

Increased Classification Accuracy and Speedup Through Pair-wise Feature Selection for Support Vector Machines

Kurt Kramer, Dmitry B. Goldgof, and Lawrence O. Hall
Dept. of Computer Science & Engineering
University of South Florida
Tampa, FL 33620
{kkramer, goldgof, hall}@cse.usf.edu

Andrew Remsen
College of Marine Science
University of South Florida
St. Petersburg, FL.
aremsen@marine.usf.edu

Abstract— Support vector machines are binary classifiers that can implement multi-class classifiers by creating a classifier for each possible combination of classes or for each class using a one class versus all strategy. Feature selection algorithms often search for a single set of features to be used by each of the binary classifiers. This ignores the fact that features that may be good discriminators for two particular classes might not do well for other class combinations. As a result, the feature selection process may not include these features in the common set to be used by all support vector machines. It is shown that by selecting features for each binary class combination, overall classification accuracy can be improved (as much as 2.1%), feature selection time can be significantly reduced (speed up of 3.2 times), and time required for training a multi-class support vector machine is reduced. Another benefit of this approach is that considerably less time is required for feature selection when additional classes are added to the training data. This is because the features selected for the existing class combinations are still valid, so that feature selection only needs to be run for the new class combinations created.

Keywords—component; Support Vector Machines, SVM, Feature Selection, Pair-wise, Wrappers, Plankton.

I. INTRODUCTION

There are applications where a relatively large number of classes are involved and additional classes are added and subtracted as the situation may warrant. For example, plankton images collected by SIPPER [1,2,3,4] may consist of 20 to 50 classes of interest out of thousands of possible classes. As data is collected and conditions change, the user may require the addition and subtraction of classes to an existing classifier. If the user wishes to keep the features and Support Vector Machine (SVM) parameters tuned, the feature selection process needs to be run again for each new set of combinations of classes, which is needed to get the best performance. With the Wrappers approach this is a lengthy procedure. Consider the case where there is a classifier consisting of 20 classes that has already been tuned and had features selected, and the user would like to add one more class. The parameter tuning and feature selection process has already been done on the 190 binary class combinations that comprise the 20 class classifier. If these features and parameters were specified for each binary class combination

separately, then for all practical purposes the procedure needs to be performed for only the 20 additional binary class combinations being created. This is preferable to processing the 210 binary class combinations that need to be evaluated when a single set of features and parameters are used, as done otherwise.

The classes vary in different ways; some differ by shape, while others have similar shapes but vary in texture. As a result features that do a good job of separating two particular classes may be ineffective in separating two other classes, thus reducing classification accuracy. This leads to the idea of selecting features by individual binary class combinations. For example, for 3 classes A, B, and C, feature selection would be performed separately for each binary combination of classes AB, AC, and AD, rather than for ABC. It will be shown that feature selection for binary combinations can result in a net reduction of features which can have the benefits of improved classification accuracy, reduced training times and faster feature selection times.

Feature selection methods can be divided into at least three different types: Filter, Embedded, and Wrappers. Filter methods work with the feature data itself without any knowledge of the learning algorithm to be used. These methods are fast, but because they are ignorant of any bias that a particular learning algorithm may have, they may produce feature subsets that are not as accurate as those produced by the other two groups. Embedded methods such as Recursive Feature Elimination (RFE) [5] perform feature selection as part of the training process of the learning algorithm. These methods are not ignorant of the learning algorithms bias and yet are relatively fast. Wrappers [6] use the learning algorithm as a Black Box often with cross validation as a heuristic to drive a search through feature space towards finding a good set of features. In all these methods a single global set of features are being selected for all classes. In some more recent papers [7, 8], authors are starting to look at the possibility of determining feature subsets and parameter settings that are optimal for a single class or combination. In some cases they still select a global set of features for all classes, but try to at least consider the consequences of a given subset of features by binary class combinations [9, 10].

In [8] the authors implemented a Wrappers-based feature selection approach where they specialized features on pair wise combinations. That is, they select a different set of features for each binary combination of classes. The premise is that there may be a subset of features that can be descriptive for a particular binary class combination, but not for the global case. As a result, a global-based feature selection schema would not select these features. They apply this logic using a Nearest Neighbors (NN) and a Bayes learner to four different datasets. They show a reduction in error rates and also a reduction in the mean number of features.

In [11] the authors do a comparison, using a rule learner, between universal and local where local implements a one-vs-all with a separate classifier for each class vs. all other classes. In the local model features were selected separately using statistical analysis. They report an improvement in precision/recall over previous studies.

We propose a new method of performing feature selection and SVM parameter tuning using Wrappers feature selection where features and parameters are chosen by individual binary class pairs, binary class feature selection (BFS), rather than by one set of features and parameters for all class pairs, multi class feature selection (MFS). It is demonstrated that significant feature reduction can be achieved while maintaining or significantly improving classification accuracy. Faster training times, Wrappers feature selection time is considerably reduced and greater flexibility in managing training libraries is provided by our approach.

II. BACKGROUND

A. Support Vector Machines

The SVM [12] is a two-class classifier that uses training examples to find a hyper-plane that separates the two classes. Since in many cases there is no linear solution that separates the two classes in feature space, the SVM projects the data into a higher dimension through the use of a kernel function. In this higher dimensional space locating a hyper-plane becomes a much more tractable problem. A SVM can be extended to handle multiple classes by implementing a one-vs-one schema where a SVM is built for every possible two-class combination. The final classification can be decided by either a popular vote or a confidence value that is assigned to each class. The specific SVM used was derived from [13]; the confidence/probability model was added separately from [14].

Equation (1) represents training data that the SVM is going to learn from. There are m examples where x_i and y_i represents respectively the feature vector and label for training example.

$$D = \{(x_i, y_i) | x_i \in \mathfrak{R}^p, y_i \in \{-1, 1\}\}_{i=1}^m \quad (1)$$

Equation (2) is the primal form of the SVM where w is a vector that is perpendicular to the separating hyper-plane, b is the distance the hyper-plane is from the origin, ξ_i is a slack variable representing the distance a training example is on the incorrect side of the margin and C is a parameter that controls the penalty to be paid for a training examples that end up on the wrong side of the decision boundary. To deal with training

examples that cannot be separated linearly feature vectors are boosted into a higher dimensional space with the function $\phi(x_i)$.

$$\text{Minimize:} \quad \left(\frac{1}{2} \langle w, w \rangle + \frac{C}{m} \sum_{i=1}^m \xi_i \right) \quad (2)$$

$$\text{Subject to:} \quad y_i(\langle w, \phi(x_i) \rangle + b) \geq 1 - \xi_i, \quad C, \xi_i > 0$$

Equation (3) represents the dual form of the SMV where the kernel function $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ represents the dot product of two training examples i and j in a higher dimensional space. The instances of α_i that are greater than 0 represent support vectors and contribute to the decision function.

$$\text{Maximize:} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad (3)$$

$$\text{Subject to:} \quad 0 \leq \alpha_i \leq \frac{C}{m}, i = 1, \dots, m, \quad \sum_{i=1}^m y_i \alpha_i = 0$$

Equation (4) computes the distance from the hyper-plane that the unknown example x is and Equation (5) is the decision function.

$$f(x) = \sum_{i=1}^m \alpha_i y_i k(x_i, x) + b \quad (4)$$

$$y = \begin{cases} -1, & f(x) \leq 0 \\ 1, & f(x) > 0 \end{cases} \quad (5)$$

Equation (6) is the kernel function utilized in this paper. It is the radial basis function (RBF) and introduces the gamma (γ) parameter.

$$k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \quad (6)$$

B. Datasets

Experiments were performed on three different datasets. Two datasets are libraries for plankton images produced by SIPPER [1,4,15]. One is for the west Florida shelf (WFS) and the other is from a 2008 eastern tropical pacific research cruise (ETP2008). The third dataset was derived from the Forest Cover dataset [16] which has more than 500,000 examples. Both plankton training datasets are highly unbalanced with respect to examples per class. The WFS training dataset class distribution ranges from 27 to 1,558 with an average of 509 examples and 5 classes containing less than 100 examples. The ETP2008 training dataset class distribution ranges from 13 to 1,000 examples with an average of 321 and 13 classes containing less than 100 examples.

Table 1 provides a summary of the datasets. Each dataset is randomly split into two parts, training and test. The training set will be used to drive both the feature selection and SVM parameter searches, and the test set is used as a final validation of results for comparison purposes between the MFS and BFS procedures. In addition the test datasets will be stratified by class for 10 folds.

All continuous feature data was z-normalized such that each feature had a mean centered on zero and unit variance.

C. Test Environment

Experiments were performed on a 64 processor cluster consisting of 8 nodes with two quad core processors per node sharing 32 GB of ram. Each of the 64 processors runs at 2.66 gigahertz. Both the BFS and MFS procedures, take advantage of the multi-processor environment.

D. Nomenclature

Classification accuracy is reported in two ways overall and class-weighted-equally. Overall accuracy is computed by dividing the number of correct predictions by the total number of test examples. Class-weighted-equally is computed by recording the accuracy for each class then averaging these accuracies to get the final accuracy. For purposes of comparing two different feature subsets or parameter sets class-weighted-equally is always used as it is more fair in the case of imbalanced classes

Longest-path-time is the longest amount of time any one individual processor spent on a given task. For example if a feature selection process requires 10,000 CPU seconds across 64 processors, but the largest amount of CPU seconds consumed by any one processor was 1,000 seconds then the longest-path-time would be considered to be 1,000 seconds.

III. METHODS

A. Binary Feature Selection Procedure.

The BFS approach consists of three major steps: initial SVM parameter tuning, binary class feature selection, and binary class SVM parameter tuning.

- 1) SVM parameter tuning. The search will be driven by classification accuracy followed by processing time. It is important that the SVM parameters are tuned before feature selection. Poor SVM parameters will have a detrimental impact on the feature selection process.

TABLE 1 DATASET DESCRIPTIONS.

Dataset Name	Number of Classes	Number of Features	Train Set Size	Test Set Size
WFS	33	82	16,807	4,199
ETP2008	55	83	17,678	23,211
Forest Cover	7	54	7,000	570,512

- 2) There are two possible methods for selecting features. If either class in the pair contain less than 25% of the average class size training examples then (i) all features will be selected otherwise (ii) a feature selection process will be used.

- (i) The thinking is that in this situation there are not enough training examples to properly represent the class in which case it is better to retain all features rather than just a subset that may not properly discriminate the class in the future.

- (ii) Using the SVM parameters determined in step 1 perform feature selection for each binary combination of classes. The feature subset selected for each binary class pair will be the subset that obtained the highest class weighted equally accuracy. Typically there will be several subsets that achieved the same high accuracy. The subset with the least number of features will be selected.

- 3) Binary class SVM Parameter tuning. For each binary class combination, perform the SVM parameter search, as done in step 1 above on the features that were selected in step 2.

The labeled feature data is divided into two datasets, training and test. The training dataset is used in both the SVM parameter tuning and feature selection processes. The test dataset allows measurement of how well the binary class feature selection process does compared to the traditional multi-class feature selection process.

B. SVM Parameter tuning.

The search for parameters is driven by two criteria: classification accuracy and processing time. The first criterion, classification accuracy, is by far the most important. After classification accuracy comes processing time. Of the sets of SVM parameters that produce the best accuracy the subset that had the fastest processing time would be selected.

The parameter search is a modified grid search similar to the parameter search done in [17]. It involves multiple passes, with the first pass being coarse, focusing on the γ parameter only with C fixed at 1. Each successive pass is finer, with the C parameter added later. Each parameter set is evaluated by performing a 5-fold cross-validation using the class-weighted-equally classification accuracy followed by processing time as selection criteria. The following pass then performs localized searches around each of these candidates using a finer level of granularity. The last grid search pass is used to test the best parameter sets located during the search. Using the two criteria listed above, the 10 best parameter sets are located and evaluated by performing a 10-fold cross-validation on the training dataset. From these final 10 parameter sets, the best parameter set is selected using the two criteria of class-weighted-equally classification accuracy and processing time.

C. Wrappers feature selection.

A Wrappers [6] based feature selection method is used for both the MFS and BFS processes. In the case of one vs. one it is used once for each possible combination of classes. It utilizes the best first strategy starting with all features selected

and reducing down by one feature at a time. The procedure keeps track of a pool (P) of feature combinations that have been evaluated but not expanded. From this pool it selects for expansion the feature combination that produced the highest classification accuracy from a 5 fold cross validation. Expansion is the process of taking a given feature combination and creating new feature combinations from it. There are two different types of expansions performed: Shrink and Grow. These are represented by the functions $ExpS(F)$ and $ExpG(F)$ respectively. $ExpS(F)$ creates new feature combinations by removing each of the features, while $ExpG(F)$ adds each feature that is not currently selected. The goal is to reduce the number of features so $ExpS(F)$ is the function primarily used. At every tenth expansion the growth expansion $ExpG(F)$ is used in addition to the shrink expansion. This is meant to make sure there is no loss of any features that may have performed badly earlier in the search process, but will do better as the feature count is reduced or as other features that it worked poorly with are removed. In addition, on every tenth expansion one feature set is selected at random for shrink expansion.

When 50 expansions are processed without locating a higher classification accuracy the search turns into a 5 wide beam search [18] which will run until the numbers of features are reduced to one.

After the feature search is completed the subset of features need to be selected. From the set of all evaluated feature combinations, select the features desired using highest class-weighted-equally accuracy (where the accuracy on each class is recorded and then averaged), smallest number of features, and fastest training time as criteria.

- a. Determine candidate feature subsets to be evaluated by a 10 fold cross validation. For each feature count select the highest accuracy subsets. For example if among all the feature subsets that consist of 6 features the highest accuracy was 90% then select all the feature subsets that consist of 6 features that had 90% accuracy. This is done for each count of features, 1 through the number of features.
- b. For each feature subset selected in previous step perform a 10 fold cross validation.
- c. From the set of all evaluated subsets in the previous step select the one with the highest class-weighted-equally classification accuracy followed by the least number of features, followed by fastest processing time. This feature subset will be the result of this feature selection process.

IV. RESULTS

A. Feature Selection Timing Results.

Tables 2 through 4 show the total CPU and longest path times in seconds used for each step of the two-feature selection methods multi class feature selection (MFS) and binary class combination feature selection (BFS).

As noted earlier, longest-Path-Time is the longest amount of time any one individual processor spent on a given task.

For example the WFS dataset required 1,546,207 CPU seconds divided amongst 64 processors to do the BFS procedure. The longest any one individual processor spent was 27,031 CPU seconds.

The top half of the tables shows the processing time for the MFS steps, while the bottom part of each table shows the processing time for the BFS steps. The time spent doing MFS parameter tuning shown in the first row is also included in the totals for the bottom BFS part. The first column provides a description for the step. The second column shows the average number of features selected as a result of the processing step. The third column shows the total CPU time in seconds involved in the processing step; this includes the time spent performing the 5-fold cross-validations used to evaluate specific SVM parameters and feature selections, plus the overhead time required in managing the feature selection process. The fourth column represents the longest time in CPU seconds of the 64 processors for the processing step the row represents. Both the top (MFS) and bottom half (BFS) have totals representing the total amount of time required to perform their respective feature selection processes.

TABLE 2 WFS - MFS & BFS SEARCH TIMES.

Description	Avg. Number Features	Total CPU Time (secs)	Longest Path (secs)
MFS – Parameter tuning	82.0	254,386	5,814
MFS – Feature selection	45.0	4,794,304	77,708
Total MFS time		5,048,690	83,522
BFS – Feature selection	33.7	1,213,275	19,900
BFS – Parameter tuning	33.7	78,546	1,317
Total BFS time		1,546,207	27,031
Speed Ups	BFS vs. MFS	3.27	3.09

TABLE 3 ETP2008 - MFS & BFS SEARCH TIMES.

Description	Avg. Number Features	Total CPU Time (secs)	Longest Path (secs)
MFS – Parameter tuning	83.0	227,322	9,361
MFS – Feature selection	40.0	3,437,356	109,091
Total MFS time		3,664,678	118,452
BFS – Feature selection	26.9	1,223,904	38,706
BFS – Parameter tuning	26.9	123,815	3,900
Total BFS time		1,575,040	51,967
Speed Ups	BFS vs. MFS	2.33	2.28

TABLE 4 FOREST COVER - MFS & BFS SEARCH TIMES.

Description	Avg. Number Features	Total CPU Time (secs)	Longest Path (secs)
MFS – Parameter tuning	54.0	123,113	2,405
MFS – Feature selection	32.0	381,848	6,620
Total MFS time		504,960	9,025
BFS – Feature selection	13.7	197,036	4,901
BFS – Parameter tuning	13.7	45,322	1,197
Total BFS time		365,471	8,503
Speed Ups BFS vs. MFS		1.38	1.06

B. Classification Accuracy Results.

Tables 5 through 7 show the classification accuracy results for each of the datasets. The first row shows results with all features after the SVM parameters have been tuned. The second and third rows show the results of the MFS and BFS produced classifiers. There are two accuracy results reported on each row. One represents the overall accuracy which is the number of test examples classified correctly divided by the total number of test examples. The other one represents class-weighted-equally. Bold indicates that BFS results are statistically different than the MFS results as per a McNemar’s test [19].

Table 5 shows the results for the WFS dataset. When comparing the BFS approach with the MFS approach, the BFS approach had a 0.63% increase in overall accuracy and a 1.69% class-weighted-equally accuracy. A McNemar’s test shows that the two classifiers are statistically significantly different. There was a speed up in training time of 1.2 times. This particular dataset consists of 33 classes where the classes are very unbalanced. In the training set the largest class consists of 1,558 examples while the smallest class contains just 27 classes.

Table 6 shows the results for the ETP2008 dataset. There was a 2.18% gain in overall classification accuracy, but a 1.11% loss in class-weighted-equally accuracy. A McNemar’s test indicates that the BFS-produced classifier is statistically significantly different from the MFS classifier as indicated by the BFS row being bold. There were 13 classes that have less than 100 examples in the training dataset, with the smallest class only having 16 examples.

TABLE 7 shows results for the Forest Cover dataset. The MFS procedure reduced the number of features from 54 to 32, while the BFS procedure reduced the feature set to a mean of 13.7. The BFS procedure produced classifier had accuracy gains of 1.98% and 0.81% for overall classification accuracy and class-weighted-equally accuracy. The BFS row in bold indicates that it is statistically significantly different from the MFS results.

C. Adding Classes to Existing Training Libraries

Table 8 shows the results of adding one class at a time using data from the ETP2008 dataset. The 13 classes that had 600 or more examples were used for this experiment. The data was then randomly split into training and test sets with 70% of the

data going to training with a maximum of 800 per class and the remainder into test. The classes were randomly ordered with the first 5 classes (Copepod Calanoid, Copepod Nauplii, Detritus Molts, Detritus Snow, and Pteropod Creseis) used as the initial starting training library. Both the MFS and BFS procedures were performed on the initial training library with the results shown in the first row of Table 8. The following classes were then added one at a time with both the MFS and BFS procedures being performed with results shown in the following rows.

The processing time for the MFS procedure consistently grows at a polynomial rate with the number of classes while the BFS procedure grows linearly. The processing time of BFS is a function of number of training examples in the additional class being added, total number of classes in the training library and the difficulty in discriminating the class being added from the classes already in the training library. The class Copepod-Calanoid-Eucalanus which is similar to Copepod-Calanoid and Copepod-Oithona required 46,769 seconds processing time compared to Tunicate-Doliolid which required only 27,380 seconds even though Tunicate-Doliolid had more training examples and a greater number of classes in the existing training library. This can be attributed to the fact that Tunicate-Doliolid was very easy to discriminate from the other classes in the training library. In general the BFS procedure had a speedup from 4.69 to 19.39 over that of the MFS procedure.

TABLE 5 WFS CLASSIFICATION RESULTS.

Description	Overall Acc.	Wtd. Acc.	Train Time(s)	Test Time(s)	Avg.# Feat’s
All Features	76.45%	69.32%	44.8	29.0	82.0
MFS Approach	76.73%	69.18%	26.0	20.0	41.0
BFS Approach	77.21%	70.35%	21.2	84.8	33.7
Accuracy Gains	0.63%	1.69%			

TABLE 6 ETP2008 CLASSIFICATION RESULTS.

Description	Overall Acc.	Wtd. Acc.	Train Time(s)	Test Time(s)	Avg.# Feat’s
All Features	83.33%	77.54%	32.4	169.5	82.0
MFS Approach	83.53%	78.05%	18.4	116.6	40.0
BFS Approach	85.35%	77.18%	17.7	649.1	26.9
Accuracy Gains	2.18%	-1.11%			

TABLE 7 FOREST COVER; MOST ACCURATE SET OF FEATURES.

Description	Overall Acc.	Wtd. Acc.	Train Time(s)	Test Time(s)	Avg.# Feat’s
All Features	69.59%	82.72%	38.8	1,348.6	54.0
MFS Approach	71.15%	84.18%	20.8	652.1	32.0
BFS Approach	72.56%	84.86%	16.2	1,772.7	13.7
Accuracy Gains	1.98%	0.81%			

V. DISCUSSION

TABLE 9 provides a summary comparing the results of the BFS method to the MFS method. Both classification accuracy and feature selection search times are shown. All three datasets had improvement in overall classification accuracy while two had improvement in class-weighted-equally classification accuracy. The ETP2008 dataset has a loss in class-weighted-equally accuracy of 1.11% but a gain of 2.18% in overall accuracy. Feature selection was faster for all three datasets ranging from a speed up of 1.38 for the forest cover to 3.27 for the WFS dataset.

The major benefits of the BFS approach have been shown to be a significant speed up in Wrapper feature selection time, a significant improvement in classification accuracy, speed up in training times, and reduction in the time to add or delete

classes from existing classifiers, giving the user greater flexibility in managing existing classifiers. One of the datasets which contained unbalanced class representation in the training data did not do well with respect to class-weighted-equally classification accuracy; but did have significantly better overall accuracy. Both the WFS and Forest Cover datasets had significantly higher overall classification accuracy and class-weighted-equally accuracy.

BFS requires a higher quality training set than MFS. It will tend to make a much tighter fit than normal feature selection and as a result may not generalize as well. However when the training set is low noise and is representative, it will result in classification accuracy that is as good as MFS or better, as shown with the Forest Cover dataset.

Wrapper-based feature selection time is faster with BFS. The feature selection process for multiple classes is considerably longer than the combined time to do all the binary class pairs feature selection and parameter tuning. The binary SVMs are more apt to reduce down to a smaller set of features, resulting in considerably less feature combinations needing evaluation (that is training and testing). This makes sense, since the MFS method has to search for a set of features that performs well for all binary classifier pairs, whereas BFS focuses on one pair of classes at a time, hence the features for which it is looking need only satisfy the requirements of two classes.

ACKNOWLEDGMENT

The research was partially supported by the United States Navy, Office of Naval Research, under grant number N00014-02-1-0266, the NSF under grant EIA-0130768, the National Institutes of Health under an SBIR award, grant number: 2R44MH076541-04 and the BP/FIO Gulf Oil Spill Prevention, Response and Recovery Grant Program.

TABLE 8 ETP2008 ADDING ONE CLASS AT A TIME.

Class Name	Class Count	Num Train	Search Time		
			MFS (sec)	BFS (sec)	Speed Up
Starting 5 classes	5	3,219	70,834	36,915	1.92
Eumalacostracan Euphausiid	6	709	73,744	12,027	6.13
Copepod copilia	7	579	109,064	7,778	14.02
Noise	8	801	101,341	8,607	11.77
Copepod Oithona	9	301	136,380	21,515	6.34
Copepod Calanoid Eucalanus	10	709	219,212	46,769	4.69
Copepod Oncaea	11	471	291,367	15,030	19.39
Protist Radiolarian	12	801	344,289	29,125	11.82
Tunicate Doliolid	13	801	483,045	27,380	17.64

TABLE 9 SUMMARY OF RESULTS.

Dataset	Accuracy Overall			Accuracy Class-Weighted-Equally			Feature Selection Time		
	MFS(s)	BFS(s)	Gain	MFS	BFS	Gain	MFS(s)	BFS(s)	Speed-up
WFS	76.73%	77.21%	0.63%	69.18%	70.35%	1.69%	5,048,690	1,546,207	3.27
ETP2008	83.53%	85.35%	2.18%	78.05%	77.18%	-1.11%	3,664,678	1,575,040	2.33
Forest Cover	71.15%	72.56%	1.98%	84.18%	84.86%	0.81%	504,960	365,471	1.38

REFERENCES

- [1] Scott Samson et al., "A system for high-resolution zooplankton imaging," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, October 2001.
- [2] Andrew Remsen, Scott Samson, Thomas Hopkins, and Kurt Kramer, "Observations of plankton and detrital particle distribution on the West Florida Shelf using SIPPER-2 and an automated classification system," *Journal of Plankton Research*, under preparation.
- [3] Andrew Walker Remsen, Evolution and Field Application of a Plankton Imaging System, 2008, Dissertation, College of Marine Science, University of South Florida.
- [4] Tang Luo et al., "Active Learning to Recognize Multiple Types of Plankton," *Journal of Machine Learning Research*, vol. 6, pp. 589 - 613, December 2005.
- [5] Isabelle Guyon, Steve Gunn, Nikravesh Masoud, and Lotfi A. Zadeh, *Feature Extraction, Foundations and Applications, Embedded Methods*, 1st ed.: Springer, August 2006.
- [6] John Ron Kohavi and H George, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1, pp. 273-324, December 1997.
- [7] F. Bruno, A. Carvalho, Rodrigo Calvo, and Renato Porfirio Ishii, "Multiclass SVM Model Selection Using Particle Swarm Optimization.," *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems*, p. 31, 2006.
- [8] Hugo Silva and Ana Fred, "Pairwise vs global multi-class wrapper feature selection," in *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'07)*, vol. 6, Corfu Island, Greece, 2007, pp. 1-6.
- [9] Xue-wen Chen, Xiangyan Zeng, and Deborah van Alp, "Multi-class feature selection for texture classification," *Yahoo Research, Technical Report YR-2008-002*, vol. 27, no. 14, pp. 1685-1691, October 2006.
- [10] Olivier Chapelle and Sathya Keerthi, "Multi-Class Feature Selection with Support Vector Machines.," *Yahoo Research, Technical Report YR-2008-002*, 2008, Yahoo Research, Technical Report YR-2008-002.
- [11] C. Apte et al., "Automated Learning of Decision Rules for Text Categorization," *ACM Transactions on Information Systems*, vol. 12, pp. 233-251, 1994.
- [12] Nello Cristianini and John Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel-based learning methods.*, 1st ed. Cambridge, United Kingdom: Cambridge University Press, March 2000.
- [13] Chih-Chung Chang and Chih-Jen Lin. (2001) LIBSVM: a library for support vector machines (version 2.3). [Online]. <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- [14] Tong Luo et al., "Recognizing Plankton Images from the Shadow Image Particle Profiling Evaluation Recorder.," *IEEE Transactions on Systems Man and Cybernetics Part B – Cybernetics*, vol. 34, no. 4, pp. 1753-1762, August 2004.
- [15] Kurt Kramer, *Dissertation, System for Identifying Plankton from the SIPPER Instrument Platform*. Tampa: University of South Florida, 2010.
- [16] Jack A. Blackard, Dr. Denis J. Dean, and Dr. Charles W. Anderson. (1998, August) Machine Learning Repository Forest Cover Data Set. [Online]. <http://archive.ics.uci.edu/ml/datasets/Covertype>
- [17] Kurt A. Kramer, Lawrence O. Hall, Dmitry B. Goldgof, Andrew Remsen, and Tong Luo, "Fast support vector machines for continuous data," *IEEE Trans Syst Man Cybern B Cybern*, vol. 39, no. 4, pp. 989-1001, August 2009.
- [18] David Furcy and Sven Koenig, "Limited Discrepancy Beam Search," in *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 125-131.
- [19] Thomas G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895-1923, October 1998.
- [20] Koby Crammer and Yoram Singer, "On the algorithmic implementation of multiclass kernel-based machines," *Journal of Machine Learning Research*, vol. 2, pp. 265-292, Dec. 2001.