Journal of Machine Learning Research 3 (2002) 723-746 Submitted 12/01; Published 12/02

The Set Covering Machine

Mario Marchand MARCHAND@SITE.UOTTAWA.CA
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ont. K1N-6N5, Canada

John Shawe-Taylor JST@CS.RHUL.AC.UK
Department of Computer Science
Royal Holloway, University of London
Egham, TW20-0EX, UK

Editors: Carla E. Brodley and Andrea Danyluk

Abstract

We extend the classical algorithms of Valiant and Haussler for learning compact conjunctions and disjunctions of Boolean attributes to allow features that are constructed from the data and to allow a trade-off between accuracy and complexity. The result is a general-purpose learning machine, suitable for practical learning tasks, that we call the *set covering machine*. We present a version of the set covering machine that uses *data-dependent balls* for its set of features and compare its performance with the support vector machine. By extending a technique pioneered by Littlestone and Warmuth, we bound its generalization error as a function of the amount of data compression it achieves during training. In experiments with real-world learning tasks, the bound is shown to be extremely tight and to provide an effective guide for model selection.

Keywords: Set Covering Machines, Computational Learning Theory, Support Vector Machines, Data-dependent Features, Feature Selection, Sample Compression, Model Selection.

1. Introduction

We may attribute the effectiveness of Support Vector Machines (Vapnik, 1998, Cristianini and Shawe-Taylor, 2000) to the fact that they combine two very good ideas. First, they map the space of input vectors onto a very high-dimensional feature space in such a way that nonlinear decision functions on the input space can be constructed by using only hyperplanes on the feature space. Second, they construct the separating hyperplane on the feature space which has the largest possible margin. Theoretical results on margin classifiers (Shawe-Taylor et al., 1998) imply that good generalization is expected whenever a large margin separating hyperplane can be found. However, for a given set of features, there might exist solutions that provide better generalization than the maximum margin solution. In particular, if the function to learn happens to depend only on a very small subset of a large set of given features, then it might be better to construct a simple decision function that depends only on a small subset of features than to find the maximum margin hyperplane on the set of all features. We propose here to investigate this possibility.

The pioneering work of Valiant (1984) and Haussler (1988) cannot be avoided when the task is to construct a simple function of few relevant features. For a given set of (data-independent) features, Valiant (1984) has proposed a very simple learning algorithm for building a conjunction from positive examples only (or building a disjunction from negative examples only). However, the obtained function might contain a lot of features and, consequently, its generalization could be much improved if their number could be substantially reduced. Following this, Haussler (1988) has shown that the training problem for conjunctions (or disjunctions) of (data-independent) features is, in fact, reducible to the minimum set cover problem(Garey and Johnson, 1979, Kearns and Vazirani, 1994). Although this problem is *NP*-complete, there exists an approximation algorithm, known as the greedy set cover algorithm, with a very good worst case guarantee (Chvátal, 1979). Therefore, Haussler (1988) has proposed to use this algorithm, on the remaining examples that are not used by the algorithm of Valiant, to reduce the number of features. Consequently, this two-step learning algorithm efficiently PAC learns the class of conjunctions (and disjunctions) of few relevant (data-independent) features.

Despite this impressive theoretical result and the fact that these papers of Valiant (1984) and Haussler (1988) are considered as basic building blocks of computational learning theory, nobody seems to have tried these algorithms in practice on "real-world" data. In fact, these algorithms have two characteristics that may make them unattractive for the practitioner. First, they are defined on Boolean attributes only and, second, they do not provide an accuracy-complexity tradeoff. The constraint of having to use Boolean attributes may be trivially relaxed by just providing a set of Boolean-valued features *i.e.*, Boolean-valued functions defined on the input domain. The second characteristic is, however, clearly a deficiency from a practical point of view. Indeed, real-world data are often noisy and, consequently, simpler functions that make some errors on the training data might be preferable to a more complex function that makes no training errors. Because the amount of noise is problem specific, a learning algorithm should provide to its user a means to control the tradeoff between the accuracy and the complexity of a classifier. For example, Support Vector Machines (SVM) provide such a control to the user via a "soft margin" parameter C. With an infinite value of C, the learning algorithm will attempt to achieve a training error of zero with the largest separating margin. But finite values of C specify quantitatively the tradeoff between the number of training errors and the margin achieved on the correctly classified examples. It is well known that finite values of C often give SVM classifiers with better generalization than those obtained with an infinite value of C.

Therefore, in Section 3, we have generalized the two-step algorithm of Valiant (1984) and Haussler (1988) in such a way that it is now possible to control quantitatively the tradeoff between the complexity and the accuracy of a classifier. But such a control is best achieved when the provided set of Boolean-valued features is built from the training data. For example, it is only for such data-dependent feature sets that it is always possible to construct a function that makes zero error on any training set (provided that there are no identical input examples with different class labels). Hence, our generalized algorithm is able to use any set of (Boolean-valued) features, including those that are constructed from the training set. In Section 4 we introduce an example of such a feature set, that we call *data-dependent balls*, and use it (in Section 6) on several real-world data sets provided by the UCI repository (Blake and Merz, 1998). We find that the performance of our algorithm compares favorably to the SVM in the sense that its generalization is often very close or better than that of the SVM but, not surprisingly, the classifier is always much *sparser* than the one returned by the SVM. This is because our algorithm tries to build the smallest conjunction (or disjunction) of features whereas the SVM constructs the largest margin hyperplane over all the features. Hence, the two algorithms are fundamentally different. We propose to call our learning algorithm the Set Covering Machine (SCM) because the training problem for conjunctions (or disjunctions) is basically the minimum set cover problem.

Unfortunately, the VC dimension (Vapnik, 1998) of the set of functions supported by the SCM is not defined whenever the SCM uses data-dependent features. Indeed, the VC dimension is a characteristic of a class of functions alone, without any reference to the training data. Consequently, we cannot use the “standard tools” of computational learning theory to bound the generalization error of SCMs with data-dependent features. However, since the SCM tries to build a small conjunction (or disjunction) of few relevant features, it is reasonable to expect good generalization whenever an SCM with few relevant features and few training errors has been found. Consequently we seek a bound on the generalization error of SCMs in terms of the sparsity achieved on the training data. A general framework for obtaining bounds in terms of what has been achieved on training data is presented by Shawe-Taylor et al. (1998), where the so-called luckiness framework is introduced. For the case of SCMs, luckiness is measured in terms of the sparsity achieved (Herbrich and Williamson, 2002) and it turns out that we could extend the compression scheme introduced by Littlestone and Warmuth (1986), and later studied by Floyd and Warmuth (1995) and Ben-David and Litman (1998), to bound the generalization error of SCMs in terms of its training error and the number of data-dependent balls present in the classifier. This theory is presented in Section 5.

In view of the recent success of SVMs, let us mention a fundamental objection that one might have about the effectiveness of a learning algorithm whose task is to construct a small conjunction (or disjunction) of few relevant features. Since conjunctions and disjunctions are (just) subsets of the set of linearly separable functions, we might believe that the maximum margin solution over a given set of features will generally exhibit better generalization than a small conjunction (or disjunction), over the same set of features, obtained from the Valiant-Haussler greedy set cover algorithm. If this is true even in the case where there exists a small conjunction (or disjunction) that makes little training error, then our proposed task would be hopeless since an SVM classifier would always be better. In the next section we provide empirical evidence that this is not the case by comparing the performance of the maximum-margin with a small conjunction returned by the Valiant-Haussler algorithm on data sets for which there exists a small conjunction that makes zero error with them. In fact, for these data sets, the generalization of the small conjunction is substantially better than the one obtained from the maximum margin.

Finally, it is worth mentioning the important recent theoretical result of Ben-David et al. (2001) who showed that an overwhelming majority of the family of concept classes of finite VC dimension cannot be embedded in half spaces of arbitrarily high dimension with a large margin. This theoretical work, along with our empirical results of Section 2, shows that other learning approaches, not based on margins, are needed in many circumstances. In this paper, we present one approach, based on sample compression, which uses a learning strategy fundamentally different than the maximum margin.

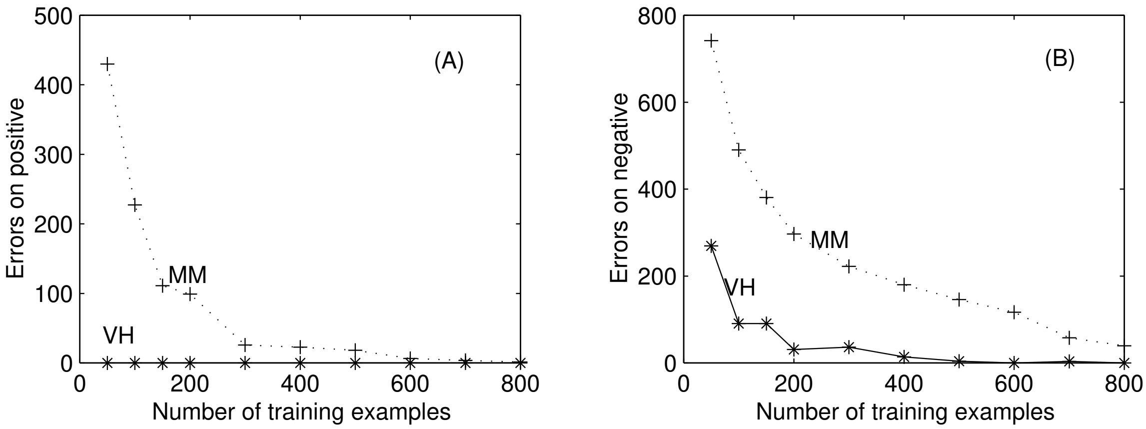

Figure 1: Comparison of the maximum margin solution (MM) with the Valiant-Haussler set covering greedy solution (VH)

2. Small Conjunctions Versus the Maximum Margin

Consider the following question. Given a set of Boolean-valued features, and given a training set for which there exists a conjunction of a small number of these features that makes zero training error, does the hyperplane (on this set of features), with the largest separating margin, *always* give better generalization that the small conjunction obtained from the greedy set cover algorithm of Valiant (1984) and Haussler (1988)?

Because both algorithms construct a function on the same set of features, let us assume w.l.o.g. that the features are the original input (Boolean) attributes. Furthermore, note that we do not engage ourselves into the ambitious task of trying to identify precisely the conditions under which one algorithm is better than the other one. Rather, we just want to show that there exist unexceptional situations for which the Valiant-Haussler algorithm is better. Hence, let us consider the case where each training and testing example is a vector of 80 Boolean attributes whose class is determined according to the conjunction $x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5$. Hence the function to learn is a conjunction of 5 attributes out of 80. Each positive example was generated by fixing x_i to 1 for $i = 1 \dots 5$ and by choosing a value for x_i in $\{0, 1\}$ uniformly and independently for $i = 6 \dots 80$. Each negative example was generated by first choosing a value for x_i in $\{0, 1\}$ uniformly and independently for $i = 1 \dots 80$ and accepting it only if at least one x_i is set to 0 for some $i \in \{1 \dots 5\}$.

The maximum-margin solution was obtained from the support vector machine program distributed by the department of Computer Science of the Royal Holloway, University of London (Saunders et al., 1998). The small conjunction returned by the Valiant-Haussler algorithm was obtained by our implementation of algorithm **BuildSCM** described in Section 3 in the case where the set of features consists of the input attributes and their negations and in the case where infinitely large values are chosen for parameters p and s. For this input of **BuildSCM**, we train until we make zero training errors.

The accuracy of both the maximum margin (MM) solution and the Valiant-Haussler (VH) solution was measured with respect to a testing set of 10000 examples—half of which are positive examples. The results are plotted on Figure 1 for various training set sizes—each containing an equal amount of positive and negative examples. For both solutions (or algorithms), the errors made on the 5000 positive testing examples are reported separately (on Figure 1a) from the the errors made on the 5000 negative testing examples (on Figure 1b). Each point denotes the average over 20 different training sets of the same size.

There is no question that the VH solution is substantially better here than the MM solution on both the positive and the negative testing examples. Hence, this shows that a small conjunction of few relevant features can indeed give better generalization than the maximum margin hyperplane on all the features.

It is therefore worthwhile to explore the learning capabilities of a general-purpose learning machine, that we call the set covering machine (SCM), which uses the Valiant-Haussler greedy algorithm as its main optimization tool to construct a conjunction (or disjunction) of a small number of relevant features.

3. The Set Covering Machine

As explained in the introduction, we will present a learning algorithm that strictly generalizes the two-step algorithm of Valiant (1984) and Haussler (1988) for learning conjunctions (or disjunctions) of Boolean attributes to the case of learning these functions over arbitrary sets of Boolean-valued features—including those that are constructed from the data. Furthermore, the learning algorithm will provide some model-selection parameters to control the tradeoff between training accuracy and the size of the conjunction (or disjunction). The final learning algorithm is presented at the end of this section.

Let $\mathbf{x}$ denote an arbitrary n-dimensional vector of the input space X. The input space X could be an arbitrary subset of $\Re^n$. We define a *feature* as an arbitrary Boolean-valued function that maps X onto $\{0,1\}$.

Consider any set $\mathcal{H} = \{h_i(\mathbf{x})\}_{i=1}^{|\mathcal{H}|}$ of Boolean-valued features $h_i(\mathbf{x})$. We will consider learning algorithms that are given any such set $\mathcal{H}$ and return a small subset $\mathcal{R} \subset \mathcal{H}$ of features. Given that subset $\mathcal{R}$, and an arbitrary input vector $\mathbf{x}$, the output $f(\mathbf{x})$ of the set covering machine (SCM) is defined to be:

$$f(\mathbf{x}) = \begin{cases} \bigvee_{i\in\mathcal{R}} h_i(\mathbf{x}) & \text{for a disjunction} \\ \bigwedge_{i\in\mathcal{R}} h_i(\mathbf{x}) & \text{for a conjunction} \end{cases}$$

Throughout the paper we will use the following definition.

Definition 1 *A function (or a feature) is said to be* consistent *with an example if it correctly classifies that example. Similarly, a function (or a feature) is said to be consistent with a set of examples if it correctly classifies all the examples in that set.*

Let us first describe the algorithm of Valiant (1984) and then Haussler (1988) in the case of conjunctions. The disjunction case immediately follows by symmetry.

The algorithm of Valiant *Given a set of m training examples and a set $\mathcal{H}$ of features, find the subset $\mathcal{C} \subseteq \mathcal{H}$ of all the features which are consistent with all the positive training examples.*

This set $\mathcal{C}$ has the property that $\bigwedge_{i\in\mathcal{C}} h_i(\mathbf{x})$ is consistent with *all* the m training examples whenever *there exists* a subset $\mathcal{E}$ of $\mathcal{C}$ such that $\bigwedge_{i\in\mathcal{E}} h_i(\mathbf{x})$ is consistent with *all* the m training examples. Indeed, since each $h_i(\mathbf{x}) \in \mathcal{C}$ is consistent with all the positive examples, $\bigwedge_{i\in\mathcal{C}} h_i(\mathbf{x})$ is consistent with all the positive training examples. Moreover, given any negative training example $\mathbf{x}$, $\bigwedge_{i\in\mathcal{E}} h_i(\mathbf{x})$ outputs 0. Hence, $\bigwedge_{i\in\mathcal{C}} h_i(\mathbf{x})$ will also output 0 since $\mathcal{E} \subseteq \mathcal{C}$. Thus $\bigwedge_{i\in\mathcal{C}} h_i(\mathbf{x})$ is consistent with all the negative training examples as well.

However, $|\mathcal{C}|$ might be very large and, as we will see below, the generalization error is expected to be small only for a classifier that contains few relevant features. Hence to reduce their number, first note that each feature in $\mathcal{C}$ is consistent with all the positive training examples and *some* negative training examples. Suppose that there exists a subset $\mathcal{E}$ of $\mathcal{C}$ such that $\bigwedge_{i\in\mathcal{E}} h_i(\mathbf{x})$ is consistent with *all* the m training examples and let Q_i denote the set of negative training examples that are consistent with feature h_i. Since $\bigwedge_{i\in\mathcal{C}} h_i(\mathbf{x})$ is consistent with the set $\mathcal{N}$ of all the negative training examples, the union $\bigcup_{i\in\mathcal{C}} Q_i$ must be equal to $\mathcal{N}$. We say that $\bigcup_{i\in\mathcal{C}} Q_i$ is a *cover* of $\mathcal{N}$. Therefore, the problem of finding the *smallest* subset of features in $\mathcal{C}$ whose conjunction is consistent with $\mathcal{N}$ is identical to the problem of finding the *smallest* collection $\mathcal{V}$ of sets Q_i for which $\bigcup_{i\in\mathcal{V}} Q_i$ is a cover of $\mathcal{N}$. This is the well known (and *NP*-complete) *Minimum Set Cover Problem*(Garey and Johnson, 1979). Hence, it is hard to find the set cover of the minimum size but, fortunately, the well known *set cover greedy algorithm* has a good worst-case bound. If z denotes the smallest number of sets Q_i that cover $\mathcal{N}$, then the set cover greedy algorithm will always find a cover of at most $z\ln(|\mathcal{N}|)$ sets (Chvátal, 1979, Kearns and Vazirani, 1994). Note that this bound has no dependence on the number of subsets (or features) in $\mathcal{C}$ and has only a logarithmic dependence on $|\mathcal{N}|$. We will therefore use this algorithm to find a small number of features in $\mathcal{C}$ that cover $\mathcal{N}$ (hence the name: set covering machine).

The set covering greedy algorithm is a very simple algorithm: first choose the set Q_i which covers the largest number of elements in $\mathcal{N}$, remove from $\mathcal{N}$ and each Q_j the elements that are in Q_i, then repeat this process of finding the set Q_k of largest cardinality and updating $\mathcal{N}$ and each Q_j until there are no more elements in $\mathcal{N}$. Therefore the following algorithm was proposed by Haussler (1988).

The algorithm of Haussler *Given a set of training examples and a set $\mathcal{H}$ of features, let $\mathcal{C}$ be the subset of $\mathcal{H}$ returned by the algorithm of Valiant. Use the set covering greedy algorithm to find a small subset $\mathcal{R} \subseteq \mathcal{C}$ of features that covers all the negative training examples.*

The hypothesis returned by this algorithm, $\bigwedge_{i\in\mathcal{R}} h_i(\mathbf{x})$, will be consistent with all the training examples and will contain a small set of features whenever *there exists* a small set of features in the initial set $\mathcal{H}$ of chosen features whose conjunction is consistent with all the training examples. However this guarantee can *generally* be achieved only for features that are constructed from the training data (like the set of data-dependent balls presented in Section 4) and only when there does not exist a pair of (contradictory) training examples having the same input vector and opposite class labels.

We can construct a disjunction (instead of a conjunction) with minimal changes in these algorithms. For the algorithm of Valiant, we just find the set $\mathcal{C}$ of all the features in $\mathcal{H}$ that are consistent with the training set of *negative* examples. For the algorithm of Haussler, we use the greedy set covering algorithm to find a small subset $\mathcal{R} \subseteq \mathcal{C}$ of features that covers all the *positive* training examples. By using the same arguments as above, we find that $\bigvee_{i \in \mathcal{R}} h_i(\mathbf{x})$ will be consistent with all the training examples whenever *there exists* a small set of features in the initial set $\mathcal{H}$ of chosen features whose disjunction is consistent with all the training examples.

However, as we have argued in the introduction, this two-step learning algorithm is unsatisfactory because it does not permit its user to control the tradeoff between the accuracy achieved on the training data and the complexity (here the size) of the classifier. Indeed, a small conjunction which makes a few errors on the training set might give better generalization than a larger conjunction (with more features) which achieves a training error of zero.

One way to include this flexibility into the SCM is to stop the set covering greedy algorithm when there remains a few more training examples to be covered. In this case, the SCM will contain fewer features and will make errors on those training examples that are not covered. But these examples all belong to the same class: negative for conjunctions, positive for disjunctions. This might be desired for asymmetric loss functions but, in general, we do need to be able to make errors on training examples of both classes. Hence, early stopping is generally not sufficient and, in addition, we will need to consider features that would not be selected by the algorithm of Valiant.

To describe our algorithm in a way which is technically valid for both conjunctions and disjunctions, we introduce the following definition.

Definition 2 *If the SCM is constructing a conjunction, let $\mathcal{P}$ be the set of positive training examples and let $\mathcal{N}$ be the set of negative training examples. If the SCM is constructing a disjunction, let $\mathcal{P}$ be the set of negative training examples and let $\mathcal{N}$ be the set of positive training examples.*

Similarly, a positive example in the conjunction case (or a negative example in the disjunction case) will be called a $\mathcal{P}$-example. Whereas, a negative example in the conjunction case (or a positive example in the disjunction case) will be called a $\mathcal{N}$-example.

Hence, for both cases, $\mathcal{P}$ refers to the subset of training examples for which each feature returned by the Valiant algorithm must be consistent with. Similarly, $\mathcal{N}$ refers to the set of examples that needs to be covered by the greedy set covering algorithm.

From the above discussion, we want to allow a feature to make a few errors on $\mathcal{P}$ provided that many more examples in $\mathcal{N}$ can be covered by this feature. Hence, for a feature h, let us denote by Q_h the set of examples in $\mathcal{N}$ covered (*i.e.*, correctly classified) by feature h and by R_h the set of examples in $\mathcal{P}$ for which h makes an error. Given that each example in R_h should decrease by some fixed *penalty* p the "usefulness" of feature h, we define the *usefulness* U_h of feature h to be the number $|Q_h|$ of examples in $\mathcal{N}$ that h covers minus the number $|R_h|$ of examples in $\mathcal{P}$ misclassified by h times some penalty p:

$$U_h \stackrel{\text{def}}{=} |Q_h| - p \cdot |R_h| \tag{1}$$

Hence, the learning algorithm for the SCM is modified as follows. In contrast with the algorithm of Valiant, we consider all the features in our initial set $\mathcal{H}$, including the features which make some errors on $\mathcal{P}$. Each feature $h \in \mathcal{H}$ is covering a subset Q_h of $\mathcal{N}$ and makes an error on a subset R_h of $\mathcal{P}$. Next, we modify the set covering greedy algorithm in the following way. Instead of using the feature that covers the largest number of examples in $\mathcal{N}$, we use the feature h that has the highest usefulness value U_h. We removed from $\mathcal{N}$ and each Q_g the elements that are in Q_h and we removed from each R_g the elements that are in R_h. Note that we update each such set R_g because a feature g that makes an error on an example in $\mathcal{P}$ does not increase the error of the machine if another feature h is already making an error on that example. We repeat this process of finding the feature h of largest usefulness U_h and updating $\mathcal{N}$, and each Q_g, and each R_g until only a few elements remain in $\mathcal{N}$ (in the case of early stopping).

Here is a formal description of our learning algorithm. The penalty p and the early stopping point s are the two model-selection parameters that give the user the ability to control the proper tradeoff between the training accuracy and the size of the function. Their values could be determined either by using a resampling method, such as k-fold cross-validation, or by computing our bound (see Section 5) on the generalization error based on what has been achieved on the training data. Note that our learning algorithm reduces to the two-step algorithm of Valiant (1984) and Haussler (1988) when both s and p are infinite and when the set of features consists of the set of input attributes and their negations.

Algorithm BuildSCM$(S, T, p, s, \mathcal{H})$

Input: A training set S of examples, a machine type T (which is either "conjunction" or "disjunction"), a penalty value p, a stopping point s, and a set $\mathcal{H} = \{h_i(\mathbf{x})\}_{i=1}^{|\mathcal{H}|}$ of Boolean-valued features.

Output: A conjunction (or disjunction) $f(\mathbf{x})$ of a subset $\mathcal{R} \subseteq \mathcal{H}$ of features.

Initialization: $\mathcal{R} = \emptyset$.

1. **If** (T = "conjunction") then let $\mathcal{P}$ be the set of positive training examples and let $\mathcal{N}$ be the set of negative training examples.
   Else then let $\mathcal{P}$ be the set of negative training examples and let $\mathcal{N}$ be the set of positive training examples.

2. For each feature $h_i \in \mathcal{H}$, let Q_i be the subset of $\mathcal{N}$ covered by h_i and let R_i be the subset of $\mathcal{P}$ for which h_i makes an error.

3. Let h_k be a feature with the largest value of $|Q_k| - p \cdot |R_k|$.

4. Let $\mathcal{R} \leftarrow \mathcal{R} \cup \{h_k\}$. Let $\mathcal{N} \leftarrow \mathcal{N} - Q_k$ and let $\mathcal{P} \leftarrow \mathcal{P} - R_k$.

5. For all i do: $Q_i \leftarrow Q_i - Q_k$ and $R_i \leftarrow R_i - R_k$.

6. **If** ($\mathcal{N} = \emptyset$ or $|\mathcal{R}| \geq s$) then go to step 7 (no more examples to cover or early stopping).
   Else go to step 3.

7. Return $f(\mathbf{x})$ where:

$$f(\mathbf{x}) = \begin{cases} \bigvee_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a disjunction} \\ \bigwedge_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a conjunction} \end{cases}$$

4. Data-dependent Balls

The above description of the SCM applies for any chosen set $\mathcal{H}$ of Boolean-valued features. However, the user has the greatest control for choosing the proper tradeoff between training accuracy and function size when the set of features is constructed from the training data. Let us examine a simple data-dependent set of features which has the property that there always exists a subset for which a conjunction will make zero training errors (provided that we do not have any pair of training examples with the same input vector and opposite class labels).

For each training example $\mathbf{x}_i$ with label $y_i \in \{0,1\}$ and (real-valued) radius ρ, we define feature $h_{i,\rho}$ to be the following *data-dependent ball* centered on $\mathbf{x}_i$:

$$h_{i,\rho}(\mathbf{x}) \stackrel{\text{def}}{=} h_\rho(\mathbf{x},\mathbf{x}_i) = \begin{cases} y_i & \text{if } d(\mathbf{x},\mathbf{x}_i) \leq \rho \\ \overline{y}_i & \text{otherwise} \end{cases}$$

where $\overline{y}_i$ denotes the Boolean complement of y_i and $d(\mathbf{x},\mathbf{x}')$ denotes the distance between $\mathbf{x}$ and $\mathbf{x}'$. Note that any metric can be used for d. So far, we have used only the L_1, L_2 and L_∞ metrics but it is certainly worthwhile to try to use metrics that actually incorporate some knowledge about the learning task. Moreover, we could use metrics that are obtained from the definition of an inner product $k(\mathbf{x},\mathbf{x}')$. Indeed, it is well known (Vapnik, 1998), that for any positive definite kernel $k(\mathbf{x},\mathbf{x}')$ there exists a mapping $\phi(\mathbf{x})$ such that $k(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})\cdot\phi(\mathbf{x}')$. In that case, we can choose $d(\mathbf{x},\mathbf{x}')$ to be the Euclidean distance between $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$:

$$d(\mathbf{x},\mathbf{x}') = ||\phi(\mathbf{x}) - \phi(\mathbf{x}')|| = \sqrt{k(\mathbf{x},\mathbf{x}) + k(\mathbf{x}',\mathbf{x}') - 2k(\mathbf{x},\mathbf{x}')}$$

Given a set S of m training examples, our initial set of features consists, in principle, of $\mathcal{H} = \bigcup_{i\in S}\bigcup_{\rho\in[0,\infty[} h_{i,\rho}$. But obviously, for each training example $\mathbf{x}_i$, we need only to consider the set of $m-1$ distances $\{d(\mathbf{x}_i,\mathbf{x}_j)\}_{j\neq i}$. This reduces our initial set $\mathcal{H}$ to $O(m^2)$ features. In fact, from the description of the SCM in the previous section, it follows that, for each greedy step, the ball h of largest usefulness value U_h always has a radius such that a $\mathcal{P}$ example is located at its border. Hence, for each example $\mathbf{x}_i$, we will consider all the balls of center $\mathbf{x}_i$ having a radius $d(\mathbf{x}_i,\mathbf{x}_j)+\epsilon$ for every $\mathbf{x}_j \in \mathcal{P}$ such that ϵ is a very small positive real number if $\mathbf{x}_i \in \mathcal{P}$ and a very small negative real number if $\mathbf{x}_i \in \mathcal{N}$. Therefore, we say that $\mathbf{x}_i$ is a ball *center* and $\mathbf{x}_j$ is a ball *border*.

In fact, we can reduce the size of $\mathcal{H}$ to exactly m features when $p=\infty$. Indeed, in that case, each feature must be consistent with all $\mathcal{P}$. This implies that a data-dependent ball centered on a $\mathcal{P}$-example must have a radius large enough to enclose all examples in $\mathcal{P}$. But among every ball that satisfies this constraint, the ball h that has the smallest radius will have the largest Q_h. Hence for each ball having a $\mathcal{P}$ center, we need only to consider the smallest radius that includes every $\mathcal{P}$-example. Similarly, for each ball having an $\mathcal{N}$ center, we need only to consider the largest radius that excludes every $\mathcal{P}$ example. Hence, for m training examples, we only need to consider exactly m data-dependent balls in the initial set $\mathcal{H}$ of features when $p=\infty$.

5. Performance Assessment

We now develop the necessary framework to enable estimates of the generalization performance of the SCM that uses data-dependent balls for its set of features. We follow the probably approximately correct (PAC) framework, which posits an underlying distribution generating the training data and defines the error of a hypothesis h as the probability that a randomly drawn test point is misclassified by h. In this framework bounds on the generalization error are required to hold with high probability, hence bounding the tail of the distribution of possible errors.

We seek a bound sensitive to the hypothesis selected. Bounds of this type are data-dependent in that they depend on properties of the solution obtained, in our case the number of features it uses. A general framework for analyzing data-dependent structural risk minimization is given in Shawe-Taylor et al. (1998), where the so-called luckiness framework is introduced. The assessment of the generalization of the SCM will require a data-dependent analysis since the features used depend on the particular training examples. The analysis, however, can be made by extending an older result due to Littlestone and Warmuth (1986). See also Floyd and Warmuth (1995) and Ben-David and Litman (1998).

Littlestone and Warmuth show that if we can *reconstruct* our hypothesis from a small subset of the training examples — the so-called *compression set* — then with high confidence the generalization can be bounded in terms of the size of this set. Here we extend the Littlestone and Warmuth technique by using a compression scheme that allows for specifying subsets of the compression set that can be treated separately for the reconstruction of the hypothesis. We begin by defining our expanded compression scheme.

Definition 3 *Let X be the input space and let $\mathcal{X} = (X \times \{0,1\})^m$ be the set of training sets of size m with inputs from X. Let A be the learning algorithm* **BuildSCM** *that uses data-dependent balls for its set of features. Given a training set $S \in \mathcal{X}$, $A(S)$ denotes the SCM classifier returned by algorithm A. If learning a conjunction then let $\mathcal{P} \subseteq S$ be the set of positive training examples and let $\mathcal{N} \subseteq S$ be the set of negative training examples, with the notations reversed when learning disjunctions (according to Definition 2).*

The compression function Λ induced by A is the map

$$\Lambda : S \longmapsto \Lambda(S) \stackrel{\text{def}}{=} (\Lambda^C_{\mathcal{P}}(S), \Lambda^C_{\mathcal{N}}(S), \Lambda^B_{\mathcal{P}}(S)),$$

such that the compression subsets $\Lambda^C_{\mathcal{P}} \subset \mathcal{P}$ and $\Lambda^C_{\mathcal{N}} \subset \mathcal{N}$ are the ball centers returned by algorithm A and $\Lambda^B_{\mathcal{P}} \subset \mathcal{P}$ is the set of ball borders returned by A that are not already contained in $\Lambda^C_{\mathcal{P}}$. Hence, each example in Λ is specified only once and a ball border can be the border of more than one ball (and can also be the center of another ball). Note also that no errors are made by $A(S)$ on examples in $\Lambda(S)$.

Consider a map Φ from arbitrary compression sets $S^C_{\mathcal{P}} \subset \mathcal{P}, S^C_{\mathcal{N}} \subset \mathcal{N}, S^B_{\mathcal{P}} \subset \mathcal{P}$ to a set F of Boolean-valued functions

$$\Phi : (S^C_{\mathcal{P}}, S^C_{\mathcal{N}}, S^B_{\mathcal{P}}) \longmapsto f \in F$$

Such a map Φ is said to be a reconstruction function relative to the compression function Λ induced by learning algorithm A iff for any training set $S \in \mathcal{X}$,

$$A(S) = \Phi(\Lambda(S)).$$

We will see below that it is the *mere existence* of a reconstruction function Φ relative to Λ that will enable us to bound the generalization error of SCMs as a function of $|\Lambda|$.

In the definition above, the splitting of $\Lambda(S)$ into three disjoint subsets represents some additional information that will be needed for Φ to reconstruct the same function as $A(S)$. Indeed, it is easy to realize that it is impossible to reconstruct $A(S)$ from a single set $\Lambda(S)$ containing no internal structure. Consider, for example, the case where $A(S)$ contains a single ball with a $\mathcal{P}$-example for its center. Then $\Lambda(S)$ would contain only two examples: a center and a border. It is impossible to construct the ball without knowing which example is the center. Hence, from the set of $\mathcal{P}$-examples in $\Lambda(S)$, Φ needs to know which examples are centers.

Lemma 4 *Let A be the learning algorithm* **BuildSCM** *that uses data-dependent balls for its set of features. Let Λ be the compression function induced by A defined in Definition 3. Then there exists a reconstruction function Φ relative to Λ if $A(S)$ correctly classifies every example in Λ for any training set S.*

Proof Given arbitrary compression subsets $S_{\mathcal{P}}^C \subset \mathcal{P}, S_{\mathcal{N}}^C \subset \mathcal{N}, S_{\mathcal{P}}^B \subset \mathcal{P}$ as defined in Definition 3, function Φ first constructs the following set of balls. For each $\mathbf{x}_i \in S_{\mathcal{P}}^C$, it creates a ball centered at $\mathbf{x}_i$ with radius $d + \epsilon$ where $d = \max_{j \in S_{\mathcal{P}}^B \cup S_{\mathcal{P}}^C} d(\mathbf{x}_i, \mathbf{x}_j)$ and ϵ is the same arbitrary small positive real number used by algorithm A. Similarly, for each $\mathbf{x}_i \in S_{\mathcal{N}}^C$, it creates a ball centered at $\mathbf{x}_i$ with radius $d - \epsilon$ where $d = \min_{j \in S_{\mathcal{P}}^B \cup S_{\mathcal{P}}^C} d(\mathbf{x}_i, \mathbf{x}_j)$. Function Φ then builds a conjunction of these balls if the $\mathcal{P}$-examples are positive examples, otherwise it builds a disjunction of these balls. This implies that the compression subsets are always correctly classified by Φ. Since, in the statement of the lemma, we have added the additional constraint that $A(S)$ always correctly classifies every example in Λ, then we have that $A(S) = \Phi(\Lambda(S))$ for any training set S. ■

Hence, the reconstruction function Φ defined above is always capable of reconstructing the function returned by A only when $A(S)$ always correctly classifies every example in Λ. Hence, this means that our bound on the generalization error will hold only if **BuildSCM** is modified in such a way to ensure that each ball border and center will be correctly classified by the returned classifier. This is easily done by imposing the constraint that, at each greedy step, we only consider balls which are consistent with every $\mathcal{P}$-example present in the centers and borders of the previous balls in $\mathcal{R}$. That constraint was imposed for the numerical experiments presented in Section 6 (releasing that constraint did not change the results significantly).

The refinement that we will present of the Littlestone and Warmuth bounds is to take account of the number of positive/negative examples in the training set. These numbers can provide tighter constraints on the choices for the sets $\Lambda_{\mathcal{P}}^C$, $\Lambda_{\mathcal{N}}^C$, and $\Lambda_{\mathcal{P}}^B$. A naive inclusion of these constraints undermines the assumptions of the proof technique. Some extra machinery is required to carry though the argument under these conditions.

Recall that, in the PAC framework, the underlying fixed (but unknown) probability distribution P generates both the training and the testing examples. The aim is to bound with high confidence the expected loss of the function returned by the learning algorithm, where by high confidence we mean that most random training samples will satisfy the bound.

For the purposes of the next theorems we consider the binary loss function $l(f,(\mathbf{x},y)) = 1$ if $f(\mathbf{x}) \neq y$, and 0 otherwise. The corresponding expected loss (*i.e.*, the generalization error) is then given by

$$\mathbb{E}[l(f)] \stackrel{\text{def}}{=} P\{(\mathbf{x},y): f(\mathbf{x}) \neq y\}.$$

We also consider the error probability separately on $\mathcal{P}$ and $\mathcal{N}$ examples:

$$\mathrm{er}_{\mathcal{P}}(f) \stackrel{\text{def}}{=} P\{f(\mathbf{x}) \neq y | (\mathbf{x},y) \in \mathcal{P}\} \text{ and } \mathrm{er}_{\mathcal{N}}(f) \stackrel{\text{def}}{=} P\{f(\mathbf{x}) \neq y | (\mathbf{x},y) \in \mathcal{N}\}.$$

Further let $\hat{\mathrm{er}}(f,S)$ denote the sample error:

$$\hat{\mathrm{er}}(f,S) \stackrel{\text{def}}{=} |\{(\mathbf{x},y) \in S: f(\mathbf{x}) \neq y\}|$$

and let $\hat{\mathrm{er}}_{\mathcal{P}}(f,S)$ denote the number of $\mathcal{P}$ points misclassified (with a similar definition for $\hat{\mathrm{er}}_{\mathcal{N}}(f,S)$).

There will be a number of parameters involved in the bounds which would make them very opaque if written out in full at each stage. We therefore introduce notation for the collection of parameters. Let $\mathbf{m}(S)$ denote the vector $(m, m_p, m_n, c_p, c_n, b_p, k_p, k_n)$, where $m \stackrel{\text{def}}{=} |S|, m_p \stackrel{\text{def}}{=} |\mathcal{P}|, m_n \stackrel{\text{def}}{=} |\mathcal{N}|, c_p \stackrel{\text{def}}{=} |\Lambda^C_{\mathcal{P}}(S)|, c_n \stackrel{\text{def}}{=} |\Lambda^C_{\mathcal{N}}(S)|, b_p \stackrel{\text{def}}{=} |\Lambda^B_{\mathcal{P}}(S)|, k_p \stackrel{\text{def}}{=} \hat{\mathrm{er}}_{\mathcal{P}}(A(S),S)$, and $k_n \stackrel{\text{def}}{=} \hat{\mathrm{er}}_{\mathcal{N}}(A(S),S)$. Finally, we denote with $B(\mathbf{m})$ the following number of ways of choosing the compression set and $\mathcal{P}$ and $\mathcal{N}$ errors:

$$B(\mathbf{m}) \stackrel{\text{def}}{=} \binom{m_p}{c_p}\binom{m_p - c_p}{b_p}\binom{m_n}{c_n}\binom{m_p - c_p - b_p}{k_p}\binom{m_n - c_n}{k_n}$$

Theorem 5 *Given the above definitions, let A be the learning algorithm* **BuildSCM** *that uses data-dependent balls for its set of features with the constraint that the returned function $A(S)$ always correctly classifies every example in the compression set. With probability $1-\delta$ over all random training sets S the expected loss $\mathbb{E}[l(A(S))]$ of $A(S)$ is bounded by*

$$\mathbb{E}[l(A(S))] \leq 1 - \exp\left\{-\frac{1}{m - c_p - b_p - c_n - k_p - k_n}\left(\ln B(\mathbf{m}(S)) + \ln\frac{1}{\delta(\mathbf{m}(S))}\right)\right\}$$

where $\delta(\mathbf{m}) \stackrel{\text{def}}{=} \left(\frac{\pi^2}{6}\right)^{-5} \cdot ((c_p+1)(c_n+1)(b_p+1)(k_p+1)(k_n+1))^{-2} \cdot \delta$

Proof We will use the notation $\mathbf{i} = (i_1, \ldots, i_d)$ for a sequence of strictly increasing indices, $0 < i_1 < i_2 < \cdots < i_d \leq m$, where with $|\mathbf{i}|$ we denote the length d of the sequence. If the sequence $\mathbf{i}'$ contains all the indices in $\mathbf{i}$, we write $\mathbf{i} \subset \mathbf{i}'$. For $S \in \mathcal{X}$, $S_{\mathbf{i}} = ((x_{i_1}, y_{i_1}), \ldots, (x_{i_d}, y_{i_d}))$. We will be using indices to select points among $\mathcal{P}$ or $\mathcal{N}$. For example, $\mathcal{P}_{\mathbf{i}}$ is the sequence whose first entry is the i_1-st $\mathcal{P}$ example. For a compression set $\Lambda(S)$ with S clear from the context we denote the indices of the sets $\Lambda^C_{\mathcal{P}}(S)$, $\Lambda^C_{\mathcal{N}}$ and $\Lambda^B_{\mathcal{P}}$ by $\mathbf{i}^c_p$, $\mathbf{i}^c_n$ and $\mathbf{i}^b_p$ denoting them collectively by $\mathbf{i}_\lambda$, where it is understood that $\Lambda^C_{\mathcal{P}}(S) = \mathcal{P}_{\mathbf{i}^c_p}$, etc.

We first argue conditional on a fixed set of integers, $\mathbf{m}(S) = \mathbf{m}$ where

$$\mathbf{m} \stackrel{\text{def}}{=} (m, m_p, m_n, c_p, c_n, b_p, k_p, k_n).$$

By the existence of a reconstruction function relative to the compression function Λ induced by learning algorithm A, and the fact that the compression set is correctly classified by its reconstruction, we can obtain the following probability bound

$$\begin{aligned} & P\left\{S \in \mathcal{X} \quad : \quad \mathbb{E}[l(A(S))] \geq \epsilon \;\middle|\; \mathbf{m}(S) = \mathbf{m}\right\} \\ & \qquad \leq B(\mathbf{m}(S)) P\Big\{S \in \mathcal{X} \colon \mathbf{i}_\lambda = \mathbf{i}_0, \mathbb{E}[l(A(S))] \geq \epsilon, \\ & \qquad\qquad \hat{\mathrm{er}}(A(S), \mathcal{P}_{\mathbf{i}_p}) = k_p, \hat{\mathrm{er}}(A(S), \mathcal{N}_{\mathbf{i}_n}) = k_n \;\Big|\; \mathbf{m}(S) = \mathbf{m}\Big\}, \end{aligned} \tag{2}$$

where $\mathbf{i}_0$, $\mathbf{i}_p$, and $\mathbf{i}_n$ are fixed disjoint index sequences with $|\mathbf{i}_p| = k_p$ and $|\mathbf{i}_n| = k_n$. Note that the two empirical error expressions imply that the mistakes are made on precisely the points in positions $\mathbf{i}_p$ and $\mathbf{i}_n$. With this choice of indices fixed, we now consider conditioning the second factor on the actual points generated in these positions, that is $\Lambda(S) = S_0$ for some fixed sequences S_0. Similarly, $S_{\mathbf{i}_p} = S_p$ and $S_{\mathbf{i}_n} = S_n$ for fixed sequences S_p and S_n. This implies that the function $A(S_0)$ is also fixed. We will bound this conditional probability independently of the particular points, hence bounding the unconditioned expression. Note that we will frequently be excluding the positive (negative) examples contained in the compression or error sets. We therefore introduce the notation $t_p \stackrel{\text{def}}{=} c_p + b_p + k_p$ for the total number of such positive examples and $t_n \stackrel{\text{def}}{=} c_n + k_n$ for the number of such negative examples.

$$\begin{aligned} & P\Big\{S \in \mathcal{X} \quad : \quad \mathbb{E}[l(A(S_0))] \geq \epsilon, \hat{\mathrm{er}}(A(S), S \setminus (S_p \cup S_n)) = 0, \\ & \qquad\qquad \Big| \mathbf{i}_\lambda = \mathbf{i}_0, \Lambda(S) = S_0, \mathcal{P}_{\mathbf{i}_p} = S_p, \mathcal{N}_{\mathbf{i}_n} = S_n, \mathbf{m}(S) = \mathbf{m}\Big\} \\ & \qquad = \quad (1 - \mathrm{er}_{\mathcal{P}}(A(S_0)))^{m_p - t_p} (1 - \mathrm{er}_{\mathcal{N}}(A(S_0)))^{m - t_n - m_p} \\ & \qquad\qquad \binom{m - t_n - t_p}{m_p - t_p} p_{\mathcal{P}}^{m_p - t_p} (1 - p_{\mathcal{P}})^{m - t_n - m_p} \\ & \qquad \leq \quad \sum_{m' = t_p}^{m - t_n} (1 - \mathrm{er}_{\mathcal{P}}(A(S_0)))^{m' - t_p} (1 - \mathrm{er}_{\mathcal{N}}(A(S_0)))^{m - t_n - m'} \\ & \qquad\qquad \binom{m - t_n - t_p}{m' - t_p} p_{\mathcal{P}}^{m' - t_p} (1 - p_{\mathcal{P}})^{m - t_n - m'} \\ & \qquad = \quad [(1 - \mathrm{er}_{\mathcal{P}}(A(S_0))) p_{\mathcal{P}} + (1 - \mathrm{er}_{\mathcal{N}}(A(S_0)))(1 - p_{\mathcal{P}})]^{m - t_n - t_p} \\ & \qquad = (1 - \mathbb{E}[l(A(S_0))])^{m - t_n - t_p} \leq (1 - \epsilon)^{m - t_n - t_p}, \end{aligned} \tag{3}$$

since the remaining points are correctly classified with probability $1 - \mathrm{er}_{\mathcal{P}}(A(S_0))$ for $\mathcal{P}$ points and $1 - \mathrm{er}_{\mathcal{N}}(A(S_0))$ for $\mathcal{N}$ points and the expected loss is $\mathrm{er}_{\mathcal{P}}(A(S_0)) p_{\mathcal{P}} + \mathrm{er}_{\mathcal{N}}(A(S_0))(1 - p_{\mathcal{P}})$. Note that although we bounded the probability using a sum over m', this is simply a manipulation and does not mean that we have varied m_p.

Let us denote by $U(\mathbf{m})$ the bound stated in the theorem. By using the bound on the conditional probability to combine equations (2) and (3), we obtain

$$P\left\{S \in \mathcal{X}\colon \mathbb{E}[l(A(S))] \geq \epsilon \;\middle|\; \mathbf{m}(S) = \mathbf{m}\right\} \leq B(\mathbf{m}(S))(1-\epsilon)^{m-t_n-t_p}.$$

$$\begin{aligned} P\left\{S \in \mathcal{X}\colon \mathbb{E}[l(A(S))] \geq U(\mathbf{m})\right\} &= \sum_{\mathbf{m}} P\left\{S \in \mathcal{X}\colon \mathbb{E}[l(A(S))] \geq U(\mathbf{m}) \;\middle|\; \mathbf{m}(S) = \mathbf{m}\right\} P\left\{S \in \mathcal{X}\colon\ \mathbf{m}(S) = \mathbf{m}\right\} \\ &= \sum_{\mathbf{m}} \delta(\mathbf{m}) P\left\{S \in \mathcal{X}\colon\ \mathbf{m}(S) = \mathbf{m}\right\} \leq \delta \sum_{m_p=0}^{m} P\left\{S \in \mathcal{X}\colon\ |S \cap \mathcal{P}| = m_p\right\} = \delta, \end{aligned}$$

where in the first inequality of the final line we have taken into account that the sum of $\delta(\mathbf{m})$ over all possible realizations of $\mathbf{m}$ for a fixed m_p and m_n is less than δ (since $\sum_{i=1}^{\infty}(1/i^2) = \pi^2/6$), while the final equality follows from the fact that the sum over the probabilities of observing m_p positive examples for different values of m_p is 1. ■

Note that the bound of Theorem 5 is presented in a non-standard format in order to obtain a tighter bound. Normally, the approximation $(1-x)^k \leq \exp(-kx)$ is used in Equation 3 to simplify the bound. This would give a bound in the more familiar form

$$\mathbb{E}[l(A(S))] \leq \frac{1}{m - c_p - b_p - c_n - k_p - k_n}\left(\ln B(\mathbf{m}(S)) + \ln \frac{1}{\delta(\mathbf{m}(S))}\right).$$

Hence, Theorem 5 shows that good generalization is expected whenever a SCM with few training errors and few data-dependent balls is found by **BuildSCM**. Moreover, in comparison with traditional bounds based on VC dimension (Vapnik, 1998), our sample compression bound is tight and non trivial (*i.e.*, always less than one). Hence, there is hope that this bound will be effective at finding the best model-selection parameters (here p and s) based on what **BuildSCM** has achieved on the training set only.

Note that the bound is larger due to fact that we have decomposed the compression set Λ into three disjoint subsets $(\Lambda_{\mathcal{P}}^C, \Lambda_{\mathcal{N}}^C, \Lambda_{\mathcal{P}}^B)$. However, we can restrict further the set of data-dependent balls by imposing that only $\mathcal{N}$-examples are allowed to be used for ball centers. In that simpler case we only need $(\Lambda_{\mathcal{N}}^C, \Lambda_{\mathcal{P}}^B)$ to be able to reconstruct the same function as the one returned by **BuildSCM**. Furthermore, for this simpler case, we have $c_p = 0$, and the number of balls is given by c_n. Hence, Theorem 5 has the following corollary.

Corollary 6 *Given the same definitions as those used for Theorem 5, let A be the learning algorithm* **BuildSCM** *that uses data-dependent balls for its set of features, with only $\mathcal{N}$-examples as centers, and with the constraint that the returned function $A(S)$ always correctly classifies every example in the compression set. With probability $1-\delta$ over all random training sets S the expected loss $\mathbb{E}[l(A(S))]$ of $A(S)$ is bounded by*

$$\begin{aligned} \mathbb{E}[l(A(S))] \leq\ & 1 - \exp\left\{\frac{-1}{m - c_n - b_p - k_p - k_n}\left[\ln\binom{m_n}{c_n} + \ln\binom{m_p}{b_p}\right.\right. \\ & \left.\left. + \ln\binom{m_n - c_n}{k_n} + \ln\binom{m_p - b_p}{k_p} + \ln\left(\frac{1}{\delta(c_n, b_p, k_p, k_n)}\right)\right]\right\} \end{aligned}$$

where:

$$\delta(c_n, b_p, k_p, k_n) = \left(\frac{\pi^2}{6}\right)^{-4} \cdot ((c_n+1)(b_p+1)(k_p+1)(k_n+1))^{-2} \cdot \delta$$

For the same number of balls and training errors, the bound of Corollary 6 is smaller than the one provided by Theorem 5. Hence, in that case, a simpler SCM that uses only $\mathcal{N}$-examples for ball centers might be preferable than the more complex SCM that uses examples of both classes for centers. However, this might not be the case if the simpler SCM makes more training errors. Because of this non-trivial tradeoff between training accuracy and function complexity, both types of SCMs should be tried in practice.

Our bounds indicate that good generalization is expected if we can find a small and simple SCM with few training errors. But *when* do we expect to find such a small and simple SCM? More specifically, we seek to find the conditions *on the training data* for which we are guaranteed to find a small and simple SCM. In the next theorem, we show that these conditions are easily identified in the case of conjunctions and disjunctions because the training problem for them reduces to the minimum set cover problem. The simple proof follows directly from Chvátal (1979) and Kearns and Vazirani (1994). We include it here for completeness.

Theorem 7 *Consider any training set $S = \mathcal{P} \cup \mathcal{N}$ of examples and any set $\mathcal{H}$ of features, possibly constructed from the data, for which there exist a subset of r features that covers all $\mathcal{N}$. Then* **BuildSCM***, with both p and s set to infinity, will return a function that contains at most $r\ln(|\mathcal{N}|)$ features.*

Proof If there exist r features of $\mathcal{H}$ that cover all $\mathcal{N}$, then **BuildSCM** will find a first feature that covers at least $|\mathcal{N}|/r$ examples of $\mathcal{N}$. Hence after l greedy steps, the number of examples from $\mathcal{N}$ that remain to be covered is at most $|\mathcal{N}|(1-1/r)^l < |\mathcal{N}|\exp(-l/r)$. Therefore, the value of $r\ln(|\mathcal{N}|)$ for l is sufficient to guarantee that less than 1 example of $\mathcal{N}$ will not be covered. ■

We can, in fact, generalize Theorem 7 to the case where there exists a smaller subset of features that partially covers $\mathcal{N}$.

Theorem 8 *Consider any training set $S = \mathcal{P} \cup \mathcal{N}$ of examples and any set $\mathcal{H}$ of features, possibly constructed from the data, for which there exists a subset of r features that covers $m_n - k$ examples of $\mathcal{N}$ (where $m_n \stackrel{\text{def}}{=} |\mathcal{N}|$ and k is a nonnegative number). Let $\beta \stackrel{\text{def}}{=} r\ln(m_n - k)$. Then* **BuildSCM***, with p set to infinity and s set to β, will return a function that makes at most k errors on $\mathcal{N}$.*

Proof We will refer to the $m_n - k$ examples covered by the r features hypothesized in the corollary statement as the main examples. The k excluded points will be referred to as the excluded examples, while the r features used will be termed the main features. Consider the point at which the i-th feature is sought during the learning algorithm. Suppose that there remain ℓ_i of the $m_n - k$ main examples still not covered. Since all of these examples can be covered by the r main features, by the pigeon hole principle there exists a feature that can cover ℓ_i/r of them. Hence, the greedy algorithm may select a feature that covers at least this many examples. It may also, however, select a feature that covers fewer of the main examples in exchange for covering some of the k excluded examples not already covered. Hence, more generally, suppose that it selects a feature that covers k_i of the excluded examples that have not been covered by features selected at an earlier stage (for some nonnegative value of k_i). In all cases, the i-th feature covers $\ell_i - \ell_{i+1} + k_i$ examples. And, by the pigeon hole principle, that number must be at least ℓ_i/r. Hence, we always have that

$$\ell_{i+1} - k_i \quad \leq \quad \ell_i \left(1 - \frac{1}{r}\right). \tag{4}$$

We will prove by induction that the number of examples not covered after stage t satisfies

$$\ell_{t+1} + k - \sum_{i=1}^{t} k_i \leq k + (m_n - k)\left(1 - \frac{1}{r}\right)^t,$$

showing that $m_n - k$ examples will be covered by $r \ln(m_n - k)$ features, since

$$(m_n - k)\left(1 - \frac{1}{r}\right)^{r \ln(m_n - k)} < (m_n - k)\exp(-\ln(m_n - k)) = 1.$$

The base case of $t = 0$ for the induction is immediate since $\ell_1 = m_n - k$. Suppose now the bound holds for $t - 1$. Using this and equation (4) we have

$$\begin{aligned} \ell_{t+1} + k - \sum_{i=1}^{t} k_i \quad &\leq \quad \ell_t \left(1 - \frac{1}{r}\right) + k - \sum_{i=1}^{t-1} k_i \\ &\leq \quad k + (m_n - k)\left(1 - \frac{1}{r}\right)^t, \end{aligned}$$

as required. ∎

The conditions on the training data for which **BuildSCM** will fail to give a small and simple SCM are easily understood. Take for example the case were *all* the training examples are located very close to the border of a two-dimensional circle. In that case **BuildSCM** will need $O(|\mathcal{N}|)$ balls to cover all the $\mathcal{N}$-examples. But if we are lucky and *there exists* a small subset of the training set from which we can construct a set of balls that will cover a large fraction of $\mathcal{N}$, then **BuildSCM** will find a good SCM.

Note that the bound on the number of features needed is independent of $|\mathcal{H}|$. But a generalization error bound based on sample compression is likely to increase with the complexity of $\mathcal{H}$. Indeed, larger compression sets or more information (*i.e.*, more compression subsets) will be required for more complex feature sets $\mathcal{H}$ in order to be able to reconstruct, from the compression set, the function returned by **BuildSCM**.

Finally the next theorem states how the running time of **BuildSCM** will increase with $|\mathcal{H}|$ and with its minimum subset r needed to cover all $\mathcal{N}$.

Theorem 9 *Consider any training set $S = \mathcal{P} \cup \mathcal{N}$ of examples and any set $\mathcal{H}$ of features, possibly constructed from the data, for which there exists a subset of r features that covers all $\mathcal{N}$. Let t be the maximum time needed to compute, on an example, the function value of a feature in $\mathcal{H}$. Then, the running time of* **BuildSCM** *will be at most*

$$\begin{array}{ll} t\cdot\ |\mathcal{H}|\cdot|\mathcal{N}|\cdot\ r & \text{for infinite } p \\ t\cdot\ |\mathcal{H}|\cdot\ \{|\mathcal{N}|\cdot r+|\mathcal{P}|\cdot(r\ln(|\mathcal{N}|)+1)\} & \text{for finite } p \end{array}$$

Proof For $p = \infty$, the time needed to find the first feature will be $|\mathcal{H}|\cdot\ |\mathcal{N}|\cdot\ t$ since every feature in $\mathcal{H}$ needs to be evaluated exactly once on every example in $\mathcal{N}$. From Theorem 7, it follows that this time must be multiplied by

$$\sum_{l=0}^{r\ln(|\mathcal{N}|)+1}\left(1-\frac{1}{r}\right)^l \ <\ r$$

in order to obtain the time needed to cover all $\mathcal{N}$ from features in $\mathcal{H}$.

For finite p, the time needed to find the first feature will be $|\mathcal{H}|\cdot\ (|\mathcal{N}|+|\mathcal{P}|)\cdot\ t$ since every feature in $\mathcal{H}$ needs to be evaluated on every example in $\mathcal{N}\cup\mathcal{P}$ to find the feature with the best usefulness value. That feature is guaranteed to cover at least $|\mathcal{N}|/r$ examples in $\mathcal{N}$. Since these examples will be removed, the time needed to find the second feature is at most $|\mathcal{H}|\cdot\ (|\mathcal{N}|(1-\frac{1}{r})+|\mathcal{P}|)\cdot\ t$. Hence, for the same reasons, the time needed to find the $(l+1)$th feature is at most $|\mathcal{H}|\cdot\ (|\mathcal{N}|(1-\frac{1}{r})^l+|\mathcal{P}|)\cdot\ t$. The summation of all these times for at most $r\ln(|\mathcal{N}|)+1$ features gives the running time bound for the case of finite p. ■

Let us use Theorem 9 to bound the running time of **BuildSCM** when it uses data-dependent balls. First note that, when ball centers of both classes are considered, we have $|\mathcal{H}| = |\mathcal{N}|+|\mathcal{P}| = m$ for infinite p and $|\mathcal{H}| = (|\mathcal{N}|+|\mathcal{P}|)\cdot\ |\mathcal{P}| = m\cdot\ |\mathcal{P}|$ for finite p. But for infinite values of p, we still need to compute $(|\mathcal{N}|+|\mathcal{P}|)\cdot\ |\mathcal{P}|$ distances to find the best radius for each center. Since it takes a time of $O(n)$ to compute a distance (with a standard metric such as L_1 or L_2) between a pair of examples of n attributes, it follows that it takes a time of $O(n\cdot\ (|\mathcal{N}|+|\mathcal{P}|)\cdot\ |\mathcal{P}|)$ to create $\mathcal{H}$ for both infinite and finite values of p. Similarly, when only $\mathcal{N}$-examples are used for ball centers, we have $|\mathcal{H}| = |\mathcal{N}|$ for infinite p and $|\mathcal{H}| = |\mathcal{N}|\cdot\ |\mathcal{P}|$ for finite p and, consequently, we need a time of $O(n\cdot\ |\mathcal{N}|\cdot\ |\mathcal{P}|)$ to create $\mathcal{H}$ for both infinite and finite values of p.

Once $\mathcal{H}$ is generated, the time t needed to evaluate a feature on an example is of $O(1)$ if the distances are cached (*i.e.*, stored in memory), or of $O(n)$ if the distances are not cached. Hence, from Theorem 9, once $\mathcal{H}$ is generated, **BuildSCM** will need at most a running time

$$\begin{array}{ll} t\cdot\ m\cdot|\mathcal{N}|\cdot\ r & \text{for infinite } p \\ t\cdot\ m\cdot\ |\mathcal{P}|\cdot\ \{|\mathcal{N}|\cdot r+|\mathcal{P}|\cdot(r\ln(|\mathcal{N}|)+1)\} & \text{for finite } p \end{array}$$

for the case where ball centers of both classes are used. But it will need only a running time of at most

$$\begin{array}{ll} t\cdot\ |\mathcal{N}|^2\cdot\ r & \text{for infinite } p \\ t\cdot\ |\mathcal{N}|\cdot\ |\mathcal{P}|\cdot\ \{|\mathcal{N}|\cdot r+|\mathcal{P}|\cdot(r\ln(|\mathcal{N}|)+1)\} & \text{for finite } p \end{array}$$

Data Set	# of exs		NNC	SVM ($C = \infty$)			SVM (finite C)			
	pos	neg	error	γ	size	error	γ	C	size	error
BreastW	239	444	29	0.07	219.7	27	0.005	2	57.7	19
Votes	18	34	7	0.05	16.6	5	0.05	15	18.2	**3**
Pima	269	499	247	0.004	543.7	243	0.002	1	526.2	203
Haberman	219	75	107	0.02	92.5	111	0.01	0.6	146.4	**71**
Bupa	145	200	124	0.02	290.4	121	0.002	0.2	265.9	107
Glass	87	76	36	0.2	47.2	42	0.8	2	91.8	34
Credit	296	357	214	0.001	406.9	205	0.0006	32	423.2	**190**

Table 1: Data sets, NNC results, and SVM results.

when ball centers are restricted to $\mathcal{N}$-examples. Hence we see that there is a substantial computational advantage at using balls with centers taken only from $\mathcal{N}$. This is especially true for imbalance class problems where $\mathcal{N} \ll \mathcal{P}$.

These running time bounds seem to be competitive with most quadratic optimizers used for SVMs. But a direct comparison is questionable in view of the fact that the running times bounds for quadratic optimization procedures are expressed in terms of quantities that do not exist in our case. On the data sets used in the next section, we have found that our implementation of **BuildSCM** gave substantially smaller running times than those given by the SVM program distributed by the department of Computer Science of the Royal Holloway, University of London (Saunders et al., 1998).

6. Empirical Results on Natural Data

We have compared the practical performance of the set covering machine (SCM) using the set of data-dependent balls with the nearest-neighbor classifier (NNC) and the support vector machine (SVM) equipped with a Gaussian kernel (also called the Radial Basis Function kernel) of variance $1/\gamma$. We have used the SVM program distributed by the Royal Holloway University of London (Saunders et al., 1998). The data sets used and the results obtained for NNCs and SVMs are reported in table 1. All these data sets where obtained from the machine learning repository at UCI (Blake and Merz, 1998), except the Glass data set which was obtained from Rob Holte, now at the University of Alberta. For each data set, we have removed all examples that contained attributes with unknown values (this has reduced substantially the "votes" data set) and we have removed examples with contradictory labels (this occurred only for a few examples in the Haberman data set). The remaining number of examples for each data set is reported in table 1. For all these data sets, we have used the 10-fold cross validation error as an estimate of the generalization error. The values reported are expressed as the total number of errors (*i.e.*, the sum of errors over all testing sets). We have ensured that each training set and each testing set, used in the 10-fold cross validation process, was the same for each learning machine (*i.e.*, each machine was trained on the same training sets and tested on the same testing sets).

For the SVM, the values of the kernel parameter γ and the soft margin parameter C are the ones that gave the smallest 10-fold cross validation error among an *exhaustive* scan

Data Set	Type	$p = \infty$, $s = \infty$		$p = \infty$, finite s		
		size	error	size	error	bound
BreastW	c	13.9	26	1	23	139
	d	16.8	34	10	34	157
Votes	c	3.7	7	1	6	23
	d	3.2	7	3	7	25
Pima	c	147.2	262	153	262	NA
	d	124.8	230	20	208	626
Haberman	c	45.5	104	1	76	202
	d	46.6	128	39	127	253
Bupa	c	80.6	144	58	142	334
	d	68.8	131	46	118	320
Glass	c	19	45	7	40	110
	d	15.9	36	11	36	104
Credit	c	137.4	255	121	253	634
	d	123.5	225	79	217	608

Table 2: SCM results, with $p = \infty$, for conjunctions (c) and disjunctions (d).

of *many* values for these parameters. We have also reported the average number of support vectors (in the "size" columns) contained in machines obtained from the 10 different training sets of 10-fold cross-validation. Note that SVMs performed significantly better than NNCs only for the soft margin case (finite values of C).

The results for the SCM are reported in tables 2 and 3. Despite the fact that we have observed that results with the L_1 metric are often better, we have only reported here the results for the L_2 metric. This was done to obtain a fair comparison with SVMs because the argument of the Radial Basis Function kernel is given by the L_2 metric between two input vectors. The L_2 metric was also used for NNCs.

We have reported in table 2 the results for SCMs trained with **BuildSCM** for $p = \infty$ and both infinite and finite values of s. Algorithm **BuildSCM** was permitted to use ball centers from both $\mathcal{N}$ and $\mathcal{P}$ for all the results of this table. For infinite s, the SCM was trained until it made zero training errors. We see that these are the largest machines and that they often have the largest generalization error. The size, expressed as the number of data-dependent balls, is the average of sizes obtained from the ten different training sets of 10-fold cross validation.

For finite s, and for one pair of training and testing set, the SCM was first trained until it made zero training errors and then tested on the testing set for all possible sizes. The size (in terms of the number of balls) that gave the least 10-fold cross validation error is reported in table 2. This is why that size is always an integer. We see that early stopping (finite s) is often effective at decreasing the generalization error. Finally note that the generalization error of SCMs trained with infinite p and finite s is generally similar to the one obtained for SVMs with infinite C.

We have reported in table 3 the results for SCMs trained with **BuildSCM** for finite p values (and finite s values). The reported values of p and s (size) are the ones that

Data Set	Type	Simple SCM				Complex SCM			
		p	size	error	bound	p	size	error	bound
BreastW	c	0.8	1	18	110	1.8	2	**15**	99
	d	0.4	1	16	108	0.4	1	16	109
Votes	c	0.8	1	7	25	1.2	1	6	23
	d	0.8	1	6	23	0.9	1	6	23
Pima	c	1.1	3	**189**	621	1.1	3	**189**	622
	d	1.1	15	206	604	4.1	16	206	624
Haberman	c	1.4	1	**71**	202	1.4	1	**71**	202
	d	1.8	1	93	230	1.4	12	**71**	NA
Bupa	c	1.5	9	109	300	1.6	20	121	NA
	d	3.0	10	**106**	292	2.8	9	107	291
Glass	c	1.0	3	35	112	0.85	4	**33**	105
	d	5.0	11	36	104	5.0	11	36	104
Credit	c	0.8	4	198	579	0.8	4	198	579
	d	1.0	4	195	572	1.2	4	194	570

Table 3: SCM results, with finite p and finite s, for conjunctions (c) and disjunctions (d).

gave the best results among an *exhaustive* scan of *many* values for these parameters. The results for "Simple SCMs" refer to those obtained when **BuildSCM** was constrained to use $\mathcal{N}$-examples for ball centers and the results for "Complex SCMs" refer to those obtained when **BuildSCM** was permitted to use examples from both $\mathcal{N}$ and $\mathcal{P}$ for ball centers. The results for "Complex SCMs" represent a substantial improvement over the case of infinite p reported in table 2. We see that finite p values generally give smaller SCMs with better generalization.

The generalization error of the "Simple SCM" is almost always very similar to the one obtained from the "Complex SCM". The two exceptions are the disjunction for the Haberman data set, were the simple SCM performed substantially worse than the complex SCM, and the conjunction on the Bupa data set, where the simple SCM performed substantially better than the complex SCM. However, for each data set, there was always either a conjunction or a disjunction that performed (almost) as well as the best complex SCM. Hence, there was no significant advantage in using the more complex SCMs on these data sets, but there was no disadvantage either (except for a degradation of time complexity).

The generalization error of SCMs trained with finite p is generally similar to the one obtained for SVMs with finite C. The main difference is in the size of the classifier. SCMs are always substantially smaller than SVMs.

We have also reported, in the "bound" columns of tables 2 and 3, the bound on the generalization error obtained by computing the r.h.s. of the inequality of Theorem 5 (with $\delta = .05$), for each of the 10 different training sets involved in 10-fold cross validation, and multiplying that bound with the size of each testing sets. A value of "NA" means that a valid bound could not be found for some SCMs of the specified size among those obtained during the 10-fold cross validation process. We see that, although the bound is not tight, it is nevertheless non-trivial. This is to be contrasted with the VC dimension bounds which

Data Set	Type	MS from 10-fold CV			MS from bound			Optimal		
		size	error	std	size	error	std	p	size	error
BreastW	c	1.7	17	11	1.8	16	12	1.8	2	15
	d	1	16	12	1.0	16	12	0.4	1	16
Votes	c	1	6	8	1	6	8	1.2	1	6
	d	1	6	8	1	6	8	0.9	1	6
Pima	c	7	195	25	9.1	192	20	1.1	3	189
	d	11.4	220	19	31.8	214	28	4.1	16	204
Haberman	c	1.5	75	13	2.1	77	14	1.4	1	71
	d	8.1	77	26	4.8	76	24	1.4	12	71
Bupa	c	9.7	122	23	8.6	125	24	1.6	20	121
	d	8.8	115	17	14.3	109	19	2.8	9	106
Glass	c	2.6	39	20	3.5	36	24	0.85	4	33
	d	7.2	41	23	6.8	38	19	5.0	11	36
Credit	c	1.6	216	40	7.3	204	38	0.8	4	197
	d	1.8	203	39	16.4	209	35	1.2	4	194

Table 4: SCM model-selection results for a fixed value of penalty p.

cannot even be applied for our case since the set of functions supported by the SCM depends on the training data.

Let us now investigate the extent to which our bound can perform SCM model-selection. More specifically, we want to answer the following question. Given a set of SCMs obtained from **BuildSCM** for various values of the model-selection parameters p and s, is our bound on the generalization error, evaluated on the training set, effective at selecting the SCM that will give the best generalization?

Note that the results reported in table 3 are, in fact, the 10-fold cross validation estimate of the generalization error that is achieved by the model selection strategy that correctly guesses the best values for p and s. This model-selection strategy is, in that sense, optimal (but not realizable). Hence, we will refer to the score obtained in table 3 as those obtained by the optimal model-selection strategy.

The results for the model-selection strategy based on our bound are reported in table 4 for the case of a "fixed" value of p set to its optimal value and in table 5 for the case of variable values of penalty p. In this latter case, we have used our bound to select the best SCM among those obtained for various penalty values among a list of fifteen penalty values (that always contained the optimal value) and for all possible sizes s. Also shown in these tables, are the results obtained for the 10-fold cross validation model selection method. This latter method is perhaps the most widely used—here, it consists of using 10-fold cross validation to find the best stopping point s and the best penalty value p (in the variable p case) on a given training set. Both model selection methods where tested by 10-fold cross validation and compared with the optimal model (taken from table 3 and reported again in table 4). In all cases **BuildSCM** could choose ball centers from examples of either $\mathcal{N}$ or $\mathcal{P}$. Finally, in addition to the error and size (as in the previous tables), we have also reported the standard deviation of the results. These were obtained by computing the standard

Data Set	Type	MS from 10-fold CV			MS from bound		
		size	error	std	size	error	std
BreastW	c	1.6	20	15	1.8	20	15
	d	1.5	27	19	1.3	28	14
Votes	c	1	6	8	1	6	8
	d	1.1	8	8	1.1	8	8
Pima	c	6.1	209	28	18.6	205	28
	d	13	219	29	25.9	215	32
Haberman	c	1.2	85	20	4.3	82	17
	d	5.7	82	25	4.9	72	18
Bupa	c	14.2	132	29	13.9	133	27
	d	8.9	120	10	13	117	19
Glass	c	3.4	38	27	4	45	25
	d	7.9	44	26	6	45	24
Credit	c	12	221	37	22.2	215	31
	d	3.8	207	33	14.9	215	36

Table 5: SCM model-selection results for variable values of penalty p.

deviation of the generalization error (per example) over the 10 different testing sets and then multiplying it by the number of examples in the data set.

Not surprisingly, the results for variable p are almost always better than those when p is fixed to its optimal value. Most importantly, we see that model selection by using our bound is generally as effective as using 10-fold cross validation. Moreover, both model selection methods are always within one standard deviation of the optimal.

7. Conclusion and Outlook

Our theoretical and experimental results indicate that the set covering machine is definitely a good candidate for practical machine learning tasks. Our bound on the generalization error indicates that good generalization is expected whenever we can find a small set covering making few training errors. Moreover, we have seen that our bound is generally as effective as 10-fold cross validation at selecting a good model.

The next important step is to generalize our bound to the case of asymmetrical loss which frequently occurs in practice. Indeed, although our model selection parameters give us the flexibility to control the amount of training errors on the positive examples independently from the training errors on the negative examples, our bound only applies to the case of symmetrical loss.

Apart from the model-selection issue, it is important to investigate several possible extensions of the set covering machine. One possibility would be to use other metrics for the data-dependent balls. Some background knowledge can sometimes be used to find some symmetries that the best metric should have, but all too often we are left with a set of unknown parameters whose best values are unknown. Intuitively, the best metric in a set of metrics is the one which needs the smallest number of balls for the covering. Hence, there is clearly a need to extend our theory to the case where the best metric is chosen from the data.

Another important issue concerns the set of features used by the SCM. Indeed we should not expect that data-dependent balls will always give the best results. We could try also to use data-dependent half-spaces. One possibility would be to construct hyperplanes from triples of examples. The weight vector would be constructed from a pair of positive and negative examples and the third example would be used to identify the threshold. This would give a larger set of features than the set of balls (since each ball is built only from two examples) but a SCM constructed from these half-spaces might be much smaller and, therefore, generalize better. Also note that, in the case of half-spaces, the learning algorithm would need only to be provided with a definition of an inner product between any pair of examples. Hence the learning algorithm would be kernel-based instead of being metric-based as in the case of balls.

Finally there is the possibility of using larger classes of functions than conjunctions and disjunctions. For this research direction, the next step is to consider decision lists (Rivest, 1987, Dhagat and Hellerstein, 1994) of data-dependent features. Indeed, for the same set of features, any conjunction and any disjunction can be expressed as a decision list. But decision lists with the presence of alternations (Dhagat and Hellerstein, 1994) cannot be expressed as a conjunction or a disjunction. However, the training problem for decision lists is not a minimum set cover problem (Dhagat and Hellerstein, 1994) and, consequently, a guarantee similar to Theorem 7 cannot be provided. Nevertheless, we can still use a greedy set cover algorithm to cover the examples of both classes and try to generate an alternating decision list that would be much smaller than a conjunction or a disjunction. That decision list would perhaps generalize better. We could also try to extend further to the strictly larger class of linear threshold functions (of data-dependent features) and use the Winnow algorithm (Littlestone, 1988) which has a good theoretical guarantee when most of the features are irrelevant. We have found however that this algorithm was simply too slow to be used in practice with data-dependent balls. Indeed, despite extensive running times, we have not been able to obtain any reasonably good solutions when using $O(m^2)$ balls for $m = 700$.

Acknowledgments

This work was supported by NSERC grant OGP0122405 and, in part, by the EU under the NeuroCOLT2 Working Group, No EP 27150.

References

Shai Ben-David, Nadav Eiron, and Hans Ulrich Simon. Limitations of learning via embeddings in Euclidean half-spaces. In *Proceedings of 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001,* Amsterdam, The Netherlands, July 2001, volume 2111 of *Lecture Notes in Artificial Intelligence*, pages 385–401. Springer, Berlin, 2001.

Shai Ben-David and A. Litman. Combinatorial variability of Vapnik-Chervonenkis classes. *Discrete Applied Mathematics*, 86:3–25, 1998.

C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases.* Department of Information and Computer Science, Irvine, CA: University of California, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

V. Chvátal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, Cambridge, U.K., 2000.

Aditi Dhagat and Lisa Hellerstein. PAC learning with irrelevant attributes. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 64–74. IEEE Computer Society Press, Los Alamitos, CA, 1994.

Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* Freeman, New York, NY, 1979.

D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36:177–221, 1988.

Ralf Herbrich and Robert C. Williamson. Algorithmic luckiness. *Journal of Machine Learning Research*, 3:175–212, 2002.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, Massachusetts, 1994.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, Santa Cruz, CA, 1986.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

C. Saunders, O. Stitson, J. Weston, L. Bottou, B. Schoelkopf, and A. Smola. Support vector machine reference manual. Technical Report CSD-TR-98-03, Department of Computer Science, Royal Holloway, University of London, London, UK, 1998.

J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44:1926–1940, 1998.

L. G. Valiant. A theory of the learnable. *Communications of the Association of Computing Machinery*, 27(11):1134–1142, November 1984.

Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley, New York, NY, 1998.