CSE-574 Introduction to Machine Learning

Programming Assignment 3

Classification and Regression

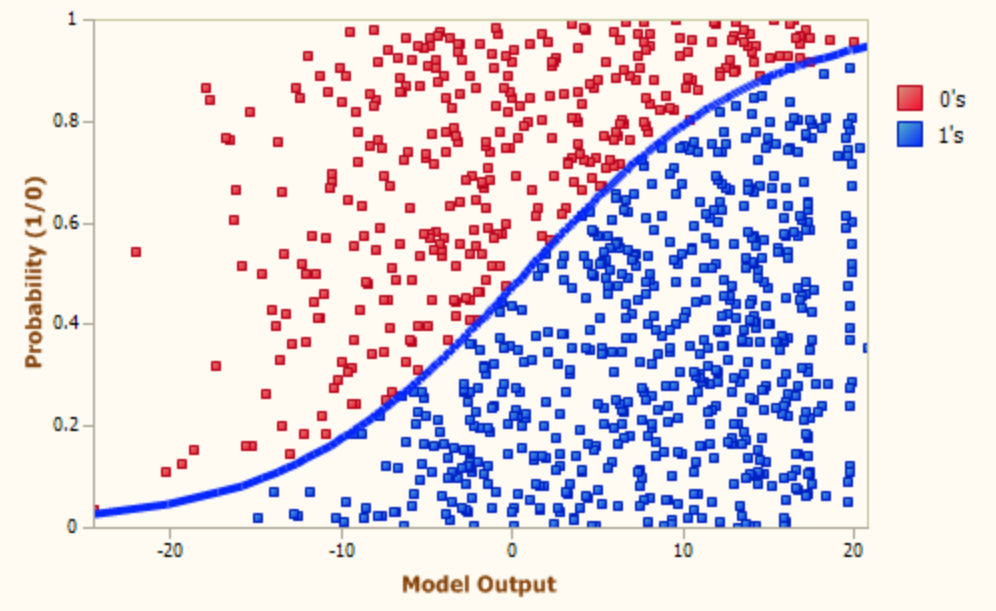
Group 150

Ashutosh Ahmad Alexandar Harish Ganesan

Problem 1.1 :

In this assignment, we will first try to implement a one-vs-all strategy in Logistic Regression to classify MNIST data into 10 separate classes, one for each digit (0-9). In particular, we are building 10 binary-classifiers (one for each class) to distinguish a given class from all other classes.

Logistic regression is a discriminative classifier which tries to learn linear boundary. Hence, it is a model for classification rather than for regression. This model is based on Log Loss where it trains data based on Maximum Likelihood Approach. It minimizes the error of the model and to achieve multiclass classification, logistic model uses One-vs-Many approach or One-VS-Other approach as discussed later.



Logistic Regression for 2 classes 0 and 1

The following screenshot shows the accuracy of the model :

```
Training set Accuracy:84.908%

Validation set Accuracy:83.74%

Testing set Accuracy:84.18%
```

This is without any regularization and using Gradient Descent Approach.

The confusion matrix output are shown below for training, validation and testing data :

```
Training set Accuracy:84.908%
[[4832    1   15    8   10   18   25    7    2    5]
 [   2 5649   33   12    3   21    4   11    0    7]
 [  36   42 4590   70   53   27   55   68    1   16]
 [  21   26  134 4658    9  149   21   45    1   67]
 [   9   20   24    5 4569   13   27   12    0  163]
 [  47   18   34  131   47 3966   89   19    8   62]
 [  26   12   28    2   19   75 4746    4    3    3]
 [  12   22   50   11   44   11    3 4972    0  140]
 [ 136  293  840 1004  193 1317  133   56   39  840]
 [  26   22   14   88  164   45    1  156    0 4433]]

Validation set Accuracy:83.74%
[[979   0   1   3   1   8   6   1   0   1]
 [  0 979   4   4   2   8   0   1   0   2]
 [ 11  18 894  22  13   5  14  15   1   7]
 [  4   9  31 892   4  26   4  13   0  17]
 [  1   5   7   2 942   3   7   0   0  33]
 [  9   9   8  40  19 886  17   2   0  10]
 [  7   3   7   0   6  15 959   2   0   1]
 [  3   4   9   1  15   0   0 926   0  42]
 [ 32  76 194 224  33 250  42   6  11 132]
 [ 10   3   5  20  23   5   1  27   0 906]]

Testing set Accuracy:84.18%
[[ 962    0    1    2    1    4    5    4    0    1]
 [   0 1123    4    1    0    2    4    1    0    0]
 [   9   12  943   21   12    4   14   13    0    4]
 [   4    1   21  929    2   23    4   13    0   13]
 [   1    2    6    3  917    0    9    2    1   41]
 [  11    4    5   37   14  783   20   10    0    8]
 [   9    4    8    2    4   20  910    1    0    0]
 [   2    9   22    5    8    2    1  951    0   28]
 [  34   42  152  199   58  259   43   20    3  164]
 [  10    8    1   18   36   13    1   25    0  897]]
```

We can calculate the class-wise accuracy as shown below :

Training Data											
Actual Class/Predicted Class	0	1	2	3	4	5	6	7	8	9	**Classwise Accuracy (%)**
0	4832	1	15	8	10	18	25	7	2	5	**98.15153362**
1	2	5649	33	12	3	21	4	11	0	7	**98.38035528**
2	36	42	4590	70	53	27	55	68	1	16	**92.57765228**
3	21	26	134	4658	9	149	21	45	1	67	**90.78152407**
4	9	20	24	5	4569	13	27	12	0	163	**94.36183395**
5	47	18	34	131	47	3966	89	19	8	62	**89.70821081**
6	26	12	28	2	19	75	4746	4	3	3	**96.50264335**
7	12	22	50	11	44	11	3	4972	0	140	**94.43494777**
8	136	293	840	1004	193	1317	133	56	39	840	**0.803957947**
9	26	22	14	88	164	45	1	156	0	4433	**89.57365124**
										Total	**84.52763103**
Validation Data											
Actual Class/Predicted Class	0	1	2	3	4	5	6	7	8	9	**Classwise Accuracy (%)**
0	979	0	1	3	1	8	6	1	0	1	**97.9**
1	0	979	4	4	2	8	0	1	0	2	**97.9**
2	11	18	894	22	13	5	14	15	1	7	**89.4**
3	4	9	31	892	4	26	4	13	0	17	**89.2**
4	1	5	7	2	942	3	7	0	0	33	**94.2**
5	9	9	8	40	19	886	17	2	0	10	**88.6**
6	7	3	7	0	6	15	959	2	0	1	**95.9**
7	3	4	9	1	15	0	0	926	0	42	**92.6**
8	32	76	194	224	33	250	42	6	11	132	**1.1**
9	10	3	5	20	23	5	1	27	0	906	**90.6**
										Total	**83.74**
Testing Data											
Actual Class/Predicted Class	0	1	2	3	4	5	6	7	8	9	**Classwise Accuracy (%)**
0	962	0	1	2	1	4	5	4	0	1	**98.16326531**
1	0	1123	4	1	0	2	4	1	0	0	**98.94273128**
2	9	12	943	21	12	4	14	13	0	4	**91.37596899**
3	4	1	21	929	2	23	4	13	0	13	**91.98019802**
4	1	2	6	3	917	0	9	2	1	41	**93.3808554**
5	11	4	5	37	14	783	20	10	0	8	**87.78026906**
6	9	4	8	2	4	20	910	1	0	0	**94.98956159**
7	2	9	22	5	8	2	1	951	0	28	**92.50972763**
8	34	42	152	199	58	259	43	20	3	164	**0.308008214**
9	10	8	1	18	36	13	1	25	0	897	**88.89990089**
										Total	**83.83304864**

Problem 1.3:

Support Vector Machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.
(https://en.wikipedia.org/wiki/Support_vectr_machine)

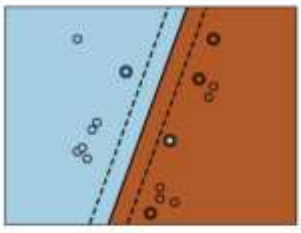
SVM parameters

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

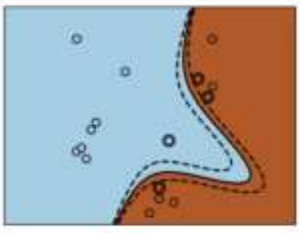
The C parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors.

Three different types of SVM-Kernels are displayed below:

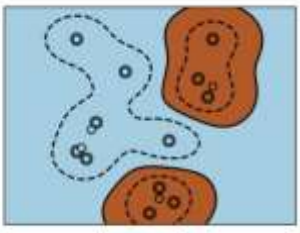
- Linear
- Polynomial
- RBF



Linear

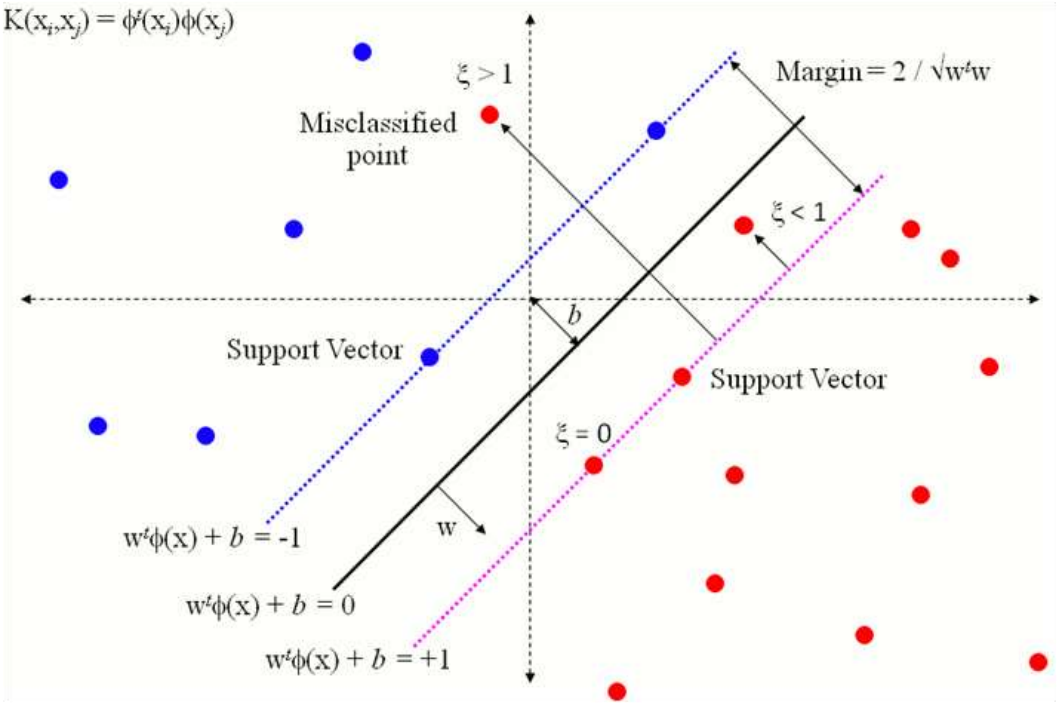


Polynomial



RBF

The polynomial and RBF are especially useful when the data-points are not linearly separable.



Shown below is the output of the SVM run on the MNIST data. The first triplet of train, validation and test data are in the order of :

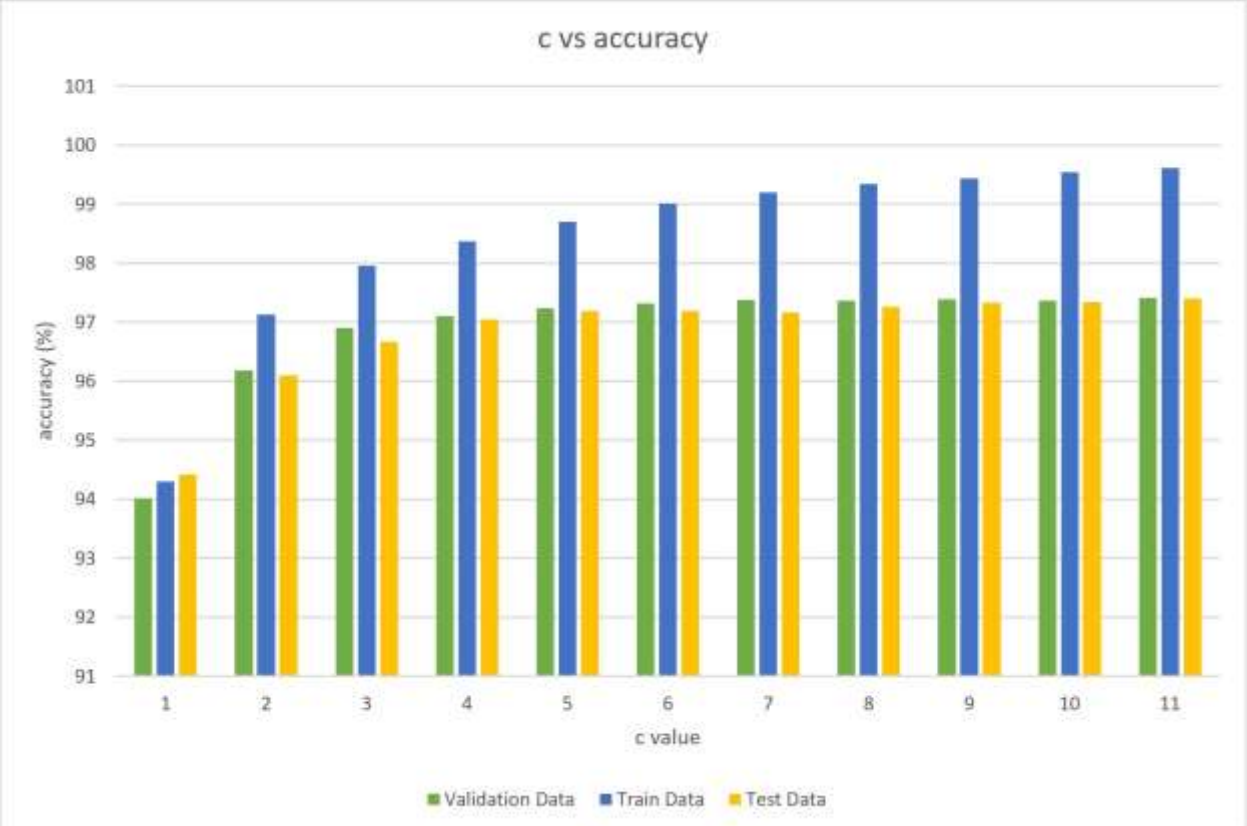
- Linear kernel (all other parameters are kept default)
- Using radial basis function with value of gamma setting to 1 (all other parameters are kept default)
- Using radial basis function with value of gamma setting to default (all other parameters are kept default)
- Using radial basis function with value of gamma setting to default and varying value of C (1, 10, 20, 30, $\cdots$, 100)

----------SVM----------

training 97.286
validation 93.64
testing 93.78
training 100.0
validation 15.48
testing 17.14
training 94.294
validation 94.02
testing 94.42

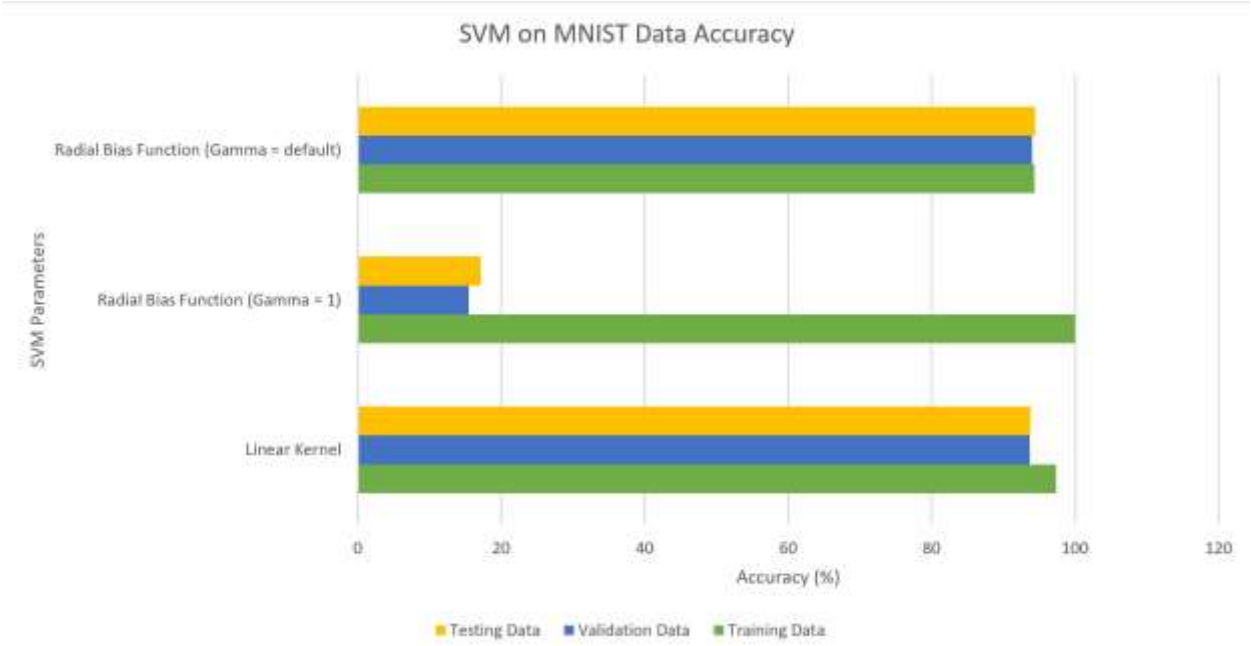
Train Accuracy for: 1 -> 94.294
Validation Accuracy for: 1 -> 94.02
Test Accuracy for: 1 -> 94.42
Train Accuracy for: 10 -> 97.132
Validation Accuracy for: 10 -> 96.18
Test Accuracy for: 10 -> 96.1
Train Accuracy for: 20 -> 97.952
Validation Accuracy for: 20 -> 96.9
Test Accuracy for: 20 -> 96.67
Train Accuracy for: 30 -> 98.372
Validation Accuracy for: 30 -> 97.1
Test Accuracy for: 30 -> 97.04
Train Accuracy for: 40 -> 98.706
Validation Accuracy for: 40 -> 97.23
Test Accuracy for: 40 -> 97.19
Train Accuracy for: 50 -> 99.002
Validation Accuracy for: 50 -> 97.31
Test Accuracy for: 50 -> 97.19
Train Accuracy for: 60 -> 99.196
Validation Accuracy for: 60 -> 97.38
Test Accuracy for: 60 -> 97.16
Train Accuracy for: 70 -> 99.34
Validation Accuracy for: 70 -> 97.36
Test Accuracy for: 70 -> 97.26
Train Accuracy for: 80 -> 99.438

Validation Accuracy for: 80 -> 97.39
Test Accuracy for: 80 -> 97.33
Train Accuracy for: 90 -> 99.542
Validation Accuracy for: 90 -> 97.36
Test Accuracy for: 90 -> 97.34
Train Accuracy for: 100 -> 99.612
Validation Accuracy for: 100 -> 97.41
Test Accuracy for: 100 -> 97.4



The above graph depicts the fact that as we increase the C values, the accuracy also gradually increases for the train, validation and test data. This is because C is a balance between max margin and wrongly classified data points. So increasing number of C will discard more wrong classifications, hence accuracy will be higher.

c value	Train Data	Validation Data	Test Data
1	94.294	94.02	94.42
10	97.132	96.18	96.1
20	97.952	96.9	96.67
30	98.372	97.1	97.04
40	98.706	97.23	97.19
50	99.002	97.31	97.19
60	99.196	97.38	97.16
70	99.34	97.36	97.26
80	99.438	97.39	97.33
90	99.542	97.36	97.34
100	99.612	97.41	97.4



From the above figure, we can see that the Gamma = 1 parameter reduces the accuracy considerably, this is due to the model overfitting the data. We can clearly observe this from the fact that the Training Accuracy in this case is a 100%. Therefore, this model does not perform favorably.

The Radial Bias Function provides **93.78%** testing accuracy, and the Radial basis function with value of gamma setting to default and varying value of C (1, 10, 20, 30, $\cdots$, 100) has a maximum accuracy of **97.4%.** This is excellent classification accuracy.

SVM Kernel Type	Training Data	Validation Data	Testing Data
Linear Kernel	97.286	93.64	93.78
Radial Bias Function (Gamma = 1)	100	15.48	17.14
Radial Bias Function (Gamma = default)	94.294	94.02	94.42

Problem 1.2 : For Extra Credit

In this part of the assignment, we implement One-VS-Other approach: In this K-Classifier, new data is fed to all the output classes and posterior probability is calculated for each class for given input. The final outcome is then decided by selecting the class with highest posterior probability.

The output of the programs with the accuracies and the confusion matrix for the train data, validation data and test data is shown below :

```
 Training set Accuracy:93.448%
[[4786    1   12    7   11   33   30    7   32    4]
 [   1 5592   26   17    6   19    2   13   58    8]
 [  23   45 4503   72   58   24   59   53  108   13]
 [  14   18   95 4654    4  148   15   39  105   39]
 [   8   20   21    7 4576    6   42   13   24  125]
 [  39   13   36  117   34 3963   68   18  102   31]
 [  23   11   29    1   24   52 4758    2   16    2]
 [   8   16   49   18   34    9    4 4989   14  124]
 [  22   75   51  103   16  113   23   16 4387   45]
 [  17   18    9   55  126   30    2  134   42 4516]]

 Validation set Accuracy:92.48%
[[975   0   1   3   2   7   3   2   6   1]
 [  0 972   3   2   1   5   0   2  13   2]
 [ 10  13 896  22  13   4  11   9  18   4]
 [  1   7  23 902   3  28   2  12  13   9]
 [  1   4   8   3 941   1  10   2   7  23]
 [  9   4   6  37  17 884  14   2  22   5]
 [  9   2   4   1   7  12 957   1   6   1]
 [  2   3   9   0   9   1   0 931   3  42]
 [ 13  17  19  27   9  20  19   2 868   6]
 [  4   3   5  14  19   4   1  24   4 922]]

 Testing set Accuracy:92.55%
[[ 960    0    0    3    0    6    6    4    1    0]
 [   0 1110    3    2    0    2    4    2   12    0]
 [   6    8  924   16   10    3   14    8   39    4]
 [   4    1   20  914    0   25    3   10   26    7]
 [   1    1    6    2  921    0    9    4    9   29]
 [  10    2    2   37   10  773   15    6   30    7]
 [   9    3    4    2    7   15  914    3    1    0]
 [   1    9   19    6    6    2    0  952    2   31]
 [   9    8    6   26    9   23   10    8  868    7]
 [  11    8    0   10   28    5    0   20    8  919]]
```

As shown below, we have the confusion matrices and the class-wise accuracies and total accuracy for training, validation and testing data for Multi-Class one-v-other Logistic Regression.

Training Data

Actual Class/Predicted Class	0	1	2	3	4	5	6	7	8	9	**Classwise Accuracy (%)**
0	4786	1	12	7	11	33	30	7	32	4	**96.33655395**
1	1	5592	26	17	6	19	2	13	58	8	**97.3876698**
2	23	45	4503	72	58	24	59	53	108	13	**90.82291246**
3	14	18	95	4654	4	148	15	39	105	39	**90.70356656**
4	8	20	21	7	4576	6	42	13	24	125	**94.50640231**
5	39	13	36	117	34	3963	68	18	102	31	**89.64035286**
6	23	11	29	1	24	52	4758	2	16	2	**96.74664498**
7	8	16	49	18	34	9	4	4989	14	124	**94.75783476**
8	22	75	51	103	16	113	23	16	4387	45	**90.43496186**
9	17	18	9	55	126	30	2	134	42	4516	**91.25075773**
										Total	**93.25876573**

Validation Data

Actual Class/Predicted Class	0	1	2	3	4	5	6	7	8	9	**Classwise Accuracy (%)**
0	975	0	1	3	2	7	3	2	6	1	**97.5**
1	0	972	3	2	1	5	0	2	13	2	**97.2**
2	10	13	896	22	13	4	11	9	18	4	**89.6**
3	1	7	23	902	3	28	2	12	13	9	**90.2**
4	1	4	8	3	941	1	10	2	7	23	**94.1**
5	9	4	6	37	17	884	14	2	22	5	**88.4**
6	9	2	4	1	7	12	957	1	6	1	**95.7**
7	2	3	9	0	9	1	0	931	3	42	**93.1**
8	13	17	19	27	9	20	19	2	868	6	**86.8**
9	4	3	5	14	19	4	1	24	4	922	**92.2**
										Total	**92.48**

Test Data

Actual Class/Predicted Class	0	1	2	3	4	5	6	7	8	9	**Classwise Accuracy (%)**
0	960	0	0	3	0	6	6	4	1	0	**97.95918367**
1	0	1110	3	2	0	2	4	2	12	0	**97.79735683**
2	6	8	924	16	10	3	14	8	39	4	**89.53488372**
3	4	1	20	914	0	25	3	10	26	7	**90.4950495**
4	1	1	6	2	921	0	9	4	9	29	**93.78818737**
5	10	2	2	37	10	773	15	6	30	7	**86.65919283**
6	9	3	4	2	7	15	914	3	1	0	**95.40709812**
7	1	9	19	6	6	2	0	952	2	31	**92.60700389**
8	9	8	6	26	9	23	10	8	868	7	**89.11704312**
9	11	8	0	10	28	5	0	20	8	919	**91.0802775**
										Total	**92.44452766**

The accuracy for training is higher than that of testing because there are more data samples to classify, and the testing data is all new data.

As we can see from the results in one-vs-many is inferior to the results given by one-v-other, as the accuracies are higher by almost 10%. The SVM results are also quite promising, but takes a very long time to finish the execution.