Unanimous Voting using Support Vector Machines

E.N. Smirnov[a] I.G. Sprinkhuizen-Kuyper[a] G.I. Nalbantov[b]

[a] IKAT, Universiteit Maastricht
[b] ERIM, Erasmus University Rotterdam

Abstract

This paper proposes a new approach to classification reliability. The key idea is to maintain version spaces containing (close approximations of) the target classifiers. In this way the unanimous-voting rule applied on these version spaces outputs reliable instance classifications.

Version spaces are defined in a hypothesis space of oriented hyperplanes. The unanimous-voting rule is implemented by testing whether version spaces are empty. Testing is done by support vector machines. Hence, the approach is called version space support vector machines.

Experiments with the approach show a 100% accuracy on the classified instances at the cost of not classifying all instances.

1 Introduction

In the last ten years machine-learning classifiers have been applied to various classification problems [5]. Nevertheless, almost no classifiers have been employed in real applications, especially in critical domains. The main reason is that it is difficult to determine whether a classification assigned to a particular instance is reliable or not.

There are several approaches to classification reliability. They first estimate some parameter(s) that are related to classification reliability. Then, the approaches learn a threshold on that parameter(s) to decide whether an instance classification is reliable. The oldest approach used the posterior probability of the predicted class as a reliability parameter [3]. Unfortunately, this rather simple approach assumes an underlying distribution of the data, which is generally unknown [6]. Newer approaches are based on the theory of randomness (cf. [5, 8, 10]). The key idea is to classify a new instance so that when the instance is added to the training data, the data show a level of randomness that is close to the level of randomness of the same data before the instance was added. The reliability parameter is inversely proportional to the difference between the two levels of randomness.

In this paper we propose a new approach to classification reliability. In contrast to the previous approaches, ours is not based on classification-reliability parameter(s) and thus does not require learning any thresholds.

The approach is applicable for binary classifiers. The key idea assumes that we can maintain version spaces [7, 11] containing (close approximations of) the target classifiers. If the assumption is correct for the training data under consideration, the classification rule of unanimous voting applied on these version spaces guarantees that the classification assigned to each instance is reliable.

We consider version spaces in a hypothesis space of oriented hyperplanes defining decision functions. Version spaces are represented by the training data [4, 11]. Our implementation of the unanimous-voting rule tests whether version spaces are empty in the hypothesis space. Testing is realized via support vector machines (SVM) [12]. Therefore, our approach is a combination of the two learning schemes and it is called version space support vector machines (VSSVM).

We conducted experiments on datasets from the UCI ML repository [1]. Our results are promising: 100% accuracy and an acceptable coverage.

The remainder of the paper is organized as follows. Section 2 formalizes the classification task. Version spaces and SVM are briefly sketched in section 3, and in section 4 we introduce VSSVM. Our initial experiments with VSSVM are given in section 5. Section 6 concludes the paper.

2 Classification Task

Assume that we have l training instances $\mathbf{x}_i$ in an n-dimensional Euclidian space R^n. Each training instance has a class label $y_i \in Y$, where $Y = \{-1, +1\}$. The class labels separate the training instances into two sets I^+ and I^- ($\mathbf{x}_i \in I^+$ iff $y_i = +1$; $\mathbf{x}_i \in I^-$ iff $y_i = -1$). Given a hypothesis space H of all possible functions h ($h : R^n \to Y$), the classification task is to find a function (classifier) h that accurately classifies future, unseen data.

3 Approaches to the Classification Task

There exist a number of approaches to the classification task. In this paper we combine two of them: version spaces and SVM.

3.1 Version Spaces

Version spaces are an established approach to the classification task [7, 11]. They are sets of functions $h \in H$ consistent with training sets I^+ and I^-:

$$VS(I^+, I^-) = \{h \in H | cons(h, \langle I^+, I^- \rangle)\},$$

where $cons(h, \langle I^+, I^- \rangle)$ is the consistency predicate defined as:

$$cons(h, \langle I^+, I^- \rangle) \leftrightarrow (\forall \mathbf{x}_i \in I^+ \cup I^-)(y_i = h(\mathbf{x}_i)).$$

The version-space classification rule is the unanimous voting. Given a version space $VS(I^+, I^-)$, an instance $\mathbf{x} \in R^n$ gets a classification $y \in Y \cup \{0\}$ as follows:

$$y = \begin{cases} +1 & \text{if } (VS(I^+, I^-) \neq \emptyset) \wedge (\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = +1) \\ -1 & \text{if } (VS(I^+, I^-) \neq \emptyset) \wedge (\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = -1) \\ 0 & \text{otherwise.} \end{cases}$$

The unanimous-voting rule assigns class $+1$ (-1) to the instance $\mathbf{x}$ if $VS(I^+, I^-)$ is nonempty and all functions h in $VS(I^+, I^-)$ assign the same class $+1$ (-1) to the instance. In all other cases, the class is unknown which is denoted by 0.

In this paper we apply the unanimous-voting rule for the training-data representation of version spaces [4, 11]. The implementation of the rule in this case requires testing whether the version space is empty. The whole process can be explained by theorem 1 [4, 11]. Assume that we have a nonempty version space $VS(I^+, I^-)$ and an instance $\mathbf{x}$ to be classified. Then, theorem 1 states that all the functions in $VS(I^+, I^-)$ assign class $+1$ (-1) to $\mathbf{x}$ if and only if the subset $VS(I^+, I^- \cup \{\mathbf{x}\})$ $(VS(I^+ \cup \{\mathbf{x}\}, I^-))$ of $VS(I^+, I^-)$ of which the functions assign class -1 $(+1)$ is empty. In all other cases, the class of $\mathbf{x}$ is unknown.

Theorem 1 *If $VS(I^+, I^-)$ is nonempty, then:*

$$(\forall \mathbf{x} \in R^n)((\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = +1) \leftrightarrow VS(I^+, I^- \cup \{\mathbf{x}\}) = \emptyset),$$
$$(\forall \mathbf{x} \in R^n)((\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = -1) \leftrightarrow VS(I^+ \cup \{\mathbf{x}\}, I^-) = \emptyset).$$

3.2 Support Vector Machines

Support vector machines (SVM) are rooted in statistical learning theory [2, 12]. We consider SVM for the classification task formalized in section 2. The space H of all possible functions for SVM is the set generated by all possible oriented hyperplanes in an n-dimensional Euclidian space R^n or in a higher dimensional feature space F obtained by a mapping $\phi(\mathbf{x})$ on the instances $\mathbf{x}$ from R^n. SVM search for an oriented hyperplane that results in a maximal margin between the two classes, while minimizing the penalty term for the training instances at the wrong side of the margin.

Given l training instances $\mathbf{x}_i$, SVM solve the following primal problem:

$$\begin{aligned} &\underset{\mathbf{w}, b, \xi}{min} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \quad \text{such that} \\ &y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \text{ and} \\ &\xi_i \geq 0, i = 1, 2, \cdots, l \end{aligned} \tag{1}$$

The term $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ is proportional to the inverse of the margin, the term $\sum_{i=1}^{l}\xi_i$ is the sum of errors, and the parameter C determines the trade-off between the margin and the sum of errors. The dual of this minimization problem is:

$$\begin{aligned} &\underset{\alpha}{max} \quad \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{such that} \\ &0 \leq \alpha_i \leq C, i = 1, 2, \cdots, l, \text{ and} \\ &\sum_{i=1}^{l} y_i\alpha_i = 0 \end{aligned} \tag{2}$$

Here $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \cdot \phi(\mathbf{x}_j)$ is a kernel function that calculates dot products of instances that are mapped into a higher dimensional space F via the mapping ϕ. The alphas are the weights associated with the training instances. All instances with nonzero weights are "support vectors". They determine the position of the optimal SVM hyperplane, $h(C, \langle I^+, I^- \rangle)$, which is given by:

$$\sum_{i=1}^{l} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b = 0.$$

The classification of a new instance $\mathbf{x}$ is found by:

$$h(C, \langle I^+, I^- \rangle)(\mathbf{x}) = sign(\sum_{i=1}^{l} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b)$$

If $\xi_i > 0$ for a training instance $\mathbf{x}_i$, then $\mathbf{x}_i$ is a training error since $\mathbf{x}_i$ is not on the right side of the margin; however, as long as $\xi_i < 1$, $\mathbf{x}_i$ will be classified correctly. If the training data $\langle I^+, I^- \rangle$ are (linearly) separable in the feature space, the SVM will classify correctly all training instances for a value of the parameter C that is sufficiently large.

4 Version Space Support Vector Machines

Version space support vector machines (VSSVM) are version spaces defined in the hypothesis spaces H of SVM (see subsection 3.2). They are defined by the training-data representation of version spaces [4, 11]. The classification algorithm of VSSVM employs SVM. VSSVM are introduced in detail in the next subsections. In subsection 4.1 we define the space of training-data representations of nonempty version spaces in the hypothesis spaces H. In subsections 4.2 and 4.3 we provide the classification algorithm of VSSVM and an example.

4.1 Space of Training-Data Representations

SVM deal with the hypothesis space H consisting of oriented hyperplanes h. In this context, the predicate *cons* holds for a hyperplane h and data sets I^+ and I^- if and only if the sets I^+ and I^- are separated by h:

$$cons(h, \langle I^+, I^- \rangle) \leftrightarrow (\forall \mathbf{x}_i \in I^+ \cup I^-)(y_i = h(\mathbf{x})).$$

Given the hypothesis space H and a value of the parameter C, we denote by $SVS(H, C)$ the set of all pairs of data sets $\langle I^+, I^- \rangle$ in R^n for which there exists a SVM hyperplane $h(C, \langle I^+, I^- \rangle)$ that separates I^+ and I^-:

$$SVS(H, C) = \{\langle I^+, I^- \rangle | (\exists h \in H)((h = h(C, \langle I^+, I^- \rangle)) \wedge cons(h, \langle I^+, I^- \rangle))\}.$$

Since the ordered pair $\langle I^+, I^- \rangle$ is a training-data representation [4], the set $SVS(H, C)$ is the space of the training-data representations of all nonempty version spaces in the hypothesis space H for a particular value of the parameter C. Thus, *a version space $VS(I^+, I^-)$ is nonempty in H for a particular value of C if and only if $\langle I^+, I^- \rangle \in \mathrm{SVS}(H, C)$, i.e. the SVM hyperplane $h(C, \langle I^+, I^- \rangle)$ is consistent with the training data $\langle I^+, I^- \rangle$.*

Input: An instance $\mathbf{x}$ to be classified;
Training data sets I^+ and I^-;
The parameter C of SVM;
Output: classification of $\mathbf{x}$;

Build a hyperplane $h(C, \langle I^+, I^- \rangle)$;
if $\neg cons(h(C, \langle I^+, I^- \rangle), \langle I^+, I^- \rangle)$ **then return** 0;
Build a hyperplane $h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$;
Build a hyperplane $h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$;
if $\neg cons(h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle), \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ and
$cons(h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle), \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$ **then return** $+1$;
if $cons(h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle), \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ and
$\neg cons(h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle), \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$ **then return** -1;
return 0.

Figure 1: The Classification Algorithm of VSSVM.

4.2 Classification Algorithm

The classification algorithm of VSSVM implements the unanimous-voting rule on version spaces in the hypothesis space H. It is based on theorem 1. To test whether version spaces are empty SVM are employed.

The classification algorithm is given in figure 1. The algorithm input is: training data sets I^+ and I^-; an instance $\mathbf{x}$ to be classified; and the parameter C of SVM. The algorithm outputs the classification of $\mathbf{x}$: $+1$, -1, or 0.

The classification algorithm starts by building a hyperplane $h(C, \langle I^+, I^- \rangle)$. If $h(C, \langle I^+, I^- \rangle)$ is inconsistent with $\langle I^+, I^- \rangle$, then the version space $VS(I^+, I^-)$ is empty in the hypothesis space H for the value of the parameter C and according to the unanimous-voting rule the algorithm returns 0; i.e., the classification of $\mathbf{x}$ is unknown. If the hyperplane $h(C, \langle I^+, I^- \rangle)$ is consistent with $\langle I^+, I^- \rangle$, then the version space $VS(I^+, I^-)$ is nonempty in H for the value of the parameter C. In this case the algorithm builds hyperplanes $h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ and $h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$. If $h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ is inconsistent with $\langle I^+, I^- \cup \{\mathbf{x}\} \rangle$ and $h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$ is consistent with $\langle I^+ \cup \{\mathbf{x}\}, I^- \rangle$, then $VS(I^+, I^- \cup \{\mathbf{x}\})$ is empty and $VS(I^+ \cup \{\mathbf{x}\}, I^-)$ is nonempty in H for the value of the parameter C. This means that all the hyperplanes in $VS(I^+, I^-)$ assign class $+1$ to $\mathbf{x}$. Thus, by theorem 1 the algorithm assigns class $+1$ to $\mathbf{x}$. If the class $+1$ cannot be assigned, the algorithm checks analogously if it can assign class -1. If both classes cannot be assigned, the algorithms returns 0; i.e., the classification of $\mathbf{x}$ is unknown.

4.3 Example

We illustrate our classification algorithm on a classification task: the space H is the set of all oriented lines in R^2, and training data consist of the sets $I^+ = \{(1,0),(2,0),(1,1),(2,1)\}$ and $I^- = \{(-1,0),(-2,0),(-1,1),(-2,1)\}$ (see figure 2). For large C ($C = +\infty$) only the points to the right of the three line segments

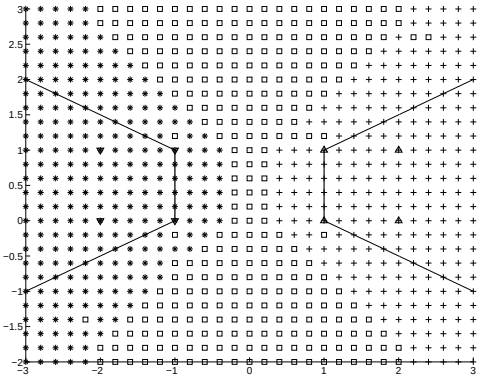

Figure 2: Illustration of the version spaces for $C = +\infty$ (bounded by the lines) and $C = 30$ (I^+ marked by $\triangle$, I^- marked by ∇, positively classified: $+$, negatively classified: $*$, and not classified: $\Box$).

through the training points $(1, 0)$ and $(1, 1)$ will be classified as positive and the corresponding region to the left of the three line segments through the training points $(-1, 0)$ and $(-1, 1)$ will be classified as negative. The version space in this case consists of all the lines not intersecting the line segments shown. Running our algorithm with $C = 30$ results in the classifications in figure 2: positively classified: $+$, negatively classified: $*$, and not classified: $\Box$. It is clear from the figure that for $C = 30$ the version space is smaller and thus the coverage is larger, than for $C = +\infty$. In the figure some unclassified points (e.g., the point $(-1, -0.1)$) are expected to be classified. Such points can correspond to solutions of the SVM where all α_i's are equal either to 0 or to C. In that case b is not uniquely determined and some solutions will result in separation while other solutions will not.

5 Experiments

We implemented VSSVM in WEKA [13] using the SMO implementation of SVM [9]. We experimented with VSSVM using polynomial function kernels (P) and radial-basis function kernels (RBF). The method for evaluation was the leave-one-out method. We searched for values of the parameters (E (exponent) and C for P and G (gamma) and C for RBF) resulting in the highest coverage for the leave-one-out method. Below we provide the best results we obtained for 7 datasets (from the UCI ML repository [1]) with VSSVM together with the results of SVM.

Table 1 shows the most important result: the accuracy of VSSVM is 100% on the data they are able to classify. Below we explain this result.

When the datasets are noise-free and the hypothesis space H contains the target hyperplane, the hyperplane is consistent with the training data; i.e., it belongs to the version space [7, 11]. Thus, the unanimous-voting classification

Data Set	Parameters	C_{vssvm}	A_{vssvm}	A_{svm}	I
Heart-Statlog	P, E=2.0, C=1730	56.3%	100%	73.0%	0.42
Heart-Statlog	RBF, G=0.2 , C=2182	40.7%	100%	73.7 %	0.24
Hepatitis	P, E=1.4, C=11.7	80.0%	100%	80.0 %	0.72
Hepatitis	RBF, G=0.02 , C=2140	69.7%	100%	81.9 %	0.39
Horse Colic	P, E=1.3, C=154.5	50.8%	100%	78.3%	0.26
Horse Colic	RBF, G=0.015 , C=12030	54.9%	100%	79.1%	0.29
Ionosphere	P, E=1.1, C=5200	77.8%	100%	89.2%	0.27
Ionosphere	RBF, G=0.05, C=4030	77.2%	100%	90.3%	0.24
Labor	P, E=1.25, C=1.17	84.2%	100%	87.7%	0.42
Labor	RBF, G=0.02, C=61	84.2%	100%	93.0%	0.21
Sonar	P, E=1.0, C=3340	70.7%	100%	74.0%	0.68
Sonar	RBF, G=0.65, C=0.664	62.5%	100%	85.6%	0.23
W. Breast Cancer	P, E=3, C=58.6	85.0%	100%	93.6%	0.19
W. Breast Cancer	RBF, G=0.8 , C=70.9	82.5%	100%	94.3 %	0.16

Table 1: Coverage and Accuracy of VSSVM and SVM. C_{vssvm} is the coverage of VSSVM. A_{vssvm} is the accuracy of VSSVM on instances it can classify. A_{svm} is the accuracy of SVM. The coverage of SVM is not given since it is always 100%. P is a polynomial kernel with the exponent E. RBF is a radial-basis function kernel with the parameter G. I is the information gain of VSSVM w.r.t. SVM.

rule produces reliable instance classifications.

When the datasets are noise-free and the hypothesis space H does not contain the target hyperplane, the version space consists of hyperplanes that approximate without mistakes the target function on the training data. Thus, the unanimous-voting classification rule produces reliable instance classifications depending how well the hyperplanes approximate the target function individually and together.

When the datasets are noisy, the situation is different. Assume that the training set I^+ (I^-) is a union of a noise-free set I_f^+ (I_f^-) and a noisy set I_n^+ (I_n^-). The noisy sets I_n^+ and I_n^- cause removal of the version space $NVS = \{h \in VS(I_f^+, I_f^-) | \neg cons(h, \langle I_n^+, I_n^- \rangle)\}$ from $VS(I_f^+, I_f^-)$. The resulting version space $VS(I^+, I^-)$[1] will continue to classify reliably instances that are classified by $VS(I_f^+, I_f^-)$, but it will err on instances in the volume of NVS. It seems that for our datasets this volume is small and does not influence the final results.

From the explanations above we may conclude that VSSVM are successfully applicable if it is possible to maintain version spaces containing the target hyperplane or its close approximations and the level of noise is low.

To measure the performance of VSSVM we computed the information gain of VSSVM w.r.t. to SVM in Table 1 [7]. The gain was computed using the proportions of correctly and incorrectly classified instances of SVM and VSSVM. All the cases in Table 1 resulted in a considerable information gain. We especially mention the Hepatitis dataset (polynomial kernel) for which the gain is 0.72 and the Sonar dataset (polynomial kernel) for which the gain is 0.68.

[1]Note $VS(I^+, I^-) = VS(I_f^+, I_f^-) \cap VS(I_n^+, I_n^-)$ and $VS(I^+, I^-) = VS(I_f^+, I_f^-) \backslash NVS$.

6 Conclusion

This paper proposes a new approach to classification reliability called version space support vector machines (VSSVM). Their main advantage compared with previous approaches to classification reliability is that they are not based on classification-reliability parameter(s) and thus do not require learning any thresholds. The experiments show that VSSVM achieve accuracy of 100% on the instances they are able to classify when it is possible to maintain version spaces containing the target hyperplane or its close approximations and the level of noise is low.

We foresee two future directions of research. The first one is to extend VSSVM for classification tasks with more than two classes. The second direction is to apply VSSVM when it is not possible to find consistent hyperplanes w.r.t. the training data. In this context we consider to apply the generalized version spaces [7].

References

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

[2] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Willey, second edition, 2000.

[4] H. Hirsh, N. Mishra, and L. Pitt. Version spaces without boundary sets. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 491–496, Menlo Park, CA, 1997. AAAI Press.

[5] M. Kukar and I. Kononenko. Reliable classifications with machine learning. In *Proceedings of the 13th European Conference on Machine Learning (ECML-2002)*, pages 219–231. Springer, 2002.

[6] T. Melluish, C. Saunders, I. Nouretdinov, and V. Vovk. Comparing the bayes and typicalness frameworks. In *Proceedings of the 12th European Conference on Machine Learning (ECML-2001)*, pages 360–371. Springer, 2001.

[7] T.M. Mitchell. *Machine learning*. McGraw-Hill, New York, NY, 1997.

[8] H. Papadopoulos, K. Proedru, V. Vovk, and A. Gammerman. Comparing the bayes and typicalness frameworks. In *Proceedings of the 13th European Conference on Machine Learning (ECML-2002)*, pages 345–356. Springer, 2002.

[9] J. Platt. A fast algorithm for training support vector machines. Technical report, Microsoft Research, 1998.

[10] K. Proedru, I. Nouretdinov, V. Vovk, and A. Gammerman. Transductive confidence machines for pattern recognition. In *Proceedings of the 13th European Conference on Machine Learning (ECML-2002)*, pages 381–390. Springer, 2002.

[11] E.N. Smirnov. *Conjunctive and disjunctive version spaces with instance-based boundary sets*. PhD thesis, Department of Computer Science, Maastricht University, Maastricht, The Netherlands, 2001.

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

[13] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.