2.5 A STORM-TYPE CLASSIFIER USING SUPPORT VECTOR MACHINES AND FUZZY LOGIC

Jennifer Abernethy*[1,2] and John K. Williams[2]
[1]University of Colorado, Boulder, Colorado
[2]National Center for Atmospheric Research†, Boulder, Colorado

1. INTRODUCTION

This paper details our methodology for entry in the 2008 American Meteorological Society's 6th Conference on Artificial Intelligence Applications to Environmental Science AI competition. We trained probabilistic Support Vector Machine models with several separations of a multi-category storm type data set. We experimented with two fuzzy logic approaches to combine the model outputs into a coherent prediction for our contest entry.

1.1 *Support Vector Machines*

The Support Vector Machine (SVM) is a popular machine learning technique for classification. Generally, a classifier is an algorithm that predicts a data classification given (presumably) relevant data features. The SVM produces a model that predicts the class label by setting parameter values of an optimization problem based on its input data (Hsu et al. 2003)

In order to learn the relationships (parameter values) between these data features and the class label, we first train a classifier by giving it many known feature/class pairs. Each pair is known as a data instance. A data instance k consists of a set of features $x_{i,k} \;\; i = 1...n$ and a target class label *y*.

The SVM is trained on many data instances called a training set. The SVM prediction accuracy is estimated using a test set of data instances with known class labels which were not used during training. Using a test set instead of the training set for accuracy estimation better reflects the SVM's ability to classify unknown data.

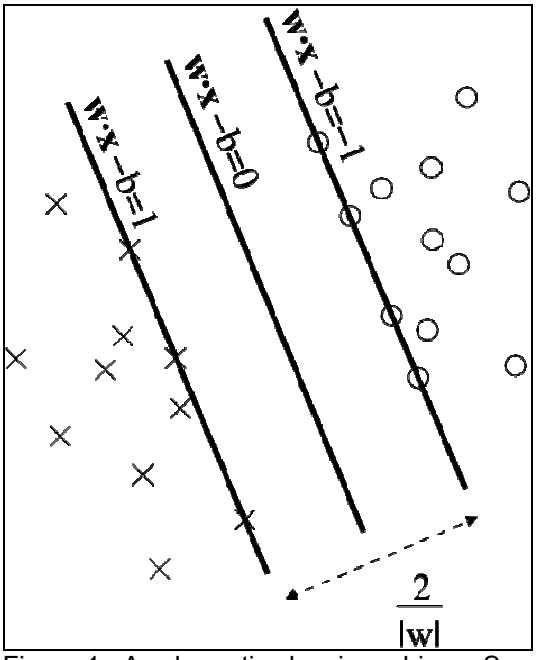

Figure 1. A schematic showing a binary Support Vector Machine classifier with a linearly separating hyperplane. The data points on the margin lines are the support vectors.

During training, each feature vector X_k is mapped into a higher dimensional space. The SVM finds a linearly separating hyperplane with the maximal margin between class means in this higher dimensional space. A schematic of this hyperplane and margins for a binary classifier is shown in Figure 1.

To classify an example, the SVM calculates the distance of that example to each class mean through a series of dot products, and classifies it in whatever class has the closest mean (Chen et al., 2003). This series of dot products is at the heart of the model and is a measure of vector similarity called a kernel function:

$$K(x_i, x_j) = \phi(x^T{}_i)\phi(x_j)$$

† The National Center for Atmospheric Research is sponsored by the National Science Foundation

*Corresponding author address: Jennifer Abernethy, University of Colorado, Department of Computer Science, 430 UCB, Boulder, CO 80309, email: aberneth@cs.colorado.edu

For implementation of the SVM, we will use the LibSVM library (Chang and Lin, 2003). LibSVM provides four basic kernels and an optional program, "grid.py", that selects the model (i.e., does a parameter search). Chang and Lin recommend the radial basis function kernel:

$$K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2), \gamma > 0$$

The radial basis function kernel only has two parameters: γ and *C*, a penalty parameter for the SVM error term.

2. METHODOLOGY

Our classification approach uses fuzzy logic to combine probabilistic output from binary and multi-category SVM models, with the goal of improving upon the accuracy of a single multi-category model. Each model was trained using a feature subset chosen for the model's particular categories using a hillclimbing subset selection search.

2.1 *Data*

This experiment used a prepared data set of four hand-classified storm types - NotSevere(0), IsolatedSupercell(1), ConvectiveLine(2), and PulseStorm(4) - with 22 attributes, described in Guilliot et al. (2008).. The training set included 1358 labeled examples, and the set of competition data contained 1071 unlabeled examples. Explanation of the data set can be found in Guillot et al. (2008) and Lakshmanan et al. (2008).

2.2 *Search*

Our search for the best subset of diagnostics is essentially the task of *feature subset selection* (Guyon and Elisseef, 2003, Kohavi, et al. 1995, 1997). We are faced with the choice between 22 features, knowing that some may not improve classification accuracy. The wrapper method in feature subset selection executes a state space search for a good feature subset, estimating prediction accuracy using an induction algorithm – here, we used SVMs. We used a simple hillclimbing search. Each state is a subset of features, and the search operator is "add a feature". The search chooses the best addition to the current subset based on the classification performance of an SVM using the current subset plus an additional feature. This approach to the search is called *forward selection*. Thus, we start with an empty subset and added features stepwise; our stopping condition was no further classification performance improvement.

For ease and time constraints, one set of kernel parameters γ and *C* were used for the searches and subsequent models. We used the LibSVM tool "grid.py" with 10-fold cross validation on the training set.
At each step of a search, sets of training data (1121 samples), testing data (121 samples), and holdout data (100 samples) were generated from the original training data set, "training.csv", containing only the current subset of features plus the proposed addition to that set. A SVM model, with deterministic output only, was trained on the training data, and the resulting model was tested on the testing data, outputting overall classification accuracy for that step of the search. We took the prediction file produced by the model and the labels in the test file and further calculated the multi-category True Skill Score (detailed in Lakshmanan, et al. 2008).

We used TSS as the scoring function as the evaluation metric for the search.
When a search completed, the model was tested on the holdout data using the chosen subset to determine the final skill of the model.

Our initial intention was to create one-against-the-rest models for each of the four categories; however, SVM models trained on 1-vs-rest and 2-vs-rest categorization of the training set showed very low skill (TSS<.1) with several different feature subsets (searches did not progress due to no improvement in TSS). We proceeded with three models: 0-vs-rest, (0,4)-vs-rest, and a multi-category model that discriminated between each of the four categories (0,1,2,4). We trained our final models for each categorization using the "-b 1" option to produce probabilistic output on the test data. LibSVM employs cross-validation to produce the probability outputs. We used each model to predict probabilities for the unlabeled competition data set.

2.3 *Fuzzy Logic*

The three SVM models produced three different sets of predictions for the competition data. We experimented with two variations of fuzzy logic

to most accurately and intelligently combine the predictions. The first algorithm, A, used a two-step approach: it combined output from 0-vs-rest model and the (0,4)-vs-rest model, and then used predictions from the (0,1,2,4) model to fill in uncertainties between the categories "1" and ""2. The second algorithm, B, considered all three models' predictions together in assigning a final classification to each competition data example.

Algorithm A's first step resolved classification of categories "0" and "4" using the predictions of models (0)-vs-rest and (0,4)-vs-rest. If (0) predicted class "0" with 95% or above probability for a test example, that example was assigned class "0". If (0) predicted "rest" and (0,4) predicted "0 or 4", the algorithm classified the example as a "4". If both models predicted "rest" then the example was temporarily classified as "either 1 or 2".

When there was ambiguity or disagreement between the models, differences in class probabilities were considered. Basic disagreements (i.e., (0) predicted "0" but (0,4) predicted "rest") were decided by the model that was most confident (highest probability) in its classification. In the case where (0) predicted "0" and (0,4) predicted "0 or 4", the algorithm overrode (0)'s prediction (thus classifying as "4") if the (0,4) confidence was 0.4 higher than the (0) confidence. We settled on 0.4 value through trial and error looking at correct and incorrect fuzzy logic predictions in the test set.

Algorithm A's second step was to resolve the class 1 and 2 classifications using the (0,1,2,4) model predictions. For examples where the prediction was "either 1 or 2" the (0,1,2,4) model's prediction was used.

Algorithm B compared all three models' predictions to determine a final classification. Instead of a weighted formula to combine the probabilities, we used a series of logic that compared the level of agreement and confidence of each model. For instance, when the (0) model predicted "0", and the (0,4) model predicted "0 or 4", both with a confidence lower than 0.8, if the (0,1,2,4) model predicted "4", the algorithm looked at the difference in probabilities between its first and second highest class probabilities. If that difference was above 0.3, indicating a much higher confidence in the "4" classification, the final prediction was "4".

In cases where the (0) and (0,4) models disagreed, the algorithm compared their probabilities to the difference in probabilities between the (0,1,2,4) model's ranking of classifications for that example. Since the 0-vs-rest and (0,4)-vs-rest models had the higher True Skill Scores than the (0,1,2,4) model, their predictions usually took precedence for class 0. Of course, (0,1,2,4) was the only model to discriminate between classes 1 and 2. Where there was ambiguity between the (0) and (0,4) models, the algorithm considered the confidence of the (0,1,2,4) model and allowed it to override the others' prediction if its confidence was high enough. Code for A and B can be requested from Abernethy. The result of the fuzzy logic step was a final prediction file for the competition data set.

We conducted repeated trials using the chosen SVM models and applying their results to both fuzzy logic algorithms, each time dividing the training set randomly into training and testing sets. From the mean and variation of the accuracies of the fuzzy logic algorithms, we chose algorithm B as most able to improve upon a single SVM model. The results of those trials are detailed in the following section.

3. RESULTS

The radial basis function kernel parameters converged upon by LibSVM's grid.py were C=4 and γ = 0.0625. Using these parameters, the forward selection subset searches were conducted for each separation of the data, 0-vs-rest, (0,4)-vs-rest, and (0,1,2,4). The 0-vs-rest model search converged on features low level shear (LLSHR), maximum expected hail size (MESH) and maximum reflectivity (MAXREF), with a TSS of 0.89 for the test set and 0.77 for the holdout set. The (0,4)-vs-rest search converged on feature set containing aspect ratio (ASPR), maximum vertical integrated liquid (MAXVIL), and orientation due north (ORN) with a TSS of 0.78 for the test set and 0.83 for the holdout set. The (0,1,2,4) search concluded on set ASPR, convective area (CONVA), maximum expected hail size over the storm's history (LMESH), location of the storm's centroid (LONC), MAXVIL and ORN. The (0,1,2,4) search found a TSS of 0.64 with this feature set for the testing set and TSS of 0.55 for the holdout set.

Means and confidence intervals for the trained models for each separation of the data are in Table 1. We also looked at improvement in the classification accuracy at each step. We recorded TSS for the first step of fuzzy logic algorithm A ("FL combo" in the Table 1) as well as the final results of both A and B.

Model/Step	Mean TSS	90% CI
0-vs-rest	0.78	(0.69, 0.88)
(0,4)-vs-rest	0.73	(0.61,0.87)
(0,1,2,4)	0.59	(0.5,0.68)
FL combo (0) and (0,4)	0.67	(0.58,0.75)
FL Algorithm A	0.612	(0.5,0.7)
FL Algorithm B	0.617	(0.51,0.71)

Table 1. True Skill Score means and confidence intervals for trials (on labeled training data set) of SVM models and fuzzy logic combination algorithms.

From the results of the trials in Table 1, we concluded that algorithm B should be used to combine the model results for our official entry in the competition. Since we knew the category distribution of the competition data set, we looked at the distributions of our predictions as reassurance of our approach. Indeed, the distribution was very close: (569,131,95, 274) for the competition set (for categories 0,1,2,4 respectively) and (565,149,93,262)for our predictions.

4. DISCUSSION

We showed that a fuzzy logic combination of multiple support vector machine models does improve upon the classification accuracy of a single multi-category model for this storm type dataset. However, tailoring the parameters of the fuzzy logic to the provided training set may have led to overfitting of the training set, causing a lower than expected accuracy on the competition data set. Although our approach predicted a very accurate distribution of categories, its TSS was a disappointing 0.55, 0.03 lower than the baseline decision tree for the competition. We hope to look into where our algorithm failed most, particularly weaknesses in the fuzzy logic. Overall, this competition was a great opportunity to explore combinations of AI techniques we might otherwise not have.

5. REFERENCES

Chang, C. and C. Lin. LIBSVM – a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chen, P., C. Lin and B. Scholkopf, 2003: A tutorial on v-support vector machines. http://kernel-machines.org.

Guillot, E., T.M. Smith, V. Lakshmanan, K.L. Elmore, D.W. Burgess and G. J. Stumpf, 2008: Tornado and Severe Thunderstorm Warning Forecast Skill and its Relationship to Storm Type. *Applications in Meteorology, Oceanography, Hydrology and Climatology, New Orleans*. CD-ROM 4 A.3.

Guyon, I. and A. Elisseef, 2003: An introduction to variable and feature selection. *J. Machine Learning Research*, **3**, 1157-1182.

Hsu, C., C. Chang and C. Lin, 2003: A practical guide to support vector classification. Published online with Libsvm documentation at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Kohavi, R., and D. Sommerfield, 1995: Feature subset selection using the wrapper method: overfitting and dynamic search space topology. *First International Conference on Knowledge Discovery in Data Mining (KDD-95).*

Kohavi, R. and G. John, 1997: Wrappers for Feature Subset Selection. *J. Artificial Intelligence*, **97**, *no1-2, 273-324.*

Lakshmanan, V., E. E. Ebert and S.E. Haupt, 2008: The 2008 Artificial Intelligence Competition. *6th Conference on Artificial Intelligence Applications to Environmental Science, New Orleans.*