An Application of Fuzzy Support Vectors

John Mill
Spokane Falls Community College
Spokane, WA 99224
E-mail: johnmi@spokanefalls.edu

Atsushi Inoue *
Department of Computer Science
Eastern Washington University
Cheney, WA 99004
E-mail: atsushi.inoue@ewu.edu

Abstract

Support Vector Machines (SVMs) are a recently introduced Machine Learning technique. SVMs approach binary classification by attempting to find a hyperplane that separates the two categories of training vectors. This hyperplane is expressed as a function of a subset of the training vectors. These vectors are called support vectors. In this paper, we present a method of fuzzifying support vectors based off of the results of an SVM induction. We then propose a method of enhancing SVM induction using these fuzzy support vectors. We finish by presenting a computational example using the IRIS data set.

1 Introduction

Support Vector Machines (SVMs) were first introduced by Vapnik in 1995[12]. An SVM employs a two part approach to binary classification. The first part is a transformation method known as the kernel function. The kernel function transforms a given set of vectors to a (possibly) higher dimensional space. The second part of an SVM, often called the induction engine, attempts to find a hyperplane to separate the transformed vectors into two regions. The hyperplane itself is often defined as a function of a subset of the vectors known as *support vectors.*

To date, SVMs have been applied in many areas and have been shown to be extremely successful[5][7][1] [3][9]. When considering SVM performance on a novel/post training vector, the distance of the point from the induced hyperplane is a natural indication of the SVM's confidence in its decisions. Points that occur nearer to the hyperplane indicate a less strong decision than points classified further from the hyperplane. Figure 1 shows two points in R^2 and their distance from a hyperplane. In previous work with SVM classification[9], it was noticed that while SVM induction was extremely successful, that it was prone to produce a certain class of error. Figure 2 shows a novel point that has been misclassified. In this figure, the positive and negative markings represent support vectors. The misclassified point occurs near to the hyperplane, and is thus a case when the SVM's decision is not all that strong. Errors such as these are called *near errors* and the ideas presented here were developed to reduce near errors. We first begin with a more detailed description of SVM induction before moving to fuzzy support vectors and a computational example.

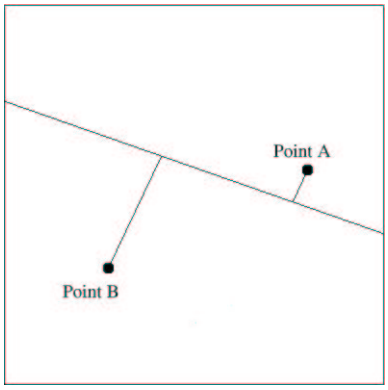

Figure 1. Two Points and their distances to the hyperplane.

1.1 Summary of SVM

The basic SVM performs binary classification[3]. It attempts to separate vectors into two separate groups, called the positive and negative categories. SVMs take as input a training set that consists of l labeled vectors $S = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), ...(\vec{x_l}, y_l)\}$ where each y_i is 1 if $\vec{x_i}$ is in the positive category, and -1 if $\vec{x_i}$ is in the negative cate-

*The Director of Inland Northwest Security Systems Initiative (INSSI) within Department of Computer Science at Eastern Washington University. This research is in part supported by the Congressional Appropriation for EWU's Technology Initiative for New Economy (TINE) and a NSF curriculum development grant (NSF 0230590).

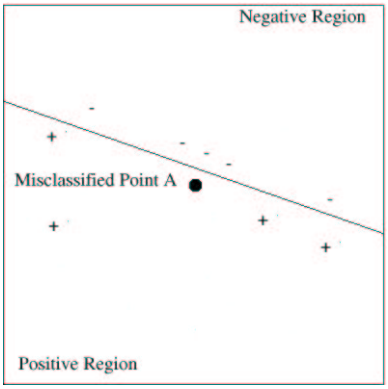

Figure 2. Misclassified Point A has a small distance from the hyperplane.

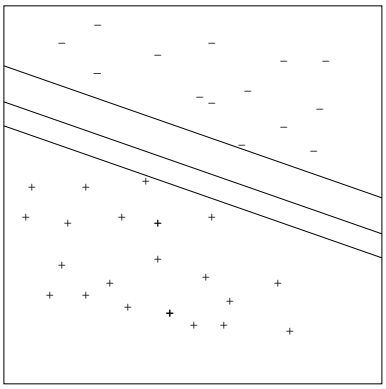

Figure 3. Three out of an infinite number of separating hyperplanes.

gory. SVMs then attempt to output a hyperplane defined as follows:

$$h(\vec{x}) = \vec{w} \cdot \vec{x} + b. \quad (1)$$

A novel point, $\vec{x_n}$ is then classified by looking at the output of $h(\vec{x_n})$. This quantity is known as the *margin of the point* or the *point margin*. The sign of the margin determines which class the point is classified into. One of the main strengths of SVM learning is that the output hypothesis is relatively inexpensive to evaluate for post training inputs.

As mentioned previously, SVMs are composed of the kernel function and the induction engine. The kernel function is based off a transformation function. Given some set of vectors S in R^n, we usually define some function ϕ that maps vectors from R^n to some other space, R^m. We call R^n the input space and R^m the feature space. With ϕ in mind, the kernel function K is defined as follows:

$$K(\vec{x}, \vec{z}) = \langle \phi(\vec{x}), \phi(\vec{z}) \rangle \quad (2)$$

The kernel function appears more prominently in the induction engine, but it also serves the purpose of mapping the input to a space where induction may be easier[3].

As mentioned previously, the SVM induction engine is designed to find a hyperplane to separate the two categories of data in a hopefully linearly separable the feature space. In doing this, the SVM induction engine must bear two things in mind. First of all, assuming that the data is linearly separable, there may be an infinite number of separating hyperplanes. As is shown in Figure 3, an infinite number of hyperplanes are possible. The figure depicts two tight fit hyperplanes and a more middle of the road hyperplane. The SVM induction engine needs to choose the hyperplane that will generalize best to the post-training data. Under most distributions of post training points, the tightest fit hyperplanes would tend to perform poorly. For example, the hyperplane that is fit snug to the positive points will misclassify any novel positive points that happen to occur on the other side of the hyperplane. Similarly with the other tightest fit hyperplane. To address this point, the Maximal Margin SVM (MMSVM) considers the *the margin of the training set*. The set margin is the value of the smallest point margin of the training set vectors. The MMSVM attempts to choose the hyperplane such that the training set points are as far away from the hyperplane as possible. They do this by solving the following optimization problem:

Given:	a training set $S = ((\vec{x_1}, y_1), ...(\vec{x_l}, y_l))$
minimize:	$\vec{w} \cdot \vec{w}$
subject to:	$y_i(\langle \vec{w} \cdot \vec{x_i} \rangle + b) \geq 1, i = 1, 2...l$

The constraint placed on the problem amounts to saying that the hyperplane must correctly classify the training set. A basic theorem of MMSVM[3] states that the $\vec{w}$ that satisfies the problem also maximizes the margin of the training set.

SVM induction engines must also concern themselves with the case when the transformed data is not linearly separable. The MMSVM cannot be used in this case since it requires that the hyperplane correctly classify the entire training set. The Soft Margin SVM (SMSVM) allows the hyperplane to mis-classify some of the training set examples. It too solves an optimization problem.

Given:	a training set $S = ((\vec{x_1}, y_1), ...(\vec{x_l}, y_l))$
minimize:	$\frac{1}{2}\vec{w} \cdot \vec{w} - C \sum_{i=1}^{l} \xi_i$
subject to:	$y_i(\langle \vec{w} \cdot \vec{x_i} \rangle + b) \geq 1 - \xi_i, i = 1, 2...l$
	$\xi_i \geq 0, i = 1, 2...l$

The ξ_i's are called the slack variables and define how much slack there is to be in the margin for each vector in the

set S. This allows the hyperplane to mis-classify some of the examples. The slack variables form the vector $\vec{\xi}$ which is known as the *margin slack vector*. The summation of the slack variables in the objective function controls the growth of the margin slack vector. The SMSVM attempts to minimize the total slack in the training set in this way. Finally, the parameter C defines how much weight is given to the minimization of the slack vector as compared to the weight vector. Fundamental theorems of SMSVMs state that the $\vec{w}$ that solves this problem provides the weight vector for a hyperplane[1] that generalizes the training set within Probably Approximately Correct acceptable bounds [3][11]. It is often computationally infeasible to straightforwardly calculate the solution to this optimization problem. So, a transformation to the dual is often performed. In this case we get the following optimization problem:

$$\begin{array}{ll} \text{Given:} & \text{a training set } S = ((\vec{x_1}, y_1), ...(\vec{x_l}, y_l)) \\ \text{minimize:} & -\sum_{i=1}^{l} \alpha_i \\ & +\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\vec{x_i}, \vec{x_j}) \\ \text{subject to:} & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & 0 \geq \alpha_i \geq C, i = 1, ..., l \end{array}$$

In this form, the problem is to now find the optimal vector $\vec{\alpha}$. Using this vector, we can compute $h(\vec{x}) = \vec{w} \cdot \vec{x}$ as follows:

$$h(\vec{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\vec{x_i}, \vec{x}) \quad (3)$$

Vectors with their $\alpha_i > 0$ are called *support vectors*. We do not need the other vectors in our computation of $\vec{w} \cdot \vec{x}$ since their $\alpha_i = 0$ and thus the product $\alpha_i y_i K(\vec{x_i}, \vec{x})$ is zero and drops out of the summation. Thus, the support vectors are the vectors that are most crucial to defining the hyperplane and the must crucial to our post training classification.

2 Fuzzy SVM Classification

Previous work on Fuzzy SVM classification has approached the problem using the point margins. In previous work[1], it is noted that the magnitude of the margin is a representation of the strength of the classification. Thus, it is in a way analogous to membership functions[8] of fuzzy sets. The fuzzy margin is then used to approach the multiclass classification problem.

In this work, we approach fuzzification of SVM classification using support vectors. The dual formulation of the problem and its search for $\vec{\alpha}$ suggests that some support vectors are better or more influential than others. That is, those support vectors whose α_i are greater hold more influence in the hyperplane (and post training classification) than vectors with small α_i. This leads us to consider creating fuzzy support vectors, or membership functions for support vectors based off $\vec{\alpha}$. These membership functions would provide higher membership values to vectors with large α_i and lower values to those vectors with small α_i. In the next subsection, we describe sample fuzzy sets for support vectors.

Once a membership function for support vectors has been decided upon, the question then becomes how to make use of them for classification. In this paper, we consider creating two membership functions over the set of post-training or test vectors: pos and neg. pos represents a novel vector's membership in the positive class, while neg represents a novel vector's membership in the negative class. When deciding in which class a novel point falls, we will choose the class for which the corresponding fuzzy set reports the highest membership. Thus, the new decision function is:

$$c(\vec{x}) = \text{ARGMAX}_{\{pos,neg\}}[\mu_{pos}(\vec{x}), \mu_{neg}(\vec{x})] \quad (4)$$

2.1 Example Membership Functions

Using $\vec{\alpha}$ the following sample membership function was considered. Define M as the value for which α_i is maximal. Then define μ_{sv} as:

$$\mu_{sv}(\vec{x_i}) = \frac{\alpha_i}{M} \quad (5)$$

With this definition in hand, we can construct membership functions for pos and neg. Very simply, for a given point $\vec{x}$, we set set $\mu_{pos}(\vec{x})$ as:

$$\mu_{pos}(\vec{x}) = \sum_{1}^{n} \frac{\mu_{sv}(n\vec{p}_i)}{d_i} \quad (6)$$

and $\mu_{neg}(\vec{x})$ as:

$$\mu_{neg}(\vec{x}) = \sum_{1}^{n} \frac{\mu_{sv}(n\vec{n}_i)}{d_i} \quad (7)$$

where $n\vec{p}_i/n\vec{n}_i$ is the ith nearest (by Euclidean distance) positive/negative support vector from $\vec{x}$ and d_i is the Euclidean distance from between $n\vec{p}_i/n\vec{n}_i$ and $\vec{x}$.

These definitions for pos and neg amount to nearest neighbor classification [4]. The difference here is that we have relied on SVM induction to identify the memorized set of points. It is hoped that near errors such as those shown in Figure 2 can be reduced by considering the relationship of a novel point to its near by support vectors.

The definitions presented in this subsection are by no means the most desirable or efficient definitions. This is clearly an avenue of future research.

[1]The offset of the hyperplane be calculated using the solutions to the optimization problem.

Table 1. Percent Accuracy: SVM vs. Fuzzy Support Vectors

Method	Setosa	Versicolour	Virginica
SVM	100%	66.67%	90.61%
Fuzzy SV	100%	94.67%	94.67

3 Experimental Setup and Results

Using the definitions of the previous section, classification using fuzzy support vectors was compared to a standard support vector implementation. Both methods were tested on the IRIS[2] dataset, a data set of three types of plants in the iris family. For the experiment, three binary tasks were created. The first involved separating examples of Setosa from the other two classes. The second task involved separating examples of Versicolour from the other two while the final task was separating examples of Virginica from the other two classes. For each task, the data was randomly split in half to form training and testing sets. Both the training and testing sets contained the same number of examples from each of the three classes.

SVM Light[6] was the SVM implementation used for the experiment. In this case, the linear kernel, the kernel whose transformation function was $\phi(\vec{x}) = \vec{x}$ was used to generate the support vectors and was then compared against the method outlined above. For μ_{pos} and μ_{neg}, n was set at ten. The percentage accuracies after training are listed in Table 1.

In two out of the three tasks, Fuzzy Support Vectors performed better than a linear SVM classifier. In the third case, both methods performed equally well.

4 Discussion

As can be seen by Table 1, the application of fuzzy support vectors proposed here possesses some utility. Of course, this is only one example using one kernel. Other application domains such as image processing[3], text classification [6][9], and network security [10] should be tested. In addition, this method should be tested using more than just the linear kernel.

Another note is that the current definitions for membership in pos and neg have a significant run time drawback with respect to standard SVMs. As discussed above, for each novel point, the n nearest positive and n nearest negative support vectors are calculated. Finding $2n$ nearest fuzzy support vectors per novel point requires more runtime than computing a dot product of a weight vector with an input vector. So, while in this limited test case, the definitions seem to be providing a benefit, it is at the cost of runtime expediency. As discussed earlier, one of the the motivations of this study was to reduce near errors. One alternative to the fuzzy classification discussed already that can help balance the increased runtime and still reduce near errors, is to only run the fuzzy classification on points whose margin's absolute value is below some threshold and otherwise accept the SVM classification. So, unless there are a great number of potential near errors, the majority of the post-training points will be processed by the efficient SVM decision function. Another alternative is to create pos and neg membership functions that are linear combinations of SVM outputs and fuzzy support vectors.

Other methods to reduce the runtime complextity of the current definitions invovle reducing the number of support vectors considered. For example support vectors with μ_{sv} below a certain threshold could be discarded. Another possibility is to combine near by fuzzy support vectors into one ideal or clustered fuzzy support vector.

Finally, we note that it is also possible to redefine support vector membership functions in terms of the slack variables output by the SMSVM. Support vectors with smaller slack variables could be considered to have higher membership values than support vectors with large slack values.

5 Conclusion

Fuzzy support vectors are introduced and initially shown to be effective when classifying the IRIS data set. In addition, many improvements to the proposed fuzzy set definitions are proposed. Finally, it is suggested that more extensive and rigorous tests are required.

References

[1] Shigeo Abe. Support vector machines for pattern classification, March 2001. Unpublished manuscript provided by A. Inoue.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] Nello Christianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[4] Cover and Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[5] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML)*. Springer, 1998.

[6] Thorsten Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[7] Thorsten Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of the International Conference on Machine Learning*, 2001.

[8] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, 1995.

[9] J. Mill. Support vector machines, n-gram kernels, and text classification. Master's thesis, Eastern Washington University, 2002.

[10] P. Miller, J. Mill, and A. Inoue. Synergistic and perceptual intrusion detection with reinforcement (spider). In *Midwest Conference on Artifical Intelligence and Cognitive Science.*, 2003.

[11] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[12] V. Vapnik. *The Nature of Statistical Learning*. Springer, 1995.