UNIVERSITY AT BUFFALO
CSE 574
INTRODUCTION TO MACHINE LEARNING
SPRING 2019

Programming Assignment 3
Classification and Regression

Project Report

Submitted by

Deepak Ranjan(dranjan)
. Rahul (rahul2)
. Siddharth Satyakam (s59)

Table of Contents

1. Binary Logistic Regression(BLR)

Set	Accuracy(%)	Error(%)
Training	92.716	7.284
Validation	91.42	8.58
Testing	91.979	8.021

Since training was done on training data so training accuracy is more than testing accuracy. Since the accuracies are quite similar, we can infer that data in training set and testing have almost same distribution.

Here is the error count from each category of the dataset:

BLR			
Class	Training	Validation	Test
0	105	23	18
1	120	34	21
2	438	117	111
3	526	116	88
4	298	63	65
5	533	133	128
6	188	42	50
7	307	77	81
8	602	157	128
9	525	96	112

Table 1 Error count for BLR different classes for training, testing and validation

The following tables shows statistics about the error count and percent that we obtain from each class categories. Statistics were collected for training, validation and testing data. Looking at the below data, we see that distribution of errors is not uniform. However, we can see that cumulative error count and percent for multi-class logistic regression is a little better than binary one vs all logistic regression.

BLR(Training)			
Class	Error	Total	Error(%)
0	107	4923	2.17
1	121	5742	2.11
2	440	4958	8.87
3	529	5131	10.31
4	300	4842	6.20
5	506	4421	11.45
6	187	4918	3.80
7	301	5265	5.72
8	613	4851	12.64
9	537	4949	10.85

MLR(Training)			
Class	Error	Total	Error(%)
0	137	4923	2.78
1	147	5742	2.56
2	453	4958	9.14
3	478	5131	9.32
4	284	4842	5.87
5	484	4421	10.95
6	207	4918	4.21
7	313	5265	5.94
8	499	4851	10.29
9	415	4949	8.39

BLR(Validation)			
Class	Error	Total	Error(%)
0	25	1000	2.5
1	31	1000	3.1
2	119	1000	11.9
3	113	1000	11.3
4	62	1000	6.2
5	128	1000	12.8
6	44	1000	4.4
7	76	1000	7.6
8	159	1000	15.9
9	97	1000	9.7

MLR(Validation)			
Class	Error	Total	Error(%)
0	22	1000	2.2
1	24	1000	2.4
2	106	1000	10.6
3	101	1000	10.1
4	60	1000	6
5	107	1000	10.7
6	52	1000	5.2
7	78	1000	7.8
8	134	1000	13.4
9	75	1000	7.5

BLR(Test)			
Class	Error	Total	Error(%)
0	19	980	1.94
1	20	1135	1.76
2	113	1032	10.95
3	93	1010	9.21
4	65	982	6.62
5	122	892	13.68
6	50	958	5.22
7	77	1028	7.49
8	128	974	13.14
9	108	1009	10.70

MLR(Test)			
Class	Error	Total	Error(%)
0	16	980	1.63
1	25	1135	2.20
2	104	1032	10.08
3	93	1010	9.21
4	64	982	6.52
5	118	892	13.23
6	51	958	5.32
7	83	1028	8.07
8	114	974	11.70
9	81	1009	8.03

2. Multi-class Logistic Regression(MLR)

Set	Accuracy(%)	Error(%)
Training	93.11	6.89
Validation	92.38	7.62
Testing	92.53	7.47

Since training was done on training data so training accuracy is more than testing accuracy. Since the accuracies are quite similar, we can infer that data in training set and testing have almost same distribution.

MLR			
Class	Training	Validation	Test
0	130	21	16
1	132	23	22
2	444	101	104
3	468	98	91
4	300	60	64
5	502	118	124
6	200	47	49
7	297	73	82
8	544	145	114
9	428	76	81

Table 2 Error count after MLR for different classes for training, testing and validation

Set	BLR Accuracy(%)	MLR Accuracy(%)
Training	92.716	93.11
Validation	91.42	92.38
Testing	91.979	92.53

Comparison between Multi-Class Logistic Regression and Binary Logistic Regression:

1. In multiclass regression we are able to classify all the classes (0-9) of mnist_all.mat data at once, whereas in the one vs all we only classify one class at a time from the whole data, so multiclass is computationally more feasible.
2. We observed the accuracy of the multiclass was better than the one vs all classification. This is due to the fact that the estimation of parameters is done interdependently in multiclass. This leads to less misclassification and more robust to outliers.

3. Support Vector Machine (SVM)

We used sci-kit learn package to train SVM and observed the following results. We tried SVM on various parameters and on the different category of the dataset that was given.

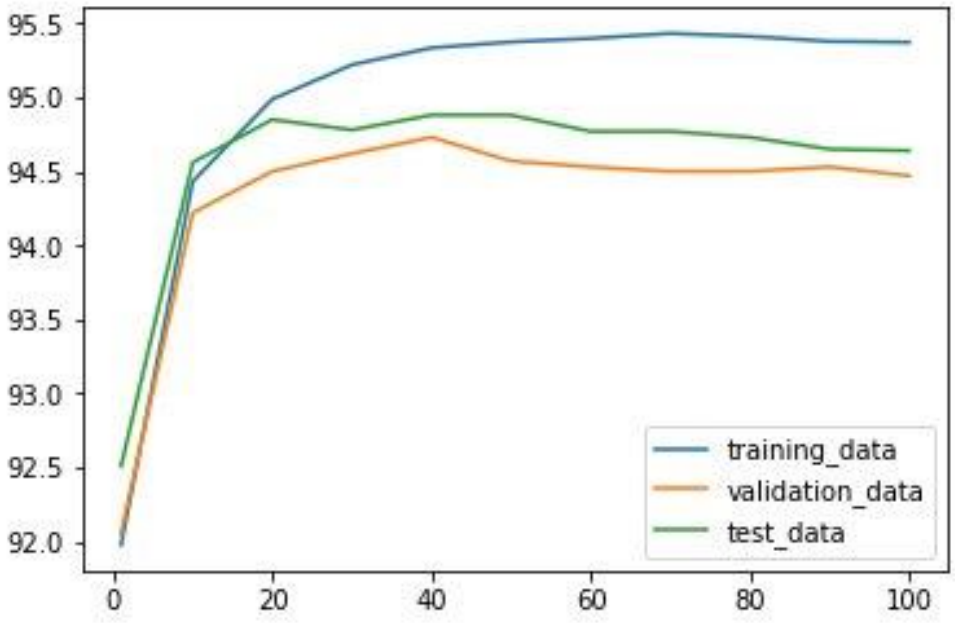
As suggested in the problem statement, we used sampled data to estimate the best kernel. We used Liner, radial basis function with gamma parameter as default and then with gamma as 1.0. We also varied C parameters for various values. Below tables shows the stats on how the model behaved.

Kernel	Gamma	Training Accuracy	Validation Accuracy	Test Accuracy
Linear	Default	92.86%	91.63%	91.72%
RBF	1.00	100%	16.70%	18.65%
RBF	Default	91.95%	92.15%	92.48%

C	Training Accuracy	Validation Accuracy	Test Accuracy
1	91.98%	92.06%	92.51%
10	94.43%	94.22%	94.56%
20	94.99%	94.50%	94.85%
30	95.22%	94.62%	94.78%
40	95.33%	94.73%	94.88%
50	95.37%	94.57%	94.88%
60	95.40%	94.53%	94.77%
70	95.43%	94.50%	94.77%
80	95.41%	94.50%	94.73%
90	95.38%	94.53%	94.65%
100	95.37%	94.47%	94.64%

After seeing the above results, the following observations were made:

- Although training accuracy of linear kernel is better than that of RBF kernel but the validation and test accuracies of RBF kernel are slightly better when gamma is default.
- In case of Gamma = 1 in RBF kernel, training accuracy is 100% and validation and test accuracies are very low showing the overfitting.
- The performance was the best for the C value of 40.



Gamma	Kernel	C	Training Accuracy	Validation Accuracy	Test Accuracy
default	RBF	40	98.71%	97.23%	97.19%
scale	RBF	40	99.95%	97.94%	97.98%

Performance between linear kernel and RBF with different gamma:

- Accuracy in linear kernel is very low when compared with RBF kernel (with penalty factor c=40) which suggests that our data is somewhat non-linear.
- Since our dataset has 10 class and therefore it will have 10 clusters in it. It is also true that it is not linearly separable and therefore using liner kernel will result in worse accuracy that the model trained with RBF.
- Moreover, using different value for gamma gives different performance and the best observed performance comes with gamma as ‘scale’.

The above plot shows the variation of accuracy with respect to penalty factor C. Penalty factor parameter is used to control the error margin and the hyperplane learned by the model. Higher value of C will lead to a hyperplane with narrower margin, while lower C value will learn a hyperplane with higher margin. We can use the stats on the data to decide what value of C should be used that optimizes the error count. Thus we can also say that increasing the value of C will increase the accuracy and then after more increase it will cause overfitting.

4. Conclusion

RBF kernel was performing the best with c = 40 and gamma as default and we were getting training accuracy = 98.71%, validation accuracy = 97.23%, test accuracy = 97.19% for the same.
For the same parameters except gamma as scale we are getting better accuracy (training accuracy = 99.95%, validation accuracy = 97.94%, test accuracy = 97.98%) as it also takes into account the variance of each feature while calculating gamma and then using it to draw the decision boundary.