Single Pass SVM using Minimum Enclosing Ball of Streaming Data

Piyush Rai*, Hal Daumé III, Suresh Venkatasubramanian
School of Computing, University of Utah
Salt Lake City - Utah 84112
{piyush,hal,suresh}@cs.utah.edu

February 8, 2008

Introduction

We present a stream algorithm for large scale classification (in the context of ℓ_2-SVM) by leveraging connections between learning and computational geometry. The stream model [1] imposes the constraint that only a single pass over the data is allowed. We study the streaming model for the problem of binary classification with SVMs and propose a single pass SVM algorithm based on the minimum enclosing ball (MEB) of streaming data [2]. We show that the MEB updates for the streaming case can be adapted to learn the SVM weight vector using simple Perceptron-like update equations. Our algorithm performs polylogarithmic computation at each example, requires very small and constant storage ($O(D)$ where D is the dimensionality of input space). 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.

The 2-class ℓ_2-SVM [3] 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 only difference between the ℓ_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 φ. We assume $\varphi(\mathbf{x_n})$ has unit norm.

Our approach is based on the equivalence of ℓ_2-SVM and the minimum enclosing balls problem [3]. A minimum enclosing ball (MEB) instance is defined by a set of points $\mathbf{x}_1, \ldots \mathbf{x}_N \in \mathbb{R}^D$ 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 $R = \max_n d(\mathbf{x}_n, \mathbf{c})$. It was shown in [3] that a solution to the ℓ_2 SVM can be obtained by solving an MEB with data points defined by $\mathbf{z}_n = y_n\hat{\varphi}(\mathbf{x}_n)$, where $\hat{\varphi}(\cdot)$ adds N-many dimensions to the input space (one per data point) all zeros except one position with value $\sqrt{1/C}$. A solution to the MEB on this data provides an exact solution to the ℓ_2 SVM, where the SVM weight vector can be recovered as the center of the MEB.

Solving the exact MEB is prohibitively expensive as it require solving a quadratic program. Thus, attention has turned to efficient approximate solutions for the MEB. A δ-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 optimal 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* [4], and running an exact MEB algorithm on this set [5]. This is the method originally employed in the CVM [3]. Note that a δ-approximation for the MEB directly yields a δ-approximation for the regularized cost function associated with the SVM problem. Unfortunately, the core-set approach requires $O(1/\epsilon)$ passes over the training data.

A Streaming Algorithm

Two single-pass streaming algorithms for the MEB problem are known. The first [6] 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 for learning problems. At the other end of the space-approximation tradeoff, the second algorithm [2] stores only the center and the radius of the current ball, requiring exactly $D+1$ floats. This algorithm yields a 3/2-approximation.

We adapt the algorithm of [2] for computing an approximate maximum margin classifier. Because we perform only a single pass over the data and the N-many additional dimensions are all mutually orthogonal, we never need to explicitly store them. It is easy to verify that the update equations for weight vector ($\mathbf{w}$) and the margin (determined

*Category: Learning Algorithms. Preference: Oral

Data Set	Dim	# Examples Train	# Examples Test	Perceptron Acc	Perceptron nSV	Pegasos Acc	Pegasos nSV	LASVM Acc	LASVM nSV	StreamSVM Acc	StreamSVM nSV
Synthetic A	2	20,000	200	95.5	1765	92.5	-	96.5	**5**	**97.0**	6
Synthetic B	3	20,000	200	68.0	7583	63.5	-	64.5	8	**68.5**	**5**
Synthetic C	5	20,000	200	77.0	3052	65.0	-	68.0	10	**87.5**	**6**
Waveform	21	4000	1000	47.4	1112	71.23	-	77.6	**9**	**78.4**	19
MNIST (0vs1)	784	12,665	2115	99.47	41	99.3	-	98.82	12	**99.71**	**8**
MNIST (8vs9)	784	11,800	1983	**95.9**	399	91.5	-	90.32	12	94.7	**6**
IJCNN	22	35,000	91,701	64.82	2111	84.50	-	74.27	**9**	**87.81**	26
w3a	300	44,837	4912	89.27	3986	74.02	-	**96.95**	**9**	89.06	12

Table 1: Single pass classification accuracy and number of support vectors found by three algorithms (using linear kernel). (For Pegasos, we use a selection set size of $k = 1$ and found that in these conditions it selected *every* point as a support vector. Data sets are listed with their dimensionality and the size of their training and test sets.)

by radius R) in our algorithm correspond to the center and radius update equations for the corresponding MEB instance [2]. Also note that the distance calculations are being done in the augmented feature space.

Our algorithm functions by reading the first data point $(y_1, \mathbf{x}_1)$ and initializing weights $\mathbf{w} = y_n\mathbf{x}_1$ and the "radius" (akin to the inverse margin) to 0. A slack variable ξ^2 is initialized to 1. As additional data points $(y_n, \mathbf{x}_n)$ arrive, a distance $d^2 = \|\mathbf{w} - y_n\mathbf{x}_n\|^2 + \xi^2 + 1/C$ is computed. If $d < R$, this point is ignored. Otherwise, the weights are updated to $\mathbf{w} + \frac{1}{2}(1 - R/d)(y_n\mathbf{x}_n - \mathbf{w})$, the radius is updated to $\frac{1}{2}(R + d)$ and the slack is updated to $\xi^2 = \xi^2\left[1 - \frac{1}{2}(1 - R/d)\right]^2 + \left[\frac{1}{2}(1 - R/d)\right]^2$. At any point the algorithm can be stopped and the current weight vector retrieved. All of the above calculations can be kernelized.

It was shown in [2] that any streaming MEB algorithm that uses only D floats of storage obtains a lower bound of $(1+\sqrt{2})/2$ on the quality of solution. In order to do better in just a single pass, the algorithm must *remember* more. We therefore extend this algorithm to simultaneously store K weight vectors (or "balls"). The space complexity of this algorithm is $K(D+1)$ floats and still makes only a single pass over the data. In the MEB setting, our algorithm choses with each arriving datapoint (that is not already enclosed) how the current $K + 1$ balls (the K balls plus the new data point) should be merged. This algorithm breaks the lower bound of $(1 + \sqrt{2})/2$. We are currently improving the upper bound.

We have implemeted a simplified version of the multiple weight vectors algorithm in which all but one ball have zero radius. Even limited as such, this algorithm works quite well in practice, as shown by preliminary experiments in Table1. Here, we compare our algorithm (we use a small value of K typically $6 - 8$) against a single pass of other state-of-the-art SVM solvers [7, 8]. Our early results have been encouraging and we expect to report a full set of experiments at the workshop.

We believe that the development of streaming algorithms for solving well-known learning models (such as SVM, logistic regression, etc.) is an attractive alternative to approaches based on stochastic subgradient [7], which tend to require multiple passes for convergence. We suggest streaming as an alternative to standard online learning methods, which typically involve designing a new learning model and algorithm: we are simply taking existing models and finding efficient stream algorithms to solve them.

References

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

[2] 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.

[3] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. volume 6, pages 363–392, Cambridge, MA, USA, 2005. MIT Press.

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

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

[6] 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.

[7] 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.

[8] 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.