Introduction to Machine Learning

Project 3

Group 14

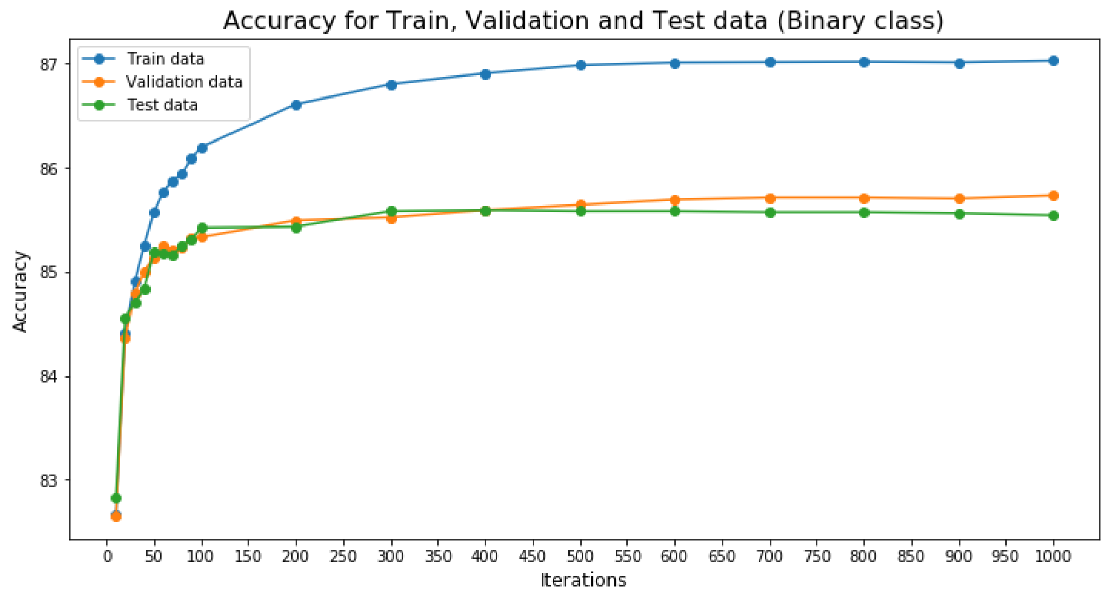
5/10/2018

Ali Alshehri
Ratnamala Korlepara

REPORT 3

- Logistic Regression

Our one-vs-all model best accuray values are 86.19%, 85.33%, and 85.42% on training, validation, and test data respectively. We ran the model with 1000 iterations. As shown in the figure below, the accuracy on training data, validation, and test data fairly levels around 500 iterations. Thus, it seems that 500 is our optimal number of iterations here.



The confusion matrix for each class on training and test data in figure 1 shows that there parallel performance across training and test data (also refer to Table 1 for precision and recall results). First, in both datasets, the model performs best on classes ‘0’, ‘1’ ‘6’ and ‘7’. Worst performance on the other hand is obtained on class ‘5’ and even worse ‘8’. The model seems to confuse 8 with many digits especially 2, 3, 5 and 9, thus the low recall value. It does better at predicting the true 8 class however. This suggests that 8 shares a lot of features with other classes making. This means that the model overfits on learning the class ‘8’. The same is true with respect to the test data. Generally, the higher overall training accuracy is expected because our models overfits the data.

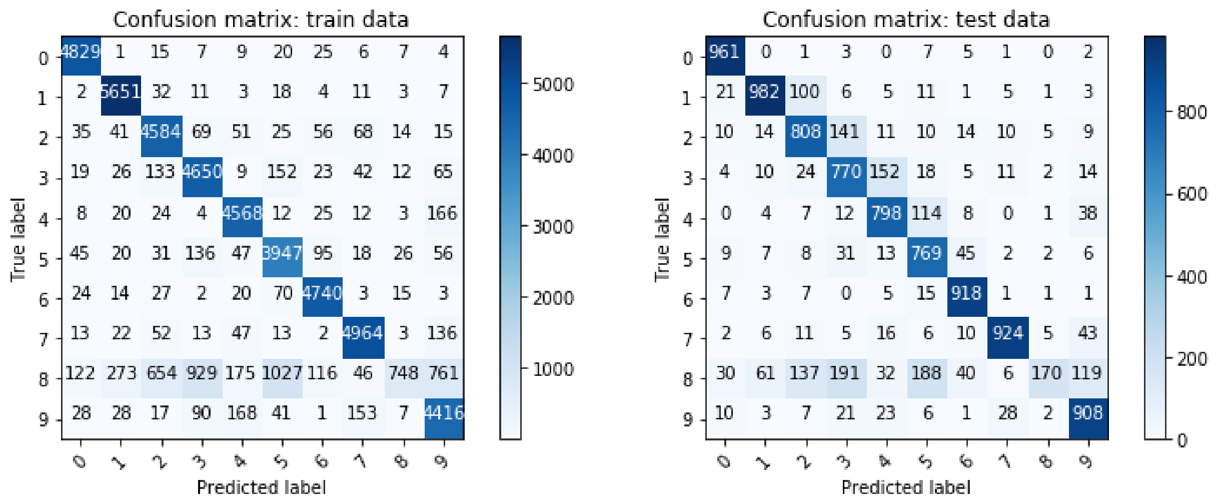


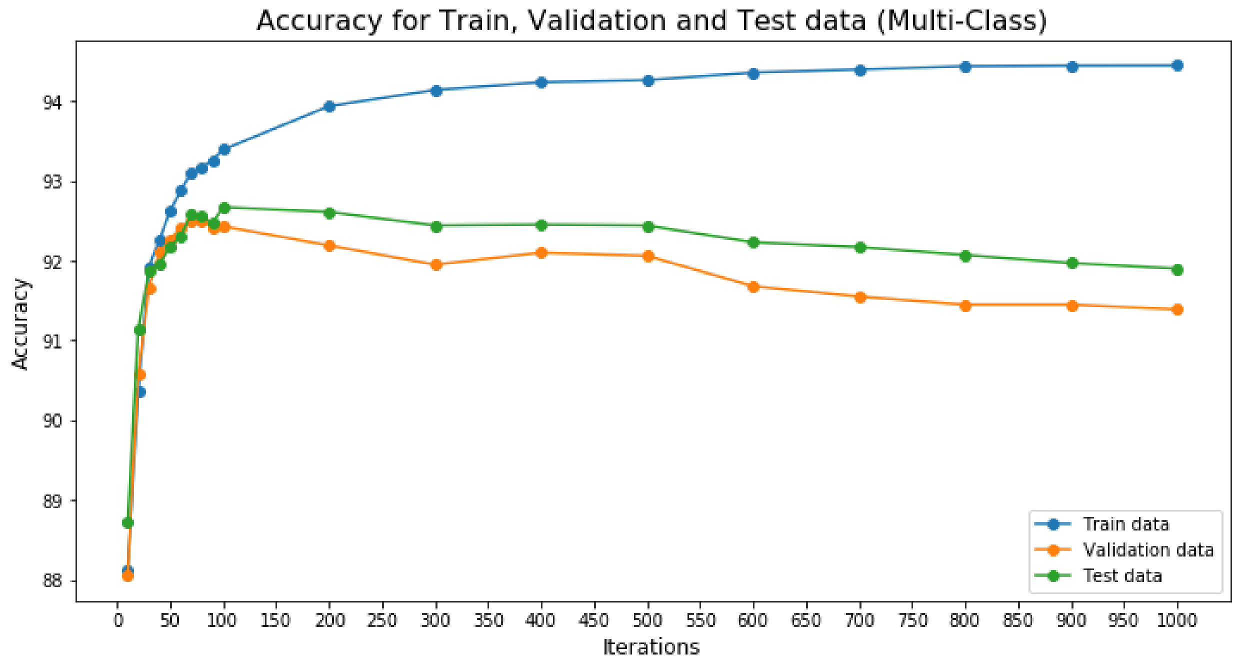
Figure 1: Confusion matrix on training (left) and test (right) data of the binary model.

Class	**Train data**		**Test data**	
	Precision	**Recall**	**Precision**	**Recall**
0	0.94	0.98	0.94	0.98
1	0.93	0.98	0.93	0.87
2	0.82	0.92	0.82	0.78
3	0.79	0.91	0.79	0.76
4	0.90	0.94	0.90	0.81
5	0.74	0.89	0.74	0.86
6	0.93	0.96	0.93	0.96
7	0.93	0.94	0.93	0.90
8	0.89	0.15	0.89	0.17
9	0.78	0.89	0.78	0.90

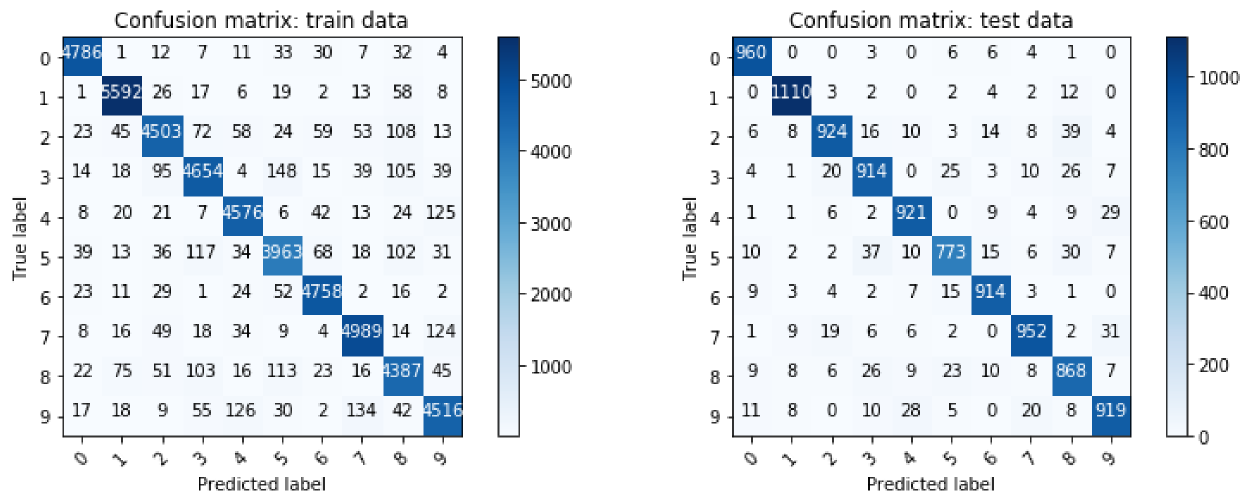
Table 1: Precision and recall wrt each class in the binary model

- **Multi-class Logistic Regression**

Our multi-class model performs better than the binary class above. Specifically, the model best accuracy values are 93.39, 92.43 and 92.67 on training, validation, and test data respectively. We ran the model with 1000 iterations. As shown in the figure below, the model seems to start to over fits after about 100 iterations. Thus, it seems that 100 is our optimal number of iterations here. Comparing these results to the binary model above, the multi-class model is better in terms of performance as well as training time.



The two figures below show the confusion matrix for each class in the training and test data. Comparing the two figures in Figure 2 below, we can see that there are comparable similarities between the performance on the training and test data. For example, it seems that the models does fairly well in terms of precision and recall on ‘0’, ‘1’, and ‘6’ classes in both training and test data (refer to table 2 for precision and recall score). The worst performance in both datasets occur with the ‘5’ class. The model confused 5 with 3 and 8 mostly. That make sense as three and 8 are very similar to 5 especially if the circles in 8 are not completely closed. Generally, the higher overall training accuracy is expected because our models overfits the data.



True label \ Predicted label (train)	0	1	2	3	4	5	6	7	8	9
0	4786	1	12	7	11	33	30	7	32	4
1	1	5592	26	17	6	19	2	13	58	8
2	23	45	4503	72	58	24	59	53	108	13
3	14	18	95	4654	4	148	15	39	105	39
4	8	20	21	7	4576	6	42	13	24	125
5	39	13	36	117	34	3963	68	18	102	31
6	23	11	29	1	24	52	4758	2	16	2
7	8	16	49	18	34	9	4	4989	14	124
8	22	75	51	103	16	113	23	16	4387	45
9	17	18	9	55	126	30	2	134	42	4516

True label \ Predicted label (test)	0	1	2	3	4	5	6	7	8	9
0	960	0	0	3	0	6	6	4	1	0
1	0	1110	3	2	0	2	4	2	12	0
2	6	8	924	16	10	3	14	8	39	4
3	4	1	20	914	0	25	3	10	26	7
4	1	1	6	2	921	0	9	4	9	29
5	10	2	2	37	10	773	15	6	30	7
6	9	3	4	2	7	15	914	3	1	0
7	1	9	19	6	6	2	0	952	2	31
8	9	8	6	26	9	23	10	8	868	7
9	11	8	0	10	28	5	0	20	8	919

Figure 2: Confusion matrix on training (left) and test (right) data of the multi-class model.

Class	**Train data**		**Test data**	
	Precision	**Recall**	**Precision**	**Recall**
0	0.97	0.97	0.95	0.98
1	0.96	0.97	0.97	0.98
2	0.93	0.91	0.94	0.90
3	0.92	0.91	0.90	0.90
4	0.94	0.95	0.93	0.94
5	0.90	0.90	0.91	0.87
6	0.95	0.97	0.94	0.95
7	0.94	0.95	0.94	0.93
8	0.90	0.90	0.87	0.89
9	0.92	0.91	0.92	0.91

Table 2: Precision and recall wrt each class in the multi-class model

To iterate, the above results show that the multi-class model performance better than the one-vs-all (binary) model. The multiclass is more suitable to our multiclass dataset. Even though that the multi-class model is slower than the binary class model, it took less iterations to obtain high performance. On the other hand, the one-vs-all model might be faster to converge, but not as good with our 10-class dataset.

- **Support Vector Machines**

  - **SVM (linear kernel)**

In this part we ran four SVM models using the 'sklearn' library. The first is an SVM model with a linear kernel; all other parameters are kept default as follows:
svm.SVC(kernel='linear').
The table below shows the accuracy of this model on training, validation and test data.

Data	**Accuracy**
Training data	**97.3%**
Validation data	**93.6%**
Test data	**93.8%**

The difference between this linear kernel and radial basis models below is that the linear kernel learns a linear decision boundary in the original space of the features. On the other hand, radial basis SVM learns a linear decision boundary in a transformed infinite dimensional space.

- **SVM (radial basis with gamma = 1.0)**

In the second SVM model, we used ‘sklearn’ library. We used gamma = 1.0 with all other parameters kept default as follows:
svm.SVC(gamma=1.0).
The table below shows the accuracy of this model on training, validation and test data. This model does poorly by overfitting the data because of the large value of gamma, which makes the radius of the area of influence of the support vectors too small.

Data	**Accuracy**
Training data	**100%**
Validation data	**15.5%**
Test data	**17.1%**

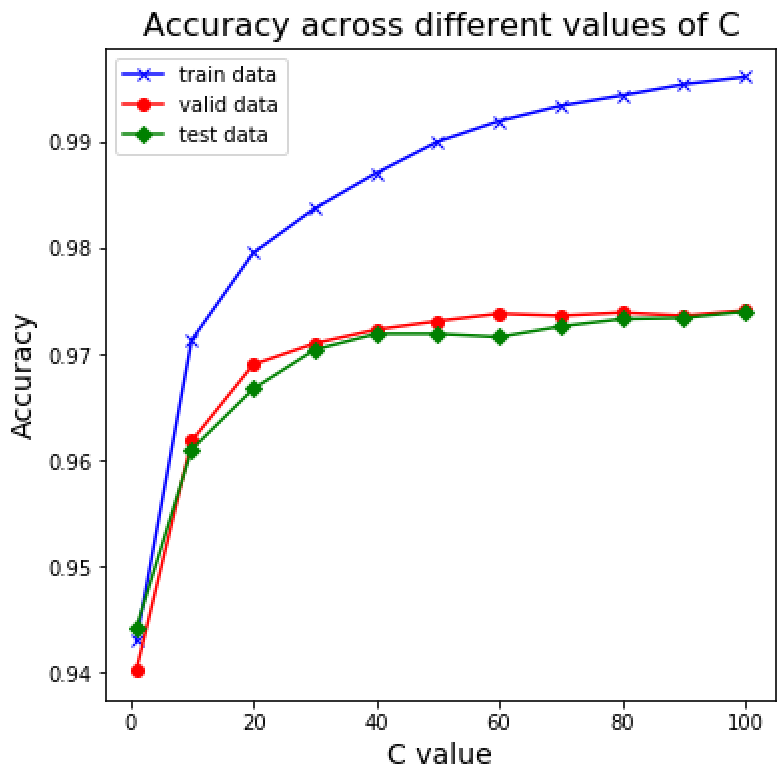
- **SVM (with radial basis of default gamma)**

In the third SVM model, we also used ‘sklearn’ library. We used a default value for gamma as well as a default value for all the other parameters:
svm.SVC(). The default gamma is very small: 1/n_features. The table below shows the accuracy of this model on training, validation and test data. Thus, compared to the gamma equal 1 above, smaller value of gamma as the one used here capture enough complexity to perform as good as 94%.

Data	**Accuracy**
Training data	**94.3%**
Validation data	**94%**
Test data	**94.4%**

- **SVM (radial basis of default gamma and C ranges between 1 and 100)**

In the fourth SVM model, we also used ‘sklearn’ library. We used the default value for gamma with 11 different values of C ranging between 1 and 100. All the other parameters were kept default. The figure below shows the accuracy of the model with different values of C. It is clear here that the model’s performance improves with higher values of C. The best accuracy we get is when C = 100, specifically we get about 99.6% on training and 97.4% on both validation and test data. It is seems however that values of C of 20 or greater performs fairly good (>96.9). Thus, choosing a smaller-margin hyperplane (large C of 100) does a better job of getting all the training points classified correctly.


Accuracy across different values of C
train data
valid data
test data
Accuracy
C value