# AdaBoost and SVM's for Unbalanced Data Sets

Matthew J. Urffer
University of Tennessee
Knoxville, Tennessee, 37916
Email: matthew.urffer@gmail.com

***Abstract*—**

## I. Introduction

A supervised machine learning problem is one which a learning algorthim is presented a set of training data and attempts to find an unkown function which maps the training values to the correct answer. Typically the training set, denoted $S$, is a set of the form $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$ where $\vec{x}_i$ is vector of some features of the problem. Examples of problem features include discrete or real valued items such as height, weight, age, zip code, grade point average, starting salary, and telephone number (as just a few) which might make up the features of a person. The $y_i$ are the class of the training feature $\vec{x}_i$ belongs to; these might be University of Tennessee students or Carnegie Mellon students. The learning algorthims job is then to find a hypothesis $h$ that corrrectly classifies a student as a Volunteer of Tartan based on the features providied. This learning process can then be defined as finding the hypothesis that has the least error (incorrect classifications) on the training data set while extending to examples outside of the training space.

### A. Support Vector Machines

Support Vector Machines (SVM) are a supervised learning technqiue in which hyperplanes are constructed in a high dimensional space to which the features are mactched. SVMs find the hyperplanes that are the farthest away from all of mapped features in order to provide excellent training performance while still maintaing the ability to generalize to new instances; i.e. SVMs are maximal margin classifiers. For a binary classification the decision function of the SVM is the dot product of the weight vector and the training example in the feature space added to a bias vector as shown in Equation 1.

$$f\left(\vec{x}\right) = \langle \vec{w}\phi(\vec{x})\rangle + \vec{b} \tag{1}$$

where $\phi(\vec{x})$ is a mapping to the higher dimensional space. The learning in the SVM is then finding the optimal values of the weight vector $\vec{w}$ and the basis $\vec{b}$.

The radial basis function (Equation 2) is a common kernal function used to map the input vector $\vec{x}$ into a higher dimension.

$$k\left(\vec{x}_i, \vec{x}_j\right) = exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|}{2\sigma^2}\right) \tag{2}$$

The maximal margin is ensured by minimizing:

$$g(\vec{w}, \eta) = \frac{1}{2}\|\vec{w}\| + C\sum_{i=1}^{N}\zeta_i \tag{3}$$

subject to:

$$y_i(\langle\vec{w}, \phi(\vec{x})\rangle + b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0 \tag{4}$$

where $\zeta_i$ is the $i$th slack varaible and C is the regulariazation parameter [1]. Something about how we are trying to elarge the margin

### B. Ensamble Classifers

Given an individiual classifier $h$, an enamble of classifers can be constructed of a set of indvidual classifers, $H = h_1, h_2, \dot{;} h_n$.

As long as the individual classifers have uncorrelated errors when any single classifer is incorrect the other classifiers in the ensamble might correctly classify the example.

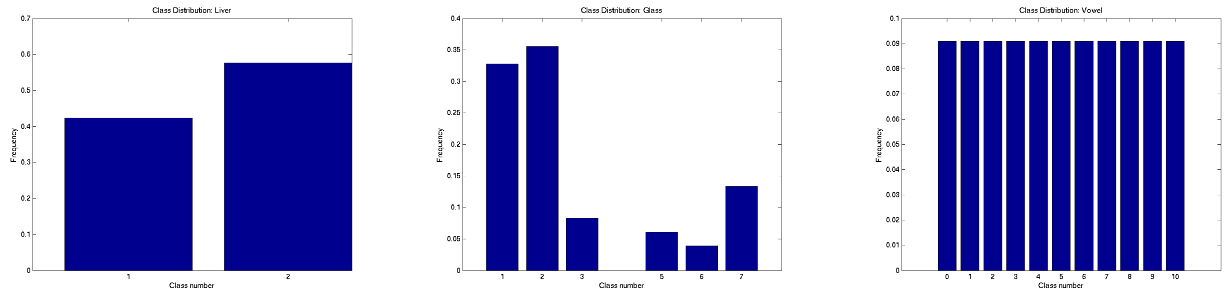
### C. Boosting

Unbalanced data sets (data sets in which a majority of the values come from one class, see Figure 1) are difficult for classification schemes to learn because the minority class is not well represented and tends to be thought as noise for the classifer. Often classifiers are trained from unblananced data sets by artifically reblaning the dataset by sampling techniques; i.e. up-sampling (sampling more from the minoarty class) and down-sampling (sampling less from the majoirty class). Boosting is an ensamble learning method in which a set of weights is maintained over the training samples and adaptively adjusted after each training itteration according to the ones that are misclasified [1]. By maintaining a weight distribution over all of the training examples, these weights could be updated to emphasize the training examples that are misclassified incorrectly. These incorrectly classified examples could then be learned in a refinement of the classifier or by training adding a new classifer to the ensamble with the new weights.

## II. Methods

### A. Support Vector Machines

Suport vector machines were implemented with the libsvm library [2]. Radial basis functions were used as the kernal space for the SVM to create RBFSVM. There are then two parameters that need to be determined, $C$ the regularization paramater and $\sigma$ the width of the kernal. These parameters




(a) Liver (b) Glass (c) Vowel

Fig. 1: Distribution of Class Data

where determined by a grid search of the parameter space. In searching for the optimal parameters with a grid search five fold cross valdation of the data was used. In addition, it was noticed for the vowel data sets that the large degree of accuracy was achieved by using a large amount of support vectors in which the SVM was essentially memorizing the training examples. This was mitigated by decreasing the error factor by passing the correct input argument to `svmtrain`.

### B. AdaBoost

AdaBoost was implemented as an ensamble of support vector machines by maintaining a weight distribution of the misclassificaiton error over all of the training examples. At each cycle $t$ AdaBoost provides the learning algorthim with training examples $\vec{x}$ and a weight distriubtion $w$ (initialized uniformly). The learning algorthim is trained to generate a classifier $h_t$ and the weight distribution is updated to reflect the predicted results; easy training examples (in which the classifier is very certain) have their weights lowered, while hard samples have their weights increased. This process continues for $T$ cycles. Finally, the AdaBoost combines all of the component classifiers into the ensamble whose signal, final hypothesis, is constructed by by weighting the individual classifers by their training errors.

The AdaBoost algorthim (Algo 1 was extended to AdaBoostM1 in order to use the RBFSVM. This algorthim uses a fairly large RBFSVM kernal (a weak learning ability) for the first classifier, $h_t$. The classifier is then retrained with a smaller $\sigma$ until the accuracy of the classifier has an accuracy of just over 50%. At this point the classifer is added to the ensamble, along with it's weight based on it's accuracy. The weights of the training samples are then updated to reflect the training examples that the classifier struggled with. This processes is then repeated for the next classifier in the ensamble until all of the ensamble is full.

The implemented algorthim only varied the value of $\sigma$ because it is known that RBFSVMs depend highly on $\sigma$ and less on $C$ so the performance of the classifer can be changed over a set $C$ by simply changing $\sigma$ [1]. In addition ensamble sizes for 5 to 150 were tested, these results are shown in Section III.

**Algorithm 1** AdaBoostSVM

**procedure** ADABOOSTSVM($\sigma_{init}, \sigma_{min}, \sigma_{step}, C, \vec{x}, \vec{y}$)
  $w_i^1 \leftarrow 1/N \forall \quad i = 1, \ldots, N$
  **while** $\sigma > \sigma_{min}$ **do**
  Train a RBFSVM component classifier
    $h_t \leftarrow$ ComponentClassifer($\vec{x}, \vec{w}$)
  Compute the error of that classifer
    $h_t \ : \ \epsilon_t = \sum_{i=1}^{N} w_i^t, y_i \neq h_t(\vec{x}_i)$
  Decrease $\sigma$ until a weak classifier
    **if** $\epsilon_t > 0.5$ **then**
      $\sigma = \sigma - \sigma_{step}$
      **go to** 3
    **end if**
  Set weight of component classifer
    $h_t \ : \ \alpha_t = \frac{1}{2} ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
  Update weights of training samples
    $w_i^{t+1} = \frac{w_i^t exp(-\alpha_i y_i h_t(\vec{x}))}{C_t}$
  $C_t$ is a normalization value, $\sum_{i=1}^{N} w_i^{t+1} = 1$
  **end while**
  **return** $f(\vec{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\vec{x})\right)$
**end procedure**

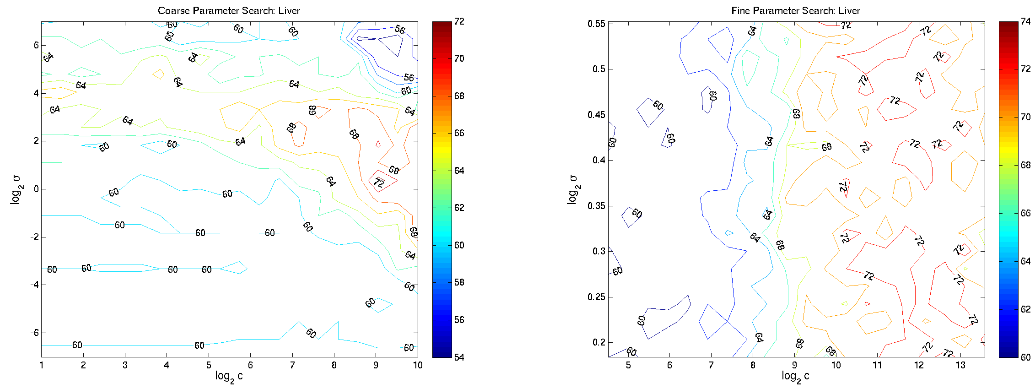
Several experiments were completed once the code base was written. First the optimal number of ensamble methods was found by investignating the classification accuracy dependance on the number of support vectors. Next the AdaBoost implementation vs the accuracy acheived by the a single parameter grid search. Finally

## III. RESULTS

The results of an SVM on the data sets is presented in Section III-A. In Section III-B the results are shown for using an ensamble methods.

### A. Parmater Search

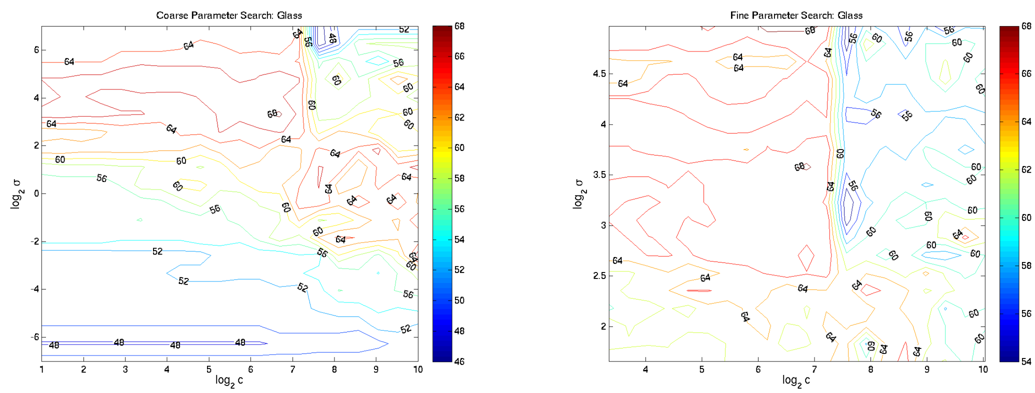
The parameter search for the optimal C and $\sigma$ parameters is shown in the contour plots of Figures 2, 3 and 3. The optimal




(a) Coarse Search

(b) Fine Search

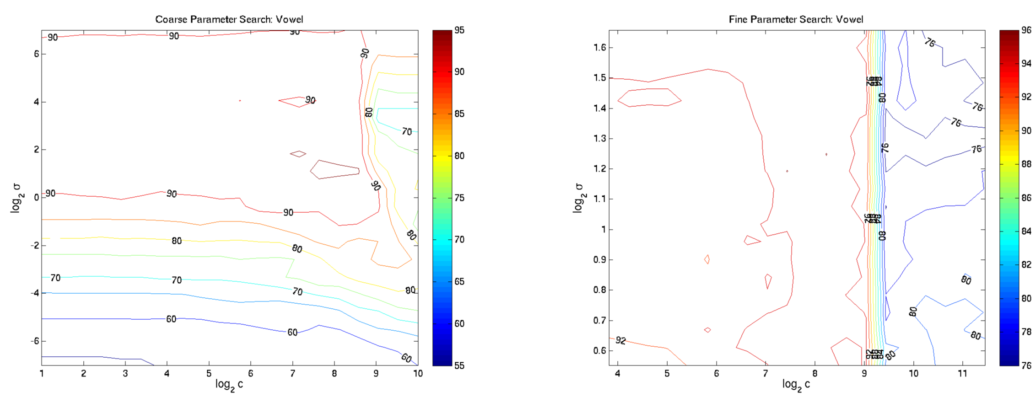
Fig. 2: Parameter search for Liver Disorder




(c) Coarse Search

(d) Fine Search

Fig. 3: Parameter search for Glass Disorder




(a) Coarse Search

(b) Fine Search

Fig. 4: Parameter search for Vowel Disorder

classifier parameters are shown for the coarse parameter search in Table I and for the fine parameter search in Table II.

Make some quantifications and

TABLE I: Coarse Optimal Classifier Parameters

| Data Set | $C_{min}$ | $C_{max}$ | $\sigma_{min}$ | $\sigma_{max}$ | $C$ | $\sigma$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| Glass | 1.00 | 10.00 | -7.00 | 7.00 | 6.68 | 3.32 | 68.33 |
| Liver | 1.00 | 10.00 | -7.00 | 7.00 | 9.05 | 0.37 | 73.00 |
| Vowel | 1.00 | 10.00 | -7.00 | 7.00 | 7.63 | 1.11 | 95.83 |

TABLE II: Fine Optimal Classifier Parameters

| Data Set | $C_{min}$ | $C_{max}$ | $\sigma_{min}$ | $\sigma_{max}$ | $C$ | $\sigma$ | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| Glass | 3.34 | 10.03 | 1.66 | 4.97 | 6.16 | 4.97 | 68.89 |
| Liver | 4.53 | 13.58 | 0.18 | 0.55 | 13.10 | 0.22 | 74.00 |
| Vowel | 3.82 | 11.45 | 0.55 | 1.66 | 8.23 | 1.25 | 96.02 |

### B. AdaBoostM1

The results using the implemented AdaBoostM1 algorthim are shown below in Table III. It is immediately observablve that the AdaBoost algorthim increased the accuarcy of the the Liver and Glass data set, but failed to increase (in fact dramatically lowered) the accuaracy for the Vowel data set.

TABLE III: AdaBoost Classifer Values

| Data Set | $T$ | $\sigma_{init}$ | $C$ | $\epsilon$ |
|---|---|---|---|---|
| Glass | 50 | 5.00 | 6.16 | 70.59 |
| Liver | 50 | 3.00 | 13.10 | 73.33 |
| Vowel | 50 | 2.00 | 10.00 | 56.49 |

| Data Set | $T$ | $\sigma_{init}$ | $C$ | $\epsilon$ |
|---|---|---|---|---|
| Glass | 100 | 5.00 | 6.16 | 73.53 |
| Liver | 100 | 3.00 | 13.10 | 73.33 |
| Vowel | 100 | 2.00 | 10.00 | 56.71 |

The effect of the number of the classifers in the ensamble is shown in Figure 5. The weight of each individual classifer for an ensamble of 150 members is shown in Figure 6

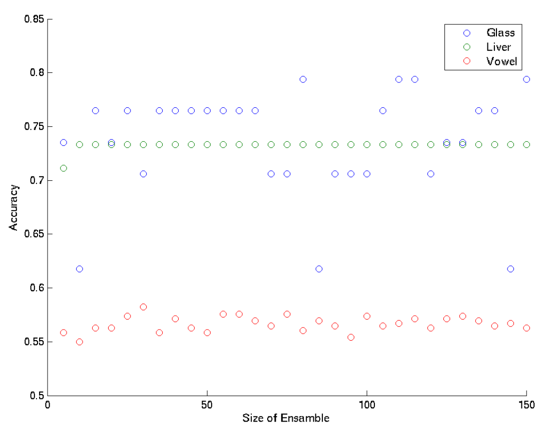



Fig. 5: Accuracy and Number of Compenents in Ensamble

## IV. Conclusions

Support vector machines were implemented on unbalanced data sets

The effect of having a large $\sigma$ can be observed by examining Figures 2, 3 and 3. Larger values of $\sigma$ tended to have a low accuracy, while too high a value tended to also have a low accuracy. Relatively large values of $\sigma$ are then suited for the ensamble methods because they tend to be represenative of an RBFSVM which a relativly weak learning ability which generalizes better.

### A. Future Work

Future work might be to change to use the decision value instead of the output class.

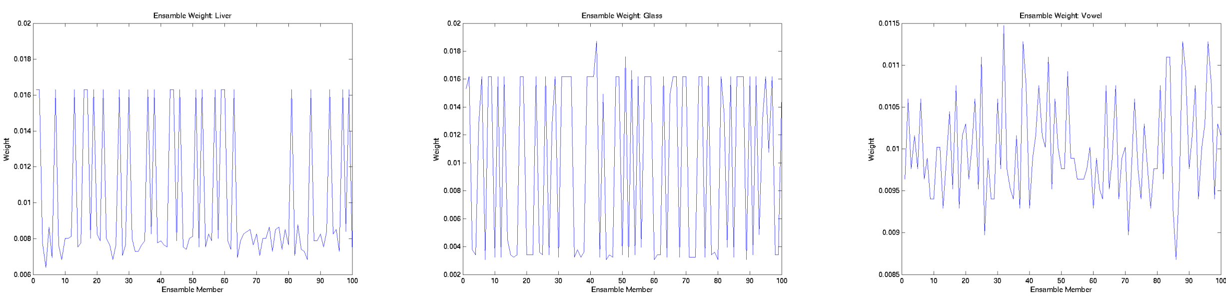
## Acknowledgements

The conversations with Alan Nam of how to best use the LIBSVM library for the use of this project where extremely helpful. In addition Mike Franklin provided valuable comments on the structure of this project.

## References

[1] X. Li, L. Wang, and E. Sung, "AdaBoost with SVM-based component classifiers," *Engineering Applications of Artificial Intelligence*, vol. 21, pp. 785–795, Aug. 2008.

[2] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.




(a) Liver (b) Glass (c) Vowel

Fig. 6: Distribution of Ensamble Weights