the accuracy starts from below 0.970 at the first round of iteration, and it increases as the number of iterations increases. When the number of iterations

goes up between 5 and 10, the accuracy increases to over 0.990 and remains the same as the number of iterations continues to increase. As indicated by the plot, future iterations do not help to increase the accuracy because the weight no longer changes at this level and the model stays the same. This is helpful when we want to decide when to stop iteration so that potential overfitting can be avoided.

In the second plot, the Log Loss decreases from the first round of iteration until the number of iterations goes up to between 5 and 10 and remains the same in future iterations. This plot is very similar to the first one in the way that when the weight stops changing, the logistic loss of the model stops decreasing and we can use it to decide to stop iteration in early phases.

1.2 Iterations vs Weight

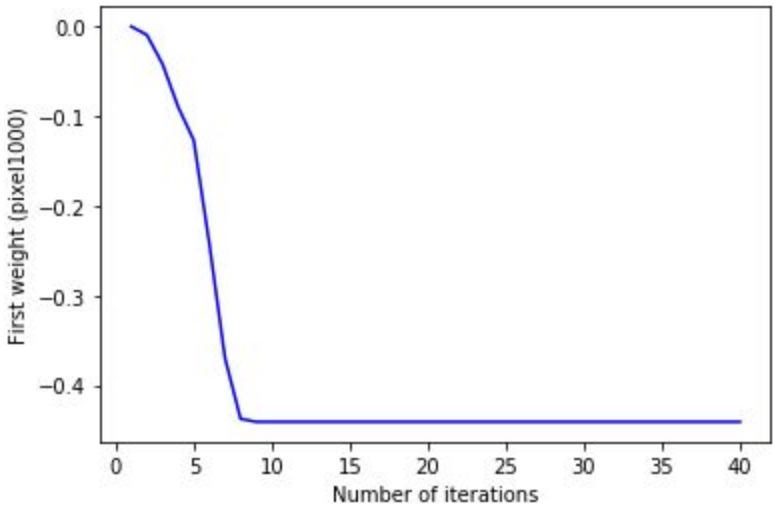


Figure 3: Number of Iterations vs First weight

In the plot, the number of iteration is plotted against first weight (pixel1000). One thing that needs attention is that the first weight of the first iteration starts from 0 and it is the initialization value. It starts to decrease as the number of iterations goes up and ends at below -0.4. This line corresponds to the convergence reached by the iterative coordinate descent algorithm described previously. After around 7 to 8 iterations, the first weight stops to decrease and remains the same. Same as the previous two plots.

1.3 Inverse Penalty and the Best Model

```
Best C-value for LR with 2-feature data: 0.032
The minimum log loss is 0.08968955643903709.
The corresponding inverse penalty value is 0.03162277660168379.
The corresponding accuracy value is 0.9808474576271187.
Predicted    0    1
True
0          942   32
1           33  976
```

Figure 4: Minimum log loss

		Predicted Output labels		
		0	1	Total
Output labels	0	942	32	974
	1	33	976	1009
	Total	975	1008	1983

Figure 5: Output labels

In this question, different C values of inverse penalty are tested on the logistic regression model with the liblinear solver used in previous questions. The gridspace is defined by np.logspace(-9, 6, 31). The minimum log loss, along with the best accuracy value are shown above. The confusion matrix for the best model is also shown.

1.4 Sample image of False Positives and False Negatives based on best model

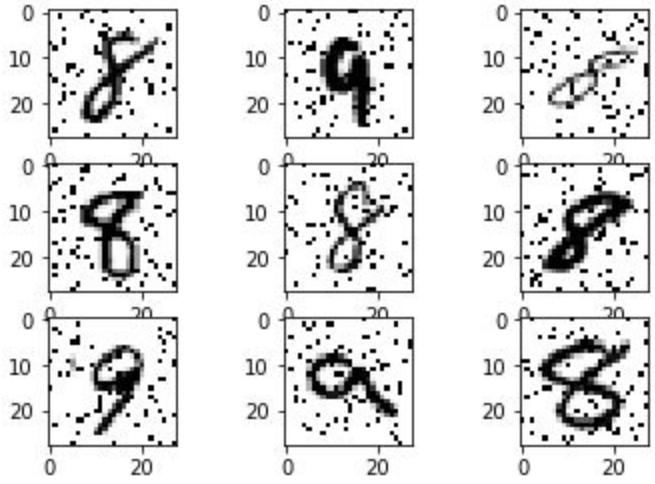


Figure 6: 9 sample images of False Positives, the productions made by the model are all “9”

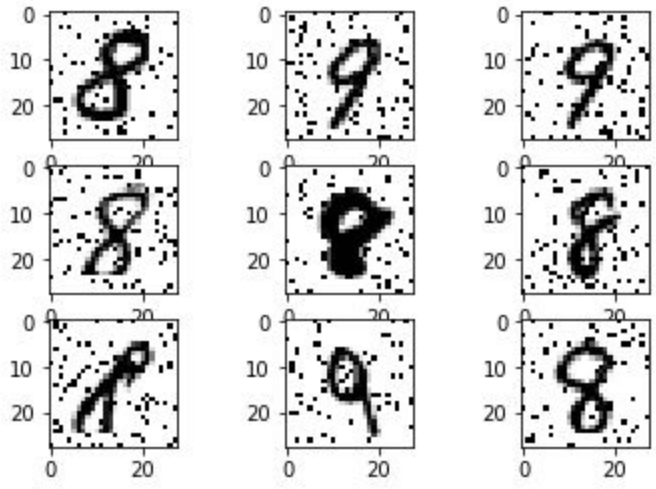


Figure 7: 9 sample images of False Negatives, the productions made by the model are all “8”

From these two 9 image sets, we can easily tell that the model makes false predictions on some of the input datas. In the first set, six “8”s are mistakenly labelled as “9” and in the second set, three “9”s are mistakenly labelled as “8”. This problem is mainly because of the threshold LogReg uses on datasets. It is set to default as 0.5 as it is possibly not the best value for this function. In this model, the classifier decides whether an image is 8 or 9 by checking the possibility is greater than or less than 0.5 which in turn generates false positives and false negatives.

1.5 weights produced by Logistic Regression with best Inverse Penalty C on a 28*28 grid

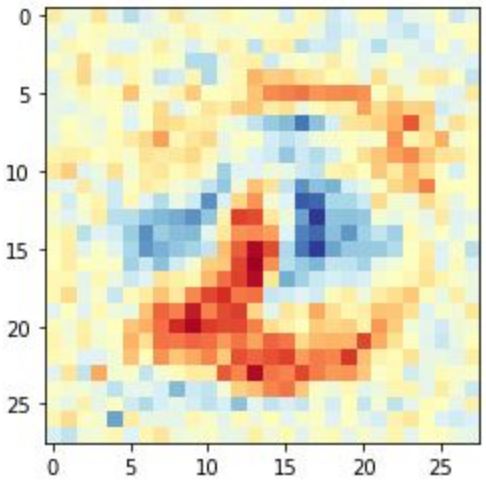
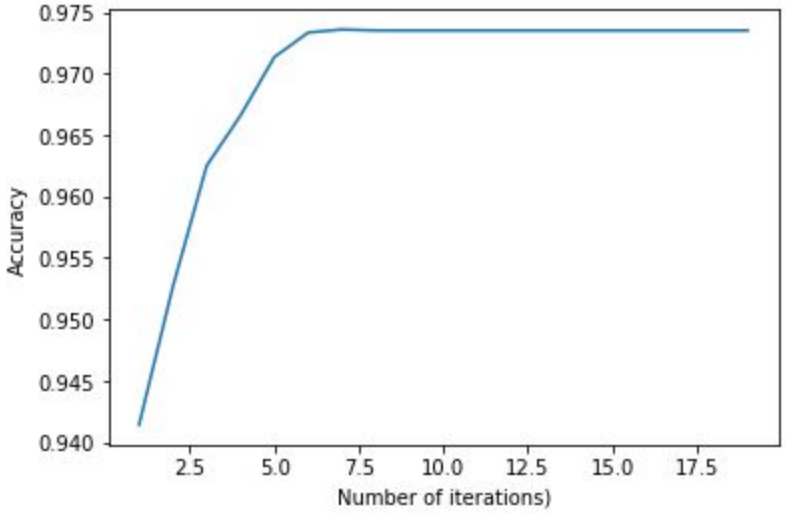


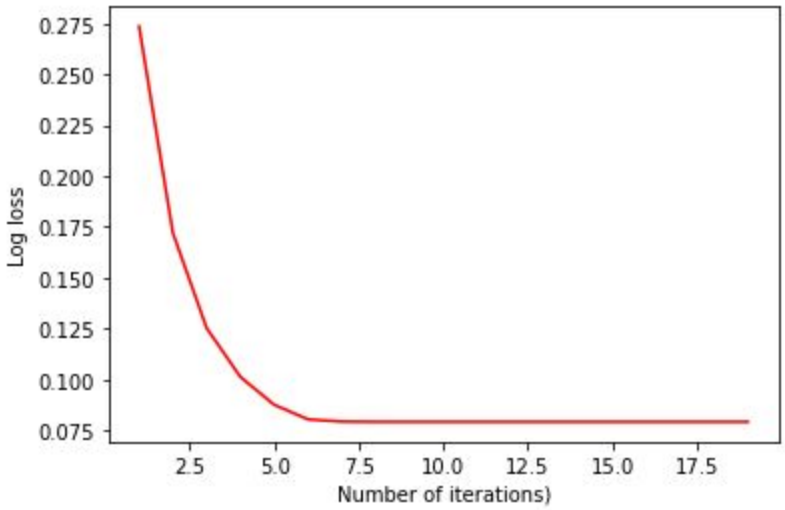
Figure 8: produced by Logistic Regression with best Inverse Penalty C on a 28*28 grid using imshow() with RdYlBu colormap

On this image, pixels related to 8 have negative weights and are colored as red and pixels related to 9 have positive weights and are colored as blue. The density of colors represents the "possibility" of being either 8 or 9. And for those pixels with color neither red nor blue, they are the empty pixels in the original dataset (as pure white pixel) and do not give weights to the prediction. And we can see that pixels with deeper red form a roughly shape of "8" while pixels with deeper blue form a roughly shape of "9". And the pixels that commonly occur in both numbers tend to stay neutral (light yellow).

Part Two: Sneakers vs Sandals

2.1 Iterations vs Accuracy and Log Loss





First, I want to see the performance of my model when it goes through a certain number of iterations so I plot the accuracy and logistic loss against iterations using my model. And I got these two plots. As shown in the first plot, the accuracy starts to climb up from first iteration at around 0.942 and goes up until it reaches around 0.974 at 6th iteration. Same thing happens to logistic loss where the loss starts to decrease from the first iteration at around 0.275 and ends at the 6th iteration at around 0.080. These show that number of iterations does have a positive effect in improving the accuracy and reducing the log loss when the number is small. However as indicated by the plots, future iterations do not help future improve the model since the weight no longer changes after several rounds of iteration and the model stays the same, and at this point we should consider to stop the iteration and prevent the potential overfitting.

2.2 Logistic Regression Model

ERROR_RATE	AUROC
0.040000000000000036	0.992858

For this question, the original logistic regression from sklearn is used with a liblinear solver and the predictions are submitted to gradescope. As shown above, the error rate is above 0.040 and the AUROC is below 0.993.

2.3 Logistic Regression Model with Gaussian RBF

ERROR_RATE	AUROC
0.04400000000000004	0.992173

For this question, the dataset is reshaped into 2D NumPy arrays. By importing gaussian_filter() from scipy, the 28*28 arrays are turned to 1D array once again. By doing this, the data is normalized to remove outliers. The datas are then added to the input dataset and make the data twice as big as the original one. The result from gradescope is shown above, with error rate at over 0.044 and AUROC under 0.993, which indicates that applying Gaussian RBF results in worse performance than Logistic Regression without any data augmentation.

2.4 Logistic Regression Model with Splitting Train and Test Sets (2:1)

ERROR_RATE	AUROC
0.04249999999999998	0.9931110000000001

For this question, the dataset is divided into two parts: Training set and Testing set and this is achieved by using train_test_split(). And 2 / 3 of the original data is used as the new training set while 1 / 3 of it is used as testing set. By doing this I want to reduce the risk of overfitting. And the result is shown above with error rate at over 0.042 and AUROC under 0.994, which is better than using Gaussian RBF and also better than the original Logistic Regression Model. And this means that overfitting is actually avoided by doing this (or I am just lucky with a good splitting). So I still need one more function to do.

2.5 Logistic Regression Model with K-Fold Cross Validation

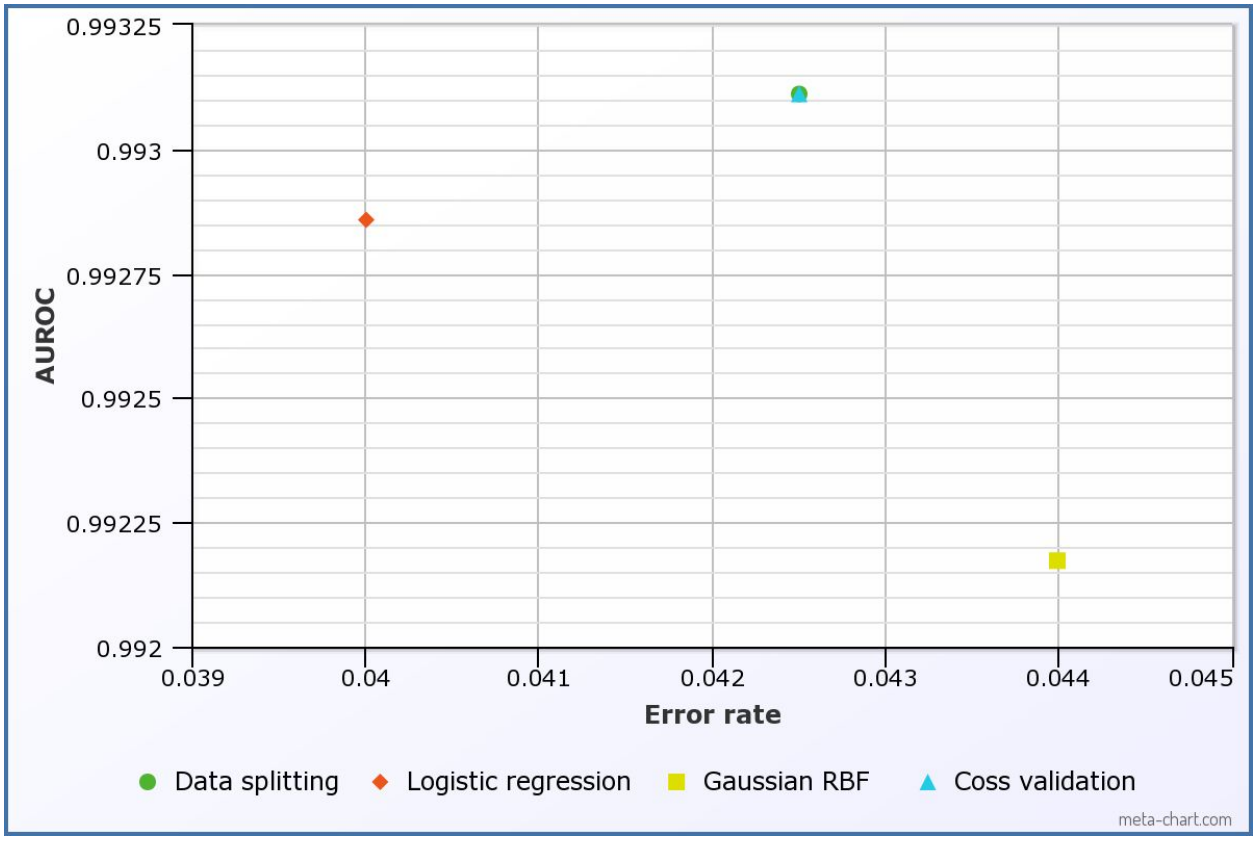
ERROR_RATE	AUROC
0.04249999999999998	0.9931110000000001

By using Cross Validation I want to once again make sure I did the right thing in the last question rather than just pure luck. So I select k = 5 and use the cross validation method in the previous homework to test my dataset and the result from gradescope is shown above. It is the same as using train_test_split() and better than Gaussian RBF and original logistic regression.

2.6 Conclusion

In part two, four methods are used on the dataset and as described above, the best methods are data splitting and cross validation. The first method: pure logistic regression without any modification can be used as the standard for other methods and it already shows a not-too-bad accuracy. The Gaussian methods actually goes against my expectation because I thought it would improve upon the original one. However after some thought, I think standardization is not guaranteed to improve the classification performance. In contrast, dataset is forced to change upon standardization and this may cause worse performance, and I find more discussion on this issue:
https://www.researchgate.net/post/Worse_SVM_results_when_standardizing_data_than_without.
And the last two methods I used are dataset splitting and cross validation and both of them give me the best performance. By doing these, I reduced the risk of overfitting and improved upon the original one. And AUROC = 0.993 is by far the best I can get. If the accuracy needs future improvement, other methods are needed.

The results from gradescope are plotted below.


0.99325
0.993
0.99275
0.9925
0.99225
0.992
AUROC
0.039
0.04
0.041
0.042
0.043
0.044
0.045
Error rate
Data splitting
Logistic regression
Gaussian RBF
Coss validation
meta-chart.com