

Maximum Margin Coresets for Active and Noise Tolerant Learning

Sariel Har-Peled and Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign
{sariel, danr}@uiuc.edu

Dav Zimak
Yahoo! Inc.
Santa Clara, CA
davzimak@yahoo-inc.com

Abstract

We study the problem of learning large margin half-spaces in various settings using coresets and show that coresets are a widely applicable tool for large margin learning. A large margin coreset is a subset of the input data sufficient for approximating the true maximum margin solution. In this work, we provide a direct algorithm and analysis for constructing large margin coresets. We show various applications including a novel coreset based analysis of large margin active learning and a polynomial time (in the number of input data and the amount of noise) algorithm for agnostic learning in the presence of outlier noise. We also highlight a simple extension to multi-class classification problems and structured output learning.

1 Introduction

Large margin techniques are the basis for both practical algorithms and theoretic analysis in machine learning. Algorithmically, the most notable example is the support vector machine (SVM) [Vap95] that finds a maximum-margin separation of a given data set. The SVM has proven very successful in practice and theoretically, a large margin separation implies good generalization performance [KS94].

The SVM has a simple representation and a straightforward implementation — find the set of *support vectors* that uniquely define the maximum margin separation. Amazingly, this approach simultaneously allows the classifier to be represented with a (potentially small) subset of the input data and, through the use of kernel functions, to utilize an arbitrarily powerful hypothesis space. If a small support set can be found, then one can guarantee high performance on unseen data when using a hypothesis class with unbounded complexity. Unfortunately, there is no guarantee that the size of the support set will be small. Additionally, the running time of the SVM algorithm to find an exact solution to the large margin problem is $O(m^3)$ time and $O(m^2)$ space using m examples and is infeasible for large datasets.

Most practical algorithms, such as chunking [Vap82], decomposition [OFG97], and sequential minimal optimization (SMO) [Pla98], reduce the problem to manageable sub-tasks are heuristic solutions which may converge slowly.

Furthermore, the running time remains crucially dependent on the number of support vectors in the solution. Alternatively, recent work has focused on on-line approaches to approximate the maximum-margin algorithms [Kow00; Gen01; LL02]. On-line algorithms are iterative solutions that add examples to the large margin solution based on various conditions — all related to the relative margin of the example under consideration. As a result, they can bound the number of examples necessary to guarantee a large margin separation.

In this paper we relax the requirement that we find the unique maximum margin separation. Specifically, we find an approximate maximum margin separation — a hyperplane that separates *all* of the input data with margin larger than $(1 - \epsilon)\rho^*$, where ρ^* is best achievable. We use the *coreset* method first described in [BC03] and extended to the maximum margin setting in [TKC05]. A *coreset* for a maximum margin separating hyperplane is a subset, $C \in D$ of examples such that the maximum margin hyperplane on C is an approximate maximum margin separating hyperplane on D . In some sense, it captures all of the necessary information for the approximation just as the set of support vectors does for the true maximum margin separating hyperplane.

This paper studies the running time of a simple coreset algorithm for binary and structured-output classification and the use of the coreset as an analysis tool for active learning and noise-tolerant learning in the agnostic setting. In previous work, the coreset was constructed as a reduction to finding a coreset for a different problem — the minimum enclosing ball [BC03]. In Section 3, we show a direct algorithm for finding a coreset of size at most $|C| = O((R/\rho_*)^2/\epsilon)$ in time $O(nd|C| + |C|T(|C|))$ where R and ρ_* measure the size of the example set and the quality of the maximum-margin classifier and $T(|C|)$ is the time to run an SVM black-box on $|C|$ examples. This improves previous bounds by a factor $1/\epsilon$ and provides the first explicit running time to the algorithm.

In Section 4 we analyze one of the most effective active learning algorithms based on the maximum-margin principal [TK02] and give a running time and a $(1 - \epsilon)$ -approximation guarantee. We show that in time $O(\frac{d|C|}{\epsilon_e} (\ln |C| + \ln \frac{1}{\delta}) + |C|T(|C|))$ one can compute a coreset C of size at most $|C| = O((R/\rho_*)^2/\epsilon^2)$ such that with high confidence, $1 - \delta$, the classifier produced will have large margin and small, ϵ_e , error. In Section 5, we analyze learn-

ing with outlier noise. Roughly speaking, a set of outliers is a small subset of the input data such that, if removed would yield the correct maximum-margin classifier. Thus if we assume there are k outliers, the best maximum margin classifier is well-defined. We show a polynomial time algorithm for learning the maximum margin separation in this setting. Finally, in Section 6 we highlight an important connection between the coresets algorithm and recent work for learning SVM for structured output [THJA04]. Indeed, we can view SVM^{struct} as a coreset algorithm.

2 Preliminaries

We are given $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_M, y_M)\}$, a labeled training set of cardinality M drawn from some distribution $\mathcal{D}_{\mathcal{X}, \mathcal{Y}}$, where $\mathbf{x}_m \in \mathcal{X}$ are the examples in a inner-product space and $y_m \in \mathcal{Y}$ are labels. For most of the paper, we assume a real valued input, $\mathcal{X} = \mathbb{R}^d$, and binary output, $\mathcal{Y} = \{-1, 1\}$. However, in Section 6 we note an important extension to the structured output domain.

In this paper, we use the maximum margin principle to discover a hypothesis $h \in \mathcal{H}$ represented by a halfspace $h(\mathbf{x}) = \arg\max_{y \in \{-1, 1\}} y(\mathbf{w} \cdot \mathbf{x})$, where $\mathbf{w} \in \mathbb{R}^d$. The *binary margin* (or *geometric margin*), $\rho(\mathbf{w}, \mathbf{x}, y) = y(\mathbf{w} \cdot \mathbf{x})$, for an example, \mathbf{x} , is defined as the distance from the example to the discriminating hyperplane $\mathbf{w} \cdot \mathbf{x} = 0$. Notice that a negative margin is indicative of a misclassified example. The margin of hypothesis h is $\rho(h, D) = \min_{(\mathbf{x}, y) \in D} \rho(\mathbf{w}, \mathbf{x}, y)$.

Therefore, given a sample, D , the maximum margin hypothesis (hyperplane) is

$$\mathcal{L}(D) = \arg\max_{\mathbf{w} \in \mathbb{R}^d, \|\mathbf{w}\|=1} \min_{(\mathbf{x}, y) \in D} y(\mathbf{w} \cdot \mathbf{x})$$

is the uniform length hyperplane with maximum margin over the data.

Definition 2.1 ((1 - ϵ)-Approximation) A hypothesis h is a $(1 - \epsilon)$ -approximation to the optimal hypothesis h^* if $\rho(h, D) \geq (1 - \epsilon)\rho(h^*, D)$.

Definition 2.2 (Maximum margin coreset) A maximum margin coreset is a set of examples $C = C(\epsilon, \rho) \subset D$ such that $h = \mathcal{L}(C)$ is a $(1 - \epsilon)$ -approximation to $h^* = \mathcal{L}(D)$.

3 Coreset Learning Algorithm

Large margin coresets were first introduced in [TKC05] to form the Core Vector Machine (CVM). In that work, they reduced finding a maximum margin hyperplane to finding the minimum enclosing ball of a set of points around the origin. For the latter task, there exists a coreset algorithm that runs in time linear in the number of points [BC03]. In this section, we provide very similar results with a slight (factor $1/\epsilon$) improvement by providing a direct algorithm and analysis.

In Figure 1, the simplified coreset algorithm is presented for learning binary labeled data in a noise-free setting. The coreset C is built iteratively. At each step, we construct the true maximum margin classifier, $h_i = \text{SVM}(C)$ and use it to find the example with the smallest (or negative) margin. This example is then added to the coreset and the process repeats.

Algorithm CORESET SVM

INPUT:

$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m))$,
Approximation parameter $\epsilon \in (0, 1)$
where $S \in \{\mathbb{R}^d \times \{-1, 1\}\}^m$

OUTPUT: A classifier $h \in \mathcal{H}$

begin

Set $C = ((\mathbf{x}_1, y_1))$

Set $R = \max_{(\mathbf{x}, y) \in S} \|\mathbf{x}\|^2$

For $i = 1 \dots T$

Set $h_i = \text{SVM}(C)$

Set $\rho_i = \rho(h_i, C)$

$(\mathbf{x}_{\min}, y_{\min}) = \arg\min_{(\mathbf{x}, y) \in S \setminus C} \rho(h_i, \mathbf{x}, y)$

if $\rho(h_i, \mathbf{x}_{\min}, y_{\min}) < (1 - \epsilon)\rho_i$

$C = C \cup (\mathbf{x}_{\min}, y_{\min})$

else

return h_i

Return $\text{SVM}(C)$

end

Figure 1: Maximum-margin learning via coresets.

It is possible to tell that h_i is a $(1 - \epsilon)$ -approximation by observing the ratio between the margin on the coreset, $\rho(h_i, C)$, and the margin on the entire data set, $\rho(h_i, D)$. If this ratio is small enough then the margin of h_i on the entire data set is sufficiently large and the algorithm halts.

Lemma 3.1 Let $\rho^* = \rho(\mathcal{L}(D), D)$ be the optimal margin for data set $D \in \{\mathbb{R}^d \times \{-1, 1\}\}^M$ of size M . Given a parameter, $\epsilon \in (0, 1)$, one can compute a coreset C of size $|C| = O((R/\rho^*)^2/\epsilon)$ in time $O(nd|C| + |C|T(|C|))$, where $R = \max_{(\mathbf{x}, y) \in D} \|\mathbf{x}\|$.

Proof sketch: We show using a simple geometric argument that each time an example is added to the coreset, the margin on the next integration decreases by at least a constant factor. That is,

$$\rho_{i+1} \leq \left(1 - \frac{\alpha_i^2}{8R^2}\right) \rho_i, \quad (1)$$

where $\alpha_i = \rho_i - \rho(h_i, \mathbf{x}_i)$ measures how much the added example violates the current margin guess using the current hypothesis h_i . Then, it can be shown that the decreasing sequence of margins will be smaller than $\rho_i \leq (1 + \epsilon)\rho$ after at most $\frac{64R^2}{\rho^2\epsilon}$ steps. Once the margin is small enough, we show that it quickly decreases and outputs a $(1 - \epsilon)$ -approximation to the maximum margin classifier¹. ■

At each step, the algorithm in Figure 1 adds the example with the smallest margin to the coreset. However, the algorithm is easily modified such that any example with margin small enough can suffice. Specifically, if we know that each example (\mathbf{x}, y) added to the coreset is such that $\rho(h_i, \mathbf{x}, y) < (1 - \epsilon)\rho_i$, but is not necessarily the example with minimum margin, the algorithm still converges, but with a larger coreset.

¹See appendix for full proof.

Corollary 3.2 Let $\rho_* = \rho(\mathcal{L}(D), D)$ be the optimal margin for data set $D \in \{\mathbb{R}^d \times \{-1, 1\}\}^M$ of size M . Given a parameter, $\epsilon \in (0, 1)$, one can compute a coreset C of size $|C| = O((R/\rho_*)^2/\epsilon^2)$ in time $O(nd|C| + |C|T(|C|))$, where $R = \max_{(x,y) \in D} \|x\|$.

Proof sketch: If, rather than add the minimum margin example at each step, we add an example with small enough margin ($\rho(h_i, \mathbf{x}, \mathbf{y}) < \rho_i \epsilon$) then according to Equation 2,

$$\rho_{i+1} \leq \left(1 - \rho_i \frac{\epsilon^2}{8R^2}\right) \rho_i.$$

By plugging this into Lemma A.1, the result follows. ■

3.1 Related Work

Central to the coreset algorithms presented here, is the idea of iteratively building a *working set* of examples by carefully selecting examples to add at each step. In the online learning literature, this idea has appeared in work related to the coreset approach.

Indeed, even the perceptron algorithm can be viewed as building a working set. At each iteration, an example, (\mathbf{x}_i, y_i) , is added to the working set if $y_i(\mathbf{w}_i \cdot \mathbf{x}_i) < 0$. The hypothesis is a linear sum of elements in the set.

Various approximate algorithms for online learning have also been proposed. In [Kow00], Kowalczyk proposed a perceptron-like update rule, with various criteria for choosing which example to update. One of them is exactly the minimum margin approach used in coresets. After only $O(\frac{R^2}{\epsilon^2} \log \frac{R}{\epsilon})$ updates, the algorithm would converge to a $(1 - \epsilon)$ -approximate classifier — a result very similar to ours, modulo $1/\epsilon$ and $\log R$ terms. At roughly the same as Kowalczyk’s algorithm, two additional algorithms were proposed: the Relaxed Online Maximum Margin Algorithm (ROMMA) and the Approximate Large Margin Algorithm (ALMA_p). ROMMA is an online algorithm that learns a $(1 - \epsilon)$ -approximate maximum margin separation. Both have similar selection criteria, and perform similarly in practice. ALMA_p also provides a mistake bound of $O(\frac{R^2}{\epsilon^2 \rho^2})$.

Recently a new algorithm, SVM-Perf, was introduced and implemented [Joa06] with similar $O(\frac{1}{\epsilon^2})$ bounds and many experimental results. In addition to providing a real-world application, this work helps support our claim that coreset-based algorithm can be practical.

4 Maximum Margin Active Learning

In active learning, the learner is presented with a set of unlabeled data, $U = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ and an oracle, $\text{ORACLE} : \mathcal{X} \rightarrow \{-1, 1\}$ that provides a label to any example $\mathbf{x} \in U$ consistent with a large margin hypothesis. The goal is to learn exactly this maximum margin separation using a limited number of oracle queries.

Recently, iterative algorithms for active learning SVM have been proposed [TK02; CCS00]. After presenting a slight modification of these algorithms using coresets and introducing an explicit stopping criteria, we show that the algorithm converges quickly to a $(1 - \epsilon)$ -approximation of the true maximum margin hypothesis that exists given all labels (with high probability).

4.1 Coreset Active Learning Algorithm

In Figure 2, the active learning algorithm from [TK02] is adapted by adding a verification stage. The algorithm runs in iterations, where at each step, the unlabeled example that is closest to the decision boundary is added to the coreset. However, if there are no examples near the decision boundary (i.e. they are further than the current large-margin guess), we may think that all labels are classified correctly and thus the algorithm can halt. Of course, since the labels are unknown, it is possible that there are still a large number of misclassified examples. At this point the algorithm enters a verification phase, $\text{VERIFY}()$, where examples are sampled uniformly at random from the entire data set according to $\text{UNIFORMRANDOM}()$ and labeled using $\text{ORACLE}()$.

It is important to note that the algorithm presented in Figure 2 repeatedly cycles over the unlabeled dataset to find the single example closest to the decision boundary. This is easily modified to two important cases when the number of examples is very large (i.e. $m \gg |C|$) or when there is an infinite stream of examples (i.e. $m = \infty$). In these cases, any example, \mathbf{x} where $|\rho(h_i, \mathbf{x}, y)| < (1 - \epsilon)\rho_i$ can be added to the coreset and the algorithm can halt after enough examples are seen without making a mistake. Indeed Lemma 4.2 below applies to these more general cases.

4.2 Analysis

Algorithm 2 seeks the true maximum margin hypothesis of the data that would be found if all of the labels were known. Here, we show that with high probability $(1 - \delta)$, it finds a $(1 - \epsilon_a)$ -approximation to this hypothesis. Unfortunately, it is impossible to guarantee error free learning (see Section 4.3), so we must accept a small, ϵ_e , prediction error.

The analysis follows from two facts. First, there exists a coreset that can be constructed by adding examples that lie close to the decision boundary. Any example with very small margin ($< (1 - \epsilon_a)\rho_i$) will improve the approximation irrespective of the actual label. Second, the verification stage will halt either because it has found an example with very small (i.e. negative) margin that will improve the coreset or because enough examples have been seen with no mistakes. In the latter case, we can apply the following lemma, adapted from [KLPV87; Ang87] that shows learning from membership queries can be used to give PAC-bounds².

Lemma 4.1 *If L is a conservative on-line algorithm with mistake bound M and access to an example oracle $\text{ORACLE}()$ drawing examples i.i.d. from distribution \mathcal{D} , then after at most $\frac{M}{\epsilon} (\ln M + \ln \frac{1}{\delta})$ calls to $\text{ORACLE}()$, with confidence $1 - \delta$, L produces a hypothesis with expected error less than ϵ on examples drawn from \mathcal{D} .*

Using Lemma 4.1 and Corollary 3.2, we bound the running time of this algorithm to converge to an approximate solution.

Lemma 4.2 *Let $D = \text{ORACLE}(U)$, be the entire labeled data set and $\rho_* = \rho(\mathcal{L}(D), D)$ be the optimal margin for data set D of size M . Given parameters, $\delta, \epsilon_e, \epsilon_a \in [0, 1]$,*

²See appendix for full proof.

```

Algorithm ACTIVE CORESET SVM
INPUT: Data  $U = (\mathbf{x}_1, \dots, \mathbf{x}_m) \in \{\mathbb{R}^d\}^m$ 
OUTPUT: A classifier  $h' : \mathbb{R}^d \rightarrow \{-1, 1\}$ 
begin
  Set  $C = \emptyset$ 
  Repeat until Halt
    Set  $h_i = \text{SVM}(C)$ 
    Set  $\rho_i = \rho(h_i, C)$ 
     $\mathbf{x}_{\min} = \text{argmin}_{\mathbf{x} \in U \setminus C} \rho(h_i, \mathbf{x}, y)$ 
    if  $\min_{y \in \{-1, 1\}} |\rho(h_i, \mathbf{x}_{\min}, y)| < (1 - \epsilon)\rho_i$ 
       $y_{\min} = \text{ORACLE}(\mathbf{x}_{\min})$ 
       $C = C \cup (\mathbf{x}_{\min}, y_{\min})$ 
    else
       $(\mathbf{x}_v, y_v) = \text{VERIFY}(U \setminus C, h_i)$ 
      if  $\mathbf{x}_v = \text{NULL}$ 
        return  $h_i$  and Halt
      else
         $C = C \cup (\mathbf{x}_v, y_v)$ 
    else
      Return SVM( $C$ ) and Halt
end

```

(a) ACTIVE CORESET SVM

```

Algorithm VERIFY
INPUT:
  Data  $U = (\mathbf{x}_1, \dots, \mathbf{x}_m) \in \{\mathbb{R}^d\}^m$ 
  A classifier  $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ 
OUTPUT:
   $(\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, 1\}$  or NULL
begin
  for  $i = 1 \dots T$  do
     $\mathbf{x} = \text{UNIFORMRANDOM}(U)$ 
     $y = \text{ORACLE}(\mathbf{x})$ 
    if  $\rho(h, \mathbf{x}, y) < 0$ 
      Return  $(\mathbf{x}, y)$ 
  Return NULL
end

```

(b) VERIFY

Figure 2: (a) Active learning using coresets. Abusing notation, $U \setminus C = (\mathbf{x} \in U | (\mathbf{x}, \text{ORACLE}(\mathbf{x})) \notin C)$. (b) Verify procedure. $\text{UNIFORMRANDOM}(U)$ returns a random example from the unlabeled set. $\text{ORACLE}(\mathbf{x})$ returns the correct label for \mathbf{x} and $\text{UNIFORMRANDOM}(U)$ selects an example from U uniformly at random.

one can compute a coreset C of size at most $O((R/\rho_*)^2/\epsilon_a^2)$ in time $O(\frac{|C|}{\epsilon_e} (\ln |C| + \ln \frac{1}{\delta}) + |C|T(|C|))$, where $R = \max_{(\mathbf{x}, y) \in D} \|\mathbf{x}\|$. The total number of calls to $\text{ORACLE}()$ is less than $O(\frac{|C|}{\epsilon_e} (\ln |C| + \ln \frac{1}{\delta}))$, and $T(m)$ is the running time of the SVM for m examples.

Proof: The coreset will be at most $|C| = O((R/\rho_*)^2/\epsilon_a^2)$ as a result of Corollary 3.2 by noticing that each time an example, (\mathbf{x}, y) is added to the coreset, $\rho(h_i, \mathbf{x}, y) < (1 - \epsilon)\rho_i$ — either because an unlabeled example is added in Line (1)

in Algorithm 2 (a) or because a labeled example is added in Line (2) in Algorithm 2 (b). In the former we know that $\rho_i - |\rho(h_i, \mathbf{x}, y)| > \epsilon\rho_i$, and in the latter we know that $\rho(h_i, \mathbf{x}, y) < 0$.

Since we know that at most $|C|$ examples will be added to the coreset, and at each iteration, the margin of the current working hypothesis decreases we can use Lemma 4.1 to bound the total number of Oracle queries in $\frac{1}{\epsilon_e} (\ln |C| + \ln \frac{1}{\delta})$ to ensure $1 - \delta$ confidence that the classifier has at most ϵ_e mistakes. Therefore, the total number of calls to $\text{ORACLE}()$ is at most $O(|C| + |C|\frac{1}{\epsilon_e} (\ln |C| + \ln \frac{1}{\delta})) = O(|C|\frac{1}{\epsilon_e} (\ln |C| + \ln \frac{1}{\delta}))$. The running time follows since SVM must be run each time an example is added to the coreset. ■

4.3 Related Work

Previously, the efficacy of maximum margin active learning algorithms were explained because by choosing the example closest to the decision boundary, the version space will be approximately halved [TK02]³. More precisely, it was argued that because at each iteration the version space is an intersection of half-spaces in the kernelized feature space. If we assume that each example is of constant size (i.e. $\|\mathbf{x}\| = 1$) then the hypothesis with maximum margin separation is a point in the version space at the center of the largest enclosed ball in this polytope. Therefore, by choosing an example with small margin, it is hoped that it comes close to bisecting the enclosed ball and also the version space.

If one could guarantee that the version space was indeed halved at each iteration, then the algorithm would converge quickly to the true maximum margin hypothesis [TK02; FS97] Unfortunately, no such guarantee can be made, either in practice or in theory, thus the “halving” argument falls short to adequately explain the practical success of choosing the minimum absolute margin example at each iteration.

In addition, as first presented in [Das05] a zero-error active learning algorithm is impossible without requesting the label of *all* examples. To see this, we consider a sample of examples spread at a constant interval on the surface of a circle in \mathbb{R}^2 . See Figure 3 for an illustration. The concept represented by Figure 3(a) is one where a single example is negative and the rest are positive. The maximum margin separation thus separates a single example from the rest. Consider any algorithm that computes the maximum margin separation of any labeled subset of this data. Unless the single negative example is included in the labeled subset, then there is no hope of achieving an approximate large-margin separation that correctly classifies all examples in the data set. Therefore, if an adversary controls the oracle, by simply answering “+” to every query until the final query, the learner is forced to ask the label of every example.

³In [TK02] it is assumed that $\|\mathbf{x}\| = 1$ for all examples. While relaxing this assumption does cause a different view of the versions space and changes the motivation, it does not affect their results. Indeed, here, we propose an alternate justification that does not depend on the version space.

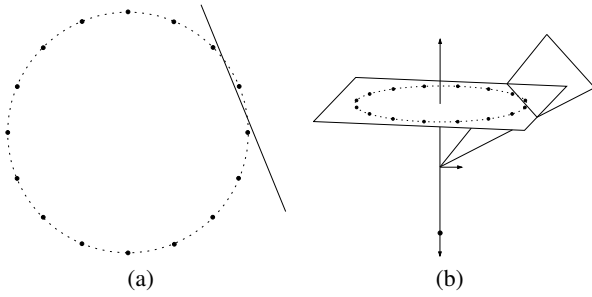


Figure 3: Impossibility of Zero-Error Approximation: (a) Active learning nightmare – all examples are positive, with a single negative. (b) Nightmare in 3D – 2 negative points, one below origin, a second on the plane of the circle.

5 Agnostic Learning with Outlier Noise

Learning in the presence of noise is of great interest. While there are many definitions of noise, such as attribute, label, and malicious noise, we consider a very general model, *outlier noise*. One can think of outlier noise in the following way: without the “noisy” examples, a “clean” function could be learned. Thus if the noisy examples could be identified a priori, we could learn the true maximum margin classifier. In some sense, all types of noise can be viewed as outlier noise, so the analysis presented in this section can be widely applied. We show a polynomial time algorithm for learning an approximate maximum margin hyperplane in the presence of outliers.

Definition 5.1 (Outlier Set) Consider a data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^M$ of binary ($y \in \{-1, 1\}$) examples. For any set of outliers, $V \subseteq D$, we can consider the maximum margin hyperplane, $h^{D'} = \mathcal{L}(D')$, on the examples $D' = D \setminus V$. Then an *outlier set* of size k is a subset, V_k , of size k that achieves maximum margin on the remaining data

$$V_k = \operatorname{argmax}_{V \in D, |V|=k} \rho(\mathcal{L}(D \setminus V), D \setminus V).$$

Because a coresset exists for the “clean” data and we know that there are at most k (or a fixed fraction) outliers, we avoid exponential running time by subsampling based on the expected coresset size. This provides an extremely simple polynomial-time algorithm for learning with noise.

Lemma 5.2 Let $\rho_k = \rho(\mathcal{L}(D \setminus V_k), D \setminus V_k)$ be the optimal margin for data set D of size M with k outliers. Given a parameter, ϵ , one can compute a separating hyperplane with margin $(1 - \epsilon)\rho_k$ on $D \setminus V_k$ in polynomial time $O(dT(c)M^{c+1} \log M)$, where $c = O((R/\rho_k)^2/\epsilon)$. and $R = \max_{(\mathbf{x}, y) \in D} \|\mathbf{x}\|$.

*Proof sketch.*⁴ Because the clean data, $D \setminus V_k$, contains a coresset of size $c = O((R/\rho_k)^2/\epsilon)$ from Lemma 3.1, it suffices to find this set and observe that there are at most k outliers. It is possible to examine *all* $\binom{M}{c}$ subsets of the input

⁴Due to space constraints, proof to appear in final version.

Algorithm OUTLIER SVM

```

INPUT:
  Data  $D = ((\mathbf{x}_m, y_m))_1^M \in \{\mathbb{R}^d \times \{-1, 1\}\}^M$ 
OUTPUT:
  A classifier  $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ 
begin
  Set  $R = \max_{(\mathbf{x}, y) \in D} \|\mathbf{x}\|$ 
  For  $i = 0, 1, 2, \dots$ 
     $\rho = R/2^i$ 
    Set  $c = \lceil 32 \frac{R^2}{\epsilon \rho^2} \rceil$ 
    For each subset  $D_s \in \binom{D}{c}$ 
       $h_s = \text{SVM}(D_s)$ 
       $h_{\max} = \operatorname{argmax}_s \rho_k(h_s, D)$ 
    if  $\rho_k(h_{\max}, D) > \rho$ 
      Return  $h_{\max}$  and Halt.
end

```

(a) OUTLIER SVM

Algorithm SIMPLE OUTLIER SVM

```

INPUT:
  Data  $D = ((\mathbf{x}_m, y_m))_1^M$ 
OUTPUT:
  A classifier  $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ 
begin
  Set  $R = \max_{(\mathbf{x}, y) \in D} \|\mathbf{x}\|$ 
  For  $i = 0, 1, 2, \dots$ 
    Set  $c = 2^i$ 
    For each subset  $D_s \in \binom{D}{c}$ 
       $h_s = \text{SVM}(D_s)$ 
       $h_{\max} = \operatorname{argmax}_s \rho_k(h_s, D)$ 
    if  $\rho_k(h_{\max}, D) > \sqrt{32 \frac{R^2}{c\epsilon}}$ 
      Return  $h_{\max}$  and Halt.
end

```

(b) SIMPLE OUTLIER SVM

Figure 4: (a) Approximate maximum-margin learning with outlier noise. $\rho_k(h, D)$ returns the margin of the example in D that is smaller than the margin of all but k examples (i.e. the $k + 1$ -th smallest margin). (b) Simple algorithm. An alternate description of the algorithm that simply doubles the size of the sub-sample sets at each iteration.

data. For each one, create the maximum margin hypothesis using an SVM black-box and measure the margin obtained by removing the k examples with minimum margin (or negative margin). Then, one of these hypotheses will have the maximum margin and have at most k outliers. Finally, since the margin is unknown a-priori, we must repeat the above procedure for exponentially decreasing guesses and stop once we see only k outliers with margin as large as the guess. ■

6 Structured Output Learning

Structured output learning is one of the most important new areas in machine learning. Structured output can be sequences, trees, rankings and general structures and are ubiq-

uitous in important applications in areas from NLP to web search to biology. Recently machine learning approaches have begun to address these problems. Here, we show a common modeling approach for structured output learning and highlight the connection to standard maximum margin learning.

Structured classifiers produce complex, structured output $\mathcal{Y} = \mathcal{Y}^1 \times \dots \times \mathcal{Y}^L$, where $\mathcal{Y}^l \in \{1, \dots, K\}$. It is common to write the decision rule as

$$h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}' \in \mathcal{Y}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}'),$$

where $\Phi(\mathbf{x}, \mathbf{y})$ represents features of each (example, label) pair. Therefore, the maximum margin hypothesis (hyperplane) is

$$\mathcal{L}^S(D) = \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^D, \|\mathbf{w}\|=1} \min_{(\mathbf{x}, \mathbf{y}) \in D} \rho(\mathbf{w}, \mathbf{x}, \mathbf{y}),$$

where

$$\rho(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \min_{\mathbf{y}' \neq \mathbf{y}} \mathbf{w} \cdot (\Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \mathbf{y}'))$$

As a result, the algorithms and analysis presented in this paper extend to the structured output setting.

6.1 Related Work

Indeed, the SVM was recently extended to the structured output domain. SVM^{struct} [THJA04] is an algorithm that runs in iterations, each time adding $(\mathbf{x}, \mathbf{y}, \mathbf{y}')$ -triples to the working set. Indeed, they use exactly the same $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{y}')$ feature vector used here. Then, the working hypothesis is updated (in the dual) by optimizing over the Lagrange multipliers, similar to the Sequential Minimal Optimization (SMO) procedure introduced by Platt [Pla98]. They also show a bound of $O(\frac{R^2}{\epsilon^2 \rho^2})$ on the size of the working set.

7 Conclusions

In this paper, we give a simple coresets algorithm with an improved bound on the coreset size. We are mainly concerned with running time analysis of various algorithms. Using coresets, we give bounds for maximum margin active learning and structured output learning. We also formulate a novel and polynomial time algorithm for learning in the agnostic (noisy) setting. Coresets are a very general tool in approximation algorithms and we have shown that they have important uses in maximum margin learning and analysis. We think that this is the tip of the iceberg, and envision that coresets will find many more applications in machine learning.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

M. Badoiu and K.L. Clarkson. Smaller core-sets for balls. In *SODA*, pages 801–802, 2003.

C. Campbell, N. Cristianini, and A.J. Smola. Query learning with large margin classifiers. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 111–118, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

S. Dasgupta. Analysis of a greedy active learning strategy. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 337–344. MIT Press, Cambridge, MA, 2005.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. First appeared in EuroCOLT '95.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

T. Joachims. Training linear svms in linear time. In L. Ungar, M. Craven, D. Gunopulos, and T. Eliassi-Rad, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2006)*, New York, 2006. The Association for Computing Machinery.

M. Kearns, M. Li, L. Pitt, and L. Valiant. Recent results on boolean concept learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 337–352, Irvine, California, 1987. (published by Morgan Kaufmann, Los Altos California).

A. Kowalczyk. Maximal margin perceptron. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge MA, 2000.

M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. In Stephen J. Hanson, George A. Drastal, and Ronald L. Rivest, editors, *Computational Learning Theory and Natural Learning Systems, Constraints and Prospect*, volume 1. The MIT Press, 1994.

Y. Li and P.M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387, 2002.

E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines, 1997.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 104, New York, NY, USA, 2004. ACM Press.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.

I.W. Tsang, J.T. Kwok, and P. Cheung. Core vector machines: Fast svm training on very large data sets. *JMLR 2005*, 6(Apr):363–392, 2005.

V.N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Series in Statistics. Springer, New York, 1982.

V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, DE, 1995.

A Appendix: Full Proofs

A.1 Full Proof of Lemma 3.1

Before presenting the proof that Algorithm 1 produces an approximate maximum margin classifier, we present a simple property of a special decreasing sequence.

Lemma A.1 *Let a_0, a_1, \dots, a_n be a sequence and $\epsilon > 0$ such that $0 < a_{i+1} \leq (1 - ca_i)a_i$, where c is a constant such that $0 < ca_0 < 1$, and $a_0 > 0$. Then, $a_{i+1} \leq a_i$, and $a_i < \epsilon$ for $i \geq \frac{8}{c\epsilon}$.*

Proof: First, $ca_0 \in (0, 1)$ implies that $ca_i \in (0, 1)$, thus $(1 - ca_i) \in (0, 1)$ and $a_{i+1} \leq a_i$. Also, $a_i \rightarrow 0$ as $i \rightarrow \infty$ because otherwise there would exist some $\epsilon > 0$ such that $a_i > \epsilon$ always, and then $a_{i+1} \leq (1 - c\epsilon)^i a_0$, which converges to 0.

More precisely,

$$a_{i+\mu} \leq (1 - ca_{i+\mu})^\mu a_i \leq e^{-(ca_{i+\mu})\mu} a_i \leq 2^{-(ca_{i+\mu})\mu} a_i$$

for any $\mu \geq 1$, since $(1 - x) \leq e^{-x}$ for all $x \in (0, 1)$.

For $\mu \geq 1/ca_{i+\mu}$, we have that $a_{i+\mu} \leq a_i/2$. There is some μ when $a_{i+\mu-1} > a_i/2$ and $a_{i+\mu} \leq a_i/2$. This $(\mu-1)$ -th step occurs after at most $1/ca_{i+\mu-1} \leq 2/ca_i$ steps (i.e., at most $\mu \geq 2/ca_i + 1$ steps).

We want to know how long until the series decreases to ϵ . Let M_j be the number of steps for the series to decrease from $a_0/2^j$ to $a_0/2^{j+1}$. From above, $M_j \leq 2^j/ca_0 + 1$. After a_0 is halved $\log(a_0/\epsilon)$ times, the series will be less than ϵ . This happens after

$$\begin{aligned} \sum_{j=0}^{\lceil \log(a_0/\epsilon) \rceil} M_j &\leq \sum_{j=0}^{\lceil \log(a_0/\epsilon) \rceil} \left(\frac{2^j}{ca_0} + 1 \right) \\ &\leq \frac{1}{ca_0} \left(\sum_{j=0}^{\lceil \log(a_0/\epsilon) \rceil} 2^j + 1 \right) \\ &\leq \frac{1}{ca_0} \sum_{j=0}^{\lceil \log(a_0/\epsilon) \rceil + 1} 2^j \\ &= \frac{1}{ca_0} \left(8 \frac{a_0}{\epsilon} \right) = \frac{8}{c\epsilon} \end{aligned}$$

steps. \blacksquare

The above property is used to determine the rate of convergence of Algorithm 1 by showing how much the margin must decrease after each example is added. The proof follows simple geometric arguments. In the following, we consider the binary case.

Proof:[Proof of Lemma 3.1] Let $C^0 = \{(\mathbf{x}_0, \mathbf{y}_0)\}$ consist of an arbitrary example $(\mathbf{x}_0, \mathbf{y}_0) \in D$. The algorithm proceeds in iterations, where in the i -th iteration, an example (with low margin) is added to the working set C^i to form C^{i+1} . Let $h^i = \mathcal{L}^{01}(C^i)$ and $\rho_i = \rho(h^i, C^i)$ be the maximum margin hypothesis and the margin, respectively, on the working set for all i .

The algorithm then finds the example $(\mathbf{x}_i, \mathbf{y}_i) \in D \setminus C^i$, with smallest (perhaps negative) margin respect to the current working hypothesis. Specifically, we require that

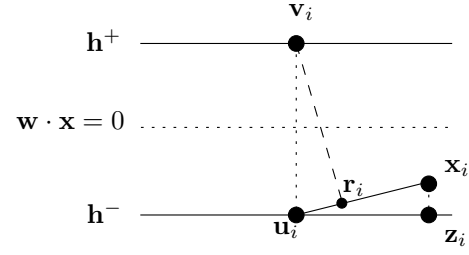


Figure 5: Proof illustration.

$\rho(h^i, \mathbf{x}_i, \mathbf{y}_i) \leq \rho(h^i, \mathbf{x}', \mathbf{y}')$ for all $(\mathbf{x}', \mathbf{y}') \in D \setminus C^i$. If the smallest margin is large enough, $\rho(h^i, \mathbf{x}_i, \mathbf{y}_i) \geq \rho_i - \epsilon\rho_i$, then the algorithm halts, and outputs coreset C^i and hypothesis h^i . In this case $\rho(h^i, \mathbf{x}, \mathbf{y}) \geq \rho_i - \epsilon\rho_i \geq (1 - \epsilon)\rho$, and h^i is an $(1 - \epsilon)$ -approximate maximum margin hyperplane.

On the other hand, if there is an example with small enough margin, then the margin on the next iteration will decrease by a significant amount.

This follows from a simple geometric argument. First, we set up some notation (see Figure A.1). Let $\mathbf{v}_i, \mathbf{u}_i$ be the two closest points on the convex-hull of the positive and negative examples in C^i , respectively. Without loss of generality, assume that \mathbf{x}_i is a negative example. Let $\mathbf{w}^i \cdot \mathbf{x} + b = 0$, where $b \in \mathbb{R}$, be the decision boundary, h^+ and h^- be the hyperplanes parallel to the decision boundary passing through \mathbf{v}_i and \mathbf{u}_i respectively (called the positive and negative boundaries)⁵ Let \mathbf{z}_i be the projection of \mathbf{x}_i onto the negative boundary and $\alpha_i = \|\mathbf{x}_i \mathbf{z}_i\|$ be the distance from \mathbf{x}_i to the negative boundary. Let \mathbf{r}_i be the closest point on the line spanning $\mathbf{u}_i \mathbf{x}_i$ to \mathbf{v}_i .

It is easy to verify that $\Delta \mathbf{v}_i \mathbf{r}_i \mathbf{u}_i$ is similar to $\Delta \mathbf{u}_i \mathbf{z}_i \mathbf{x}_i$. Furthermore, $\|\mathbf{z}_i \mathbf{u}_i\| \leq \|\mathbf{x}_i \mathbf{u}_i\| \leq 2R$. Now, we have

$$\begin{aligned} 2\rho_{i+1} &\leq \text{dist}(\mathbf{v}_i, \mathbf{x}_i \mathbf{u}_i) = \|\mathbf{v}_i \mathbf{r}_i\| = \frac{\|\mathbf{v}_i \mathbf{u}_i\|}{\|\mathbf{u}_i \mathbf{x}_i\|} \|\mathbf{u}_i \mathbf{z}_i\| \\ &= \frac{\|\mathbf{v}_i \mathbf{u}_i\|}{\|\mathbf{u}_i \mathbf{x}_i\|} \sqrt{\|\mathbf{u}_i \mathbf{x}_i\|^2 - \|\mathbf{x}_i \mathbf{z}_i\|^2} \\ &= \|\mathbf{v}_i \mathbf{u}_i\| \sqrt{1 - \frac{\|\mathbf{x}_i \mathbf{z}_i\|^2}{\|\mathbf{u}_i \mathbf{x}_i\|^2}} \\ &= 2\rho_i \sqrt{1 - \frac{\alpha_i^2}{\|\mathbf{u}_i \mathbf{x}_i\|^2}} \\ &\leq 2\rho_i \sqrt{1 - \frac{\alpha_i^2}{4R^2}} \leq 2\rho_i \left(1 - \frac{\alpha_i^2}{8R^2} \right) \end{aligned}$$

Thus,

$$\rho_i - \rho_{i+1} \geq \frac{\alpha_i^2}{8R^2} \rho_i. \quad (2)$$

Another observation, is that $\rho_i - \alpha_i \leq \rho$, since otherwise, we would have a separating hyperplane of margin larger than

⁵For the purposes of the proof, it is more convenient to consider hypotheses of the form $\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ to ensure that the margin corresponds to the Euclidean distance between the convex hulls defined by the positive and negative point sets. This is easily extended to the $\text{sign}(\mathbf{w} \cdot \mathbf{x})$ setting.

ρ . This implies, that $\alpha_i \geq \rho_i - \rho$. Setting $\rho_i = (1 + \epsilon_i)\rho$, we get that $\alpha_i \geq \epsilon_i\rho$. Substituting into the previous equation,

$$\begin{aligned}\rho_i - \rho_{i+1} &\geq \frac{\alpha_i^2}{8R^2}\rho_i \\ (1 + \epsilon_i)\rho - (1 + \epsilon_{i+1})\rho &\geq \frac{\epsilon_i^2\rho^2}{8R^2}(1 + \epsilon_i)\rho \\ \epsilon_i - \epsilon_{i+1} &\geq \epsilon_i^2\frac{\rho^2}{8R^2} + \epsilon_i^3\frac{\rho^2}{8R^2} \\ \epsilon_{i+1} &\leq \epsilon_i - \epsilon_i^2\frac{\rho^2}{8R^2} - \epsilon_i^3\frac{\rho^2}{8R^2}.\end{aligned}$$

Thus, we have that

$$\epsilon_{i+1} \leq \left(1 - \epsilon_i\frac{\rho^2}{8R^2}\right)\epsilon_i.$$

Using Lemma A.1, after $\frac{64R^2}{\rho^2\epsilon}$ iterations, $\epsilon_i \leq \epsilon$ and $\rho_i \leq (1 + \epsilon)\rho$.

Finally, we show (by contradiction) that this state (where $\rho_i \leq (1 + \epsilon)\rho$) for only a limited number of steps before the algorithm halts. Recall that $\alpha_i \geq \epsilon\rho_i \geq \epsilon\rho$, so after the above number of iterations (i.e. $i > 16R^2/(\rho^2\epsilon)$), we have from Equation 2 that $\rho_{i+1} \leq (1 - \frac{\alpha_i^2}{8R^2})\rho_i \leq (1 - \frac{\epsilon\rho^2}{8R^2})\rho_i$. Therefore, setting $\mu = \frac{8R^2}{\epsilon\rho^2}$,

$$\begin{aligned}\rho_{i+\lceil 2\mu \rceil} &\leq \left(1 - \frac{1}{\mu}\right)^{\lceil 2\mu \rceil} \rho_i \leq (1 - \epsilon)\rho_i \\ &< (1 - \epsilon)(1 + \epsilon)\rho = (1 - \epsilon^2)\rho \leq \rho,\end{aligned}$$

where we have used the fact that $(1 - \epsilon/\mu)^\mu \leq (e^{-\epsilon/\mu})^\mu = e^{-\epsilon} \leq (1 - \epsilon/2)$ for $\epsilon, \mu \in (0, 1)$ and that $\rho_i \leq (1 + \epsilon)\rho$. The last inequality is a contradiction, since $\rho_i \geq \rho$ for all i . ■

It is easy to verify that Lemma 3.1 holds not only for hypotheses of the form $h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$, but also for those where $b = 0$ ($h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$). First, construct a data set by reflecting all points through the origin – $D^R = \{(\mathbf{x}, y), (-\mathbf{x}, -y) | (\mathbf{x}, y) \in D\}$ for binary data where $y \in \{-1, 1\}$ and by using the Kesler construction for more complex output problems. Then, the maximum margin hypothesis $h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ over D^R passes through the origin and $b = 0$. Also, each example in the coreset C^R constructed for D^R is associated with an example in D which is in the coreset C for D . Therefore the coreset algorithm run on D^R will produce a coreset C for D with no bias term.

A.2 Full Proof of Lemma 4.1

Proof: The algorithm works in at most M stages, drawing $Q = \frac{1}{\epsilon}(\ln M + \ln \frac{1}{\delta})$ examples from ORACLE() at each stage. If, at some stage, L makes no mistakes, the current hypothesis is output and the algorithm halts.

To bound the probability of error, we have to observe the probability that the algorithm halted with a “bad” hypothesis that has error larger than ϵ . The probability that the bad hypothesis correctly classifies Q *i.i.d.* examples, given its’ error is larger than ϵ is less than $(1 - \epsilon)^Q$. Thus, the probability that the algorithm halted at all, given its’ error is larger than ϵ is less than $M(1 - \epsilon)^Q$. We want this probability to be at most

δ . Thus, $M(1 - \epsilon)^Q \leq Me^{-\epsilon Q} \leq \delta$, and after some algebra, $Q \geq \frac{1}{\epsilon}(\ln M + \ln \frac{1}{\delta})$. Finally, the algorithm will draw at most $\frac{M}{\epsilon}(\ln M + \ln \frac{1}{\delta})$, examples from ORACLE(). ■

A.3 Full Proof of Lemma 5.2

Proof: If we knew V_k a priori, then we could use Algorithm 1 to find a $(1 - \epsilon)$ -approximate hyperplane, $h_k^* = \mathcal{L}^{01}(D \setminus V_k)$ with margin $\rho_k^* = \rho(h_k^*, D \setminus V_k)$. Thus, from Lemma 3.1, we know there exists a coreset of size $c_k^* \leq 128(R/\rho_k^*)^2/\epsilon$ that produces a $(1 - \epsilon)$ -approximation to h_k^* . Therefore, if c_k^* was known, as we will see below, by sampling all subsets of size c_k^* from D , we can discover at least one $(1 - \epsilon)$ -approximation. Indeed, for any $c > c_k^*$ this approach works.

First, we describe a procedure where we assume some fixed $\rho \in (0, 2R)$ and $c = 128(R/\rho)^2/\epsilon$. Then, we can compute a set of hyperplanes by considering all subsets of the data D of size c . There are $\binom{M}{c} \leq M^c$ such example sets, $\{D_s\}_{s=1}^S$. For each D_s , we can define $h_s = \text{SVM}(D_s)$ as the maximum margin separating (or non-separating) hyperplane on data set D_s . In addition, we define $\rho_s = \rho_k(h_s, D)$ to be the margin of h_s on the original data D after removing the k smallest margin examples from D according to h_s . For each subset s , finding the max margin classifier takes at most $O(cd^2c)$ and finding ρ_s takes $O(dM \log M)$ time to classify and sort all examples. Then, $h_{\max} = \text{argmax}_s \rho_k(h_s, D)$ is the output of this procedure and the guess of an approximate maximum-margin classifier. Therefore, given ρ and c , this procedure takes $O(cd^2 2^c M^{c+1} \log M)$.

If, in the above procedure, $\rho \leq \rho_k^*$ (and $c \geq c_k^*$), then we are guaranteed that at least one of h_s is a $(1 - \epsilon)$ -approximation to $D \setminus V_k$ since one of the subsets will contain a coreset and thus produce an approximate classifier. However ρ_k^* is unknown a priori, and must be discovered by the algorithm. By starting with an initial estimate, $\rho = 2R$, and repeating the above procedure for decreasing values of ρ , eventually it will produce approximate classifier. Specifically, we know to halt when the classifier with the largest margin (minus the k potential outliers) is larger than the current estimate, i.e. when $\rho_{\max} = \max_s \rho_s \geq \rho$, then we are guaranteed that $\rho_k^* \geq \rho$ since we know that $\rho_k^* \geq \rho_{\max}$ always. Thus, the estimated coreset size, $c \geq c_k^*$, and h_{\max} is guaranteed to be a $(1 - \epsilon)$ -approximation.

By decreasing ρ exponentially fast, the algorithm will converge quickly and the running time is dominated by the final iteration. Specifically, if, $\rho = 2R/2^i$ in round i , then the procedure is guaranteed to halt after at most $\lceil \log(2R/\rho_k^*) \rceil$ rounds and the total running time is

$$\sum_{i=0}^{\lceil \log(2R/\rho_k^*) \rceil} O(c_i d^2 2^{c_i} M^{c_i+1} \log M) = O(c_I d^2 2^{c_I} M^{c_I+1} \log M)$$

where $c_i = 128(\frac{R}{R/2^i})^2 = 128(2^{2i})$ and $I = \lceil \log(2R/\rho_k^*) \rceil + 1$. The above expression is easily verified by noticing that each term in the summation on the left more than doubles the previous term and is dominated by the final term (just as $\sum_{i=0}^n 2^i = 2^{n+1}$). Putting everything together, $c_I = 32(2^{2I}) = 2048(R/\rho_k^*)^2$, and the total running time follows. ■