AI201 PA 4 Report
Comparison of Boosted Perceptrons and SVM

Jemima Bian T. Anila (2015-05643)

Date of Submission: May 9, 2024

1. INTRODUCTION

Support Vector Machines (SVM) is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space. It is used to solve classification and regression problems.

Ensemble learning is the method of combining different classifiers, called inducers, to achieve a performance that is better than an individual inducer. Adaptive Boosting (AdaBoost) is the use of weak learners - classifiers that are slightly better than random guessing with errors less than 50% - in an ensemble to produce errors that are arbitrarily close to zero.

This paper attempts to compare the performance of Adaboost and SVM in classifying banana and splice datasets. The Adaboost uses perceptron classifiers as its weak learner and uses the pocket algorithm since the data is not linearly separable. Pocket Algorithm is a variant of the perceptron algorithm wherein a new set of weights that can give a better error rate than previous iterations are stored.

SVM can have different kernel types such as a linear kernel for linearly separable data, polynomial kernel for non-linear data wherein the degree of the polynomial is specified as the degree parameter, sigmoid kernel which maps the input into a higher-dimensional space, and Gaussian/RBF kernel which applies a radial basis function for non-linear data wherein the width of the kernel is specified as the gamma parameter.

In SVM, a higher degree parameter for a polynomial kernel leads to more complex decisions boundaries which can potentially cause overfitting without proper regularization. Similarly, a low gamma parameter for a Gaussian/RBF kernel assigns more influence to points far away from the decision boundary leading to a smoother boundary.

2. OBJECTIVES

The objectives of the assignment are the following:

1. Implement a perceptron learner and test its performance on a binary classification problem.
2. Construct an ensemble of perceptrons that use Adaboost and plot training and test accuracies vs K for the banana and splice datasets.
3. Run the SVM classifier on the banana dataset and find the kernel and kernel parameters with the best performance.
4. Compare the performance of SVM and Boosted Perceptron for the banana and splice datasets in terms of test accuracy, training speed, and test speed.

3. METHODOLOGY

3.1 Perceptron Classifier Construction

First, I defined a function that allows me to generate synthetic data for binary classification. The data consists of 100 samples for each class from two different normal distributions centered at [0,0] and [10,10], respectively. The function returns shuffled data and labels ready for training and testing.

Next, I defined the classify function to train a perceptron using the pocket algorithm with the given number of iterations. This returns the best pocket weights found during training.

Then I defined the predict function which uses the perceptron model to make predictions and calculate the sum of squared errors for the test set.

Finally, I constructed a main function that allows me to generate my synthetic data, train the perceptron and retrieve the best weights, and predict and calculate the sum of squared errors on the test set.

3.2 Adaboost Construction and Evaluation

Here I first created a function `adabtrain` that allows me to sample a new training set St from S with replacement according the instance weights. Then I train the weak learner on St to obtain a hypothesis ht using the *classify* function. The training error et of ht on the full dataset S is then calculated and all learners with error rate greater than 50% are removed from the ensemble. The inducer's weight, *alphat*, is calculated using its error rate and the instance weights for the next iteration are updated. This whole function returns the list of classifiers and their corresponding weights.

Next I created a function `adapredict` that provides the final prediction by taking the sign of the weighted predictions from all classifiers in the ensemble.

Then I create a function `load_dataset` that loads data from a CSV file into a DataFrame, extract features and labels,

and converts them to 0 to -1 to fit the Adaboost label requirements, and returns the processed features and their corresponding labels.

Finally, I run the Adaboost algorithm on the banana and splice data sets keeping only the first 400 samples as training data for banana set, first 1000 samples as training data for splice set, and the rest as testing samples. I plotted the accuracy and the time it took to train and test over a range of number of learners K to compare performance.

3.3 SVM Classifier

I first created a function to read the LibSVM-formatted data, converting the feature index to an array index. I then created a function to split the dataset into training and testing sets, separating the features from the labels. I used GridSearchCV to find the best hyperparameters for the different kernel types. I then created functions to train and test the model, and evaluate its performance from the sum of its correct predictions. Finally, I run the SVM model over the two datasets and the different kernel types and return the best parameters and kernel type. The different kernel types used are linear, polynomial, sigmoid, and Gaussian (RBF).

4. ANALYSIS AND DISCUSSION OF RESULTS

The initial perceptron classifier yielded an SSE of 12.0 on the synthetic test data.

Figures 1 to 4 show the accuracy and time for the banana and splice dataset using Boosted Perceptrons (AdaBoost). Despite only having 400 training samples, the banana dataset's training accuracy was high for varying number of learners. This means the model has effectively learned from the training data. However, its testing accuracy only reached 54.59% which means it is only slightly better than random guessing (50% accuracy). This might be due to the smaller number of training samples for the banana dataset (400) which is less than half its total sample size (4900). It is worth noting the high variance in training times despite the obvious increasing pattern. At certain numbers of learners, the training time increases by 1 second from the usual trend.

As shown in Figure 3, there is an obvious increase in training accuracy as the number of learners increase for the splice dataset. The testing accuracy was higher than the banana dataset at around 81.97% which means it performs better than random guessing. There are also no abrupt changes in the training times for the splice dataset and only took up to a maximum of 1 second for the largest number of learners.

Out of all four kernels, the Gaussian or RBF Kernels at the specified parameters in Figure 5 and 6 yielded better results than the other kernel types in both banana and splice datasets.

In the banana dataset, SVM was able to reach a higher testing accuracy than Adaboost even with the use of its least accurate kernel (SVM Linear Kernel: 55.31% vs AdaBoost: 54.59%). In the splice dataset, the difference in performance

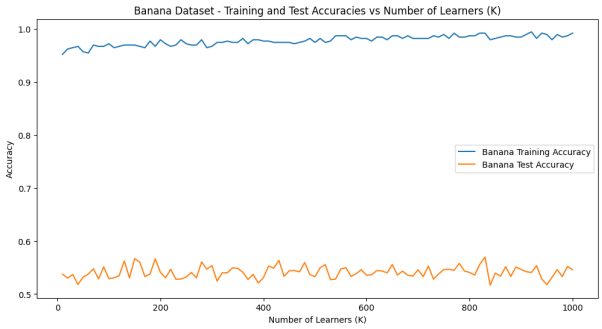

Figure 1: Banana Dataset: Accuracy slightly increases from 95% to 99.25% as the number of learners increase from 0 to 1000 during training but remains relatively low and the same around 54.59% during testing.

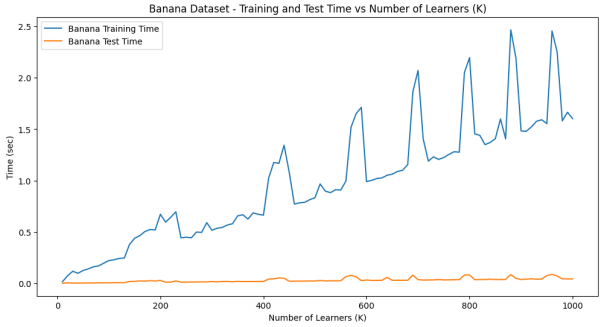

Figure 2: Banana Dataset: It takes slightly longer from 0 to 1.6 seconds as the number of learners increase from 0 to 1000 during training but remains negligible during testing.

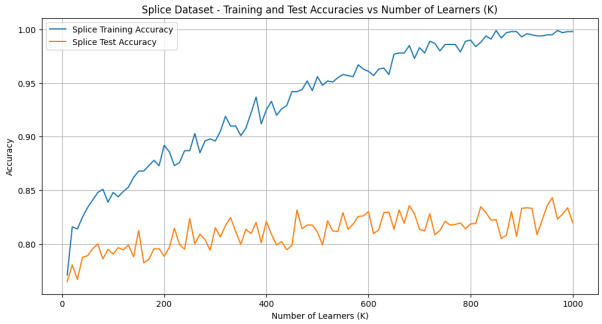

Figure 3: Splice Dataset: Accuracy significantly increases from 80% to 99.8% as the number of learners increase from 0 to 1000 during training and remains relatively high at around 81.97% during testing.

is not that significant (SVM Linear Kernel: 83.02% vs AdaBoost: 81.97%).

In terms of training time, AdaBoost reached up to 2.5 seconds in some learner sizes with the banana dataset but in general took less than 2 seconds to train. SVM takes slightly longer to train and at one point reached 75.61 seconds (lin-

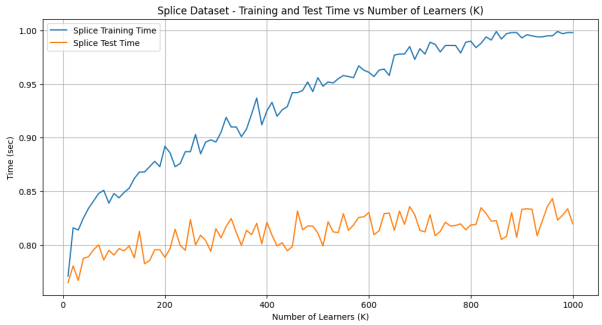

Figure 4: Splice Dataset: There is not much increase from 0.8 to 1 second as the number of learners increase from 0 to 1000 during training. There is also not much change during testing.

```
Banana Dataset:

Linear Kernel:
Best parameters for Linear Kernel: {'C': 0.1}
Accuracy: 55.31 %
Train Time: 1.68 seconds
Test Time: 0.08 seconds

Sigmoid Kernel:
Best parameters for Sigmoid Kernel: {'C': 0.1, 'gamma': 0.001}
Accuracy: 55.31 %
Train Time: 0.96 seconds
Test Time: 0.07 seconds

Polynomial Kernel:
Best parameters for Polynomial Kernel: {'C': 1, 'degree': 2, 'gamma': 1}
Accuracy: 67.12 %
Train Time: 5.89 seconds
Test Time: 0.07 seconds

Gaussian Kernel:
Best parameters for Gaussian Kernel: {'C': 1, 'gamma': 1}
Accuracy: 89.16 %
Train Time: 0.89 seconds
Test Time: 0.06 seconds
```

Figure 5: Banana Dataset: The best performing kernel is the Gaussian kernel with parameters C=1 and gamma=1 which achieved an accuracy of 89.16%, training time of 0.89 seconds, and testing time of 0.06 seconds.

```
Splice Dataset:

Linear Kernel:
Best parameters for Linear Kernel: {'C': 0.1}
Accuracy: 83.02 %
Train Time: 75.61 seconds
Test Time: 0.04 seconds

Sigmoid Kernel:
Best parameters for Sigmoid Kernel: {'C': 10, 'gamma': 0.001}
Accuracy: 83.73 %
Train Time: 3.06 seconds
Test Time: 0.13 seconds

Polynomial Kernel:
Best parameters for Polynomial Kernel: {'C': 0.1, 'degree': 3, 'gamma': 0.1}
Accuracy: 83.12 %
Train Time: 12.64 seconds
Test Time: 0.08 seconds

Gaussian Kernel:
Best parameters for Gaussian Kernel: {'C': 10, 'gamma': 0.01}
Accuracy: 88.90 %
Train Time: 4.77 seconds
Test Time: 0.19 seconds
```

Figure 6: Splice Dataset: The best performing kernel is the Gaussian kernel with parameters C=10 and gamma=0.01 which achieved an accuracy of 88.90%, training time of 4.77 seconds, and testing time of 0.19 seconds.

ear kernel with the splice dataset). There can be issues with this when scaling, but for the datasets in this study, it was not a significant issue.

In the cases in this study, SVM model performed better in terms of accuracy but slightly worse in terms of training time.

Note that the Boosted Perceptron shows a tendency to overfit the training data, given the very high training accuracies compared to the test accuracies. Overfitting means that the model has learned the training data too well, including its noise and outliers, which can negatively impact the performance on the test set.

5. CONCLUSION

Adaboost and SVM, two high-performance classification methods, were used on banana and splice datasets and their accuracy and training times were compared.

SVM performed better than Adaboost in terms of accuracy but Adaboost trained data slightly faster.

6. REFERENCES

[1] Prospero C. Naval, Jr. 2024. AI 201 AY 2023-2024 Lectures Slides: Support Vector Machines and Classifier Ensembles.