Reducing the Storage Requirements of 1-v-1 Support Vector Machine Multi-classifiers

P. Lingras[1] and C.J. Butz[2]

[1] Department of Math and Computer Science, Saint Mary's University
Halifax, Nova Scotia, Canada, B3H 3C3.
E-mail: Pawan.Lingras@stmarys.ca

[2] Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
E-mail: butz@cs.uregina.ca

Abstract. The methods for extending binary support vectors machines (SVMs) can be broadly divided into two categories, namely, 1-v-r (one versus rest) and 1-v-1 (one versus one). The 1-v-r approach tends to have higher training time, while 1-v-1 approaches tend to create a large number of binary classifiers that need to be analyzed and stored during the operational phase. This paper describes how rough set theory may help in reducing the storage requirements of the 1-v-1 approach in the operational phase.

1 Introduction

While support vector machines (SVMs) improve traditional perceptrons [6,10] by using a higher dimensional space and identifying planes that provide maximal separation between two classes, they are essentially binary classifiers. In order to increase the applicability of SVMs, it is necessary to extend them to multi-classification. Vapnik [11] proposed the 1-v-r (one versus rest) approach, which involves constructing a binary classifier for each class that separates objects belonging to one class from objects that do not. If we assume that there are N classes, the 1-v-r approach will create N binary classifiers. Knerr, et al. [4] suggested the use of 1-v-1 (one versus one) approach, whereby a separate SVM is constructed for every pair of classes. This approach has been further investigated by many researchers. The 1-v-1 approach involves creating binary classifiers for each pair of classes, thus creating $N \times (N-1)/2$ classifiers. Platt, et al. [9] proposed the use of directed acyclic graphs, referred to as DAGSVM, to reduce the number of computations when deciding classification of objects during the testing and operational phases. Similar to the classical 1-v-1 approach, however, DAGSVMs need to store $N \times (N-1)/2$ SVMs [1]. Lingras and Butz [5] provided interpretation of the binary classification resulting from a SVM in terms of interval or rough sets. This paper extends the formulation proposed by Lingras and Butz for multi-classification using the 1-v-1 approach. The paper also explores

the advantages of such a formulation, which include the same time complexity during operational phase as the DAGSVM, but significantly lower storage requirements.

This paper is organized as follows. Section 2 reviews support vector machines for binary classification. A rough sets approach to support vector machine binary classification is given in Section 3. In Section 4, we present a rough set approach to 1-v-1 support vector machine multi-classifiers. Conclusions are made in Section 5.

2 Support Vector Machines for Binary Classification

Let $\mathbf{x}$ be an input vector in the input space X. Let y be the output in $Y = \{+1, -1\}$. Let $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots\}$ 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:

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

for all $(\mathbf{x}, \mathbf{y}) \in S$. 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:

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

for all $(\mathbf{x}, \mathbf{y}) \in S$. 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 $\mathbf{w} = (w_1, w_2)$:

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

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 [2,11]: minimize $< \mathbf{x}, \mathbf{w} >$ such that

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

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

The problem of multi-classification, especially for systems like SVMs, does not present an easy solution [9]. It is generally simpler to construct classifier theory and algorithms for two mutually-exclusive classes than it is for N mutually-exclusive classes. Platt [8] claimed that constructing N-class SVMs is still an unsolved research problem. The standard method for N-class SVMs [11] is to construct N SVMs. The ith SVM will be trained with all of the examples in the ith class with positive labels, and all other examples with negative labels. Platt et al. [9] refer to SVMs trained in this way as 1-v-r SVMs (short for one versus rest). The final output of the N 1-v-r SVMs is the class that corresponds to the SVM with the highest output value. Platt et al. [9] list the disadvantages of 1-v-r approach as follows. There is no bound on the generalization error for the 1-v-r SVM, and the training time of the standard method scales linearly with N.

Another method for constructing N-class classifiers from SVMs is derived from previous research into combining two-class classifiers. Knerr et al. [4] suggested constructing all possible two-class classifiers from a training set of N classes, each classifier being trained on only two out of N classes. Thus, there would be $N \times (N-1)/2$ classifiers. Platt et al. [9] refer to this as 1-v-1 SVMs (short for one-versus-one). Platt et al. [9] proposed DAGSVM, which uses directed acyclic graphs to reduce the number of SVMs that need to be used during the testing and operational phase. Chang et al. [1] studied one-against-one and DAGSVM. In the training phase, both methods require solving $N \times (N-1)/2$ binary classification problems. In the testing phase, the one-against-one technique conducts $N \times (N-1)/2$ classifications, while the DAGSVM technique employs a directed acyclic graph that has $N \times (N-1)/2$ nodes and N leaves, reducing the number of classifications to $N-1$. Both methods are subject to the drawback that, when the number of classes N is large, they incur exhaustive amount of training time and produce an extremely large set of support vectors [1].

The 1-v-r approach creates N SVMs as opposed to $N \times (N-1)/2$ SVMs created and stored for 1-v-1 and DAGSVM methods. The training of 1-v-1 is computationally less expensive, since only a subset (corresponding to the pair

of classes involved) of the training sample is used. This paper describes a rough set based scheme that makes it possible to reduce the training set size, provides a possible semantic description for the multiclassification, and makes the testing and operational phase more streamlined.

3 Rough Sets based on Binary Support Vector Machine Classification

This section describes a rough set interpretation of SVM binary classification proposed by Lingras and Butz [5]. A certain familiarity with *rough set theory* [7] 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: $\mathbf{y} \times [\ < \mathbf{x}, \mathbf{w} > + b_1\ ] \geq 0$, for all $(\mathbf{x}, \mathbf{y}) \in S$, and there exists at least one training example $(\mathbf{x}, \mathbf{y}) \in S$ such that $y = 1$ and $\mathbf{y} \times [\ < \mathbf{x}, \mathbf{w} > + b_1\ ] = 0$. Similarly, b_2 is defined as: $\mathbf{y} \times [\ < \mathbf{x}, \mathbf{w} > + b_2\ ] \geq 0$, for all $(\mathbf{x}, \mathbf{y}) \in S$, and there exists at least one training example $(\mathbf{x}, \mathbf{y}) \in S$ such that $y = -1$ and $\mathbf{y} \times [\ < \mathbf{x}, \mathbf{w} > + b_2\ ] = 0$. 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 $+1$.	(R1)
If $< \mathbf{x}, \mathbf{w} > + b_2 \geq 0$, classification of $\mathbf{x}$ is -1.	(R2)
Otherwise, classification of $\mathbf{x}$ is uncertain.	(R3)

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 (R1)-(R3) 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 $(\mathbf{x}, \mathbf{y}) \in S$, if $< \mathbf{x}, \mathbf{w} > + b_1 \geq 0$, then y must be $+1$. Similarly, b_2 can be chosen such that, for an $(\mathbf{x}, \mathbf{y}) \in S$, 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 example,

b_1 can be chosen in such a way that, for an $(\mathbf{x}, \mathbf{y}) \in S$, 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 $(\mathbf{x}, \mathbf{y}) \in S$, 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.

4 Rough sets based on the 1-v-1 approach

The terminologies for rules (R1)-(R3) from the previous section need be slightly modified for pairwise classifications between classes i and j. Let us assume that +1 corresponds to class i and -1 corresponds to class j. We can define equivalence classes corresponding to rules (R1)-(R3). Let $E_{ij}(i)$ be the set of $\mathbf{x}$ (or region) that follows rule (R1), $E_{ij}(j)$ be the set of $\mathbf{x}$ that follows rule (R2), and $E_{ij}(BND)$ be the set of $\mathbf{x}$ that follows rule (R3). The lower bound for class i will be $E_{ij}(i)$ and the upper bound will be $E_{ij}(i) \cup E_{ij}(BND)$. Similarly, the lower bound for class j will be $E_{ij}(j)$ and the upper bound will be $E_{ij}(j) \cup E_{ij}(BND)$. In some cases, this would mean the use of soft margin classifiers.

Figure 1 shows a classification problem with three classes. It is assumed that the objects have already been mapped using the same mapping Φ to transform the problem to a linear separable case. Figure 2 shows the 1-v-1 classification for classes (1,2). Similarly, Figures 3 and 4 show the 1-v-1 classifications for the pairs (1,3) and (2,3), respectively. The equivalence classes for each pair (i, j) will be used to calculate the overall lower bounds of N classes as shown in Eq. (4).

$$\underline{A}(class_i) = \bigcap_{j=1}^{N} E_{ij}(i), \tag{4}$$

where $j \neq i$.

We now show that the lower bounds of every class are mutually exclusive. Consider any pair of classes (i, j). By definition, $E_{ij}(i) \cap E_{ij}(j) = \emptyset$. By Eq. (4), we can conclude that $\underline{A}(class_i) \subseteq E_{ij}(i)$ and $\underline{A}(class_j) \subseteq E_{ij}(j)$. Therefore, $\underline{A}(class_i) \cap \underline{A}(class_j) = \emptyset$.

The boundary region for each class will be a union of boundary regions from each pairwise classification. However, such a boundary region will include lower bounds of other classes. Therefore, it is necessary to delete the union of lower bounds from such a boundary class. Hence, the boundary region, $\overline{A}(class_i) - \underline{A}(class_i)$, for each class i is given by:

$$\overline{A}(class_i) - \underline{A}(class_i) = \bigcup_{j=1}^{N} E_{ij}(BND) - \bigcup_{k=1}^{N} \underline{A}(class_k), \tag{5}$$

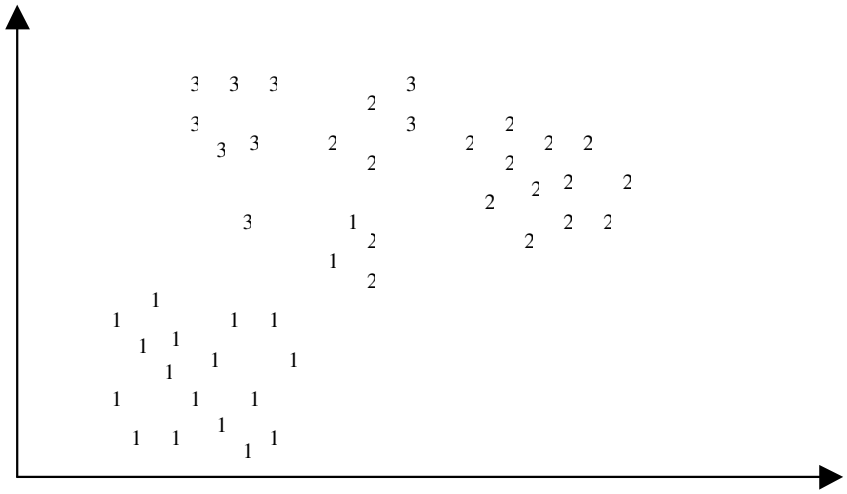

Fig. 1. A classification problem involving three classes 1, 2 and 3

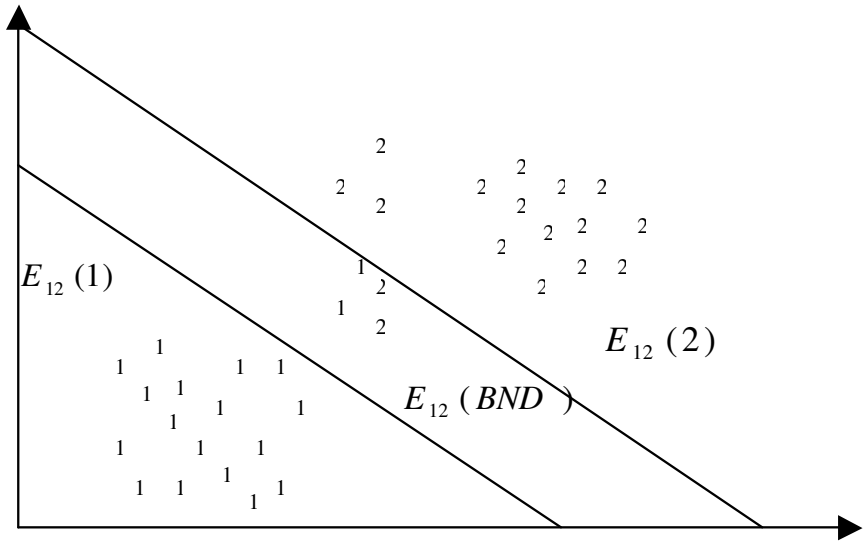

Fig. 2. A rough set approach to 1-v-1 classification for classes 1 and 2

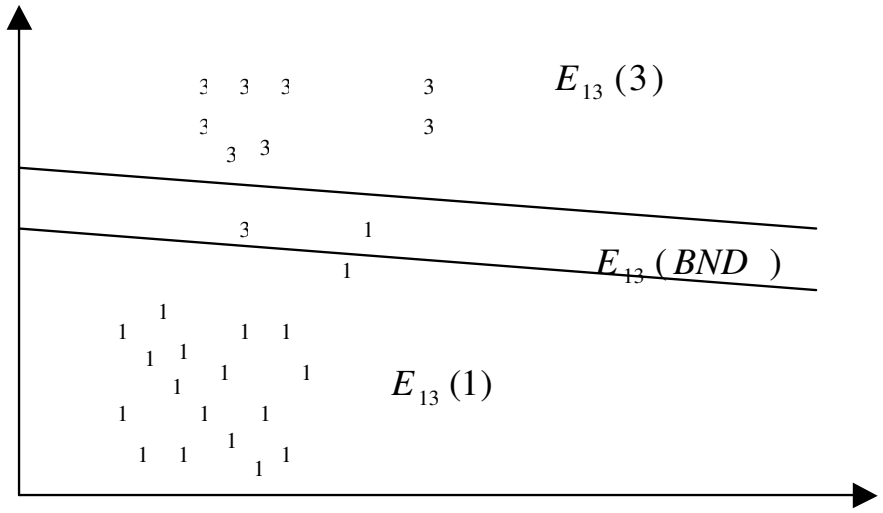

Fig. 3. A rough set approach to 1-v-1 classification for classes 1 and 3

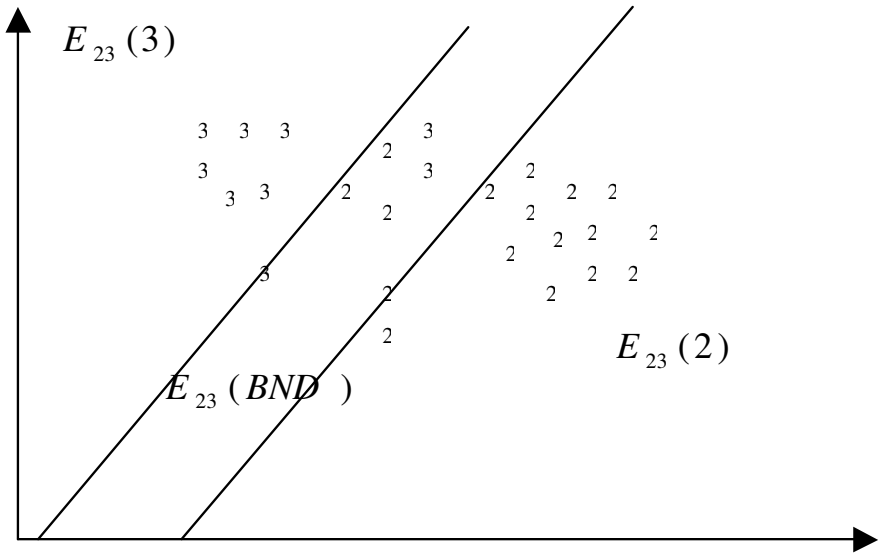

Fig. 4. A rough set approach to 1-v-1 classification for classes 2 and 3

where $j \neq i$. From Eq. (5), the upper bound, $\overline{A}(class_i)$, for a class i is given by:

$$\begin{aligned}
\overline{A}(class_i) &= \bigcup_{j=1}^{N} E_{ij}(BND) - \bigcup_{k=1}^{N} \underline{A}(class_k) + \underline{A}(class_i) \\
&= \bigcup_{j=1}^{N} E_{ij}(BND) - \bigcup_{j=1}^{N} \underline{A}(class_j) \\
&= \bigcup_{j=1}^{N} (E_{ij}(BND) - \underline{A}(class_j)), \qquad (6)
\end{aligned}$$

where $j \neq i$.

The approach proposed in this paper uses the same training time as the classical 1-v-1 approach or DAGSVM. However, only two rules need to be stored for each class, one corresponding to the lower bound $\underline{A}(class_i)$, given by Eq. (4), and another corresponding to the upper bound $\overline{A}(class_i)$, given by Eq. (6). Therefore, a total of $2N$ rules are stored for the testing and operational phases, as opposed to $N \times (N-1)/2$ SVMs stored by 1-v-1 and DAGSVM approaches. Moreover, during the operational phase, the determination of membership of an object in a class will involve simply testing which lower and/or upper bounds the object belongs to. The time requirement for classification in the operational phase will be $O(N)$, the same as DAGSVM. The rough set representation also provides the possibility of specifying an uncertainty in the classification. For example, it will be possible to say that while precise classification of the object is not known, it belongs to the upper bounds of a list of classes, i.e., a list of possible classifications. Finally, the rules corresponding to lower and upper bounds may be able to provide better semantic interpretations of the multi-classification process than the other SVM approaches, which have been regarded as black-box models.

5 Conclusion

The approaches for extending binary classifications obtained from support vector machines (SVMs) to multi-classification can be divided into two categories, 1-v-r (one versus rest) and 1-v-1 (one versus one). 1-v-r classification technique involves creating a separate SVM for every class using the members of the class as positive instances and non-members as negative instances. This approach requires a large training time. The 1-v-1 approach involves creating and storing $N \times (N-1)/2$ SVMs. The time requirement in the operational phase for 1-v-1 approach can be reduced using directed acyclic graphs leading to DAGSVMs. However, the storage requirement for DAGSVM is still the same as the classical 1-v-1 approach, i.e., $N \times (N-1)/2$. This paper describes an extension of 1-v-1 approach using rough or interval sets. The use of rough sets may make it possible to provide a semantic interpretation of the classification process using rule-based approach, which may be an advantage over the black-box SVM models. It is

shown that during the operation phase, the proposed approach has the same $O(N)$ time requirement as DAGSVM. However, it only requires storage of $2N$ rules as opposed to $N \times (N-1)/2$ SVMs stored by the DAGSVM and the classical 1-v-1 approach. Our approach also makes it possible to introduce uncertainty in describing classifications of objects.

References

1. Chang, F., Chou, C-H., Lin, C-C, and Chen, C-J: A Prototype Classification Method and Its Application to Handwritten Character Recognition. Proceedings of IEEE International Conference on Systems, Man and Cybernetics (2004) 4738-4743
2. Cristianini, N.: Support Vector and Kernel Methods for Pattern Recognition. http://www.support-vector.net/tutorial.html (2003)
3. Hoffmann, A.: VC Learning Theory and Support Vector Machines. http://www.cse.unsw.edu.au/∼cs9444/Notes02/Achim-Week11.pdf (2003)
4. Knerr, S., Personnaz, L., and Dreyfus, G.: 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. Lingras P. and Butz, C.J.: Interval Set Classifiers using Support Vector Machines. Proceedings of 2004 conference of the North American Fuzzy Information Processing Society, Banff, AB., June 27-30 (2004) 707-710
6. Minsky, M.L. and Papert, S.A.: Perceptrons, The MIT Press, Cambridge, MA. (1969)
7. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers (1992)
8. Platt, J.C.: Support Vector Machines. http://research.microsoft.com/users/jplatt/svm.html (2003)
9. Platt, J.C., Cristianini, N., and Shawe-Taylor, J.: Large margin DAG's for multiclass classification. In: Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, (2000) 547-553
10. Rosenblatt, F.: The perceptron: A perceiving and recognizing automaton. Technical Report 85-460-1, Project PARA, Cornell Aeronautical Lab (1957)
11. Vapnik, V.: Statistical Learning Theory. Wiley, NY. (1998)