# Streamed Learning: One-Pass SVMs

**Piyush Rai, Hal Daumé III, Suresh Venkatasubramanian**
University of Utah, School of Computing
{piyush,hal,suresh}@cs.utah.edu

## Abstract

We present a streaming model for large-scale classification (in the context of $\ell_2$-SVM) by leveraging connections between learning and computational geometry. The streaming model imposes the constraint that only a single pass over the data is allowed. The $\ell_2$-SVM is known to have an equivalent formulation in terms of the minimum enclosing ball (MEB) problem, and an efficient algorithm based on the idea of *core sets* exists (CVM) [Tsang *et al.*, 2005]. CVM learns a $(1+\varepsilon)$-approximate MEB for a set of points and yields an approximate solution to corresponding SVM instance. However CVM works in batch mode requiring multiple passes over the data. This paper presents a single-pass SVM which is based on the minimum enclosing ball of streaming data. We show that the MEB updates for the streaming case can be easily adapted to learn the SVM weight vector in a way similar to using online stochastic gradient updates. Our algorithm performs polylogarithmic computation at each example, and requires very small and constant storage. Experimental results show that, even in such restrictive settings, we can learn efficiently in just one pass and get accuracies comparable to other state-of-the-art SVM solvers (batch and online). We also give an analysis of the algorithm, and discuss some open issues and possible extensions.

## 1 Introduction

Learning in a streaming model poses the restriction that we are constrained both in terms of time, as well as storage. Such scenarios are quite common, for example, in cases such as analyzing network traffic data, when the data arrives in a streamed fashion at a very high rate. Streaming model also applies to cases such as disk-resident large datasets which cannot be stored in memory. Unfortunately, standard learning algorithms do not scale well for such cases. To address such scenarios, we propose applying the *stream model* of computation [Muthukrishnan, 2005] to supervised learning problems. In the stream model, we are allowed only one pass (or a small number of passes) over an ordered data set, and polylogarithmic storage and polylogarithmic computation per element.

In spite of the severe limitations imposed by the streaming framework, streaming algorithms have been successfully employed in many different domains [Guha *et al.*, 2003]. Many of the problems in geometry can be adapted to the streaming setting and since many learning problems have equivalent geometric formulations, streaming algorithms naturally motivate the development of efficient techniques for solving (or approximating) large-scale batch learning problems.

In this paper, we study the application of the stream model to the problem of maximum-margin classification, in the context of $\ell_2$-SVMs [Vapnik, 1998; Cristianini and Shawe-Taylor, 2000]. Since the support vector machine is a widely used classification framework, we believe success here will encourage further research into other frameworks. SVMs are known to have a natural formulation in terms of the minimum enclosing ball problem in a high dimensional space [Tsang *et al.*, 2005; 2007]. This latter problem has been extensively studied in the computational geometry literature and admits natural streaming algorithms [Zarrabi-Zadeh and Chan, 2006; Agarwal *et al.*, 2004]. We adapt these algorithms to the classification setting, provide some extensions, and outline some open issues. Our experiments show that we can learn efficiently in just one pass and get competetive classification accuracies on synthetic and real datasets.

## 2 Scaling up SVM Training

Support Vector Machines (SVM) are maximum-margin kernel-based linear classifiers [Cristianini and Shawe-Taylor, 2000] that are known to provide provably good generalization bounds [Vapnik, 1998]. Traditional SVM training is formulated in terms of a quadratic program (QP) which is typically optimized by a numerical solver. For a training size of $N$ points, the typical time complexity is $O(N^3)$ and storage required is $O(N^2)$ and such requirements make SVMs prohibitively expensive for large scale applications. Typical approaches to large scale SVMs, such as chunking [Vapnik, 1998], decomposition methods [Chang and Lin, 2001] and SMO [Platt, 1999] work by dividing the original problem into smaller subtasks or by scaling down the training data in some manner [Yu *et al.*, 2003; Lee and Mangasarian, 2001]. However, these approaches are typically heuristic in nature: they may converge very slowly and do not provide rigorous guarantees on training complexity [Tsang *et al.*, 2005]. There has been a recent surge in interest in the online learning literature

for SVMs due to the success of various gradient descent approaches such as stochastic gradient based methods [Zhang, 2004] and stochastic sub-gradient based approaches[Shalev-Shwartz *et al.*, 2007]. These methods solve the SVM optimization problem iteratively in steps, are quite efficient, and have very small computational requirements. Another recent online algorithm LASVM [Bordes *et al.*, 2005] combines online learning with active sampling and yields considerably good performance doing single pass (or more passes) over the data. However, although fast and easy to train, for most of the stochastic gradient based approaches, doing a single pass over the data does not suffice and they usually require running for several iterations before converging to a reasonable solution.

## 3   Two-Class Soft Margin SVM as the MEB Problem

A minimum enclosing ball (MEB) instance is defined by a set of points $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^D$ and a metric $d : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^{\geq 0}$. The goal is to find a point (the *center*) $\mathbf{c} \in \mathbb{R}^D$ that minimizes the radius $R = \max_n d(\mathbf{x}_n, \mathbf{c})$.

The 2-class $\ell_2$-SVM [Tsang *et al.*, 2005] is defined by a hypothesis $f(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x})$, and a training set consisting of $N$ points $\{\mathbf{z}_n = (\mathbf{x}_n, y_n)\}_{n=1}^N$ with $y_n \in \{-1, 1\}$ and $\mathbf{x}_n \in \mathbb{R}^D$. The primal of the two-classs $\ell_2$-SVM (we consider the unbiased case one—the extension is straightforward) can be written as

$$\min_{\mathbf{w}, \xi_i} ||\mathbf{w}||^2 + C \sum_{i=1,m} \xi_i^2 \qquad (1)$$

$$s.t. \ y_i(\mathbf{w}'\varphi(\mathbf{x}_i)) \geq 1 - \xi_i, \ i = 1, ..., N \qquad (2)$$

The only difference between the $\ell_2$-SVM and the standard SVM is that the penalty term has the form $(C \sum_n \xi_n^2)$ rather than $(C \sum_n \xi_n)$.

We assume a kernel $K$ with associated nonlinear feature map $\varphi$. We further assume that $K$ has the property $K(\mathbf{x}, \mathbf{x}) = \kappa$, where $\kappa$ is a fixed constant [Tsang *et al.*, 2005]. Most standard kernels such as the isotropic, dot product (normalized inputs), and normalized kernels satisfy this criterion.

Suppose we replace the mapping $\varphi(\mathbf{x}_n)$ on $\mathbf{x}_n$ by another nonlinear mapping $\tilde{\varphi}(\mathbf{z}_n)$ on $\mathbf{z}_n$ such that (for unbiased case)

$$\tilde{\varphi}(\mathbf{z}_n) = \left[ y_n \varphi(\mathbf{x}_n); C^{-1/2} \mathbf{e}_n \right]^\top \qquad (3)$$

The mapping is done in a way that that the label information $y_n$ is subsumed in the new feature map $\tilde{\varphi}$ (essentially, converting a supervised learning problem into an unsupervised one). The first term in the mapping corresponds to the feature term and the second term accounts for a regularization effect, where $C$ is the misclassification cost. $\mathbf{e}_n$ is a vector of dimension $N$, having all entries as zero, except the $n^{\text{th}}$ entry which is equal to one.

It was shown in [Tsang *et al.*, 2005] that the MEB instance $(\tilde{\varphi}(\mathbf{z}_1), \tilde{\varphi}(\mathbf{z}_2), \ldots \tilde{\varphi}(\mathbf{z}_N))$, with the metric defined by the induced inner product, is dual to the corresponding $\ell_2$-SVM instance (1). The weight vector $\mathbf{w}$ of the maximum margin hypothesis can then be obtained from the center $\mathbf{c}$ of the MEB using the constraints induced by the Lagrangian [Tsang *et al.*, 2007].

## 4   Approximate and Streaming MEBs

The minimum enclosing ball problem has been extensively studied in the computational geometry literature. An instance of MEB, with a metric defined by an inner product, can be solved using quadratic programming[Boyd and Vandenberghe, 2004]. However, this becomes prohibitively expensive as the dimensionality and cardinality of the data increases; for an $N$-point SVM instance in $D$ dimensions, the resulting MEB instance consists of $N$ points in $N + D$ dimensions.

Thus, attention has turned to efficient approximate solutions for the MEB. A $\delta$-approximate solution to the MEB ($\delta > 1$) is a point $\mathbf{c}$ such that $\max_n d(\mathbf{x}_n, \mathbf{c}) \leq \delta R^*$, where $R^*$ is the radius of the true MEB solution. For example, A $(1 + \epsilon)$-approximation for the MEB can be obtained by extracting a very small subset (of size $O(1/\epsilon)$) of the input called a *core-set* [Agarwal *et al.*, 2005], and running an exact MEB algorithm on this set [Bădoiu and Clarkson, 2002]. This is the method originally employed in the CVM [Tsang *et al.*, 2005]. [Har-Peled *et al.*, 2007] take a more direct approach, constructing an explicit core set for the (approximate) maximum-margin hyperplane, without relying on the MEB formulation. Both these algorithms take linear training time and require very small storage. Note that a $\delta$-approximation for the MEB directly yields a $\delta$-approximation for the regularized cost function associated with the SVM problem.

Unfortunately, the core-set approach cannot be adapted to a streaming setting, since it requires $O(1/\epsilon)$ passes over the training data. Two one-pass streaming algorithms for the MEB problem are known. The first [Agarwal *et al.*, 2004] finds a $(1 + \epsilon)$ approximation using $O((1/\varepsilon)^{\lfloor D/2 \rfloor})$ storage and $O((1/\varepsilon)^{\lfloor D/2 \rfloor} N)$ time. Unfortunately, the exponential dependence on $D$ makes this algorithm impractical. At the other end of the space-approximation tradeoff, the second algorithm [Zarrabi-Zadeh and Chan, 2006] stores only the center and the radius of the current ball, requiring $O(D)$ space. This algorithm yields a 3/2-approximation to the optimal enclosing ball radius.

### 4.1   The StreamSVM Algorithm

We adapt the algorithm of [Zarrabi-Zadeh and Chan, 2006] for computing an approximate maximum margin classifier. The algorithm initializes with a single point (and therefore an MEB of radius zero). When a new point is read in off the stream, the algorithm checks whether or not the current MEB can enclose this point. If so, the point is discarded. If not, the point is used to suitably update the center and radius of the current MEB. All such selected points define a core set of the original point set.

Let $\mathbf{p}_i$ be the input point causing an update to the MEB and $\mathbf{B}_i$ be the resulting ball after the update. From figure 1, it is easy to verify that the new center $\mathbf{c}_i$ lies on the line joining the old center $\mathbf{c}_{i-1}$ and the new point $\mathbf{p}_i$. The radius $\mathbf{r}_i$ and the center $\mathbf{c}_i$ of the resulting MEB can be defined by simple update equations.

$$r_i = r_{i-1} + \delta_i \qquad (4)$$

$$||\mathbf{c}_i - \mathbf{c}_{i-1}|| = \delta_i \qquad (5)$$

Here $2\delta_i = (\|\mathbf{p}_i - \mathbf{c}_{i-1}\| - r_{i-1})$ is the closest distance of the new point $\mathbf{p}_i$ from the old ball $\mathbf{B}_{i-1}$. Using these, we can define a closed-form analytical update equation for the new ball $\mathbf{B}_i$:

$$\mathbf{c}_i = \mathbf{c}_{i-1} + \frac{\delta_i}{\|\mathbf{p}_i - \mathbf{c}_{i-1}\|}(\mathbf{p}_i - \mathbf{c}_{i-1}) \qquad (6)$$
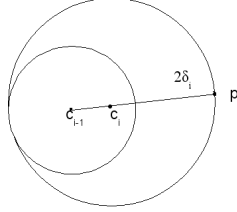


Figure 1: Ball updates

It can be shown that, for adversarially constructed data, the radius of the MEB computed by the algorithm has a lower-bound of $(1 + \sqrt{2})/2$ and a worst-case upper-bound of $3/2$ [Zarrabi-Zadeh and Chan, 2006].

We adapt these updates in a natural way in the augmented feature space $\tilde{\varphi}$ (see Algorithm 1). Each selected point belongs to the *core set* for the MEB. The support vectors of the corresponding SVM instance come from this set. It is easy to verify that the update equations for weight vector ($\mathbf{w}$) and the margin ($R$) in StreamSVM correspond to the center and radius updates for the ball in equation 7 and 4 respectively. The $\xi^2$ term is the distance calculation is included to account for the fact that the distance computations are being done in the $D + N$ dimensional augmented feature space $\tilde{\varphi}$ which, for the linear kernel case, is given by:

$$\tilde{\varphi}(\mathbf{z}_n) = \left[ y_n\mathbf{x}_n; C^{-1/2}\mathbf{e}_n \right]^\top. \qquad (7)$$

Also note that, because we perform only a single pass over the data and the $\mathbf{e}_n$ components are all mutually orthogonal, we never need to explicitly store them. The number of updates to the weight vector is limited by the number of core vectors of the MEB, which we have experimentally found to be much smaller as compared to other algorithms (such as Perceptron). The space complexity of StreamSVM is small since only the weight vector and the radius need be stored.

## 4.2 Kernelized StreamSVM

Although our main exposition and experiments are with linear kernels, it is straightforward to extend the algorithm for nonlinear kernels. In that case, algorithm 1, instead of storing the weight vector $\mathbf{w}$, stores an $N$-dimensional vector of Lagrange coefficients $\alpha$ initialized as $[y_1, \ldots, 0]$. The distance computation is line 5 are replaced by $d^2 = \sum_{n,m} \alpha_n \alpha_m k(\mathbf{x}_n, \mathbf{x}_m) + k(\mathbf{x}_n, \mathbf{x}_n) - 2y_n \sum_m \alpha_m k(\mathbf{x}_n, \mathbf{x}_m) + \xi^2 + 1/C$, and the weight vector updates in line 7 can be replaced by Lagrange coefficients updates $\alpha_{1:n-1} = \alpha_{1:n-1}(1 - \frac{1}{2}(1 - R/d))$, $\alpha_n = \frac{1}{2}(1 - R/d)\,y_n$.

---

**Algorithm 1** StreamSVM

1: **Input:** examples $(\mathbf{x}_n, y_n)_{n \in 1 \ldots N}$, slack parameter $C$
2: **Output:** weights ($\mathbf{w}$), radius ($R$), number of support vectors ($M$)
3: Initialize: $M = 1; R = 0; \xi^2 = 1, \mathbf{w} = y_1\mathbf{x}_1$
4: **for** $n = 2$ to $N$ **do**
5:     Compute distance to center:
    $d = \sqrt{\|\mathbf{w} - y_n\mathbf{x}_n\|^2 + \xi^2 + 1/C}$
6:     **if** $d \geq R$ **then**
7:         $\mathbf{w} = \mathbf{w} + \frac{1}{2}(1 - R/d)(y_n\mathbf{x}_n - \mathbf{w})$
8:         $R = R + \frac{1}{2}(d - R)$
9:         $\xi^2 = \xi^2\left[1 - \frac{1}{2}(1 - R/d)\right]^2 + \left[\frac{1}{2}(1 - R/d)\right]^2$
10:        $M = M + 1$
11:     **end if**
12: **end for**

---

**Algorithm 2** StreamSVM with lookahead L

**Input:** examples $(\mathbf{x}_n, y_n)_{n \in 1 \ldots N}$, slack parameter $C$, lookahead parameter $L \geq 1$
**Output:** weights ($\mathbf{w}$), radius ($R$), upper bound on number of support vectors ($M$)

1: Initialize: $M = 1; R = 0; \xi^2 = 1; \mathbf{S} = \emptyset; \mathbf{w} = y_1\mathbf{x}_1$
2: **for** $n = 2$ to $N$ **do**
3:     Compute distance to center:
    $d = \sqrt{\|\mathbf{w} - y_n\mathbf{x}_n\|^2 + \xi^2 + 1/C}$
4:     **if** $d \geq R$ **then**
5:         Add example $n$ to the active set:
        $\mathbf{S} = \mathbf{S} \cup \{y_n\mathbf{x}_n\}$
6:         **if** $|\mathbf{S}| = L$ **then**
7:             Update $\mathbf{w}, R, \xi^2$ to enclose the ball $(\mathbf{w}, R, \xi^2)$ and all points in $\mathbf{S}$
8:             $M = M + L$ ; $\mathbf{S} = \emptyset$
9:         **end if**
10:     **end if**
11: **end for**
12: **if** $|\mathbf{S}| > 0$ **then**
13:     Update $\mathbf{w}, R, \xi^2$ to enclose the ball $(\mathbf{w}, R, \xi^2)$ and all points in $\mathbf{S}$
14:     $M = M + |\mathbf{S}|$
15: **end if**

---

## 4.3 StreamSVM approximation bounds and extension to multiple balls

It was shown in [Zarrabi-Zadeh and Chan, 2006] that any streaming MEB algorithm that uses only $O(D)$ storage obtains a lower-bound of $(1 + \sqrt{2})/2$ and an upper-bound of $3/2$ on the quality of solution (i.e., the radius of final MEB). Clearly, this is a conservative approximation and would affect the obtained margin of the resulting SVM classifier (and hence the classification performance). In order to do better in just a single pass, one possible conjecture could be that the algorithm must *remember* more. To this end, we therefore extended algorithm-1 to simultaneously store $L$ weight vectors (or "balls"). The space complexity of this algorithm is $L(D + 1)$ floats and it still makes only a single pass over the

data. In the MEB setting, our algorithm chooses with each arriving datapoint (that is not already enclosed in any of the balls) how the current $L+1$ balls (the $L$ balls plus the new data point) should be merged, resulting again into a set of $L$ balls. At the end, the final set of $L$ balls are merged together to give the final MEB. A special variant of the $L$ balls case is when all but one of the $L$ balls are of zero radius. This amounts to storing a ball of non-zero radius and to keeping a *buffer* of $L$ many data-points (we call this the *lookahead* algorithm - Algorithm 2). Any incoming point, if not already enclosed in the current ball, is stored in the buffer. We solve the MEB problem (using a quadratic program of size $L$) whenever the buffer is full. Note that algorithm 1 is a special case of algorithm 2 with $L=1$, with the MEB updates available in a closed analytical form (rather than having to solve a QP).

Algorithm 1 takes linear time in terms of the input size. Algorithm 2 which uses a lookahead of $L$ solves a quadratic program of size $L$ whenever the buffer gets full. This step takes $O(L^3)$ times. The number of such updates is $O(N/L)$ (in practice, it is considerably less than $N/L$) and thus the over all complexity for the lookahead case is $O(NL^2)$. For small lookaheads, this is roughly $O(N)$.

## 5    Experiments

We evaluate our algorithm on several synthetic and real datasets and compare it against several state-of-the-art SVM solvers. We use 3 crieria for evaluations: a) Single-pass classification accuracies compared against single-pass of on-line SVM solvers such as iterative sub-gradient solver Pegasos [Shalev-Shwartz *et al.*, 2007], LASVM [Bordes *et al.*, 2005], and Perceptron [Rosenblatt, 1988]. b) Comparison with CVM [Tsang *et al.*, 2005] which is a batch SVM algorithm based on the MEB formulation. c) Effect of using lookahead in StreamSVM. For fairness, all the algorithms used a linear kernel.

### 5.1    Single-Pass Classification Accuracies

The single-pass classification accuracies of StreamSVM and other online SVM solvers are shown in table-1. Details of the datasets used are shown in table-1. To get a sense of how good the single-pass approximation of our algorithm is, we also report the classification accuracies of batch-mode (i.e., all data in memory, and multiple passes) libSVM solver with linear kernel on all the datasets. The results suggest that our single-pass algorithm StreamSVM, using a small reasonable lookahead, performs comparably to the batch-mode libSVM, and does significantly better than a single pass of other online SVM solvers.

### 5.2    Comparison with CVM

We compared our algorithm with CVM which, like our algorithm, is based on a MEB formulation. CVM is highly efficient for large datasets but it operates in batch mode, making one pass through the data for each core vector. We are interested in knowing how many passes the CVM must make over the data before it achieves an accuracy comparable to our streaming algorithm. For that purpose, we compared the accuracy of our single-pass StreamSVM against two and more

passes of CVM to see how long does it take for CVM to beat StreamSVM (we note here that CVM requires at least two passes over the data to return a solution). We used a linear kernel for both. Shown in Figure 2 are the results on MNIST 8vs9 data and it turns out that it takes several hundreds of passes of CVM to beat the single pass accuracy of StreamSVM. Similar results were obtained for other datasets but we do not report them here due to space limitations.
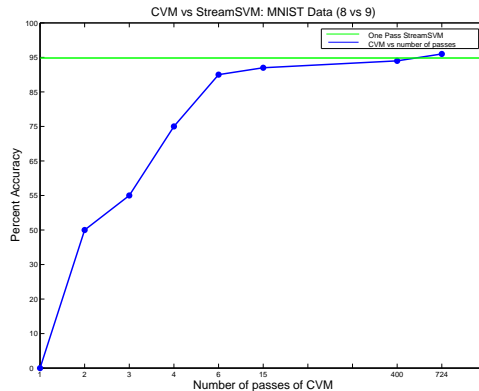


Figure 2: MNIST 8vs9 data: Number of passes CVM takes before achieving comparable single-pass accuracy of StreamSVM. X axis represents number of passes of CVM and Y axis represents the classification accuracy.
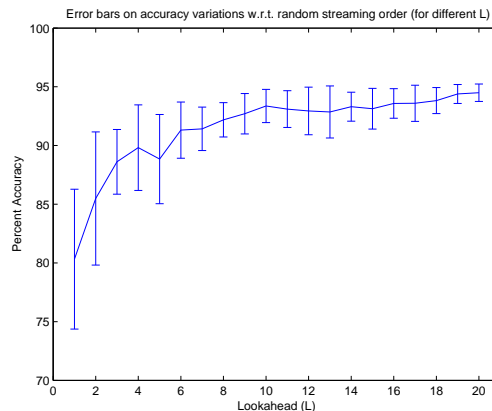


Figure 3: Single-pass with varying lookahead on MNIST 8vs9 data: Performance w.r.t random ordering of streaming. X axis represents the lookahead parameter and Y axis represents classification accuracy. Verticle bars represent the standard deviations in accuracies for a given lookahead.

### 5.3    Effect of Lookahead

We also investigated the effect of doing higher-order lookaheads on the data. For this, we varied $L$ (the lookahead parameter) and, for each $L$, tested Algorithm 2 on 100 random permutations of the data stream order, also recording the standard deviation of the classification accuracies with respect to

|  |  | # Examples | | libSVM | Perceptron | Pegasos | | LASVM | StreamSVM | |
| Data Set | Dim | Train | Test | (batch) |  | k = 1 | k = 20 |  | Algo-1 | Algo-2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Synthetic A | 2 | 20,000 | 200 | 96.5 | 95.5 | 83.8 | 89.9 | 96.5 | 95.5 | **97.0** |
| Synthetic B | 3 | 20,000 | 200 | 66.0 | 68.0 | 57.05 | 65.85 | 64.5 | 64.4 | **68.5** |
| Synthetic C | 5 | 20,000 | 200 | 93.2 | 77.0 | 55.0 | 73.2 | 68.0 | 73.1 | **87.5** |
| Waveform | 21 | 4000 | 1000 | 89.4 | 72.5 | 77.34 | 78.12 | 77.6 | 74.3 | **78.4** |
| MNIST (0vs1) | 784 | 12,665 | 2115 | 99.52 | 99.47 | 95.06 | 99.48 | 98.82 | 99.34 | **99.71** |
| MNIST (8vs9) | 784 | 11,800 | 1983 | 96.57 | **95.9** | 69.41 | 90.62 | 90.32 | 84.75 | 94.7 |
| IJCNN | 22 | 35,000 | 91,701 | 91.64 | 64.82 | 67.35 | 88.9 | 74.27 | 85.32 | **87.81** |
| w3a | 300 | 44,837 | 4912 | 98.29 | 89.27 | 57.36 | 87.28 | **96.95** | 88.56 | 89.06 |

Table 1: Single pass classification accuracies of various algorithms (all using linear kernel). The synthetic datasets (A,B,C) were generated using normally distributed clusters, and were of about 85% separability. libSVM, used as the absolute benchmark, was run in batch mode (all data in memory). StreamSVM Algo-2 used a small lookahead ($\sim$10). Note: We make the Pegasos implementation do a single sweep over data and have a user chosen block size $k$ for subgradient computations (we used k=1, and k=20 akin to using a lookahead of 20). Perceptron and LASVM are also run for a single pass and do not need block sizes to be specified. All results are averaged over 20 runs (w.r.t. random orderings of the stream)

the data-order permutations. Note that the algorithm still performs a single pass over the data. Figure 3 shows the results on the MNIST 8vs9 data (similar results were obtained for other datasets but not shown due to space limitations). In this figure, we see two effects. Firstly, as the lookahead increase, performance goes up. This is to be expected since in the limit, as the lookahead approaches the data set size, we will solve the exact MEB problem (albeit at a high computational cost). The important thing to note here is that even with a small lookahead of 10, the performance converges. Secondly, we see that the standard deviation of the result decreases as the lookahead increases. This shows experimentally that higher lookaheads make the algorithm less susceptible to badly ordered data. This is interesting from an empirical perspective, given that we can show that in theory, any value of $L < N$ cannot improve upon the 3/2-approximation guaranteed for $L = 1$.

## 6 Analysis, Open Problems, and Extensions

There are several open problems that this work brings up:

1. Are the $(1 + \sqrt{2})/2$ lower-bound and the $3/2$ upper-bound on MEB radius indeed the best achievable in a single pass over the data?

2. Is it possible to use a richer geometric structure instead of a ball and come up with streaming variants with provably good approximation bounds?

We discuss these in some more detail here.

### 6.1 Improving the Theoretical Bounds

One might conjecture that storing more information (i.e., more points) would give better approximation guarantees in the streaming setting. Although the empirical results showed that such approaches do result in better classification accuracies, this is not theoretically true in many cases.

For instance, in the adversarial stream setting, one can show that *neither* the lookahead algorithm *nor* its more general case (the multiple balls algorithm) improves the bounds given by the simple no-lookahead case (Algorithm-1). In particular, one can prove an identical upper- and lower-bound

for the lookahead algorithm as for the no-lookahead algorithm. To obtain the $3/2$-upper bound result, one can show a nearly identical construction as to [Zarrabi-Zadeh and Chan, 2006] where $L - 1$ points are packed in a small, carefully constructed cloud the boundary of the true MEB.

Alternatively, one can analyze these algorithms in the random stream setting. Here, the input points are chosen adversarially, but their *order* is permuted randomly. The lookahead model is not strengthened in this setting either: we can show both that the lower bound for no-lookahead algorithms, as well as the 3/2-upper bound for the specific no-lookahead algorithm described, generalize. For the former, see Figure 4. We place $(N - 1)/2$ points around $(0, 1)$ and $(N - 1)/2$ points around $(0, -1)$ and one point at $(1 + \sqrt{2}, 0)$. The algorithm will only beat the $(1 + \sqrt{2})/2$ lower bound if the singleton appears in the first $L$ points, where $L$ is the lookahead used. Assuming the lookahead is polylogarithmic in $N$ (which must be true for a streaming algorithm), this means that as $N \longrightarrow \infty$, the probability of a better bound tends toward zero. Note, however, that this applies only to the lookahead model, not to the more general multiple balls model, where it *may* be possible to obtain a tighter bounds in the random stream setting.
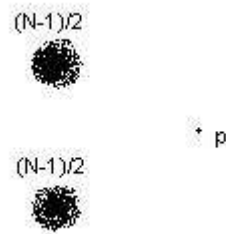


Figure 4: An adversarially constructed setting.

### 6.2 Ellipsoidal Balls

Instead of using a minimum enclosing ball of points, an alternative could be to use a minimum volume ellipsoid (MVE)

[Kumar *et al.*, 2005]. An ellipsoid in $\mathbb{R}^D$ is defined as follows: $\{\mathbf{x} : (\mathbf{x} - \mathbf{c})'\mathbf{A}(\mathbf{x} - \mathbf{c}) <= 1\}$ where $\mathbf{c} \in \mathbb{R}^D$, $\mathbf{A} \in \mathbb{R}^{DxD}$, and $\mathbf{A} \succeq 0$ (positive semi-definite).

Note that a ball, upon inclusion of a new point, expands equally in all dimensions which may be unnecessary. On the other hand, an ellipsoid can have several axes and scales of variations (modulated by the covariance matrix $\mathbf{A}$). This allows the ellipsoid to expand only along those directions where needed. In addition, such an approach can also be seen along the lines of confidence weighted linear classifiers [Dredze *et al.*, 2008]. The confidence weighted (CW) method assumes a Gaussian distribution over the space of weight vectors and updates the mean and covariance parameters upon witnessing each incoming example. Just as CW maintains the models uncertainty using a Gaussian, an ellipsoid generaization can model the uncertainty using the covariance matrix $\mathbf{A}$. Recent work has shown that there exist streaming possibilities for MVE [Mukhopadhyay and Greene, 2008]. The approximation gaurantees, however, are very conservative. It would be interesting to come up with improved streaming algorithms for the MVE case and adapt them for classification settings.

# 7 Conclusion

Within the streaming framework for learning, we have presented an efficient, single-pass $\ell_2$-SVM learning algorithm using a streaming algorithm for the minimum enclosing ball problem. We have also extended this algorithm to use a *lookahead* to increase robustness against poorly ordered data. Our algorithm, StreamSVM, satisfies a proven theoretical bound: it provides a $\left(\frac{3}{2}\right)$-approximation to the optimal solution. Despite this conservative bound, our algorithm is experimentally competitive with alternative techniques in terms of accuracy, and learns much simpler solutions. We believe that a careful study of stream-based learning would lead to high quality scalable solutions for other classification problems, possibly with alternative losses and with tighter approximation bounds.

# References

[Agarwal *et al.*, 2004] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. volume 51, pages 606–635, New York, NY, USA, 2004. ACM Press.

[Agarwal *et al.*, 2005] P. Agarwal, S. Har-Peled, and K. Varadarajan. Geometric approximations via coresets. *Combinatorial and Computational Geometry - MSRI Publications*, 52:1–30, 2005.

[Bordes *et al.*, 2005] Antoine Bordes, Seyda Ertekin, Jason Weston, and Leon Bottou. Fast kernel classifiers with online and active learning. volume 6, Cambridge, MA, USA, 2005. MIT Press.

[Boyd and Vandenberghe, 2004] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[Bădoiu and Clarkson, 2002] Mihai Bădoiu and Kenneth L. Clarkson. Optimal core-sets for balls. In *Proc. of DIMACS Workshop on Computational Geometry*, 2002.

[Chang and Lin, 2001] Chih-Chung Chang and Chih-Jen Lin. *LIB-SVM: a library for support vector machines*, 2001.

[Cristianini and Shawe-Taylor, 2000] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, 2000.

[Dredze *et al.*, 2008] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 264–271, New York, NY, USA, 2008. ACM.

[Guha *et al.*, 2003] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[Har-Peled *et al.*, 2007] Sariel Har-Peled, Dan Roth, and Dav Zimak. Maximum margin coresets for active and noise tolerant learning. In *IJCAI*, 2007.

[Kumar *et al.*, 2005] P. Kumar, E. A. Yildirim, and Communicated Y. Zhang. Minimum volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126:1–21, 2005.

[Lee and Mangasarian, 2001] Yuh-Jye Lee and Olvi L. Mangasarian. Rsvm: Reduced support vector machines. In *Proc. of Symposium on Data Mining (SDM)*, 2001.

[Mukhopadhyay and Greene, 2008] Asish Mukhopadhyay and Eugene Greene. A streaming algorithm for computing an approximate minimum spanning ellipse. *18th Fall Workshop on Computational Geometry*, 2008.

[Muthukrishnan, 2005] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[Platt, 1999] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, Cambridge, MA, USA, 1999. MIT Press.

[Rosenblatt, 1988] F. Rosenblatt. *The perception: a probabilistic model for information storage and organization in the brain*. MIT Press, 1988.

[Shalev-Shwartz *et al.*, 2007] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proc. ICML*, pages 807–814, New York, NY, USA, 2007. ACM Press.

[Tsang *et al.*, 2005] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. volume 6, Cambridge, MA, USA, 2005. MIT Press.

[Tsang *et al.*, 2007] Ivor W. Tsang, Andras Kocsor, and James T. Kwok. Simpler core vector machines with enclosing balls. In *Proc. ICML*, pages 911–918, New York, NY, USA, 2007. ACM Press.

[Vapnik, 1998] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

[Yu *et al.*, 2003] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svms with hierarchical clusters. In *Proc. ACM KDD*, pages 306–315, New York, NY, USA, 2003. ACM Press.

[Zarrabi-Zadeh and Chan, 2006] Hamid Zarrabi-Zadeh and Timothy M. Chan. A simple streaming algorithm for minimum enclosing balls. In *Proc. of Canadian Conference on Computational Geometry (CCCG)*, 2006.

[Zhang, 2004] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 116, New York, NY, USA, 2004. ACM.