# Multiclass Support Vector Machines


## Abstract

For my project I did a review and comparison of three Multiclass Support Vector Machine algorithms. For the algorithms I compared One-vs-All, an algorithm developed by Lin, Lee, and Wahba, and finally an algorithm developed by Crammer and Singer. I ran some experiements using different sizes of training sets and report the findings.


## 1 Introduction

Support Vector Machines (SVM's) are an excellent tool used for classification of data. However, as the techniques have been research more and more, there have been developments leading to the creation of Multiclass Support Vector Machines. For my project I considered some different approaches to the multiclass problem. The simplest version of this problem is when one has two classes, this is the standard SVM problem. However, when one is considering more than two classes is when the problem extends into the realm of multiclass SVM's.

I looked at three specific algorithms developed to solve multiclass SVM's. The first I looked at was the one-vs-all or one-vs-rest approach. This was a natural first choice and provided me with a baseline in how I should consider the next approaches. The second approach I looked at was an algorithm developed by Lin, Lee, and Wahba. This approach provided a more complex was of considering the multiclass problem. The final approach I considered was developed by Crammer and Singer. This was the most mathematically intensive approach I looked at over the course of my project.

## 2 Background

In order to understand the multiclass SVM problem, one must first understand the standard SVM problem. In this section I will briefly cover standard SVM's using binary classification and its benefits.

SVM's work to solve the problem of how one can classify a data set into two different classes. This do this by training to develop a decision boundary and then classify points based on this decision boundary. The decision boundary is found as a hyperplane through the input space of the data set. The hyperplane can be generalized into nonlinear functions. The desired function is of the form $f(x) = h(x) + b$, where $h(x)$ exists in a reproducing kernel Hilbert Space, and $b$ is a constant minimizing (figure 1).

$$\frac{1}{n}\sum 1 - y_i f(x_i))_+ + \lambda||h||^2$$

Figure 1: Hinge Loss function

Where $(x)_+ = max(x, 0)$ and $||h||^2$ is a square norm function.

MIT 9.520/6.860: Statistical Learning Theory and Applications (Fall 2016).

Multiclass SVM's look to extend the utility of SVM's by adding the ability to look at multiple classes when considering the problem of classifying a data set. Multiclass SVM's have the same purpose as SVM's in the sense that they want to classify a data set into multiple different classes. The natural starting point for this is to look at the case where we continue to use a binary classifier, but instead of one binary classifier, we look at many binary classifier and consider how to get the most accurate results.

## 3 Review of Algorithms

In this section I will review the algorithms I considered for MSVM's. For my project I focused on One-vs-All (OVA), Lin,Lee, and Wahba (LLW), and Crammer and Singer (CS).

### 3.1 One-vs-All

One-vs-all is the first algorithm that I explored. This algorithm also represents the simplest of the algorithms that I looked at. The basic idea is that one construct $N$ binary classifies and then uses each of these classifiers to classify the data points. The classifiers are trained such that $n$-th binary classify is trained using data from the $n$-th class as positive examples and the remaining classes are negative examples. Then on the test set the classifier that is used is the classifier that returns the highest output. This approach leads to the following classifier (figure 2).

$$f(x) = argmax(f_i(x))$$

Figure 2: Classification Function for OVA

This approach is very simplistic due to the fact that is still uses binary classifiers instead of any other method. It works in the multiclass case due to the use of multiple binary classifiers.



### 3.2 Lin, Lee, and Wahba

The next approach I looked at was developed by Lin, Lee, and Wahba. In their approach they wanted to capitalize off the idea that SVM's use a Bayes Decision rule in their classification. They concluded that if one takes the asymptotical bounds of the standard SVM solution, one will get the equation in figure 3 [1].

$$f(x) = sign(p(x) - \frac{1}{2})$$

Figure 3: Asymptotical Solution

From this one can see that OVA works well for regions where the $p > \frac{1}{2}$ , but won't be able to return a definite class in other regions. To help solve this problem LLW developed the following idea: define $v_i$ as an $N$ dimensional vector with 1 in the $i$-th coordinate and $\frac{-1}{N-1}$ everywhere else. This vector acts as the desired vector for points in the $i$-th class. So, for a give $x$ one wants $f_i = \frac{-1}{N-1}$ for $j \neq y_i$. This leads to the equation in figure 4.

$$min_{f_1,..,f_N \in H_K} \frac{1}{l} \sum_{i=1}^{l} \sum_{j=1, j \neq y_i} (f_j(x_i) + \frac{1}{N-1})_+ + \lambda \sum_{j=1}^{N} ||f_j||_K^2$$

Figure 4: Minimization Problem

Which taken asymptomatically results in the following classifier (figure 5).

$$f(x) = argmax(p_j(x))$$

Figure 5: Asymptomatical Classifier

### 3.3 Crammer and Singer

The final algorithm that I looked at was developed by Crammer and Singer. To start their algorithm, they changed how they consider the misclassification error of an example to a piecewise linear bound (figure 6).

$$max\{\bar{M}_r\bar{x} + 1 - \delta_{y,r}\} - \bar{M}_y\bar{x}$$

Figure 6: Piecewise linear bound

Where $\delta$ is equal to 1 if $p = q$ and 0 otherwise. With this bound they suffered loss which was linearly proportional to the difference between the confidence of the right label and the maximum of the confidence of the rest of the labels [2]. Using this bound and taking the $l_2$ norm of the matrix $M$, they derived the following equation (figure 7).

$$min\frac{1}{2}\beta||M||_2^2 + \sum_{i=1}^{M}\epsilon$$

Figure 7: $l_2$ norm of $M$

They then had to solve the optimization problem, so they took the Lagrangian and derived the following classifier (figure 8).

$$H(\bar{x}) = argmax_{r=1}^{k}\sum_{i}\tau_{i,r}K(\bar{x},\bar{x}_i)$$

Figure 8: Crammer and Singer Classification Rule

Where $\tau$ is the difference between the distribution point and the distribution obtained by the optimization problem.



## 4 Numerical Analysis

### 4.1 Experiment

For the testing of these algorithms I created a data set that has two feature vectors with three possible classes. The feature vectors were chosen at random within a range so that they would be all clustered around the same area, but to also allow overlapping in where the features are in the plot. The following figure (figure 9) is the full 3000 data points with their respective classes show in red, blue, and green.

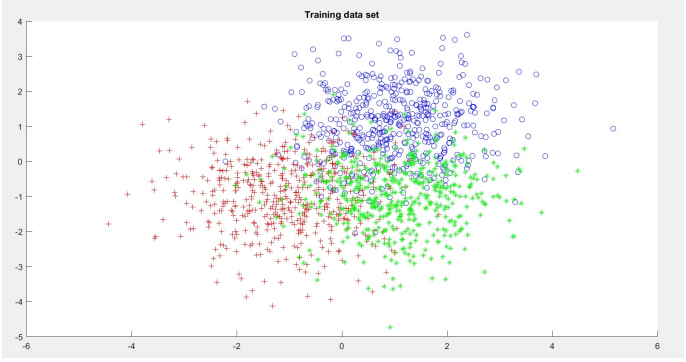


Figure 9: Training Set

Figures 10, 11, and 12 all show the different outputs of the respective multiclass algorithms. They were all trained using 300 points and then used to classify the rest of the training set.

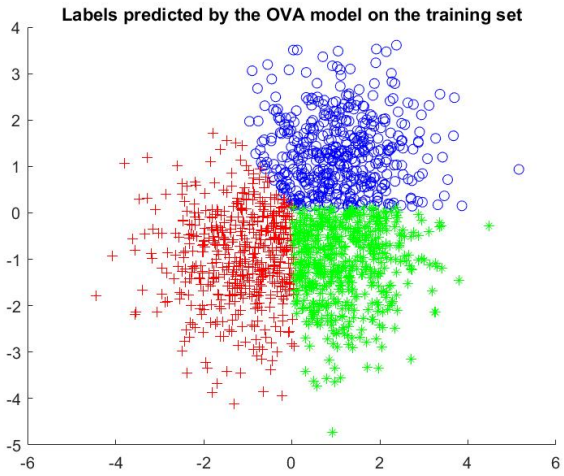


Figure 10: One-vs-All

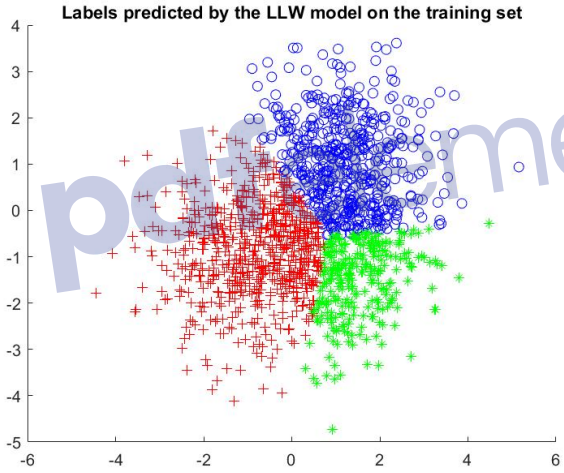


Figure 11: Lin, Lee, and Wahba

The data was then split into subsets where the number of training points were varied in order to get an idea of the effectiveness of each of these algorithms as the number of training points were varied (figure 13 and 14). I measured effectiveness by looking at the loss found by doing a k-fold analyses on the model produced by the algorithm. The algorithms were run using a linear kernel.

The algorithms were implemented using a Matlab Library called MSVMpack1.5. I used this library to implement Lin, Lee, and Wahba along with Crammer and Singer. For OVA, I used a built in MatLab function.

On the graph blue is One-vs-All, red is Lin, Lee, and Wahba, and green is Crammer and Singer. For the first trial I started the training set at 300 points and increased it by 30 for every subsequent trial. The y-axis is the error reported by each algorithm using a k-fold algorithm to test. For the k-fold I used 3 folds to try to ensure accuracy while compansating for the lack of processor power my laptop could provide.

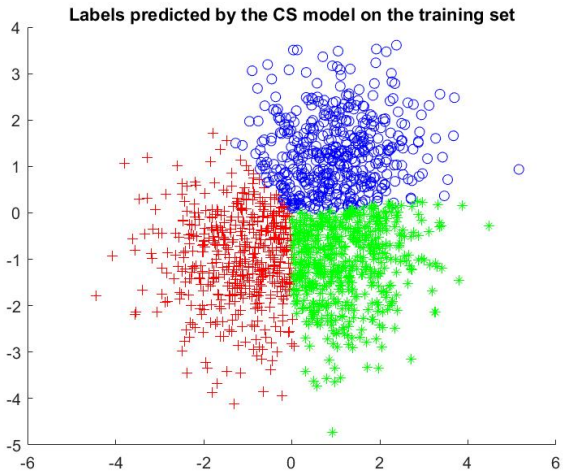


Figure 12: Crammer and Singer

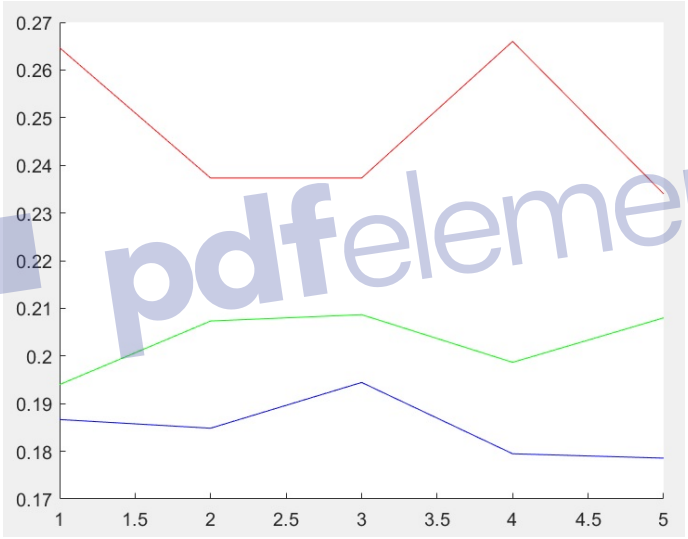




Figure 13: Graph of The Error for 5 Trials

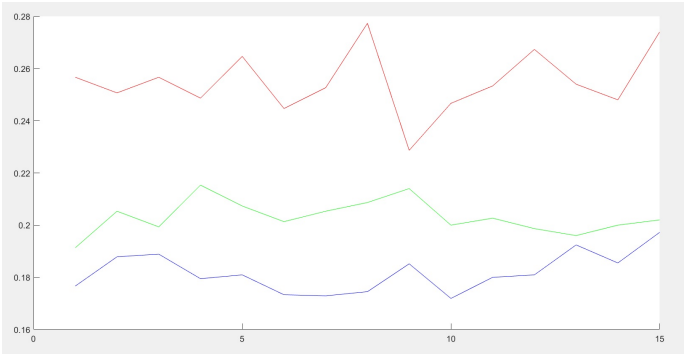


Figure 14: Graph of The Error for 15 Trials

### 4.2 Comparison

#### 4.2.1 Error

Inspecting the graph (figure 10), one can see that OVA (blue) consistantly has the lowest error of the three algorithms. However in the figure 11, one can start to see OVA having a general trend up as the number of training points is increased. CS (green) tends to stay around the same error while LLW (red) seems to vary more.

#### 4.2.2 Time

From a perforance stand point, I used my laptop to run these algorithms in Matlab. I used a library that has an implementation of LLW and CS in it. From my observations, OVA performed the fastest. For OVA I used an implementation that Matlab already has in its standard libraries. After OVA, LLW performed next fastest with CS coming in at the slowest.

## 5 Conclusion

Overall, OVA seems to work the best. It constistantly has the lowest error and the quickest run time. It also comes in at the simplest of the three algorithms I looked at. In the paper by Crammer and Singer, they claim that their algorithm outpreforms OVA on the data set that they tested [2]. My finds do not seem to agree with their finding. Lin, Lee, and Wahba don't make any claims about the performance of their algorithm. All-in-all One-vs-all seems to be the better of the algorithms I looked at.

**Acknowledgments**

None.



## References

[1] Lee, Y., Lin, Y. & Wahba, G. (2004) Multicategory Support Vector Machines: Theory and Applicaiton to the Classification of Microarray Data and Satellite Radiance Data.*American Statistical Association* pp. 67-81.

[2] Crammer, K. & Singer, Y. (2001) On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines *Jouranal of Machine Learning Research.* pp. 265-292