Name: Taiman Siddiqui **HW 7 Report**

Text Classification

Spam vs. non-spam:

Accuracy of SVM algorithm with linear kernel: 97.70%.

Accuracy of perceptron algorithm: 97.80%

Accuracy of average perceptron algorithm: 98.10%.

Atheism vs. Religion:

Accuracy of linear svm on test dataset version 1: 59.474%.

Accuracy of linear svm on test dataset version 2: 59.474%.

Accuracy of perceptron on test dataset version 1: 59.474%.

Accuracy of perceptron on test dataset version 2: 59.298%.

Accuracy of average perceptron on test dataset version 1: 60.877%.

Accuracy of average perceptron on test dataset version 2: 61.579%.

We observe that the linear SVM performs similar to the perceptron algorithm and slightly worse than the average perceptron algorithm in both cases.

Digit Recognition

Linear Perceptron:

Training the linear perceptron on development data for epochs set to $T \in \{1,2,...,20\}$ to obtain T with best performance:

Epoch: 1, Accuracy: 92.80

Epoch: 2, Accuracy: 94.00

Epoch: 3, Accuracy: 93.70

Epoch: 4, Accuracy: 94.40

Epoch: 5, Accuracy: 94.70

Epoch: 6, Accuracy: 93.20

Epoch: 7, Accuracy: 94.40

Epoch: 8, Accuracy: 95.30

Epoch: 9, Accuracy: 93.80

Epoch: 10, Accuracy: 93.80

Epoch: 11, Accuracy: 94.70

Epoch: 12, Accuracy: 94.10

Epoch: 13, Accuracy: 95.00

Epoch: 14, Accuracy: 94.10

Epoch: 15, Accuracy: 94.70

Epoch: 16, Accuracy: 93.80

Epoch: 17, Accuracy: 94.60

Epoch: 18, Accuracy: 93.70

Epoch: 19, Accuracy: 95.30

Epoch: 20, Accuracy: 95.30

Maximum accuracy in epoch: 8

Result of training and testing the linear perceptron using the appropriate examples:

Accuracy of linear perceptron = 92.71%

Time taken by linear perceptron = 1.27 seconds

Number of support vectors = 640

Confusion matrix for linear perceptron:

[[170 0 0 0 1 7 0 0 0 0]

[0 147 14 0 5 0 1 0 8 7]

[0 0 177 0 0 0 0 0 0 0]

[0 0 4 165 0 3 0 1 4 6]

[0 0 0 0 178 0 0 0 3 0]

[0 0 1 1 0 179 0 0 0 1]

[1 1 0 0 2 1 176 0 0 0]

[0 0 0 0 3 11 0 158 2 5]

[0 9 0 3 1 6 0 0 153 2]

[0 1 0 0 5 2 0 0 9 163]]

Polynomial kernel perceptron:

Testing degree d ∈ {2,3,4,5,6} on development data:

Degree: 2, Accuracy: 96.4

Degree: 3, Accuracy: 97.1

Degree: 4, Accuracy: 98.3

Degree: 5, Accuracy: 97.4

Degree: 6, Accuracy: 98.6

Maximum accuracy in polynomial degree d = 6

Testing trained model on test examples using tuned degree d:

Accuracy of polynomial kernel perceptron = 97.162%

Number of support vectors = 28230

Time taken by poly kernel perceptron = 9.91 minutes

Confusion matrix for poly kernel perceptron:

```
[[178   0   0   0   0   0   0   0   0   0]
 [  2 177   0   0   0   0   2   0   1   0]
 [  1   3 167   3   0   0   0   0   3   0]
 [  0   0   0 175   0   2   0   1   4   1]
 [  0   1   0   0 178   0   0   0   0   2]
 [  1   0   0   0   1 177   2   0   0   1]
 [  0   0   0   0   0   0 180   0   1   0]
 [  1   0   0   0   0   0   0 172   1   5]
 [  3   4   0   0   0   0   0   0 166   1]
 [  0   0   0   2   0   1   0   0   1 176]]
```

Gaussian kernel perceptron:

Testing sigma ∈ {0.1,0.5,2.0,5.0,10.0} on development data:

Sigma: 0.1, Accuracy: 96.2

Sigma: 0.5, Accuracy: 96.2

Sigma: 2.0, Accuracy: 98.5

Sigma: 5.0, Accuracy: 96.6

Sigma: 10.0, Accuracy: 94.9

Maximum accuracy for sigma = 2.0

Testing trained model on test examples using tuned sigma:

Accuracy of gaussian kernel perceptron = 98.275%

Number of support vectors = 28230

Time taken by gaussian kernel perceptron = 18.11 minutes

Confusion matrix for gaussian kernel perceptron:

[[178 0 0 0 0 0 0 0 0 0]

[0 182 0 0 0 0 0 0 0 0]

[0 2 175 0 0 0 0 0 0 0]

[0 0 1 175 0 1 0 2 2 2]

[0 0 0 0 181 0 0 0 0 0]

[0 0 0 1 1 179 0 0 0 1]

[1 1 0 0 0 0 179 0 0 0]

[0 0 0 0 0 0 0 177 0 2]

[0 7 0 0 0 1 0 0 165 1]

[0 0 0 2 0 1 0 0 2 175]]

Linear svm:

Accuracy of linear svm: 95.771%

Time taken by linear svm: 0.24 seconds

Number of support vectors for each class = [28 55 49 49 58 56 40 44 77 70]

Total number of support vectors = 526

Confusion matrix for linear svm:

```
[[178   0   0   0   0   0   0   0   0   0]
 [  0 177   0   0   0   0   3   0   1   1]
 [  0   2 173   0   0   0   1   1   0   0]
 [  0   0   6 169   0   2   0   3   2   1]
 [  0   0   0   0 179   0   0   0   2   0]
 [  0   0   1   0   0 179   0   0   0   2]
 [  0   0   0   0   0   0 179   0   2   0]
 [  0   0   0   0   2   7   0 166   0   4]
 [  0  11   1   3   1   2   0   0 155   1]
 [  1   2   0   4   2   3   0   0   2 166]]
```

Polynomial SVM:

Testing degree d ∈ {2,3,4,5,6} on development data:

Degree: 2, Accuracy: 98.5

Degree: 3, Accuracy: 98.5

Degree: 4, Accuracy: 98.6

Degree: 5, Accuracy: 98.7

Degree: 6, Accuracy: 98.7

Maximum accuracy in polynomial degree d: 5

Testing trained model on test data using tuned degree d:

Accuracy for poly kernel svm: 97.774%

Time taken by poly kernel svm: 0.29 seconds

Number of support vectors for each class = [24 59 58 66 67 74 41 53 85 80]

Total number of support vectors = 607

Confusion matrix for poly svm:

[[178 0 0 0 0 0 0 0 0 0]

[0 181 0 0 0 0 1 0 0 0]

[0 1 176 0 0 0 0 0 0 0]

[0 0 1 176 0 1 0 2 1 2]

[0 1 0 0 180 0 0 0 0 0]

[0 0 0 0 0 181 0 0 0 1]

[0 0 0 0 0 0 181 0 0 0]

[0 0 0 0 0 2 0 170 0 7]

[0 6 0 1 0 1 0 0 160 6]

[0 0 0 2 0 3 0 0 1 174]]

Gaussian SVM:

Testing sigma ∈ {0.1,0.5,2.0,5.0,10.0} on development data:

Degree: 0.1, Accuracy: 9.4

Degree: 0.5, Accuracy: 9.4

Degree: 2.0, Accuracy: 9.4

Degree: 5.0, Accuracy: 36.8

Degree: 10.0, Accuracy: 96.7

Maximum accuracy for sigma = 10.0

Testing trained model on test data using tuned sigma:

Accuracy for gaussian svm: 94.769%

Time taken by gaussian svm: 2.21 seconds

Number of support vectors for each class = [131 213 216 219 233 235 176 215 263 250]

Total number of support vectors = 2151

Confusion matrix for gaussian svm:

```
[[171   0   0   0   1   0   0   0   6   0]
 [  0 173   0   0   0   0   0   0   9   0]
 [  0   0 163   0   0   0   0   0  14   0]
 [  0   0   0 168   0   1   0   0  14   0]
 [  0   0   0   0 177   0   0   0   4   0]
 [  0   0   0   0   0 177   0   0   4   1]
 [  0   0   0   0   0   0 172   0   9   0]
 [  0   0   0   0   0   0   0 156  19   4]
 [  0   1   0   0   0   0   0   0 172   1]
 [  0   0   0   2   0   1   0   0   3 174]]
```

Observations:

If we observe all the confusion matrices, we that the greatest spread of misclassifications in general is for digit 9. However, the number of misclassifications for each digit is relatively low for all cases, and the confusion matrices are very diagonal heavy. The best performance is achieved by the Gaussian kernel perceptron using sigma = 2. The polynomial kernel perceptron and Gaussian kernel perceptron algorithms are slower at training time. This is because of the greater or more complex computational requirements of these algorithms; the polynomial kernel computes degree 6 polynomial for each training example (corresponding to a large feature map) whereas the gaussian kernel uses the exponential function to compute the same.