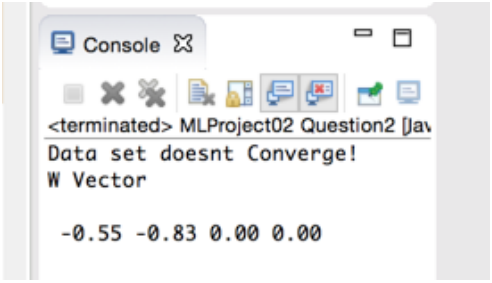
1. Introduction

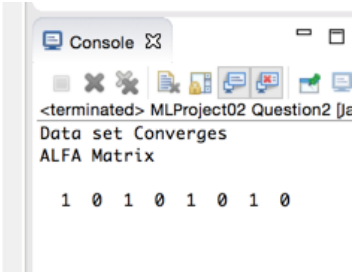
The project aims at implementation of the four versions of perceptron algorithms and evaluation of the digit recognition data set using these four algorithms and Support Vector machines to analyze and compare the performance of the algorithms and its results. The first 1000 samples of "optDigits.tra" training data file is used for development data and rest of the data is used for training. The models are tested with "optdigits.tes" data file.

2. Results and Analysis using Perceptron Algorithms

2.1 Question B

The data set given in the question 2 of the assignment is evaluated with the implemented Linear Perceptron algorithm. The result proved that the data doesn't converge at this feature space with number of Epochs set to 2, 5, 10. The same data set converges with Polynomial Kernel Perceptron Algorithm with degree 2 and No of Epochs as 3.





2.2 Question C

For Digit Recognition, The training file "optdigits.tra" has features for the samples of digits 0 to 9. So, it is divided into 9 training files each for a digit setting the class to 1 for instances of the digit and -1 for instances of the other digits. The same has been done for the development and test data to use them for tuning and testing respectively.

2.3 Question D

Development data has been used to decide the optimal parameters; Epochs, Sigma and Exponent. The tuning is done on the Linear Perceptron algorithm to decide an optimal value of No of Epochs whereas to decide upon optimal values for exponent and sigma, Polynomial Kernel and Gaussian Kernel Perceptron Algorithms are used respectively.

2.3.1 Optimal Parameters

The Linear perceptron algorithm is trained with the training data and tested with the development data to obtain a best value for T among {1,5,10,20}. The best value for T that is obtained on the development data is 5 with an accuracy of 90.8%. The values of Epochs and their respective accuracies are tabulated in Table 1.1

Table 1.1

Accuracy %'s for Different values of T				
Epochs	1	5	10	15
Accuracy(%)	88	90.8	87.9	89.6

The Polynomial Kernel Perceptron algorithm is trained with the training data and tested with the development data to yield a best value for d as 5 with an accuracy of 98.3%. Refer table 1.2 for values of d and their accuracies.

Table 1.2

Accuracy %'s for Different values of d					
Exponent(d)	2	3	4	5	6
Accuracy(%)	94.69	96.5	97	98.3	97.39

The Gaussian Kernel Perceptron Algorithm trained with training data and tested with development data resulted the best sigma as 10 with and Accuracy of 98.2%. Refer table 1.3 for values of sigma and their respective accuracies.

Table 1.3

Accuracy %'s for Different values of Sigma					
Sigma	0.5	2	3	5	10
Accuracy(%)	65.9	97.1	97.2	97.2	98.2

All The kernels and Weight Vectors that are used for the above computations are normalized except Gaussian Kernel as it is not subjective to Normalization.

2.3.2 Training Times and Support Vectors

Table 1.4 shows the Training times for different Models and the values reveals that Averaged models is taking more time when compared to the normal models and this is because of the additional steps of logic that been added to the averaged version in all the algorithms.

Table 1.4

Model	Training Time(Milli Sec's)
Linear Perceptron	242
Avg. Linear Perceptron	714
Polynomial Kernel Perceptron	18507
Avg. Poly Kernel Perceptron	32746
Gaussian Kernel Perceptron	18774
Avg. Gaussian Kernel Perceptron	28857

Table 1.5 shows the count of support vectors for different Models implementing different versions of Perceptron Algorithm. It is observed that the no of support vectors are same for Normal and respective averaged versions of the algorithm and this is because the we are averaging the same support vector that we have got in the Normal Version of the Algorithm.

Table 1.5

Model for Digit	Polynomial Kernel Peceptron	Avg. Kernel Polynomial Perceptron	Gaussian Kernel Perceptron	Avg. Gaussain Kernel Perceptron
0	22	22	26	26
1	103	103	80	80
2	46	46	41	41
3	88	88	80	80
4	45	45	56	56
5	50	50	59	59
6	46	46	38	38
7	42	42	33	33
8	145	145	87	87
9	141	141	107	107

2.3.3 Comparison of Confusion Matrices on Test Data

1. Gaussian and Averaged Gaussian Kernel Perceptron

Gaussian Kernel Perceptron Confusion Matrix											
	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	178	0	0	2	0	0	0	0	0	1	181
SD1	0	180	1	0	0	0	2	0	4	0	187
SD2	0	0	173	0	0	0	1	0	0	0	174
SD3	0	0	0	165	0	0	0	0	2	1	168
SD4	0	0	1	2	178	0	2	0	1	0	184
SD5	0	0	0	0	1	181	0	0	0	1	183
SD6	0	1	2	5	0	0	175	0	0	1	184
SD7	0	0	0	4	0	0	1	169	2	1	177
SD8	0	1	0	4	2	0	0	0	164	3	174
SD9	0	0	0	1	0	1	0	10	1	172	185
Sum	178	182	177	183	181	182	181	179	174	180	1797
%	100.00	98.90	97.74	90.16	98.34	99.45	96.68	94.41	94.25	95.55	
Average Gaussain Kernel Perceptron Confusion Matrix											
	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	178	0	0	2	0	0	0	0	0	1	181
SD1	0	180	1	0	0	0	1	0	5	0	187
SD2	0	0	174	0	0	0	1	0	0	0	175
SD3	0	0	0	165	0	0	0	0	2	1	168
SD4	0	0	1	2	178	1	2	0	0	1	185
SD5	0	0	0	0	1	180	0	0	0	1	182
SD6	0	1	1	5	0	0	175	0	0	0	182
SD7	0	0	0	4	0	0	2	171	2	1	180
SD8	0	1	0	4	2	0	0	0	164	2	173
SD9	0	0	0	1	0	1	0	8	1	173	184
Sum	178	182	177	183	181	182	181	179	174	180	1797
%	100.00	98.90	98.30	90.16	98.34	98.90	96.68	95.53	94.25	96.11	

The Confusion Matrix clearly shows that both Gaussian Kernel Perceptron Algorithm and Averaged Gaussian Kernel Perceptron Algorithm find digit 3 classification hard.

2. Linear Perceptron and Averaged Linear Perceptron

	Perceptron Confusion Matrix										
	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	169	0	0	0	0	0	0	0	0	0	169
SD1	0	146	2	0	0	0	0	0	11	1	160
SD2	0	12	174	3	0	1	0	0	0	0	190
SD3	0	0	1	166	0	0	0	1	1	0	169
SD4	1	1	0	0	178	0	2	5	0	4	191
SD4	8	5	0	4	0	180	3	11	12	4	227
SD5	0	3	0	0	0	0	175	0	1	0	179
SD6	0	0	0	0	0	0	0	150	0	0	150
SD7	0	8	0	3	3	0	1	7	146	7	175
SD8	0	7	0	7	0	1	0	5	3	164	187
SD9	178	182	177	183	181	182	181	179	174	180	1797
%	94.94	80.02	98.30	90.70	98.34	98.90	97.20	83.79	83.90	91.10	
	Average Perceptron Confusion Matrix										
	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	174	0	0	0	0	0	0	0	0	0	174
SD1	0	161	1	1	3	0	2	0	17	3	188
SD2	0	9	173	2	0	0	0	0	0	0	184
SD3	0	0	1	168	0	1	0	0	1	3	174
SD4	1	0	0	0	176	0	2	1	0	4	184
SD4	3	1	0	3	0	179	0	7	4	3	200
SD5	0	0	0	0	0	1	176	0	1	0	178
SD6	0	0	2	2	1	0	0	167	0	0	172
SD7	0	5	0	3	1	0	1	2	148	9	169
SD8	0	6	0	4	0	1	0	2	3	158	174
SD9	178	182	177	183	181	182	181	179	174	180	1797
%	97.75	88.46	97.74	91.80	97.23	98.35	97.23	93.29	85.05	87.77	

The Confusion Matrix clearly shows that the using Linear Perceptron Algorithm its difficult to classify digit 1 and for Averaged Linear Perceptron Algorithm its difficult to classify digit 8. The results are decided upon the percentage values shown in the table.

3. Polynomial and Averaged Polynomial Kernel Perceptron

Kernel Perceptron Confusion Matrix

	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	
SD0	178	0	0	0	0	1	0	0	0	0	179
SD1	0	182	4	0	7	2	0	1	7	1	204
SD2	0	0	170	0	0	0	0	1	0	0	171
SD3	0	0	0	176	0	1	0	0	0	2	179
SD4	0	0	0	0	172	1	2	0	0	0	175
SD5	0	0	0	2	0	175	0	0	0	1	178
SD6	0	0	0	0	0	0	178	0	0	0	178
SD7	0	0	1	0	0	0	0	173	0	0	174
SD8	0	0	2	5	2	0	1	1	166	2	179
SD9	0	0	0	0	0	2	0	3	1	174	180
Sum	178	182	177	183	181	182	181	179	174	180	1797
%	100.00	100.00	96.04	96.17	95.02	96.15	98.34	96.64	95.40	96.66	

Average Kernel Perceptron Confusion Matrix

	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	178	0	0	0	0	1	0	0	0	0	179
SD1	0	181	5	0	3	0	0	0	7	0	196
SD2	0	0	172	1	0	0	0	0	0	0	173
SD3	0	0	0	173	0	0	0	0	0	1	174
SD4	0	0	0	0	176	1	2	0	0	0	179
SD5	0	0	0	2	0	179	0	2	0	1	184
SD6	0	0	0	0	0	0	178	0	0	0	178
SD7	0	0	0	1	1	0	0	173	0	0	175
SD8	0	1	0	4	1	0	1	1	166	1	175
SD9	0	0	0	2	0	1	0	3	1	177	184
Sum	178	182	177	183	181	182	181	179	174	180	1797
%	100.00	99.45	97.17	94.50	97.23	98.35	98.34	96.64	95.40	98.33	

The Confusion Matrix clearly shows that Polynomial Kernel Perceptron Algorithm finds digit 4 as toughest to classify whereas Averaged polynomial Kernel Perceptron Algorithm finds digit 3 is hard to classify.

2.3.4 Overall Performance on Test Data

The overall performance of the the six models are computed and tabulated in Table 1.6. Models having polynomial and Gaussian Kernels did the best as we know that they change the feature space into high dimensionality to make them converge.

Table 1.6

Model	Correctly Classified Samples	Test data Set Size	Accuracy %
Perceptron Algorithm	1648	1797	91.7
Avg. Perceptron	1680	1797	93.48
Polynomial Kernel Perceptron	1744	1797	97.05
Avg. Polynomial Kernel Perceptron	1753	1797	97.55
Gaussian Kernel Perceptron	1735	1797	96.54
Avg. Gaussian Kernel Perceptron	1738	1797	96.71

3. Results and Analysis using Support Vector Machines

3.1 Question E

SVM Light package has been used with Linear Kernel, polynomial Kernel with exponent 5 and C value as 1, Gaussian Kernel with Sigma as 10. The Learning times and the test results are tabulated for each algorithm and compared with the Perceptron Algorithms respectively. Training and Test data has been converted to make it SVM Light package compatible and a Java Class "*SVMConfusionMatrices.java*" has been written to compute the Confusion Matrices based upon the models generated by the SVM package.

3.2 Training times and Support Vectors

The support vectors for the 3 different Kernels is captured in Table 1.7. A simple observation that one could make out is there are more number of support vectors in Gaussian Kernel when compared to others. The Support vectors are generated using *generateModels.sh* script that generates the models for each of the digits. The Learning times are comparatively very less for these SVM Models when compared to the Perceptron models and this is because SVM uses quadratic optimization problem solvers.

Kernel Type	Training Time(SEC's)
Linear Kernel SVM	1.1
Polynomial Kernel SVM	1.4
Gaussian Kernel SVM	9.8

Table 1.7 Support Vectors Count for SVM's

Model	Linear Perceptron	Polynomial Kernel Perceptron	Gaussian Kernel Perceptron
Digit 0 Model	124	62	2037
Digit 1 Model	238	145	2038
Digit 2 Model	202	120	2050
Digit 3 Model	251	173	2087
Digit 4 Model	212	126	2064
Digit 5 Model	234	142	2086
Digit 6 Model	155	99	2035
Digit 7 Model	157	111	2051
Digit 8 Model	395	242	2082
Digit 9 Model	378	232	2084

3.3 Confusion Matrices

Confusion Matrices of SVM Models for the 3 different kernels reveal there is an increase in overall performance of the models when compared to their respective perceptron versions.

3.3.1 Confusion Matrix for Linear Kernel SVM

Linear Kernel SVM Confusion Matrix											
	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	174	0	0	0	0	1	0	0	0	0	175
SD1	0	152	1	1	7	0	2	0	15	3	181
SD2	0	16	174	4	0	0	0	0	0	1	195
SD3	0	0	1	167	0	0	0	0	1	4	173
SD4	1	0	0	0	172	0	1	1	0	5	180
SD5	3	1	0	3	0	179	0	7	6	3	202
SD6	0	1	0	0	0	1	177	0	2	0	181
SD7	0	0	1	3	1	0	0	167	0	0	172
SD8	0	3	0	3	1	0	1	1	143	4	156
SD9	0	9	0	2	0	1	0	3	7	160	182
Sum	178	182	177	183	181	182	181	179	174	180	1797
%	97.75	83.51	98.30	91.25	95.02	98.35	97.79	93.29	82.10	88.88	

The confusion matrix says that the hardest digit to classify by the Linear Kernel SVM is 8 with an accuracy of 82.10%. The Overall Accuracy of the Linear Kernel SVM Model is 1665/1797= 92.65%.

3.3.2 Confusion Matrix for Polynomial Kernel SVM

Polynomial Kernel SVM Confusion Matrix											
	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	178	0	0	0	0	0	0	0	0	0	178
SD1	0	182	1	0	1	0	0	0	5	1	190
SD2	0	0	176	0	0	0	0	0	0	0	176
SD3	0	0	0	178	0	0	0	0	0	1	179
SD4	0	0	0	0	180	0	0	0	0	0	180
SD5	0	0	0	1	0	181	0	1	1	1	185
SD6	0	0	0	0	0	0	180	0	0	0	180
SD7	0	0	0	1	0	0	0	174	0	0	175
SD8	0	0	0	1	0	0	1	0	166	2	170
SD9	0	0	0	2	0	1	0	4	2	175	184
Sum	178	182	177	183	181	182	181	179	174	180	1797
%	100.00	100.00	99.43	97.26	99.44	99.45	99.44	97.20	95.40	97.22	

The confusion matrix says that the hardest digit to classify by the Polynomial Kernel SVM is 8 with an accuracy of 95.40%. The Overall Accuracy of the Polynomial Kernel SVM Model is 1770/1797= 98.49%.

3.3.3 Confusion Matrix for Gaussian Kernel SVM

Gaussian Kernel SVM Confusion Matrix											
	TD0	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	TD9	Sum
SD0	177	0	0	0	0	0	0	0	0	0	177
SD1	0	180	1	0	1	0	0	0	6	1	189
SD2	0	0	172	0	0	0	0	0	0	0	172
SD3	0	0	0	176	0	0	0	0	1	2	179
SD4	1	0	0	0	179	1	0	0	0	0	181
SD5	0	0	0	1	0	179	1	0	0	1	182
SD6	0	1	0	0	0	0	179	0	0	0	180
SD7	0	0	1	3	0	0	0	174	0	0	178
SD8	0	1	3	3	1	0	1	2	165	2	178
SD9	0	0	0	0	0	2	0	3	2	174	181
Sum	178	182	177	183	181	182	181	179	174	180	1797
%	99.43	98.90	97.17	96.17	98.89	98.35	98.89	97.20	94.82	96.66	

The confusion matrix says that the hardest digit to classify by the Gaussian Kernel SVM is 8 with an accuracy of 94.82%. The Overall Accuracy of the Polynomial Kernel SVM Model is 1755/1797= 97.66%.