Incremental Support Vector Machine Classification

Glenn Fung[*] *and Olvi L. Mangasarian*[†]

Abstract **Using a recently introduced proximal support vector machine classifier [4], a very fast and simple incremental support vector machine (SVM) classifier is proposed which is capable of modifying an existing linear classifier by both *retiring* old data and *adding* new data. A very important feature of the proposed single-pass algorithm , which allows it to handle massive datasets, is that huge blocks of data, say of the order of millions of points, can be stored in blocks of size $(n+1)^2$, where n is the usually small (typically less than 100) dimensional input space in which the data resides. To demonstrate the effectiveness of the algorithm we classify a dataset of 1 billion points in 10-dimensional input space into two classes in less than 2.5 hours on a 400 MHz Pentium II processor.**

Keywords ***incremental classifier, massive data classification, support vector machines***

1 Introduction

Recently [4] a fast simple classification algorithm was proposed that classifies points by assigning them to the closest of two parallel planes (in input or feature space) that are pushed apart as far as possible. This formulation, which can be interpreted as regularized least squares and can be considered in the much more general context of regularized networks [2, 3], leads to an extremely fast and simple algorithm for generating a linear or nonlinear classifier that merely requires the solution of a single

[*]Computer Sciences Department, University of Wisconsin, Madison, WI 53706. *gfung@cs.wisc.edu*.
[†]Computer Sciences Department, University of Wisconsin, Madison, WI 53706. *olvi@cs.wisc.edu*, corresponding author.

system of linear equations. In contrast, standard support vector machines (SVMs), which assign points to one of two halfspaces, solve a quadratic or a linear program that require considerably longer computational time. By taking advantage of the structure of the $(n+1) \times (n+1)$ positive definite matrix constituting the system of linear equations, where n is the usually small dimensional input space of the proximal SVM classifier [4], we are able to propose an incremental algorithm that has the following capabilities:

(i) Old data can be easily retired while new data can just as easily be incorporated into the classifier.

(ii) Storage capacity required is of the order $(n+1)^2$.

(iii) Computational time is of order $(n+1)^3$.

(iv) Extremely large datasets, of the order of 10^9, can be classified incrementally.

(v) Data compression of large data block of size $s \times n$ into data block of size $(n+1)^2$ is easily carried out for use by the incremental SVM, where s can be as large as 10^9.

We briefly summarize the contents of the paper now. In Section 2 we describe the proximal linear support vector machine and state the Linear Proximal Algorithm 2.1 on which our incremental algorithm is based. In Section 3 we introduce our Linear Incremental SVM Algorithm 3.1 based on solving the linear proximal SVM formulation of Section 2 incrementally. Section 4 contains numerical test results for our linear incremental classification algorithm including the classification of 1-billion points in 10-dimensional input space in 2 hours and 26 minutes on a Pentium II 400 MHz machine. Section 5 concludes the paper.

A word about our notation and background material. All vectors will be column vectors unless transposed to a row vector by a prime superscript $'$. For a vector x in the n-dimensional input space R^n, the sign function $sign(x)$ is defined as $sign(x)_i = 1$ if $x_i > 0$ else $sign(x)_i = -1$ if $x_i \leq 0$, for $i = 1, \ldots, n$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$ and the 2-norm of x will be denoted by $\|x\|$. For a matrix $A \in R^{m \times n}$, A_i is the ith row of A which is a row vector in R^n, while $A_{\cdot j}$ is the jth column of A. A column vector of ones of arbitrary dimension will be denoted by e. The identity matrix of arbitrary dimension will be denoted by I.

2 The Linear Proximal Support Vector Machine

We consider the problem, depicted in Figure 1, of classifying m points in the n-dimensional input space R^n, represented by the $m \times n$ matrix A, according to membership of each point A_i in the class $A+$ or $A-$ as specified by a given $m \times m$ diagonal matrix D with plus ones or minus ones along its diagonal. For this problem, the standard support vector machine with a linear kernel [13, 1] is given by the following quadratic program with parameter $\nu > 0$:

$$
\begin{array}{lrcl}
\min\limits_{(w,\gamma,y)\in R^{n+1+m}} & \nu e'y + \frac{1}{2}w'w & & \\
\text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\
 & y & \geq & 0.
\end{array}
\tag{1}
$$

As depicted in Figure 1, w is the normal to the bounding planes:

$$
\begin{array}{rcl}
x'w & = & \gamma + 1 \\
x'w & = & \gamma - 1,
\end{array}
\tag{2}
$$

that bound most of the sets $A+$ and $A-$ respectively. The constant γ determines their location relative to the origin. When the two classes are strictly linearly separable, that is when the error variable $y = 0$ in (1) (which is not the case shown in Figure 1), the plane $x'w = \gamma + 1$ bounds all of the class $A+$ points, while the plane $x'w = \gamma - 1$ bounds all of the class $A-$ points as follows:

$$
\begin{array}{rcll}
A_i w & \geq & \gamma + 1, & \text{for} \quad D_{ii} = 1, \\
A_i w & \leq & \gamma - 1, & \text{for} \quad D_{ii} = -1.
\end{array}
\tag{3}
$$

Consequently, the plane:

$$
x'w = \gamma,
\tag{4}
$$

midway between the bounding planes (2), is a separating plane that separates $A+$ from $A-$ completely if $y = 0$, else only approximately as depicted in Figure 1. The quadratic term in (1), which is twice the reciprocal of the square of the 2-norm distance $\frac{2}{\|w\|}$ between the two bounding planes of (2) (see Figure 1), maximizes this distance, often called the "margin". Maximizing the margin enhances the generalization capability of a support vector machine [13, 1]. If the classes are linearly inseparable, which is the case shown in Figure 1, then the two planes bound the two classes with a "soft margin" (i.e. bound approximately with some error) determined by the nonnegative error variable y, that is:

$$
\begin{array}{rcrcll}
A_i w & + & y_i & \geq & \gamma + 1, & \text{for } D_{ii} = 1, \\
A_i w & - & y_i & \leq & \gamma - 1, & \text{for } D_{ii} = -1.
\end{array}
\tag{5}
$$

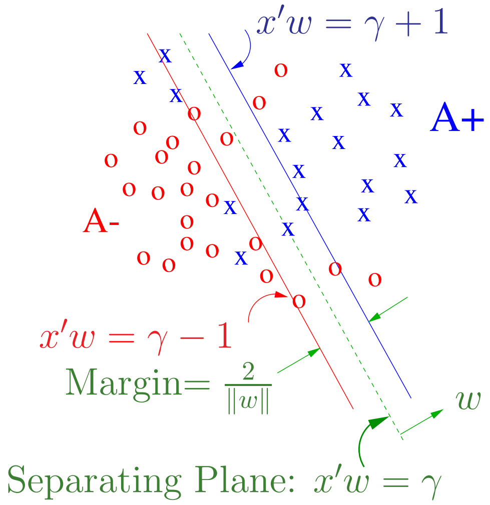

Figure 1. **The Standard Support Vector Machine Classifier in the w-space of R^n: The approximately bounding planes of equation (2) with a soft (i.e. with some error) margin $\frac{2}{\|w\|}$, and the plane of equation (4) approximately separating $A+$ from $A-$.**

The 1-norm of the error variable y is minimized parametrically with weight ν in (1), resulting in an approximate separating plane (4) as depicted in Figure 1. This plane acts as a linear classifier as follows:

$$sign(x'w - \gamma\ ) \begin{cases} = 1, & \text{then } x \in A+, \\ = -1, & \text{then } x \in A-. \end{cases} \tag{6}$$

To generate our proximal SVM we first modify the standard SVM formulation (1) in a similar manner to that of [9, 10], where the optimization problem (1) is replaced by the following problem:

$$\begin{array}{rl} \displaystyle\min_{(w,\gamma,y)\in R^{n+1+m}} & \frac{\nu}{2}\|y\|^2 + \frac{1}{2}(w'w + \gamma^2) \\ \text{s.t.} & D(Aw - e\gamma) + y \ \ \geq \ \ e \end{array} \tag{7}$$

Note that no explicit nonnegativity constraint is needed on y, because if any component y_i is negative then the objective function can be decreased by setting that $y_i = 0$ while still satisfying the corresponding inequality constraint. Note further that the 2-norm of the error vector y is minimized instead of the 1-norm, and the margin between the bounding planes is maximized with respect to both orientation w and relative location to the origin γ. Extensive computational experience, as in [8, 9, 10, 6, 5] indicates that this formulation is just as good as the classical formulation (1) with some added advantages such as strong convexity of the objective function. The key idea of the proximal SVM is to make a very simple, but fundamental, change in the formulation (7), namely replace the inequality constraint by an equality as follows:

$$\begin{array}{rrcl} \displaystyle\min_{(w,\gamma,y)\in R^{n+1+m}} & \frac{\nu}{2}\|y\|^2+\frac{1}{2}(w'w+\gamma^2) & & \\ \text{s.t.} & D(Aw-e\gamma)+y & = & e \end{array} \tag{8}$$

This modification, even though very simple, changes the nature of optimization problem significantly. In fact it turns out that we can write an explicit exact solution to the problem in terms of the problem data as we show below, whereas it is impossible to do that in standard SVM formulations because of their combinatorial nature. Geometrically the formulation (8) is depicted in Figure 2 which can be interpreted as follows. The planes $x'w-\gamma=\pm 1$ are not bounding planes anymore, but can be thought of as "proximal" planes, around which the points of each class are clustered and which are pushed as far apart as possible by the term $(w'w+\gamma^2)$ in the objective function which is nothing other than the reciprocal of the 2-norm distance squared between the two planes in the (w,γ) space of R^{n+1}.

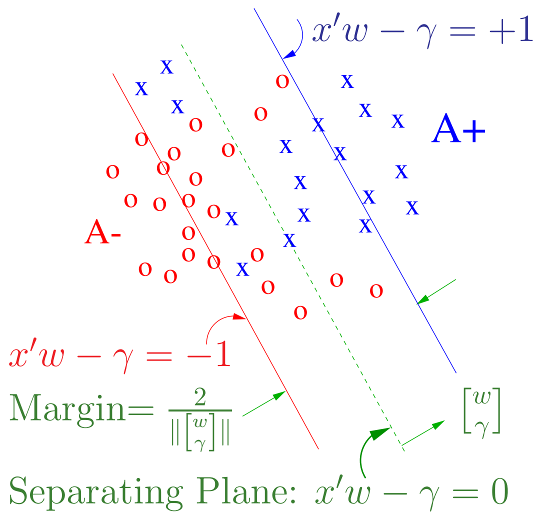

Figure 2. The Proximal Support Vector Machine Classifier in the (w,γ)-space of R^{n+1}: The planes $x'w-\gamma=\pm 1$ around which points of the sets $A+$ and $A-$ cluster and which are pushed apart by the optimization problem (8).

Substituting for y from the constraint in terms of $\begin{bmatrix} w \\ \gamma \end{bmatrix}$, we have the unconstrained optimization problem:

$$\min_{(w,\gamma)\in R^{n+1}} \quad \frac{\nu}{2}\|D(Aw-e\gamma)-e\|^2+\frac{1}{2}(w'w+\gamma^2). \tag{9}$$

Setting the gradient with respect to $\left[\begin{smallmatrix} w \\ \gamma \end{smallmatrix}\right]$ gives:

$$\begin{array}{rcl} \nu A'D(D(Aw - e\gamma) - e) + w & = & 0, \\ -\nu e'D(D(Aw - e\gamma) - e) + \gamma & = & 0. \end{array} \tag{10}$$

Noting that $DD = I$ and solving for $\left[\begin{smallmatrix} w \\ \gamma \end{smallmatrix}\right]$ gives:

$$\begin{bmatrix} w \\ \gamma \end{bmatrix} = \begin{bmatrix} A'A + \frac{I}{\nu} & -A'e \\ -e'A & \frac{1}{\nu} + m \end{bmatrix}^{-1} \begin{bmatrix} A'De \\ -e'De \end{bmatrix} = \left[\ \frac{I}{\nu} + \left[\begin{smallmatrix} A' \\ -e' \end{smallmatrix}\right] [A \quad -e] \ \right]^{-1} \begin{bmatrix} A'De \\ -e'De \end{bmatrix}. \tag{11}$$

Defining

$$E = [A \quad -e], \tag{12}$$

the solution given in (11) can be written as:

$$\begin{bmatrix} w \\ \gamma \end{bmatrix} = (\frac{I}{\nu} + E'E)^{-1}E'De. \tag{13}$$

This expression for $\left[\begin{smallmatrix} w \\ \gamma \end{smallmatrix}\right]$ involves the solution of a system of linear equations determined by the small dimensional $(n+1) \times (n+1)$ matrix, $\frac{I}{\nu} + E'E$. For concreteness we explicitly state our simple algorithm.

Algorithm 2.1. Linear Proximal SVM *Given m data points in R^n represented by the $m \times n$ matrix A and a diagonal matrix D of ± 1 labels denoting the class of each row of A, we generate the linear classifier (6) as follows:*

(i) Define E by (12) where e is an $m \times 1$ vector of ones. Compute $\left[\begin{smallmatrix} w \\ \gamma \end{smallmatrix}\right]$ by (13) for some positive ν. Typically ν is chosen by means of a tuning (validating) set.

(ii) Classify a new x by using (6) and the solution $\left[\begin{smallmatrix} w \\ \gamma \end{smallmatrix}\right]$ from Step (i) above.

For standard SVMs, support vectors consist of all data points which are the complement of the data points that can be dropped from the problem without changing the separating plane (4) [14, 7]. Thus, for the standard SVM formulation (1), support vectors correspond to data points for which the Lagrange multipliers are nonzero because, solving (1) with these data points only will give the same answer as solving it with the entire dataset. In our proximal formulation (8) however, the Lagrange multipliers u (which are not computed explicitly here because the constraint is eliminated before solving the problem) can be shown [4, Equation (11)] to be merely a multiple of the error vector y, that is, $u = \nu y$. Consequently, because all components of y are typically nonzero, since none of the data points usually lie on the proximal planes $x'w = \pm 1$, the concept of support vectors was modified in [4] as follows. We define the concept of *ϵ-support vectors* as those data points A_i for which error vector y_i is less than ϵ in absolute value. We typically pick ϵ small enough such that about 1% of the data are ϵ-support vectors. Re-solving our proximal SVM problem (8) with these data points and a ν adjusted (typically upwards) by a tuning set gives test set correctness that is essentially identical to that obtained by using the entire dataset.

We turn our attention now to our incremental algorithm that is based on the proximal SVM formulation described above.

3 Incremental SVM Classification

We describe now how a simple procedure can be applied to the proximal SVM, described in the previous section, that will allow it to generate a new linear classifier (6) by retiring old data while simultaneously adding new data. To do that let us augment the input vector $x \in R^n$ by -1 and define an augmented input vector $z \in R^{n+1}$ as follows:

$$z := \begin{bmatrix} x \\ -1 \end{bmatrix}. \tag{14}$$

Coupling this definition with the solution expression for $\left[\begin{smallmatrix} w \\ \gamma \end{smallmatrix}\right]$ given by (13), the linear classifier (6) can be written explicitly in terms of the input data as follows:

$$sign(z'(\frac{I}{\nu} + E'E)^{-1}E'De\ ) \begin{cases} = 1, & \text{then } x \in A+, \\ = -1, & \text{then } x \in A-, \end{cases} \tag{15}$$

where, as before, $E = [A \quad -e]$. Two key properties of the classifier (15) are:

(i) The size of the data to be stored is of order of $(n+1)^2$ for $E'E$ and of order $(n+1)$ for $E'De$, even if the number of data points is of the order of millions.

(ii) Time to solve the system of linear equations determined by the positive definite matrix $\frac{I}{\nu} + E'E$ is of order $(n+1)^3$.

Since n is typically less than 100 and often around 10, the above classifier is extremely effective for massive datasets with millions of data points. Based on these simple facts we describe an incremental algorithm that is capable of retiring any desired portion of the data while at the same time adding new data to generate an appropriately altered classifier. To keep the notation simple assume that the current classifier is based on an input dataset $E \in R^{m\times(n+1)}$ and a corresponding diagonal matrix $D \in R^{m\times m}$ of ± 1. Suppose that an "old" subset of this data represented by the submatrix $E^1 \in R^{m^1\times(n+1)}$ of E and a corresponding diagonal submatrix $D^1 \in R^{m^1\times m^1}$ of ± 1 of D needs to be retired and removed from the characterization of our classifier, leaving its complement in E to determine the new classifier together with new data. The "new" set of data points is represented by a new matrix $E^2 \in R^{m^2\times(n+1)}$ and a corresponding diagonal matrix $D^2 \in R^{m^2\times m^2}$ of ± 1 that needs to be added to the characterization of our new classifier. The classifier (15) can be updated to reflect the retirement of E^1 and the addition of E^2 as stated below in our incremental algorithm.

Algorithm 3.1. Linear Incremental SVM *Given m data points in R^n represented by the $m \times n$ matrix A and a diagonal matrix D of ± 1 labels denoting the class of each row of A, we generate an incremental linear classifier by retiring old data represented by the submatrix $E^1 \in R^{m^1 \times (n+1)}$ of $E = [A \quad -e]$ and a corresponding diagonal submatrix $D^1 \in R^{m^1 \times m^1}$ of D of ± 1 and adding new data represented by a new matrix $E^2 \in R^{m^2 \times (n+1)}$ and a corresponding diagonal matrix $D^2 \in R^{m^2 \times m^2}$ of ± 1 as follows:*

$$sign(z'(\frac{I}{\nu} + E'E - E^{1'}E^1 + E^{2'}E^2)^{-1}(E'De - E^{1'}D^1e + E^{2'}D^2e\,))\begin{cases} = 1, & then\ x \in A+, \\ = -1, & then\ x \in A-. \end{cases} \tag{16}$$

Note that for each block of data, say $E^i \in R^{m^i \times (n+1)}$, all we need to store is the relatively small $(n+1) \times (n+1)$ matrix $E^{i'}E^i$ and the $(n+1) \times 1$ vector $E^{i'}D^ie$. These quantities, which are of order than $(n+1)^2$ and $n+1$ respectively, allow us to retire and add data to classifiers as often as we wish. Furthermore, this incremental process allows us to handle arbitrarily large datasets, by successively adding blocks of data in the form of $E^{i'}E^i$ and $E^{i'}D^ie$, as will be demonstrated by the numerical results of the next section. Note also that $E^{i'}E^i$ which is of order $(n+1)^2$ can be considered to be a compression of the much larger dataset E^i which is of order $m^i(n+1)$. Since it takes $2(n+1)^2m^i$ operations to compute $E^{i'}E^i$ and $2(n+1)m^i$ operations to compute $E^{i'}D^ie$, the amount of computation of the incremental algorithm only grows linearly in the number of data points $m = \sum m^i$ as shown in Figure 4.

4 Numerical Testing

All our computations were performed on the University of Wisconsin Data Mining Institute "locop1" machine, which utilizes a 400 Mhz Pentium II and allows a maximum of 2 Gigabytes of memory for each process. This computer runs on Windows NT server 4.0, with MATLAB 6 installed. Even though "locop1" is a multiprocessor machine, only one processor was used for all the experiments since MATLAB is a single threaded application and does not distribute any load across processors [11].

Because we are principally interested in massive problems, we focus our numerical tests on two datasets synthetically generated and totally stored on disk. These sets are generated by Musicant's NDC (normally distributed clustered) dataset generator [12]. The data is generated as clusters of normally distributed points in R^n with an adjustable linear separability.

The first dataset consists of 1 billion points in 10-dimensional input space. The purpose of this dataset is to demonstrate the capability of our incremental SVM algorithm to obtain linear classifier by 500 increments of 2 million points each. The 1-billion dataset was generated first, and stored in 500 subsets of size 2 million points each. A testing set of size 10 million, was also generated with

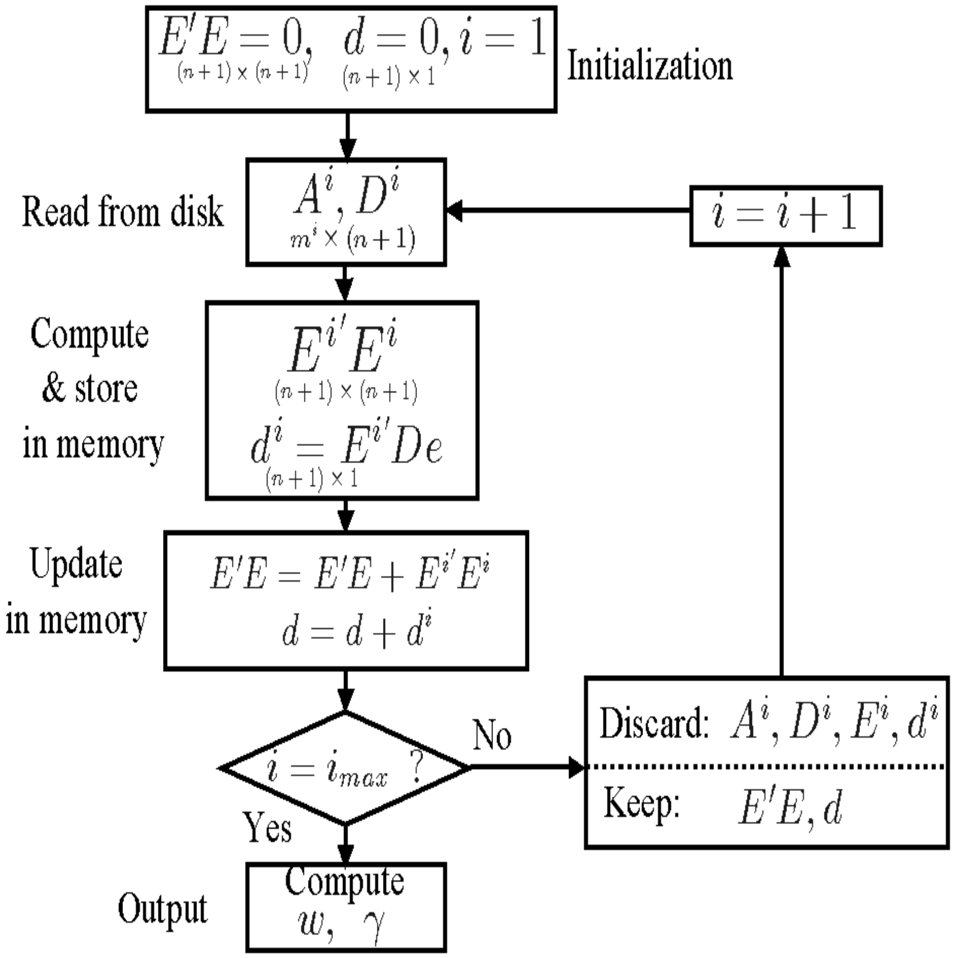

Figure 3. **Flow chart for the Linear Incremental Algorithm 3.1.**

the same distribution used to generate the 1 billion dataset and hence, resulting in essentially the same linear separability. It is important to note that every block of data is read from disk only once, and that once the new block of data is processed, the only data that has to be kept in memory is a matrix of size 11×11 and a vector of size 11×1 in our case here. Every time a new block is read from disk, new incoming data is processed by the algorithm, the 11×11 matrix and 11×1 vector are updated, and after this, all the processed data is discarded, leaving the memory ready for new incoming data. This process is graphically depicted in Figure 3.

The proposed incremental algorithm solved this 1-billion point problem in less than 2 hours and 26 minutes (8747 seconds). About 43 minutes, slightly less than 30% of the total time, was spent reading data from disk. Training set correctness was 90.78 % while testing set correctness was 90.79 %. A graph exhibiting the linear dependence of computational time of the algorithm on the number of data points used to generate the solution $\begin{bmatrix} w \\ \gamma \end{bmatrix}$ of (13) is presented in Figure 4.

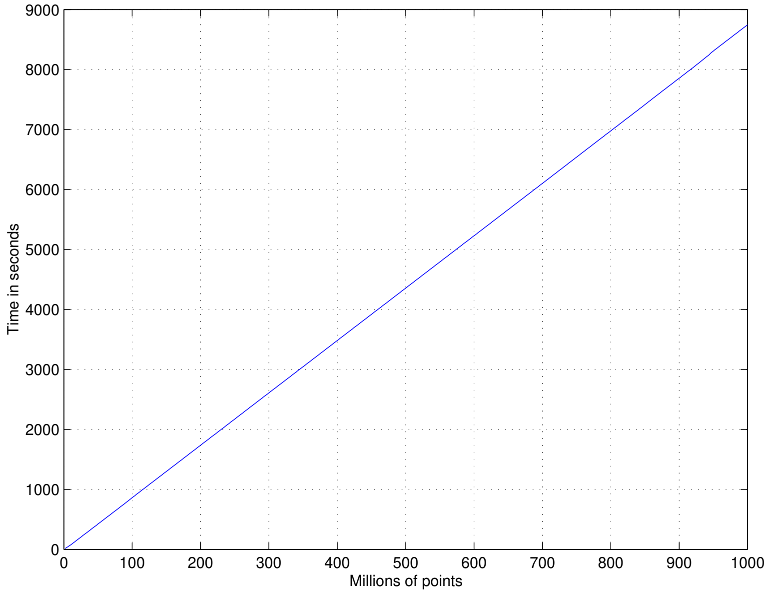

Figure 4. Computational time of the Linear Incremental Algorithm 3.1 as a function of the number of data points used to compute the solution $\begin{bmatrix} w \\ \gamma \end{bmatrix}$ of (13) for the 1-billion point dataset.

The second dataset of 60 million points in 10-dimensional input space is intended to simulate two months of data. We use this dataset to demonstrate how our algorithm can incrementally retire 1 million old points and add 1 million new points in 30 successive steps which simulate daily retirement of the oldest data and addition of the newest data. In the previous experiment all the data was assumed to be on disk from the beginning and just one final separating hyperplane was required. In this case at the beginning we assume that we just have 30 days of data and every day the oldest block of data has to be retired (1-million points) and one new block of data (1-million points) corresponding to the new data just collected has to be added. Furthermore, we assume that at every time that an update of the data is made, a new separating hyperplane has to be calculated. The data was generated in such a way that the starting and final separating hyperplanes differ considerably. This means that the incoming data is constantly changing and the separating criteria to be learned changes dynamically with respect to time. A gradual change in the resulting hyperplane after incrementally retiring 1 million old points and adding 1 million new points can be observed. In order to show this gradual change, we

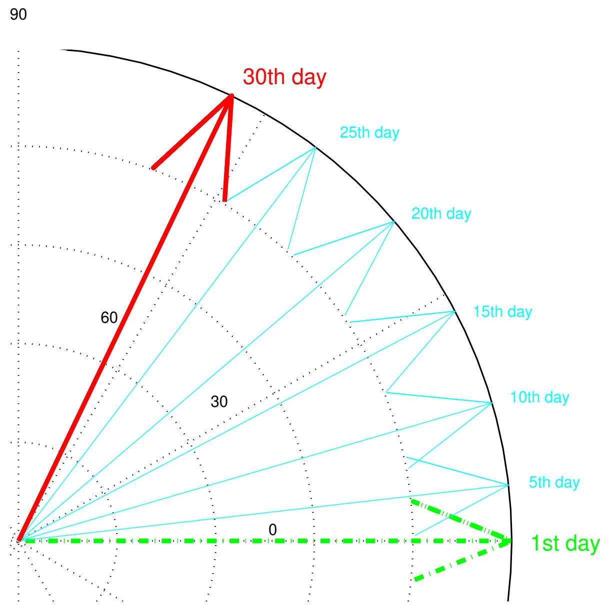

Figure 5. **Normals to the separating hyperplanes $x'w^i = \gamma^i,\ i = 1, \ldots, 30$, corresponding to 5-day intervals.**

measure the differences between planes by calculating the angle α between their normals. Thus, given two hyperplane normals $w^i, w^j \in R^n$ we compute:

$$\cos\,\alpha = \frac{|{w^i}'w^j|}{\|w^i\|\|w^j\|}$$

A plot showing the angles between the normals to the hyperplanes as the data changes is depicted in Figure 5.

5 Conclusion

We have proposed a very fast and simple incremental support vector machine classifier based on proximity of each class of points to one of two parallel planes that are pushed apart appropriately. The principal features of the proposed algorithm are its ability to retire old data and add new data very easily, its effectiveness and speed in handling massive datasets incrementally, and its ability to compress large datasets in a compression ratio of order $\frac{mn}{n^2} = \frac{m}{n}$, where m can be of the order of millions, and n is of the order 10 to a 100. These features coupled with its simplicity will hopefully make it a useful classification tool.

Acknowledgements

The research described in this Data Mining Institute Report 01-08, September 2001, was supported by National Science Foundation Grants CCR-9729842 and CDA-9623632, by Air Force Office of Scientific Research Grant F49620-00-1-0085 and by the Microsoft Corporation.

Bibliography

[1] V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods*. John Wiley & Sons, New York, 1998.

[2] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[3] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 171–203, Cambridge, MA, 2000. MIT Press.

[4] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings KDD-2001: Knowledge Discovery and Data Mining, August 26-29, 2001, San Francisco, CA*, pages 77–86, New York, 2001. Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps.

[5] Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. Technical Report 00-07, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, July 2000. Proceedings of the First SIAM International Conference on Data Mining, Chicago, April 5-7, 2001, CD-ROM Proceedings. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-07.ps.

[6] Yuh-Jye Lee and O. L. Mangasarian. SSVM: A smooth support vector machine. Technical Report 99-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, September 1999. *Computational Optimization and Applications* 20(1), October 2001, to appear. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-03.ps.

[7] O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

[8] O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-18.ps.

[9] O. L. Mangasarian and D. R. Musicant. Active support vector machine classification. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 577–583. MIT Press, 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-04.ps.

[10] O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-06.ps.

[11] MATLAB. *User's Guide.* The MathWorks, Inc., Natick, MA 01760, 1994-2001. http://www.mathworks.com.

[12] D. R. Musicant. NDC: normally distributed clustered datasets, 1998. www.cs.wisc.edu/∼musicant/data/ndc/.

[13] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[14] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, second edition, 2000.