Precision and Recall in Rough Support Vector Machines

P. Lingras and C.J. Butz

***Abstract*— Rough support vector machines (RSVMs) supplement conventional support vector machines (SVMs) by providing a better representation of the boundary region. Increasing interest has been paid to the theoretical development of RSVMs, which has already lead to a modification of existing SVM implementations as RSVMs. This paper shows how to extend the use of precision and recall from a SVM implementation to a RSVM implementation. Our approach is demonstrated in practice with the help of Gist, a popular SVM implementation.**

I. Introduction

Support vector machines (SVMs) [15] and multilayer neural networks are two approaches that are popularly used to overcome the linear separability limitations of perceptrons [7], [13]. Neural networks use hidden layers to introduce non-linearity. SVMs, on the other hand, use non-linear transformations for separating non-linearly separable objects. In addition, SVMs maximize the margin between two classes, and use the hyperplane bisecting the margin for separating the two classes. It is postulated that with the help of an appropriate kernel function, every binary classification problem can be modeled using an SVM. However, such a kernel function may lead to very high dimensional feature space, and as such, may be computationally infeasible. Therefore, SVMs must allow for soft margins. SVMs with soft margins allow for classification errors. Most real world classification problems are modeled by SVMs with soft margins. The margins used by SVMs, in general, and, soft margins in particular, provide a natural link between SVMs and *rough set theory* (RST) [10].

Lingras and Butz [5] first proposed *rough support vector machines* (RSVMs) as a link between binary SVMs and RST. Subsequently, they extended their approach to multi-classifications and also provided some experimental findings [6]. Since then other researchers have also contributed to the theoretical and experimental development of RSVMs [16].

The quality of classifications resulting from SVMs can be measured using the notions of precision and recall. The measures are defined for a binary classification of "yes" and "no" with a particular emphasis on the objects classified as "yes." Precision measures the percentage of objects that are correctly classified as "yes" as a ratio of all the objects that should truly be "yes." Recall describes what proportion of objects that should be classified as "yes" are in fact classified as "yes."

Pawan.Lingras@StMarys.CA Department of Mathematics and Computing Science, Saint Mary's University Halifax, Nova Scotia, B3H 3C3, Canada. butz@cs.uregina.ca Department of Computer Science, University of Regina, Regina, Regina, Saskatchewan, S4S 0A2, Canada.

This paper adapts the notions of precision and recall to binary SVM classifiers, where both class labels are meaningful, that is, one class is not simply the negation of the other class. The precision and recall measures are then used to in RSVM classification. A popular implementation of SVM, called Gist, will be used to demonstrate the proposed implementation of RSVM.

The remainder of this paper is organized as follows. Sections 2 and 3 briefly review SVMs and RSVMs, respectively. In Sections 4 and 5, we respectively show how to use precision and recall in SVMs and RSVMs. Section 6 contains our conclusions.

II. Support Vector Machines

In this paper, we will restrict our attention to binary classification problems. Let $\mathbf{x}$ be an input vector in the input space X. Let y be the output in $Y = \{+1, -1\}$. Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_i, y_i), \ldots\}$ be the training set used for supervised classification. Let us define the inner product of two vectors $\mathbf{x}$ and $\mathbf{w}$ as:

$$< \mathbf{x}, \mathbf{w} > \quad = \quad \sum_j x_j \times w_j,$$

where x_j and w_j are components of the vectors $\mathbf{x}$ and $\mathbf{w}$, respectively. If the training set is linear separable, the perceptron learning algorithm will find the vector $\mathbf{w}$ such that:

$$y \times [\ < \mathbf{x}, \mathbf{w} > + b\ ] \quad \geq \quad 0, \qquad (1)$$

for all $(\mathbf{x}, y) \in S$. SVMs overcome the shortcomings of linear separability in the perceptron approach by using a mapping Φ of the input space to another feature space with higher dimension. Equation (1) for perceptrons is then changed as follows:

$$y \times [\ < \Phi(\mathbf{x}), \Phi(\mathbf{w}) > + b\ ] \quad \geq \quad 0, \qquad (2)$$

for all $(\mathbf{x}, y) \in S$. Usually, a high dimensional transformation is needed in order to obtain reasonable classification [2]. Computational overhead can be reduced by not explicitly mapping the data to feature space, but instead just working out the inner product in that space. In fact, SVMs use a kernel function K corresponding to the inner product in the transformed feature space as: $K(\mathbf{x}, \mathbf{w}) =< \Phi(\mathbf{x}), \Phi(\mathbf{w}) >$. Polynomial kernel is one of the popular kernel functions. Let us derive the polynomial kernel function of degree 2 for two dimensional input space. Let $\mathbf{x} = (x_1, x_2)$ and

$\mathbf{w} = (w_1, w_2)$:

$$\begin{aligned} K(\mathbf{x}, \mathbf{w}) &= < \mathbf{x}, \mathbf{w} >^2 \\ &= (x_1 w_1 + x_2 w_2)^2 \\ &= (x_1^2 w_1^2 + x_2^2 w_2^2 + 2 x_1 w_1 x_2 w_2) \\ &= < x_1^2 + x_2^2 + \sqrt{2} x_1 x_2, \ w_1^2 \\ &\quad + w_2^2 + \sqrt{2} w_1 w_2 > \\ &= < \Phi(\mathbf{x}), \Phi(\mathbf{w}) > . \end{aligned}$$

The dimensionality rises very quickly with the degree of polynomial. For example, Hoffmann [3] reports that for an original input space with 256 dimensions, the transformed space with second degree polynomials was approximately 33,000, and for the third degree polynomials the dimensionality was more than a million, and fourth degree led to a more than billion dimension space. This problem of high dimensionality will be discussed later in the paper.

The original perceptron algorithm was used to find one of the possibly many hyperplanes separating two classes. The choice of the hyperplane was arbitrary. SVMs use the size of margin between two classes to search for an optimal hyperplane. The problem of maximizing the margin can be reduced to an optimization problem [2], [15]: minimize $< \mathbf{x}, \mathbf{w} >$ such that

$$y \times [\ < \mathbf{x}, \mathbf{w} > + b\ ] \quad \geq \quad 0, \tag{3}$$

for all $(\mathbf{x}, y) \in S$. SVMs attempt to find a solution to such an optimization problem.

III. Rough Support Vector Machines

This section describes *rough support vector machines* (RSVMs), proposed by Lingras and Butz [5]. A certain familiarity with RST is assumed. We will first consider the ideal scenario, where the transformed feature space is linear separable and the SVM has found the optimal hyperplane by maximizing the margin between the two classes. There are no training examples in the margin. The optimal hyperplane gives us the best possible dividing line. However, if one chooses to not make an assumption about the classification of objects in the margin, the margin can be designated as the boundary region. This will allow us to create rough sets as follows.

Let us define b_1 as: $y \times [\ < \mathbf{x}, \mathbf{w} > + b_1\ ] \geq 0$, for all $(\mathbf{x}, y) \in S$, and there exists at least one training example $(\mathbf{x}, y) \in S$ such that $y = 1$ and $y \times [\ < \mathbf{x}, \mathbf{w} > + b_1\ ] = 0$. Similarly, b_2 is defined as: $y \times [\ < \mathbf{x}, \mathbf{w} > + b_2\ ] \geq 0$, for all $(\mathbf{x}, y) \in S$, and there exists at least one training example $(\mathbf{x}, y) \in S$ such that $y = -1$ and $y \times [\ < \mathbf{x}, \mathbf{w} > + b_2\ ] = 0$. It can be easily seen that b_1 and b_2 correspond to the boundaries of the margin. The modified SVM classifier can then be defined as follows:

If $< \mathbf{x}, \mathbf{w} > + b_1 \geq 0$, classification of $\mathbf{x}$ is $+1$.
If $< \mathbf{x}, \mathbf{w} > + b_2 \geq 0$, classification of $\mathbf{x}$ is -1.
Otherwise, classification of $\mathbf{x}$ is uncertain.

The proposed classifier will allow us to create three equivalence classes, and define a rough set based approximation space. This simple extension of an SVM classifier provides a basis for a more practical application, when the SVM transformation does not lead to a linear separable case. Cristianini [2] list disadvantages of refining feature space to achieve linear separability. Often this will lead to high dimensions, which will significantly increase the computational requirements. Moreover, it is easy to overfit in high dimensional spaces, i.e., regularities could be found in the training set that are accidental, which would not be found again in a test set. The soft margin classifiers [2] modify the optimization problem to allow for an error rate. The rough set based rules given by the above rules can still be used by empirically determining the values of b_1 and b_2. For example, b_1 can be chosen in such a way that, for an $(\mathbf{x}, y) \in S$, if $< \mathbf{x}, \mathbf{w} > + b_1 \geq 0$, then y must be +1. Similarly, b_2 can be chosen such that, for an $(\mathbf{x}, y) \in S$, if $< \mathbf{x}, \mathbf{w} > + b_2 \leq 0$, then y must be -1. Assuming there are no outliers, such a choice of b_1 and b_2 would be reasonable. Otherwise, one can specify that the requirements hold for a significant percentage of training examples. For instance, b_1 can be chosen in such a way that, for an $(\mathbf{x}, y) \in S$, if $< \mathbf{x}, \mathbf{w} > + b_1 \geq 0$, then in at least 95% of the cases y must be +1. Similarly, b_2 can be chosen in such a way that, for an $(\mathbf{x}, y) \in S$, if $< \mathbf{x}, \mathbf{w} > + b_2 \leq 0$, then in at least 95% of the cases y must be -1.

The extension proposed by Lingras and Butz [5] can be easily implemented after the soft margin classifier determines the value of $\mathbf{w}$. All the objects in the training sample will be sorted based on the values of $< \mathbf{x}, \mathbf{w} >$. The value of b_1 can be found by going down (or up if the positive examples are below the hyperplane) in the list until 95% of the positive examples are found. The value of b_2 can be found by going up (or down if the positive examples are below the hyperplane) in the list until 95% of the negative examples are found.

IV. Precision and Recall in SVMs

Precision and recall are complementary measures most commonly used by information retrieval theorists and practitioners [14]. They can also be used to measure the effectiveness of a classifier. Let us assume that we have a classification problem, where we need to decide whether or not an object belong to a single class. If an object belongs to a class we will call that object positive; otherwise, it will be deemed negative. Let R be the set of objects that belong to the class. Let us assume that our classifier has identified A as the set of objects as belonging to the class. Precision is defined as:

$$precision \ = \ \frac{\|R \bigcap A\|}{\|A\|}, \tag{4}$$

where $\|X\|$ denotes the cardinality of a set X. Eq. (4) tells us what proportion of objects that are identified as postitives are actually positive. Recall is defined as:

$$recall \ = \ \frac{\|R \bigcap A\|}{\|R\|} \tag{5}$$

Eq. (5) gives us the proportion of actual positive objects that are correctly identified as positives by our classifier. Precision

and recall are competing measures. An attempt to maximize precision usually leads to lower recall values, and vice versa.

It should be noted that while the precision and recall measures are technically defined for binary classification system, they in fact deal with "yes" and "no" answers for a single class. A typical binary classification problem may actually want to separate objects into two different classes, such as class 1 and class 2. In this case, we should extend the notion of precision and recall by defining separate values for the two classes. Let p_1 and r_1 be the precision and recall for class 1. Similarly, let p_2 and r_2 be the precision and recall for class 2. Let R_1 be the set of objects belonging to class 1 and R_2 the set of objects belonging to class 2. Let us assume that our classifier has identified A_1 and A_2 as the sets of objects belonging to class 1 and class 2, respectively. By definition, the precision p_1 and recall r_1 of class 1 are:

$$p_1 = \frac{\|R_1 \bigcap A_1\|}{\|A_1\|}, \tag{6}$$

$$r_1 = \frac{\|R_1 \bigcap A_1\|}{\|R_1\|}, \tag{7}$$

while for class 2, we have:

$$p_2 = \frac{\|R_2 \bigcap A_2\|}{\|A_2\|}, \tag{8}$$

$$r_2 = \frac{\|R_2 \bigcap A_2\|}{\|R_2\|}. \tag{9}$$

When a classifier precisely classifies an object as either class 1 or 2, then

$$R_1 \bigcup R_2 \ = \ A_1 \bigcup A_2.$$

Thus,

$$p_1 \ = \ r_1 \ = \ p_2 \ = \ r_2 \ = \ 1.$$

However, the relationship between p_1, p_2, r_1, and r_2 is more complicated. A higher p_1 (r_1) does not necessary imply a higher p_2 (r_2). On the other hand, a higher value of p_1 (p_2) will usually result in a higher value of r_2 (r_1).

Lemma 1 p_1 is proportional to r_2, and p_2 is proportional to r_1.

Proof: A large value of p_1 implies that most of the elements in A_1 belong to class 1. This means that there are very few class 2 objects in A_1. This, in turn, implies that most of the class 2 objects will belong to A_2. Hence, a higher r_2 is obtained. A similar argument hold for the second claim.

The interaction between these four values becomes even more complex, once we introduce the notion of boundary region from RST.

V. Precision and Recall in RSVMs

In this section, we will discuss the operational details of implementing precision and recall in a binary RSVM.

Our implementation utilizes the Gist software tools [8], [9] for SVM classification that are downloadable from http://microarray.cpmc.columbia.edu/gist/. The SVM portion of Gist is also available via an interactive web server at http://svm.sdsc.edu [8]. The web interface for Gist expects the following three input three files [9]:

1) Training data: This file should be a tab-delimited text file. The first row should contain feature names, and the first column should contain example names. The rest of the file should consist of a matrix of numbers with each row of the matrix corresponding to an example.
2) Class labels: This file should be a tab-delimited text file containing two columns. The first column should contain the same example identifiers (as the training set file) and in the same order. The second column should contain class labels (1 for the positive class, and -1 for the negative class).
3) Test data: This file is similar in format to the training set. It should contain the same number of features as the training set, but may contain different examples. If you do not have a separate test file, you can always use the training data file here. The class labels have to be modified to take the values of 1 and -1. For example, when we run Gist for classification of classes 1 and 2, we specify 1 for class 1 and -1 for class 2.

Once these three files have been specified, you can choose various options, including the Kernel function, and then submit the data. After Gist has completed the training and testing phases, the results are returned.

As shown in Fig. 1, we created a synthetic feature space with two dimensions and consisting of 100 objects. These 100 objects belong to two classes; each class containing 50 objects. As can be seen from Fig. 1, it is possible to separate most of the objects from these two classes using hyperplanes, but there will be a certain percentage of false positives and negatives that should belong to the boundary regions.

Table 1 shows partial results of SVM classification obtained from Gist for classification of classes 1 and 2. The first column in the table is Object ID, the second column gives the actual classification of the object. The weight shown in third column is an indication of whether the object is a support vector or not. The predicted class appears in the fourth column. The fifth column gives the values of discriminants, which can be used as surrogates of the distances from the hyperplane separating the positives from negatives. Usually, a positive discriminant corresponds to the object above the hyperplane, and negative discriminant indicates an object below the hyperplane. Therefore, we do not have to explicitly determine b_1 and b_2 [6]. If we sort the objects in descending order first according to the actual class and then according to the discriminant value, as shown in Table 1, we will be able to identify the boundary region using the precision and recall values.

Fig. 2 shows the precision and recall values for the objects from Table 1. As we can see that the values of p_1 are positively correlated to r_2, and p_2 is positively correlated with r_1. This provides experimental evidence for Lemma 1.

Based on the precision and recall graph, we can draw the cut-offs for lower bounds of both the classes. Let us assume that we are aiming for a precision of 0.8 and recall of 0.7.

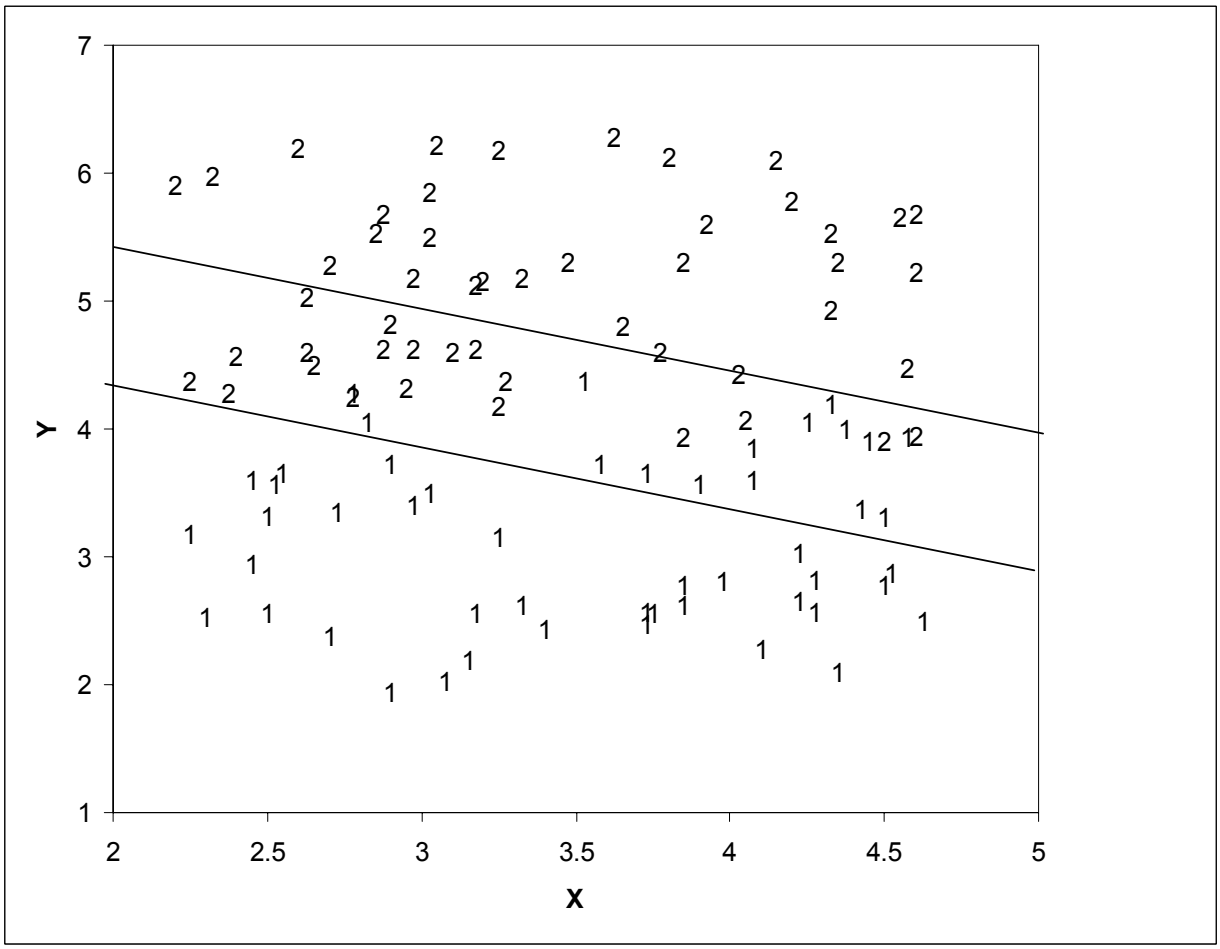

Fig. 1. Synthetic training data of 100 two-dimensional objects.

c	class	weight	train classification	train discriminant
41	1	0	1	1.335
24	1	0	1	1.192
⋮	⋮	⋮	⋮	⋮
4	1	1.294	1	0.2862
40	1	1.308	1	0.2796
77	-1	-2.312	1	0.2757
⋮	⋮	⋮	⋮	⋮
36	1	2.683	-1	-0.4806
⋮	⋮	⋮	⋮	⋮
61	-1	-1.109	-1	-0.3897
81	-1	-1.099	-1	-0.3949
⋮	⋮	⋮	⋮	⋮
69	-1	0	-1	-1.145
57	-1	0	-1	-1.191

TABLE I

PARTIAL GIST OUTPUT FOR THE TRAINING DATA IN FIG. 1.

The boundaries drawn in Fig. 2 provide a reasonable balance between precision and recall, which describe the lower and upper bounds of classes 1 and 2 as follows. The lower bound of class 1 is drawn at a precision, denoted $\underline{p_1}$, of 0.86, while the recall, denoted $\underline{r_1}$, is 0.74. The upper bound of class 1 has precision $\overline{p_1}$ = 0.74 and recall $\overline{r_1}$ = 0.86. For class 2, the lower bound is represented with precision $\underline{p_2}$ = 0.84 and recall $\underline{r_2}$ = 0.72, while the upper bound corresponds to precision $\overline{p_2}$ = 0.76 and recall $\overline{r_2}$ = 0.88.

It is important to note that the end of the lower bound of class 1 in a precision and recall graph signifies the beginning of upper bound of class 2, and vice versa. The precision for an upper bound will be lower than that for the lower bound. As can be seen from Fig. 2, the precision for the upper bound of class 1 is $\overline{p_1}$ = 0.74, which is less than the precision of the lower bound $\underline{p_1}$ = 0.86. On the contrary, the recall for upper bound will be higher than that of the lower bound. For instance, the recall $\overline{r_1}$ = 0.86 of the upper bound is greater than $\underline{r_1}$ = 0.74 for the lower bound. Similarly, the precision $\overline{p_2}$ = 0.76 is less than $\underline{p_2}$ = 0.84, while the recall $\overline{r_2}$ = 0.88

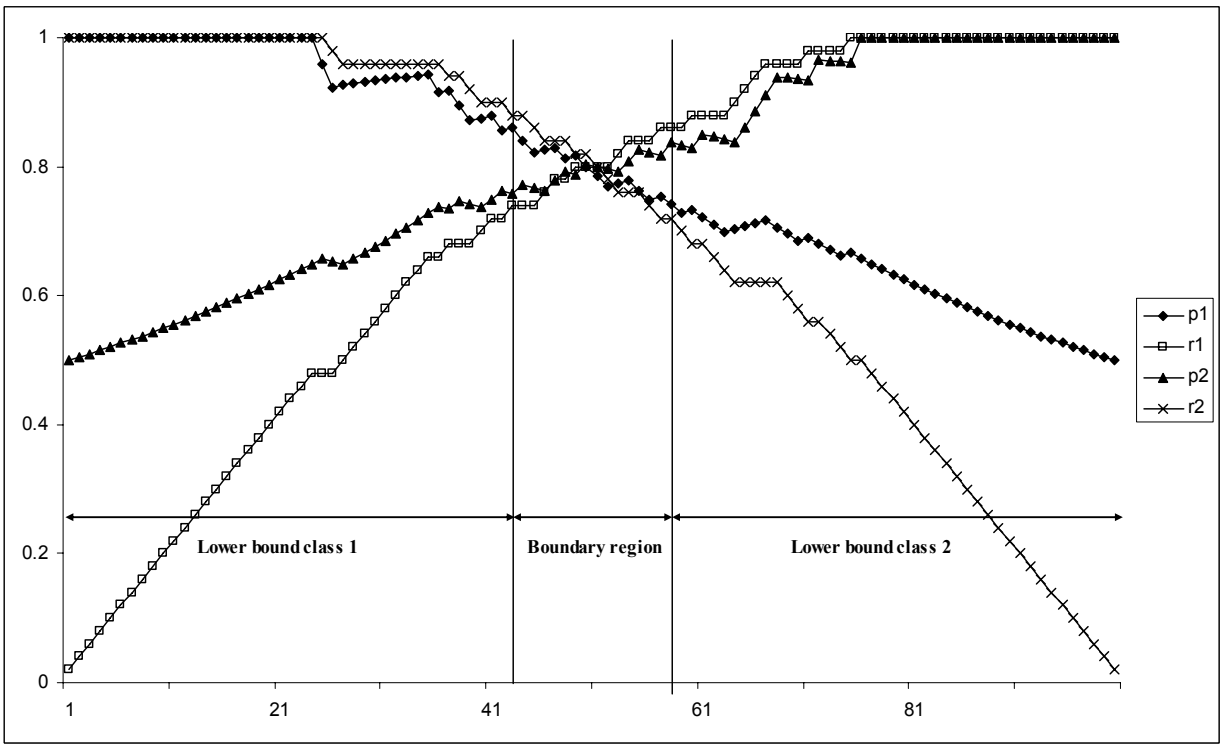

Fig. 2. A RSVM precision and recall graph for the Gist output in Table I.

is greater than $\underline{r_2}$ = 0.72.

VI. Conclusion

This paper describes a precision and recall based implementation of RSVMs. The conventional precision and recall values are based on whether an object belongs to a class or not. We suggest using two different precision and recall values for binary SVM classification. Lemma 1 shows that the precision of one class is correlated to the recall of the other class.

The paper uses results from an implementation of SVM called Gist. The graphs of precision and recall for binary SVM classification of a synthetic dataset is used to illustrate how to draw the boundaries for lower bounds of both classes. We will be applying these precision and recall based implementation for multi-classification. The results of our work will be presented in a future publication.

References

[1] F. Chang, C-H. Chou, C-C. Lin, and C-J. Chen, "A Prototype Classification Method and Its Application to Handwritten Character Recognition," *Proceedings of IEEE International Conference on Systems, Man and Cybernetics,* 2004, pp. 4738-4743.

[2] N. Cristianini, "Support Vector and Kernel Methods for Pattern Recognition," http://www.support-vector.net/tutorial.html, 2003.

[3] A. Hoffmann, "VC Learning Theory and Support Vector Machines" http://www.cse.unsw.edu.au/ cs9444/Notes02/Achim-Week11.pdf, 2003.

[4] S. Knerr, L. Personnaz, G. Dreyfus, "Single-layer learning revisited: A stepwise procedure for building and training a neural network" In Fogelman-Soulie and Herault, editors, *Neurocomputing: Algorithms, Architectures and Applications,* NATO ASI. Springer, 1990.

[5] P. Lingras, C.J. Butz, "Interval Set Classifiers using Support Vector Machines," *Proceedings of 2004 conference of the North American Fuzzy Information Processing Society,* Banff, AB., June 27-30, pp. 707-710, 2004.

[6] P. Lingras, C.J. Butz, "Rough Set based 1-v-1 and 1-v-r Approaches to Support Vector Machine Multi-classification," *to appear in Information Sciences*, 2007.

[7] M.L. Minsky, S.A. Papert, *Perceptrons*, The MIT Press, Cambridge, MA., 1969.

[8] W. S. Noble and P. Pavlidis, Gist Support Vector Machine, http://svm.sdsc.edu/svm-intro.html, 2006.

[9] W. S. Noble and P. Pavlidis, Gist Support Vector Machine: Web Interface, http://svm.sdsc.edu/cgi-bin/nph-SVMsubmit.cgi, 2006.

[10] Z. Pawlak, *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1992.

[11] J.C. Platt, "Support Vector Machines," http://research.microsoft.com/users/jplatt/svm.html, 2003.

[12] J.C. Platt, N. Cristianini, J. Shawe-Taylor, "Large margin DAG's for multiclass classification," *Advances in Neural Information Processing Systems,* MIT Press, Cambridge, MA, (2000) 547-553

[13] F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," *Technical Report 85-460-1*, Project PARA, Cornell Aeronautical Lab, 1957.

[14] G. Salton, M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, NY., 1983.

[15] V. Vapnik, *Statistical Learning Theory*. Wiley, NY., 1998.

[16] T. Wakaki, H. Itakura, M. Tamura, H. Motoda, and T. Washio, "A Study on Rough Set-Aided Feature Selection for Automatic Web-Page Classification," *Web Intelligence and Agent Systems: An International Journal*, 4(3), 431-441, 2006.