Jackson Cook

CS4641 – Machine Learning

Assignment 1: Supervised Learning

Classification Problems/Datasets

Vehicle:

This dataset was collected with the goal of classifying a vehicle by its silhouette as one of four types. By extracting a variety of scale independent features, the vehicle would be classified as OPEL, SAAB, BUS, or VAN. Two sets of 60 images were taken for each vehicle covering a full 360 degrees of rotation. This dataset contains 846 instances and each has 19 attributes.

Vehicle is interesting because vehicle identification is practical in life, especially in law enforcement. Classification will be very difficult as geometry of vehicles of all types tend to follow a similar trend. The VAN and BUS are expected to be easily distinguished between each other and amongst the OPEL and SAAB. The 4 OPEL and SAAB are similar in geometry and will likely be more difficult to classify. Overfitting is a major concern because of this, as many geometric features will probably be shared between all four vehicle types. Boosting will probably be very valuable on the dataset as it seems some of the instances may be much more easily classified than others.

Semeion:

This dataset has the results of a study where handwritten digits from approximately 80 people were scanned, expanded into a 16 pixel by 16 pixel square, and given a Boolean for each pixel in the box if the pigment was at a certain threshold. There are 1593 instances in this dataset with 256 attributes each.

Semeion is valuable as character recognition is extremely useful in many fields especially in translation from handwritten to digital. Correct classification of numbers and in the future letters can provide tremendous utility to many people. Issues to expect with classification will likely be linked to peoples' different handwriting which varies from person to person. The dataset is also quite small given how many digits there are and how many instances per digit there are for the algorithm to learn. Overfitting may be useful when classifying this data, so I expect the neural network to excel on this dataset.

Decision Trees

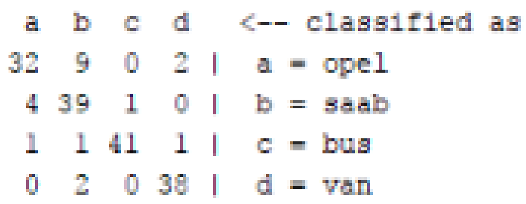
For the decision tree analysis, I used many different levels of pruning (confidence levels of .01, .05, .1, .25, and .5) as well as one unpruned tree.

Vehicle:

Input		**Output**					
Pruned?	**Confidence Level**	**Tree Size**	**Total Time(s)**	**Training Correct**	**Testing Correct**	**Training Correct w/ X-Validation**	**Testing Correct w/ X-Validation**
Yes	0.01	63	0.02	83.432%	70%	71.0059%	83.0409%
Yes	0.05	95	0.03	88.6095%	69.4118%	71.8935%	87.1345%
Yes	0.1	109	0.02	90.2369%	67.6471%	71.1538%	87.7193%
Yes	0.25	127	0	91.568%	68.2353%	71.3018%	88.8889%
Yes	0.5	137	0.02	92.3077%	68.2353%	71.5976%	90.0585%
No	-	147	0.01	92.6036%	67.6471%	71.7456%	90.0585%

Table V1: Accuracy with varying parameters of decision tree on *Vehicle* dataset

An easily seen trend in the decision tree is that the more heavily pruned the tree is, the better it performs at classifying the testing set. This is very likely due to overfitting of the decision tree as the pruning decreases. Decisions trees frequently overfit and will make a whole branch for an outlier. As expected, the gap between the testing and the training data increases as the amount of pruning decreases. The accuracy of the classifier on the training set increased as pruning was decreased due to overfitting and memorization of the training set.



```
  a  b  c  d   <-- classified as
 32  9  0  2 |  a = opel
  4 39  1  0 |  b = saab
  1  1 41  1 |  c = bus
  0  2  0 38 |  d = van
```

Image V1: Confusion matrix of test set with confidence level 0.1

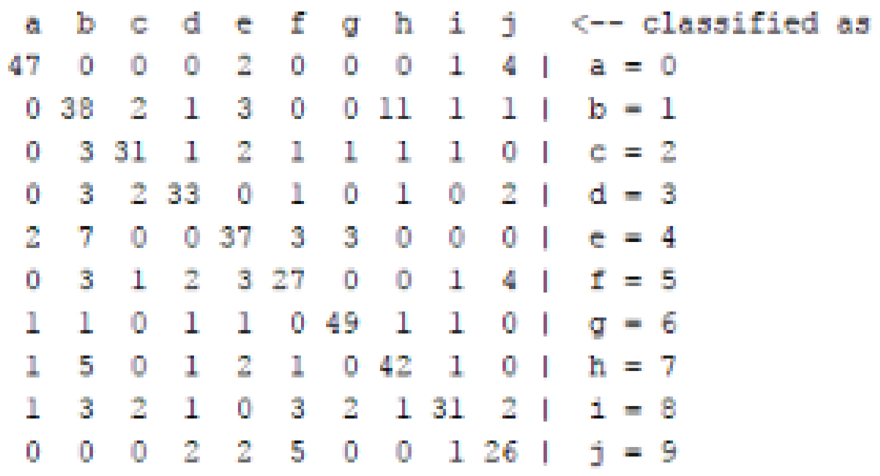
In the confusion matrix, it can be seen that the most commonly confused vehicles were the OPEL and the SAAB. This was expected before the experiment was conducted. This inaccuracy could likely be solved with boosting, as otherwise there were few errors, and boosting would have allowed for the classifier to take on these more difficult cases.

Semeion:

Input		Output					
Pruned?	**Confidence Level**	**Tree Size**	**Total Time(s)**	**Training Correct**	**Testing Correct**	**Training Correct w/ X-Validation**	**Testing Correct w/ X-Validation**
Yes	0.01	213	0.01	91.129%	73.3753%	72.9391%	73.3753%
Yes	0.05	227	0.01	92.0251%	74.4235%	72.8495%	74.4235%
Yes	0.1	249	0	93.2796%	75.0524%	72.7599%	75.0524%
Yes	0.25	259	0	93.2796%	75.6813%	72.7599%	75.6813%
Yes	0.5	137	0.02	93.9964%	75.4717%	72.6703%	75.4717%
No	-	147	0.01	94.1756%	75.2621%	72.491%	75.2621%

Table S1: Accuracy with varying parameters of decision tree on *Semeion* dataset

Once again there is an increase in accuracy of the classifier on the training set as the pruning decreases due to overfitting. Interestingly, the accuracy of the decision tree on the testing set increases for a while as pruning decreases but then began to decreases after a confidence level of 0.25 showing that an optimal amount of pruning exists for this dataset around this confidence level.



```
  a  b  c  d  e  f  g  h  i  j   <-- classified as
 47  0  0  0  2  0  0  0  1  4 |  a = 0
  0 38  2  1  3  0  0 11  1  1 |  b = 1
  0  3 31  1  2  1  1  1  1  0 |  c = 2
  0  3  2 33  0  1  0  1  0  2 |  d = 3
  2  7  0  0 37  3  3  0  0  0 |  e = 4
  0  3  1  2  3 27  0  0  1  4 |  f = 5
  1  1  0  1  1  0 49  1  1  0 |  g = 6
  1  5  0  1  2  1  0 42  1  0 |  h = 7
  1  3  2  1  0  3  2  1 31  2 |  i = 8
  0  0  0  2  2  5  0  0  1 26 |  j = 9
```

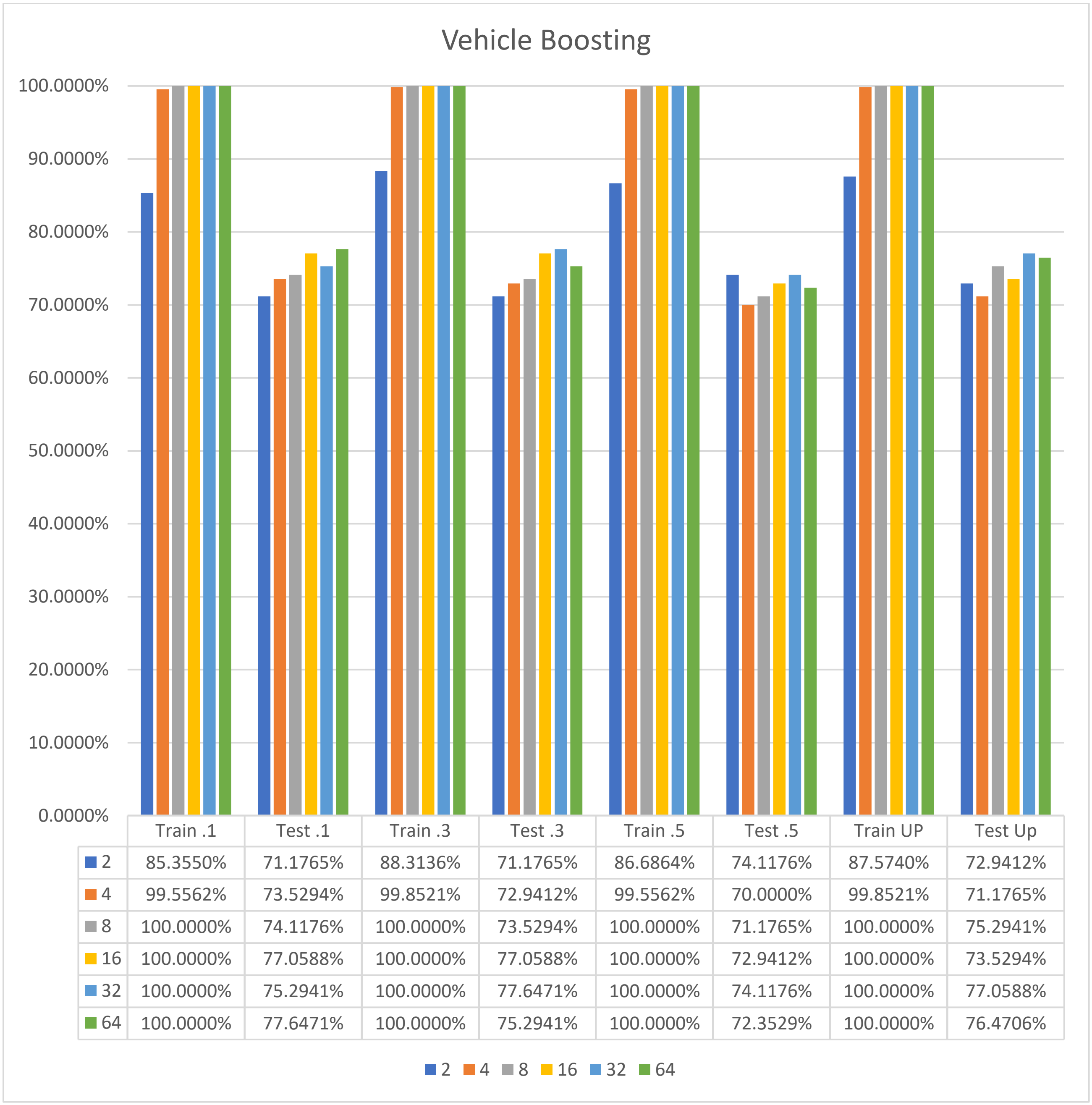
Image S1: Confusion matrix of test set with confidence level 0.25

In **Image S1** above, a confusion matrix of the most accurate test on the testing set lies. Many digits were often misclassified as a 1 as a very common error. Another common error that can be seen is the confusion of the classifier between 1(b) and 7(h). With an increase in instances, it would be expected for this error to dissipate as this is likely caused by a small number of training instances.

Boosting

I used Adaboost on a decision tree for the boosting analysis. Three separate confidence levels were used on the decision tree as well as an unpruned tree. Also, a different number of iterations of the boosting algorithm were run (2, 4, 8, 16, 32, and 64) for each tree.

Vehicle

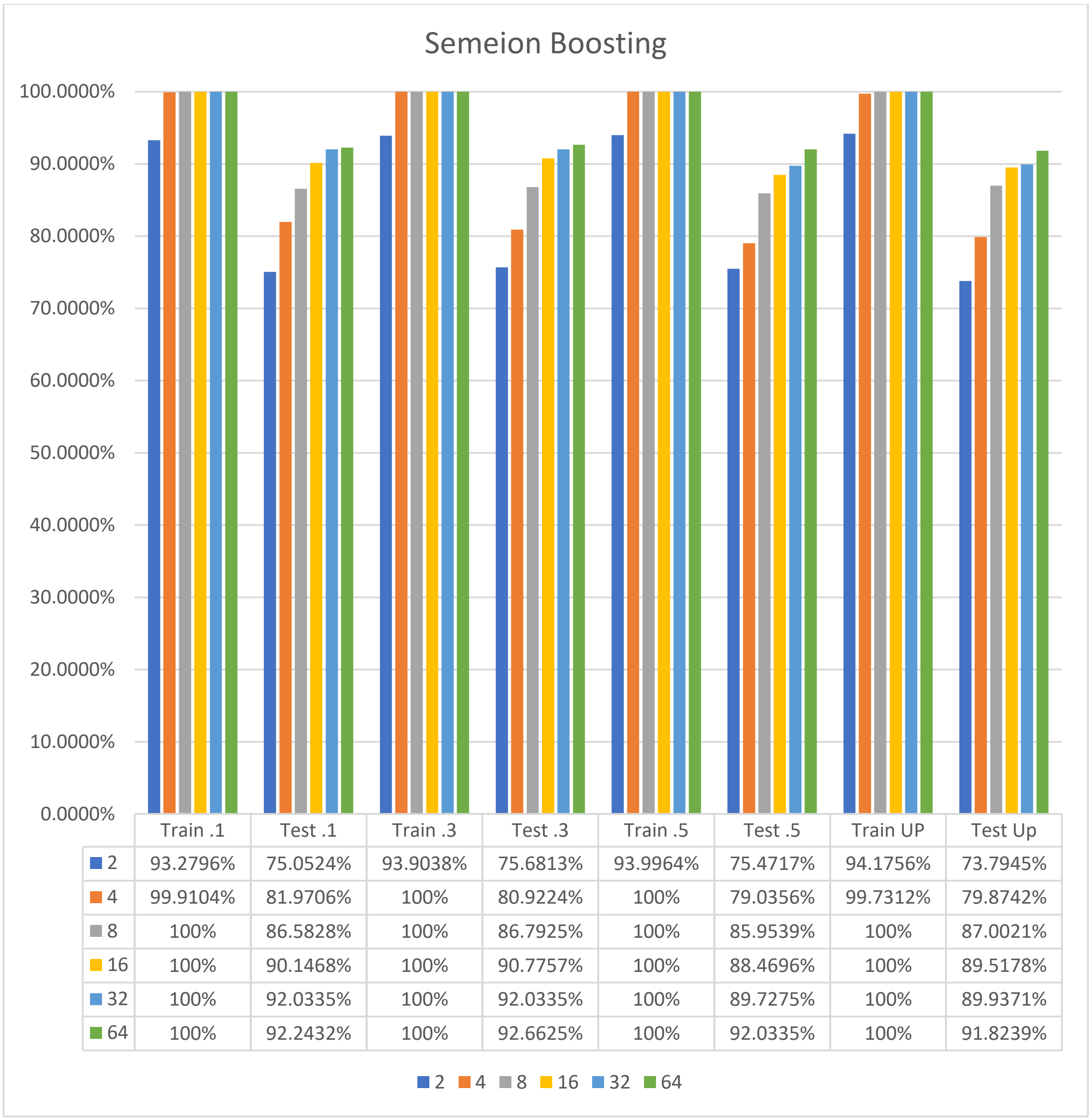


	Train .1	Test .1	Train .3	Test .3	Train .5	Test .5	Train UP	Test Up
2	85.3550%	71.1765%	88.3136%	71.1765%	86.6864%	74.1176%	87.5740%	72.9412%
4	99.5562%	73.5294%	99.8521%	72.9412%	99.5562%	70.0000%	99.8521%	71.1765%
8	100.0000%	74.1176%	100.0000%	73.5294%	100.0000%	71.1765%	100.0000%	75.2941%
16	100.0000%	77.0588%	100.0000%	77.0588%	100.0000%	72.9412%	100.0000%	73.5294%
32	100.0000%	75.2941%	100.0000%	77.6471%	100.0000%	74.1176%	100.0000%	77.0588%
64	100.0000%	77.6471%	100.0000%	75.2941%	100.0000%	72.3529%	100.0000%	76.4706%

Table V2: Accuracy with varying parameters of boosting on decision tree for *Vehicle* dataset

The boosting increased the accuracy of the decision tree on the testing set by about 10 percent. It was most accurate on the testing set with high pruning (confidence level of .1) and at the highest number of iterations at 64. Boosting quickly pushed the accuracy of the classifier on the training set to 100 after eight iterations with any level of pruning. Overfitting did seem to occur on less pruned trees after 32 iterations of boosting.

Semeion



	Train .1	Test .1	Train .3	Test .3	Train .5	Test .5	Train UP	Test Up
🟦 2	93.2796%	75.0524%	93.9038%	75.6813%	93.9964%	75.4717%	94.1756%	73.7945%
🟧 4	99.9104%	81.9706%	100%	80.9224%	100%	79.0356%	99.7312%	79.8742%
⬜ 8	100%	86.5828%	100%	86.7925%	100%	85.9539%	100%	87.0021%
🟨 16	100%	90.1468%	100%	90.7757%	100%	88.4696%	100%	89.5178%
🟦 32	100%	92.0335%	100%	92.0335%	100%	89.7275%	100%	89.9371%
🟩 64	100%	92.2432%	100%	92.6625%	100%	92.0335%	100%	91.8239%

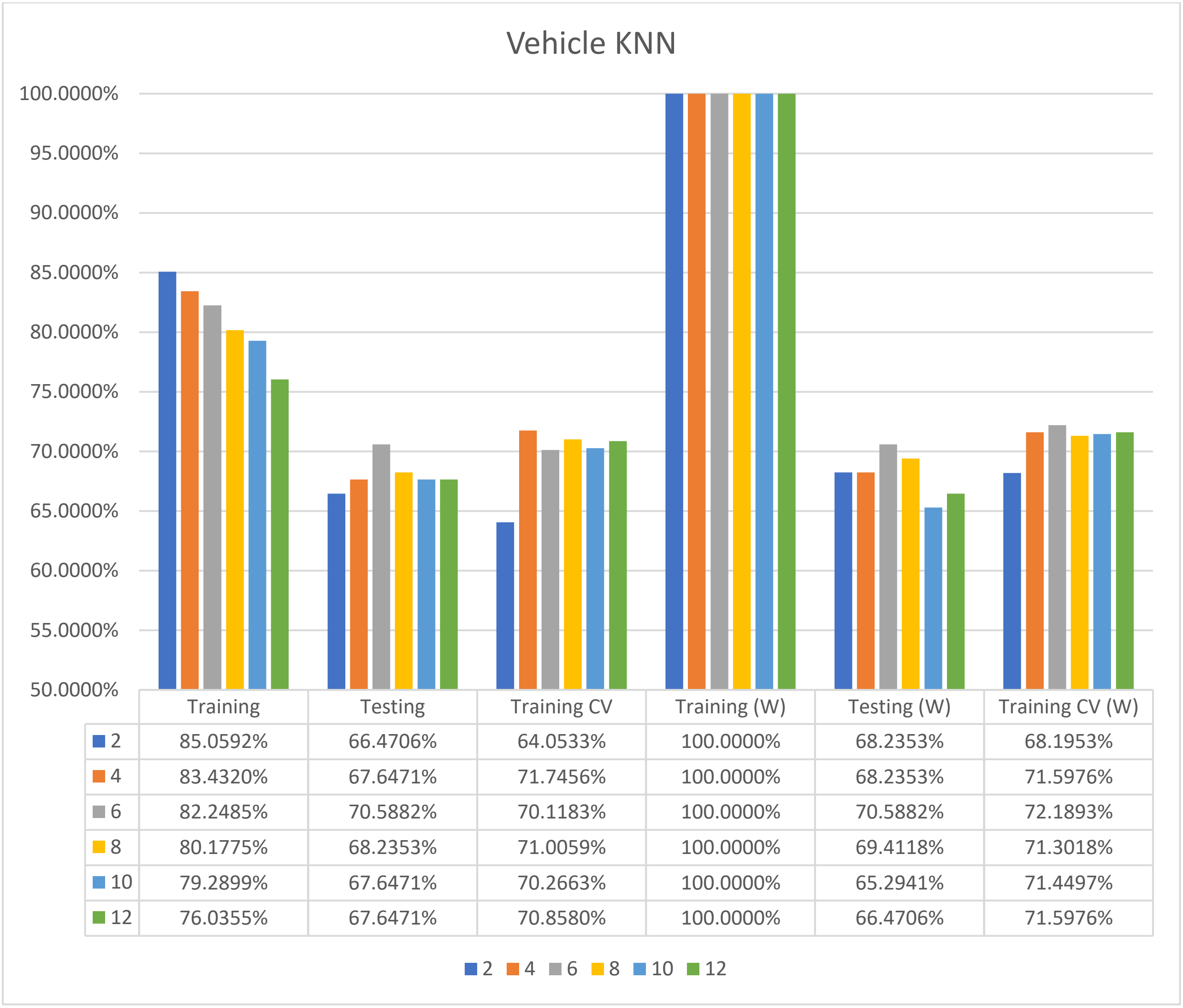
Table S2: Accuracy with varying parameters of boosting on decision tree for *Semeion* dataset

As can be seen consistently up to 64, the accuracy of the classifier increased with the number of iterations of boosting. The boosting seemed to be more effective on moderately pruned trees. This can be seen by looking at how when a confidence level of .3 pruning was used, it was always better than .1, .5, and the unpruned tree. The optimal level of pruning is likely for the boosting is likely between .1 and .3. As expected, the boosting greatly increased the accuracy of the classifier on the testing set.

K Nearest Neighbors

For KNN, I used a variety of different numbers of neighbors ranging from 2 to 12. I did it once with no weighting and once with a weighting of 1/distance.

Vehicle

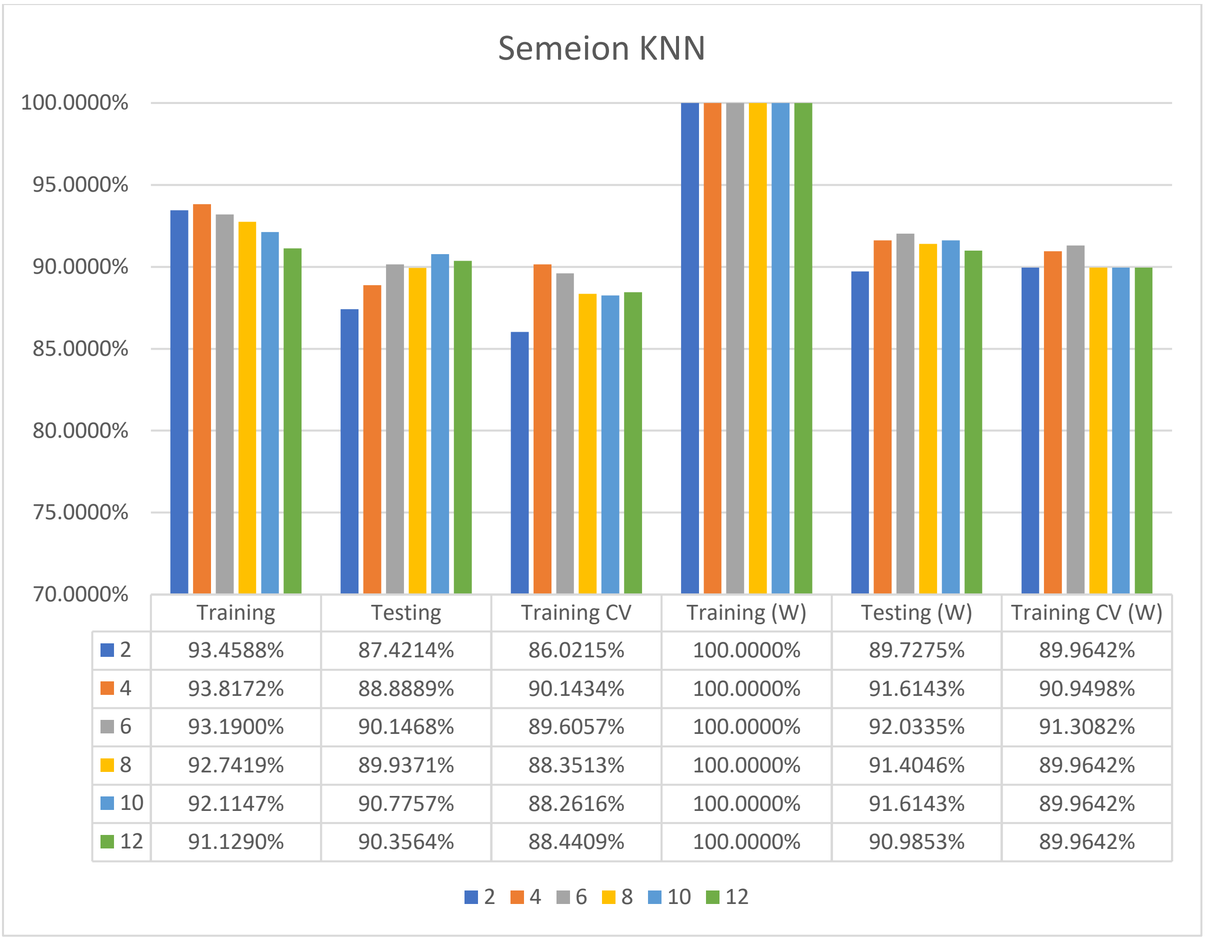


	Training	Testing	Training CV	Training (W)	Testing (W)	Training CV (W)
2	85.0592%	66.4706%	64.0533%	100.0000%	68.2353%	68.1953%
4	83.4320%	67.6471%	71.7456%	100.0000%	68.2353%	71.5976%
6	82.2485%	70.5882%	70.1183%	100.0000%	70.5882%	72.1893%
8	80.1775%	68.2353%	71.0059%	100.0000%	69.4118%	71.3018%
10	79.2899%	67.6471%	70.2663%	100.0000%	65.2941%	71.4497%
12	76.0355%	67.6471%	70.8580%	100.0000%	66.4706%	71.5976%

Table V3: Accuracy with varying parameters of KNN on *Vehicle* dataset

KNN did not do well classifying this dataset. The weighted and the unweighted classifiers performed very similarly on the testing set probably due to the very similar instances (the OPEL and the SAAB). Accuracy peaked at 6 nearest neighbors for both. This is due to the classifier overfitting after 6 neighbors. The classifier seems to be too weak to distinguish between very similar instances.

Semeion



	Training	Testing	Training CV	Training (W)	Testing (W)	Training CV (W)
2	93.4588%	87.4214%	86.0215%	100.0000%	89.7275%	89.9642%
4	93.8172%	88.8889%	90.1434%	100.0000%	91.6143%	90.9498%
6	93.1900%	90.1468%	89.6057%	100.0000%	92.0335%	91.3082%
8	92.7419%	89.9371%	88.3513%	100.0000%	91.4046%	89.9642%
10	92.1147%	90.7757%	88.2616%	100.0000%	91.6143%	89.9642%
12	91.1290%	90.3564%	88.4409%	100.0000%	90.9853%	89.9642%

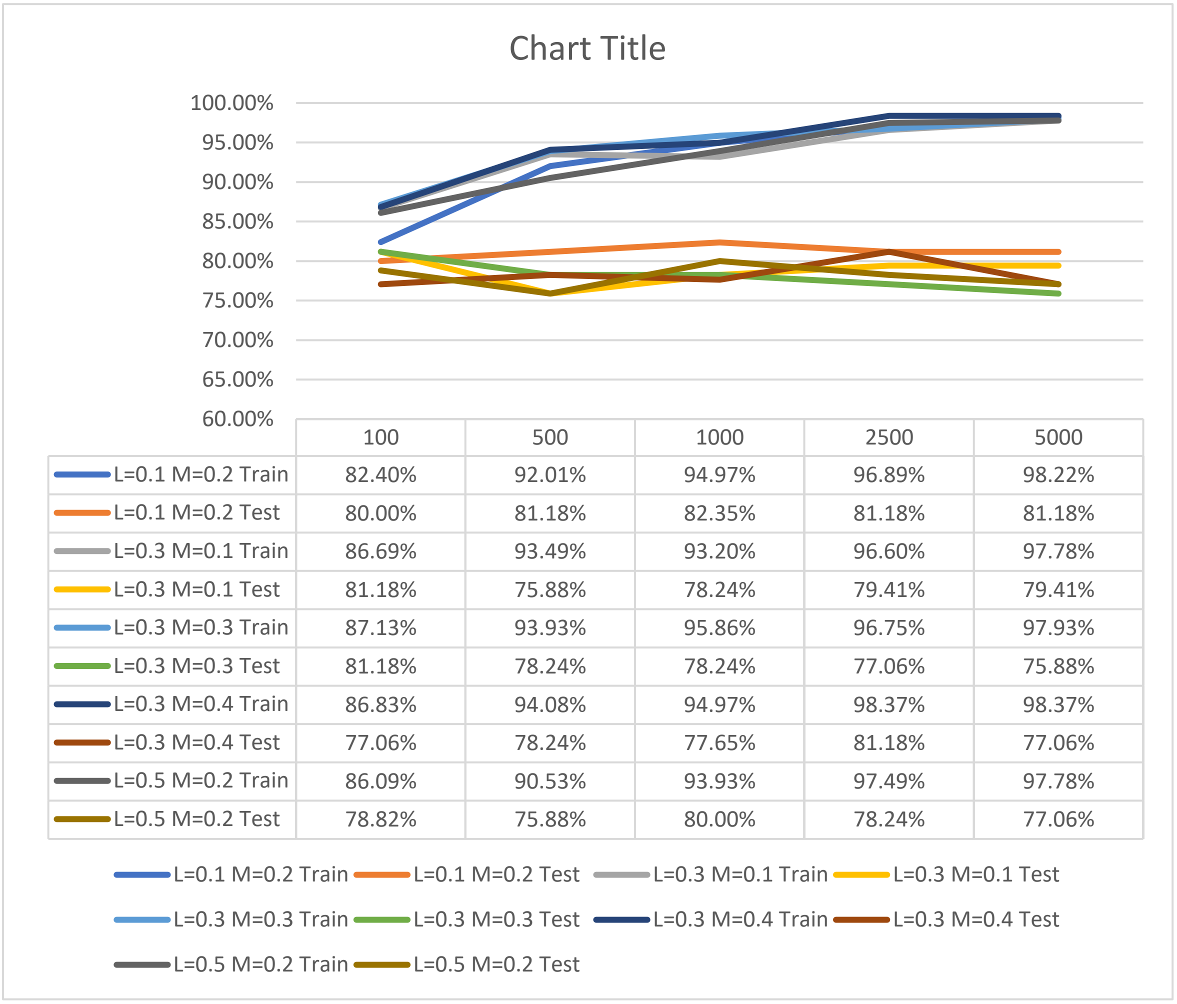
Table S3: Accuracy with varying parameters of KNN on *Semeion* dataset

KNN did a great job classifying this dataset. The accuracy of the classifier without weighting peaked at 12 nearest neighbors and peaked at 6 nearest neighbors with the 1/distance weighting. The discrepancies between the accuracy on the testing and the training decreases as the number of neighbors increases. The weighting added value to the classifier and was able to increase its accuracy. The weighting filtered out some of the noise that would exist due to different handwriting styles.

Neural Net

Five experiments were run on both the training set and the testing set. The momentum and the learning rates were varied so it could be seen how each of those constants affected the accuracy of the algorithm. Runtime was very high on both datasets and especially on the *Semeion* dataset due to the large number of attributes for each instance. Not many epochs were needed to classify the *Semeion* data so the epochs I used were 1, 5, 10, 50, and 100. The *Vehicle* dataset has less attributes so I used many more epochs (100, 500, 1000, and 2500).

Vehicle

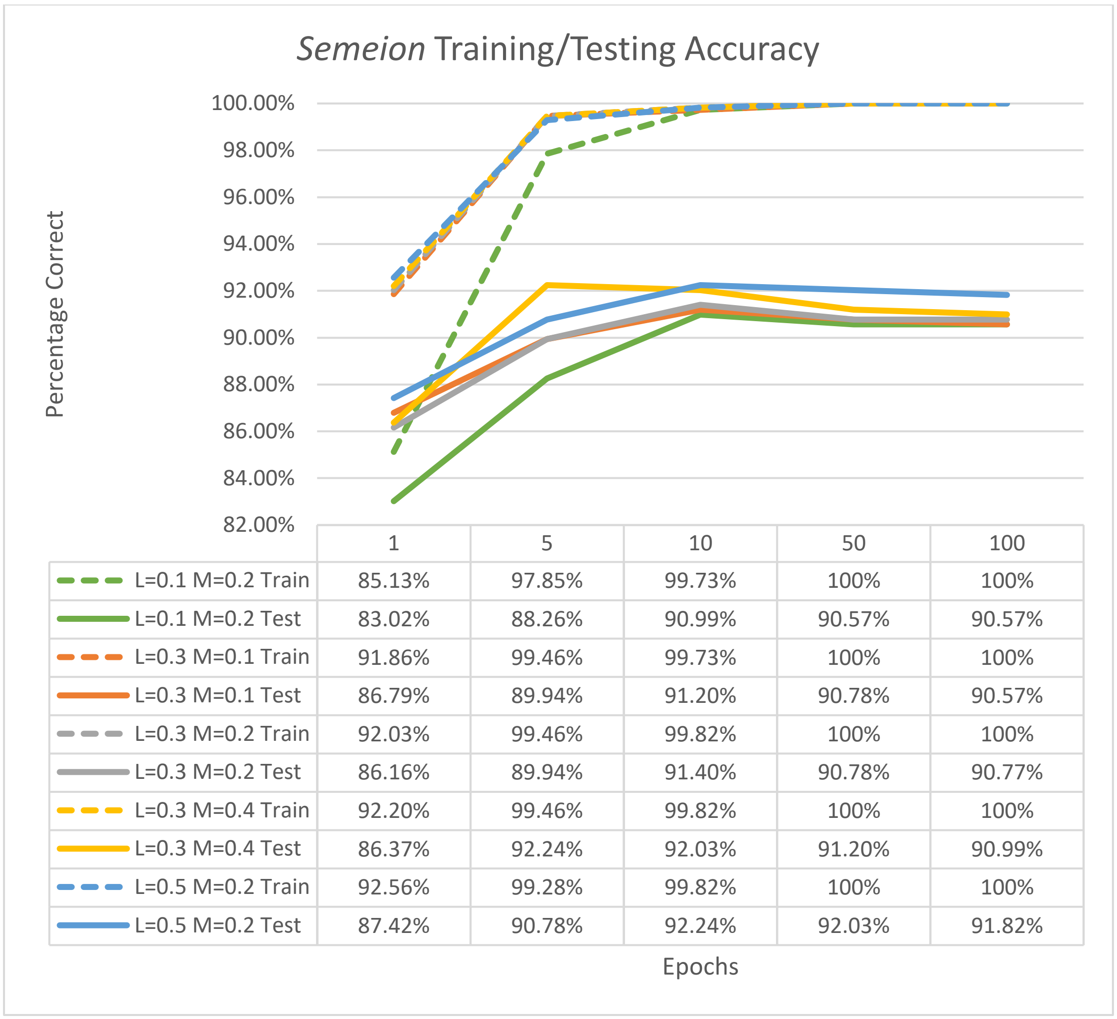


	100	500	1000	2500	5000
L=0.1 M=0.2 Train	82.40%	92.01%	94.97%	96.89%	98.22%
L=0.1 M=0.2 Test	80.00%	81.18%	82.35%	81.18%	81.18%
L=0.3 M=0.1 Train	86.69%	93.49%	93.20%	96.60%	97.78%
L=0.3 M=0.1 Test	81.18%	75.88%	78.24%	79.41%	79.41%
L=0.3 M=0.3 Train	87.13%	93.93%	95.86%	96.75%	97.93%
L=0.3 M=0.3 Test	81.18%	78.24%	78.24%	77.06%	75.88%
L=0.3 M=0.4 Train	86.83%	94.08%	94.97%	98.37%	98.37%
L=0.3 M=0.4 Test	77.06%	78.24%	77.65%	81.18%	77.06%
L=0.5 M=0.2 Train	86.09%	90.53%	93.93%	97.49%	97.78%
L=0.5 M=0.2 Test	78.82%	75.88%	80.00%	78.24%	77.06%

Table V4: Accuracy with varying parameters of a neural network on *Vehicle* dataset

The neural net did well classifying this dataset. The momentum very much influenced when the neural net began to overfit. The higher the momentum, the more epochs it took for the neural net to overfit. With the fixed momentum of .2, the neural net began to overfit after 1000 in all cases but one (learning rate of .3) where it overfit after 100 epochs. As the learning rate increased, the accuracy of the classifier on the testing set decreased. The classifier with learning rate of .1 and momentum of .2 classified this data the best. The neural network classified this dataset the best of all of the classifiers likely due to the data's complexity and the network's ability to fit complex functions.

Semeion



	1	5	10	50	100
L=0.1 M=0.2 Train	85.13%	97.85%	99.73%	100%	100%
L=0.1 M=0.2 Test	83.02%	88.26%	90.99%	90.57%	90.57%
L=0.3 M=0.1 Train	91.86%	99.46%	99.73%	100%	100%
L=0.3 M=0.1 Test	86.79%	89.94%	91.20%	90.78%	90.57%
L=0.3 M=0.2 Train	92.03%	99.46%	99.82%	100%	100%
L=0.3 M=0.2 Test	86.16%	89.94%	91.40%	90.78%	90.77%
L=0.3 M=0.4 Train	92.20%	99.46%	99.82%	100%	100%
L=0.3 M=0.4 Test	86.37%	92.24%	92.03%	91.20%	90.99%
L=0.5 M=0.2 Train	92.56%	99.28%	99.82%	100%	100%
L=0.5 M=0.2 Test	87.42%	90.78%	92.24%	92.03%	91.82%

Table S4: Accuracy with varying parameters of a neural network on *Semeion* dataset

The neural net did well classifying the data and quickly reached 90% accuracy as epochs increased with all learning rate and momentum variations. Accuracy peaked mostly at 10 epochs but also at 5 epochs in some instances. For epochs greater than that the classifier probably overfit the data. An increased learning rate with a constant momentum seemed to increase the accuracy of the classifier on the testing set. Having fixed learning rate and increasing the momentum showed that less epochs were needed to reach a peak with a higher momentum. The higher momentum also caused the classifier to overfit at lower epochs. The neural network with the learning rate equal to .5 and the momentum equal to .2 seemed to perform the best on the testing set. The high learning rate could have hurt the classifier if the data was more noisy but since it is not very noisy, learning the data quickly was advantageous to the classifier. Overall, the neural network did a great job classifying relative to other classifiers on the *Semeion* dataset.

Support Vector Machine

I ran two experiments on each dataset. One of the experiements was with polynomial kernel with varying exponents and the other was with RBF kernel with a varying gamma value. For the polynomial kernel I chose values of 1, 2, 2.5, 3, 3.5, 4, and 5 in order to find the best exponent.

Vehicle

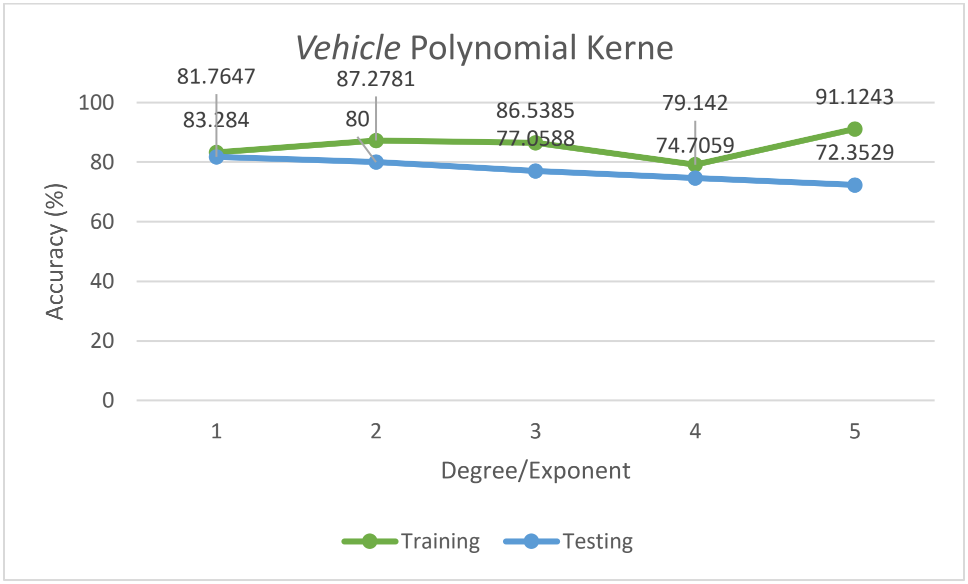


Table V5: Accuracy with varying parameters of a polynomial kernel SVM on *Vehicle* dataset

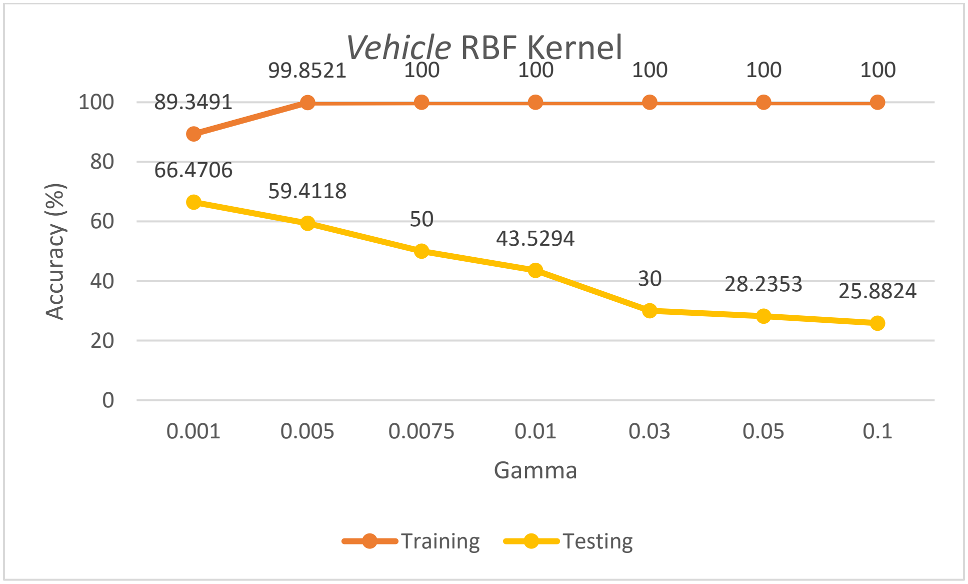


Table V6: Accuracy with varying parameters of a RBF kernel SVM on *Vehicle* dataset

Support vector machines performed well on the dataset. Due to the small number of attributes, both of the classifiers used quickly overfit. The polynomial classifier was the most accurate when the exponent was lower with the highest at 1. The RBF kernel performed worse but did its best with the low gamma of 0.001. The polynomial kernel seems to be much better suited for this data set's classification.

Semeion

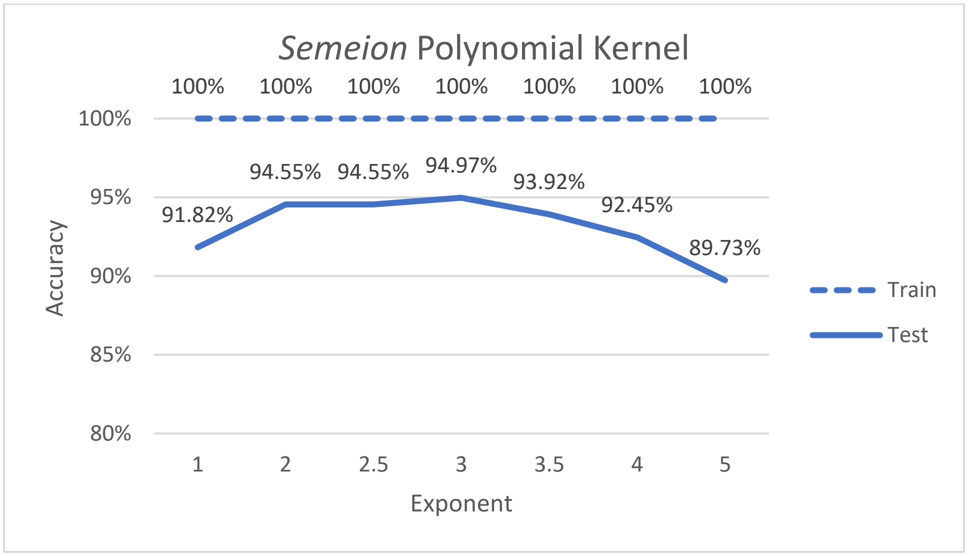


Table S5: Accuracy with varying parameters of a polynomial kernel SVM on *Semeion* dataset

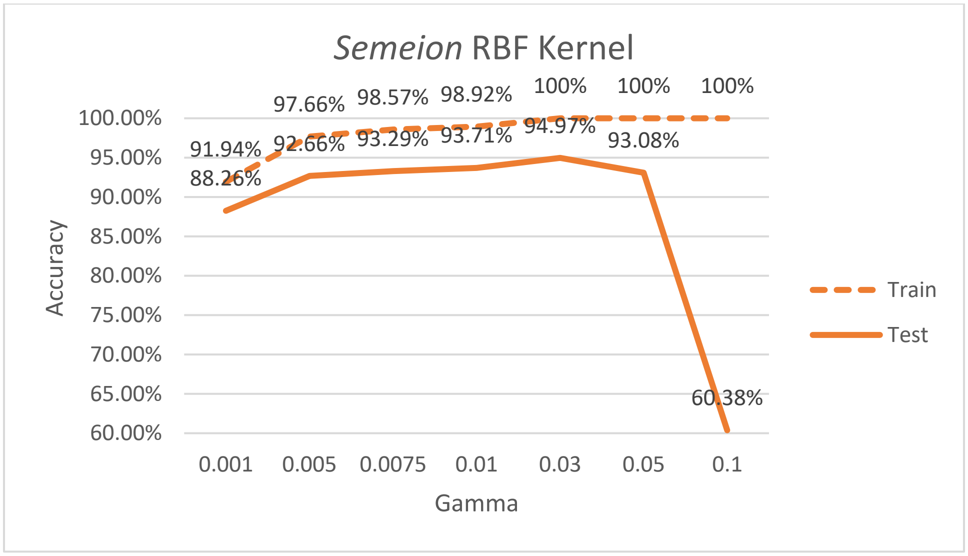


Table S5: Accuracy with varying parameters of a RBF kernel SVM on *Semeion* dataset

Support vector machines outperformed all the other classifiers on this dataset. Three was the exponent that yielded the best accuracy for the polynomial kernel test. The polynomial kernel began to overfit slowly after exponent 3, while the RBF kernel quickly overfit after a gamma of 0.05 as it approached 0.1. The large amount of attributes this dataset had is what I believe made the support vector machine so accurate. It is interesting to note that the RBF and the polynomial kernel both had the same peak accuracy of 94.97%.

Conclusion

The support vector machine performed the best on the *Semeion* dataset and on the *Vehicle* dataset closely followed by the neural net. Neural networks and support vector machines also took by far the longest time to run, sometimes exceeding a minute to run on the *Semeion* dataset.

Cross validated classifiers always performed the same on the testing set as the un cross validated classifiers; however, using the cross validated classifier on the testing set always yielded an accuracy extremely close to the accuracy of the un cross validated classifier on the testing set. This is interesting as it can be used to predict the accuracy of the classifier on a testing set. I included this here instead of in the individual analyses of each classifier as it was shared across every case. Cross validation also took extremely long times especially when combined with an expensive classifier such as a neural net or a support vector machine.