Incremental Generalized Eigenvalue Classification on Data Streams

Mario R. Guarracino, Salvatore Cuciniello, Davide Feminiano

High Performance Computing and Networking Institute
National Research Council, Italy

Abstract

As applications on massive data sets are emerging with an increasing frequency, we are facing the problem of analyzing the data as soon as they are produced. This is true in many fields of science and engineering: in high energy physics, experiments have been done to transfer data at a sustained rate of 150 gigabits per second. In Y2007, that speed will enable the delivery to users of data continuously produced by the LHC particle accelerator located at CERN. Other examples can be found in network traffic analysis, telecommunications data mining, discrimination of data from sensors that monitor pollution and biological hazards, video and audio surveillance. In all cases, computational procedures have to deal with a large amount of data that are delivered in form of data streams. Traditional data mining techniques assume that the dataset is static and, to increment knowledge, random samples are extracted from the dataset. In this study, we use Incremental Regularized Generalized Eigenvalue Classification (I-ReGEC), a supervised learning algorithm, to continuously train a classification model from a data stream. The advantage of this technique is that the classification model can be update incrementally. The algorithm online decides which are the points that contain new information and updates the available classification model. We show through numerical experiments, on a synthetic dataset, the method performance, highlighting its behavior with respect to the number of incremental training set, the accuracy classification and the throughput of the data stream.

1 Introduction

The even-increasing amount of data generated in scientific and technical applications imposes new ways to extract knowledge from information. In high energy physics, for example, LHC particle accelerator located at CERN is going to produce tens of terabytes of data per second, which can be filtered down to hundreds of gigabytes per seconds by hardware preprocessing. Experiments have shown it is possible to transmit data at a sustained rate of 150 gigabits per second, in order to start delivering, in Y2007, the data continuously produced by the new generation accelerators, to users. Other examples can be found in computer networks traffic, bank transactions, web search and click streams and video/audio sensors. All those situations impose to rethink to the way in which information is acquired, stored and transmitted and to provide a shift in paradigms for data analysis.

The analysis of data streams is actually recognized in all those applications where data are not well represented by a persistent collection of items, but rather by an evolving data stream. The particular behavior of data streams consists in its continuous unpredictable arrival at highly rate. This relevant feature leads to a data stream model in which data are not randomly available in memory, but rather they appear as a continuous flow, where each *window* of items is available online only once. Furthermore, the period of time in which each window will be available for processing, depends on the stream speed and the capacity of the buffer used for temporary storage. The process of extracting knowledge from data streams reflects the mentioned features. Indeed, any mining task needs to be able to be performed online, that means data need to be processed on fly, at the speed in which they are transmitted. Furthermore, algorithms need to sample data, in such a way that models do not over fit data and are detailed enough to describe the phenomena. Finally, algorithms need to change their behavior during time, accordingly to the nature and gradient of the data. An interesting aspect of data stream analysis is that once a data item has been processed it can be either discarded or stored for its relevant informative contribute. In fact, even legacy data bases and warehouse containing terabytes of data could take advantage from this kind of analysis. If data can not be loaded in main memory, they could be accessed as a data stream [13].

In the case of supervised classification, which is the task of separating data into classes, starting from a set for which discrimination is known, standard methods rely on the persistence of a complete training set to build the discrimination model.

Applications of supervised classification on data streams are numerous. A classification model can be continuously trained over time with data regarding loan requests, and delivered to bank branches to predict the ability of a client to pay back. A video surveillance system can learn and detect suspicious behaviors of moving objects. A mail server can detect spam in the incoming stream of mails, dynamically changing its classification model to reflect the changes in spam flows. An online classifier can accurately identify the malicious applications associated with a TCP flow. Unfortunately, classification algorithms suffer of large memory requirements, when trained in batch mode. Therefore, data have to be partitioned in subsequences and analyzed in windows.

Among existing classification algorithms available for data streams there are Support Vector Machines (SVM). SVM algorithms [20] are the state-of-the-art for the existing classification methods. These methods classify the points from two linearly separable sets in two classes, finding an optimal separating hyperplane between two classes. This hyperplane maximizes the distance from the convex hulls of each class. SVM can be extended to the nonlinear cases by embedding the data in a nonlinear space using *kernel functions* [18].

There are also efficient algorithms that exploit the special structure of a slightly different optimization problem, such as Generalized Proximal SVMs (GEPSVM) [12], in which the binary classification problem can be formulated as a generalized eigenvalue problem. This formulation differs from SVM since, instead of finding one hyperplane that separates the two classes, it finds two hyperplanes that approximate the two classes. The prior study requires the solution of two different eigenvalue problems, while a classifier that uses a new regularization technique, known as Regularized General Eigenvalue Classifier (ReGEC) requires the solution of a single eigenvalue problem to find both hyperplanes [6].

In this work we propose SI-ReGEC, a novel technique based on I-ReGEC [4], that can be used to classify data streams. It is based on an incremental learning technique applied to a generalized eigenvalue classifier. With respect to other available algorithms, it determines classification models based on a very small sample of the stream, and it provides accuracy results that are comparable with other methods.

The notation used in the paper is as follows. Scalar product of two vectors x and y in R^n will be denoted by $x'y$, 2-norm of x will be denoted by $\|x\|$ and the unit vector will be denoted by e. The traspose of a matrix C is C^T.

The remainder of the the papers is organized as follows. In Section 2 related work is presented. In Section 3 the SI-ReGEC algorithm is detailed. In Section 4, numerical results are reported and discussed. Finally, in Section 5, conclusions are drawn and future work is proposed.

2 Related work

As machine learning becomes a part of data intensive computational systems, updating the learning system becomes intractable in many cases. Therefore, incremental methods that require some minimal computational burden to build and update classification models are strongly preferred. For this purpose, several methods, especially for kernel-based nonlinear classification, have been proposed to reduce the size of the training set, and thus, the related kernel [3, 5, 10, 11, 16]. All of these methods show that a sensible data reduction is possible while maintaining a comparable level of classification accuracy.

The general purpose methods are only suitable for small size problems, whereas for large problems, chunking subset selection [14] and decomposition methods [15] use subsets of points. SVM-Light [9] and LIBSVM [8] are among the most preferred implementations that use chunking subset selection and decomposition methods efficiently. SVM algorithm is used in different areas and its variations have been applied to classify also data stream.

Here we refer to five different techniques based on SVM, that are those used for performance comparison.
Batch technique (B). This technique uses SVM model with complete data set at once, therefore we denote it as batch mode.
Error-driven technique (ED). This technique is a variation of the method introduced in [13]. This technique defines two sets. It randomly takes k samples and stores them in the first set; all the others are stored in the second set. It then builds the classifier using KNNR [7] and it classifies the points of the other set. If a point is well classified, it remains in the set, otherwise it is moved in the other set. The procedure ends when there are no more points to transfer from one set to the other. The points used for incremental training are a percentage of both misclassified points.

Fixed-partition technique (FP). This method is introduced in [19] and it previews to divide the training set in batches of fixed size. This partition permits to add points to current support vector machine accordingly to the loaded in memory.
Exceeding-margin technique (EM) [5]. Given the model SVM_t at the time t and for each new point the algorithm checks if the new data exceeds the margin evaluated by SVM_t. The point is added to the incremental training, if the margin is exceeded, otherwise it is discarded. If the points is added then the new model SVM_{t+1} is calculated.
Fixed-margin+errors technique (EM+E) [5]. This technique is similar to the previous method, but in this case the new point is added to the incremental training either if it exceeds the margin or it is misclassified.

Since SI-ReGEC shares several common features with all those techniques, a comparison among them is meaningful and fair.

3 Algorithm

3.1 Classification based on Generalized Eigenvalues

Consider two matrices $A \in R^{n\times m}$ and $B \in R^{k\times m}$, that represent on each row the samples of the two classes. [12] proposes to classify these two sets of points A and B using two hyperplanes in the feature space, each closest to one set of points, and furthest from the other. In case of non linearly separable datasets, we can take advantage of kernel techniques to achieve greater separability among classes. In that case, the initial problem is non-linearly transformed into a space of greater dimension and the discrimination is found in that space.

This nonlinear mapping can be done implicitly by kernel functions [17], which represent the inner product of the elements in a nonlinear space.
In this study we use the *Gaussian kernel*,

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma}}. \tag{1}$$

In (1), x_i and x_j denote two points in the feature space. This technique usually allows to obtain better separation among classes, as shown in several applications. Results regarding nonlinearly separable problems [1, 2] still hold and a formulation for the eigenvalues problem can easily be derived.

In the nonlinear case, we use the kernel matrix $K(A, B)$, where each element is defined as:

$$K(A, B)_{i,j} = e^{-\frac{\|A_i - B_j\|^2}{\sigma}}. \tag{2}$$

where A_i and B_j are the i^{th} and j^{th} rows of the matrices A and B, respectively.

The two hyperplanes

$$K(x, C)u_1 - \gamma_1 = 0, \quad K(x, C)u_2 - \gamma_2 = 0 \tag{3}$$

can be obtained using the new regularization method *ReGEC*, proposed by [6], by solving the following generalized eigenvalue problem:

$$\min_{w,\gamma \neq 0} \frac{\|K(A, C)u - e\gamma\|^2 + \delta\|\tilde{K}_B u - e\gamma\|^2}{\|K(B, C)u - e\gamma\|^2 + \delta\|\tilde{K}_A u - e\gamma\|^2}. \tag{4}$$

where $C^T = \begin{bmatrix} A^T & B^T \end{bmatrix}$ and δ is the regularization parameter.

Here $\tilde{K}_A$ and $\tilde{K}_B$ are diagonal matrices with the diagonal entries from the matrices $K(A, C)$ and $K(B, C)$. The new regularization leads to a problem which provides accuracy results comparable to the original method.

The eigenvectors related to minimum and maximum eigenvalues obtained from the solution of (4) provide the proximal planes (3) P_i, $i = 1, 2$ to classify the new points. The distance of a point x from hyperplane P_i is:

$$dist(x, P_i) = \frac{|K(x, C)u - \gamma|}{\|u\|}, \tag{5}$$

and the class of a point x is determined as

$$class(x) = argmin_{i=1,2}\{dist(x, P_i)\}. \tag{6}$$

3.2 Incremental subsets selection

Incremental subset selection permits to construct a small set of points that retains the information of the entire training set and provides comparable accuracy results. A kernel built from a smaller subset is computationally more efficient in predicting new elements, compared to the one that uses the entire training set. Furthermore, a smaller set of points reduces the probability of over-fitting the problem. Finally, as new points are available, the cost to retrain the algorithm decreases if the influence of those new points on classification is only evaluated with respect to that subset, rather than the whole training set.

The algorithm takes an initial set of points $C \supset C_0 = A_0 \cup B_0$ and the entire training set C as input, where A_0 and B_0 are sets of points in C_0 that belong to the two classes A and B. We refer to C_0 as the *incremental subset*. Let $\Gamma_0 = C \setminus C_0$ be the initial set of points that can be included in the incremental subset. ReGEC classifies all of the points in the training set C using the kernel from C_0. Let P_{A_0} and P_{B_0} be the hyperplanes found by ReGEC, R_0 be the classification accuracy and M_0 be the points that are misclassified. Then, among the points in $\Gamma_0 \cap M_0$ the point that is farthest from its respective hyperplane is selected, i.e.

$$x_1 = x_i : \max_{x \subset \{\Gamma_0 \cap M_0\}} \left\{ dist(x, P_{class(x)}) \right\}, \tag{7}$$

where $class(x)$ returns A or B depending on the class of x. This point is the candidate point to be included in the incremental subset. This choice is based on the idea that a point very far from its plane either is either needed in the classification subset to improve accuracy, or it is an outlier. We update the incremental set as $C_1 = C_0 \cup \{x_1\}$. Then, we classify the entire training set C using the points in C_1 to build the kernel. Let the classification accuracy be R_1. If $R_1 > R_0$ then we keep the new subset; otherwise we reject the new point, that is $C_1 = C_0$. In both cases $\Gamma_1 = \Gamma_0 \setminus \{x_1\}$. The algorithm repeats until $|\Gamma_k| = 0$ at some k^{th} iteration. The s initial points are the training points closest the centroids determined by a simple *k-means* algorithm applied to each class. In [4] is showed that the k-mean based selection criteria gives the best performance in term of stability and accuracy, with respect to random selection of initial points.

3.3 Incremental learning on streams

In order to describe the use of the previous method on data streams, we will fix our attention to a single window. At the beginning, when incremental subset selection is performed, the initial points are determined by a simple k-mean algorithm. In the following steps, the initial points are taken from the incremental subset returned by the previous step. The procedure repeats until there are points in the stream . In Algorithm 1, we detail the procedure. Let C be the *wsize* points that are initially captured from the *stream*. Let C_0 the $2km$ centroids determined by kmeans algorithm. Every time new data is loaded from the *stream*, the previously determined incremental subset is used as C_0.

Algorithm 1 SI-ReGEC(km,$stream$,$wsize$)

```
1: $C = load(stream, wsize)$
2: $C_0 = kmean(C, km)$
3: $\Gamma_0 = C \setminus C_0$
4: $\{R_0, M_0\} = Classify(C, C_0)$
5: repeat
6:    $k = 1$
7:    while $|\Gamma_k| > 0$ do
8:       $x_k = x : \max_{x \in \{M_k \cap \Gamma_{k-1}\}} \{dist(x, P_{class(x)})\}$
9:       $\{R_k, M_k\} = Classify(C, \{C_{k-1} \cup \{x_k\}\})$
10:      if $R_k > R_{k-1}$ then
11:         $C_k = C_{k-1} \cup \{x_k\}$
12:      end if
13:      $\Gamma_k = \Gamma_{k-1} \setminus \{x_k\}$
14:      $k = k + 1$
15:   end while
16:   $C_0 = C_0 \cup C_k$
17:   $\Gamma_0 = load(stream, wsize)$
18: until $|\Gamma_0| = 0$
```

4 Numerical results

Performance results are calculated using an Intel Pentium 4 3.00GHz, 1GB RAM running Windows XP with Matlab 7.1. Matlab function *eig*, for the solution of the generalized eigenvalue problem, is used for ReGEC.

SI-ReGEC is tested on *Large-noisy-crossed-norm* data set. It has 200.000 points with 20 features equal divided in 2 classes. 100.000 points are used as training set and the remaining 100.000 to test the classifier. Each class is drawn from a multivariate normal distribution with unit covariance matrix. One class has mean $\mu_1 = 2/\sqrt{20}$ along each attribute and the other has mean $\mu_2 = -2/\sqrt{20}$. A Gaussian kernel is used for SI-ReGEC classifier with $\sigma = 120$ value of the best kernel parameter, $km = 2$ for the k-means method. The km value for each dataset is empirically determined as follows: first, the best σ value is determined for $km = 2$ using ten-fold cross-validation; then, the best km value is determined by gradually increasing its value.

In Table 1, classification error, number of samples in the incremental subset and window size for different methods applied to Large-noisy-crossed-norm data set. SI-ReGEC performances, expressed as percentage of clas-

Parameter	B	ED	FP	EM	EM+E	SI-ReGEC
Error (%)	3.2	9.1	3.2	4.5	6.7	**2.88**
subset train	8321	4172	8452	1455	5308	413
window size	100000	500	500	500	500	500

Table 1: Classification error, dimension of the incremental subset and window size for Large-noisy-crossed-norm

Wsize	Acc.	Growth Rate	Avg. Time	Time
500	96.13%	15.75	4.25s	8.5e-4s
1000	96.92%	4.31	15.16s	1.5e-3s
2000	96.50%	2.63	61.79s	3.1e-3s
4000	97.45%	1.81	232.49s	7.0e-3s

Table 2: Accuracy, growth rate, average time for each step and time for a single point.

sification error, show to compare well with the other incremental methods considered. It can be seen that SI-ReGEC has the lower error and it uses the smaller incremental set.

Table ?? shows for different values of the window size: classification accuracy, growth rate of incremental subset, average elapsed time for the execution on single window, average time for model update for a single sample, average ratio between incremental subset and training dataset. Results refer to a 10% sample of the dataset. We note that the accuracy is not influenced by window dimension *wsize*. The growth rate of the incremental subset decreases as the window size increases, leading to a smaller subsets for larger windows, although the execution time is four fold for a two fold window size. The better average time for a single sample processing is obtained for $wsize = 500$ and data is processed at 1.43Mbs.

In order to understand the influence of *wsize* on the incremental set, in Figure 1 we show the number of points of the incremental set on the first window and after the classification of 10% of samples in the dataset, for an increasing size of *wsize*. We note when *wsize* increases, the same happens to the number of initial points, whereas the final dimension of the incremental dataset decreases. This results confirms when a faster execution is needed, more storage space needs to be allocated to the incremental set samples.

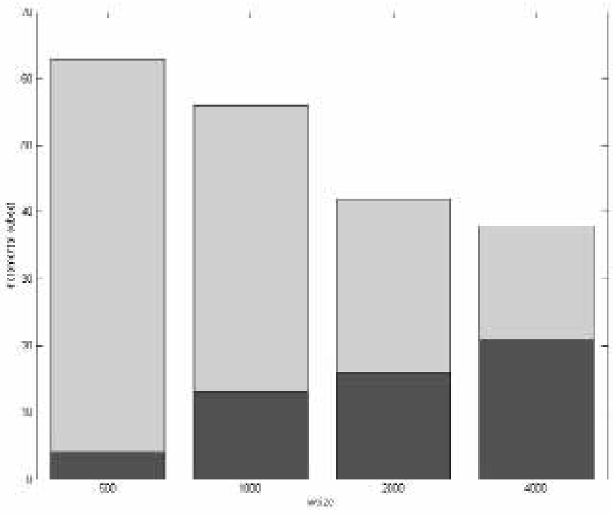

Figure 1: Incremental set size for *wsize* and 10.000 samples

5 Conclusions

In this study, we have introduced SI-ReGEC, an incremental algorithm that reduces the dimension of the training sets and, for each execution window of the stream, adds only few points to update the classifier. The proposed method *i*) achieves a classification accuracy that compares well with other incremental learning and batch mode methods and *ii*) produces smaller incremental training sets. In future, we will investigate how to adaptive window dimension to stream rate and storage requirements.

6 Acknowledgments

This work has been partially supported by Italian National Research Council and Centro Ricerche Enrico Fermi.

References

[1] K. Bennet and C. Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13, 2000.

[2] K. Bennett and O. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[3] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2000.

[4] C. Cifarelli, M.R. Guarracino, O. Seref, S. Cuciniello, and P.M. Pardalos. Incremental classification with generalized eigenvalues. Technical Report RT-ICAR-NA-2006-12, ICAR-CNR, June 2006.

[5] C. Domeniconi and D. Gunopulos. Incremental support vector machine construction. In *First IEEE International Conference on Data Mining (ICDM'01)*, pages 589–593, 2001.

[6] M. R. Guarracino, C. Cifarelli, O. Seref, and P. M. Pardalos. A classification algorithm based on generalized eigenvalue problems. *Optimization Methods and Software, in print*, 22(1):73–81, 2007.

[7] P. Hart. The condensed nearest neighbor rule. *IEEE Trans. on Inform. Th.*, (14):515–516, 1968.

[8] C.W. Hsu, C.C. Chang, and C.J. Lin. A practical guide to support vector classification. http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf, 2004.

[9] T. Joachims. *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning. MIT-Press, 1999.

[10] Y.J. Lee and O.L. Mangasarian. Rsvm: Reduced support vector machines. In *First SIAM International Conference on Data Mining*, 2001.

[11] K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 6(14):1449 – 1459, 2003.

[12] O. L. Mangasarian and E. W. Wild. Multisurface proximal support vector classification via generalized eigenvalues. Technical Report 04-03, Data Mining Institute, September 2004.

[13] P. Mitra, C.A. Murthy, and S. K. Pal. Data condensation in large databases by incremental learning with support vector machines. *ICPR*, 02:2708, 2000.

[14] R.F.E. Osuna and F. Girosi. An improved training algorithm for support vector machines. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, 1997.

[15] J. Platt. *Advances in Kernel Methods: Support Vector Learning*, chapter Fast training of SVMs using sequential minimal optimization, pages 185–208. MIT press, Cambridge, MA, 1999.

[16] L. Ralaivola. Incremental support vector machine learning: A local approach. *Lecture Notes in Computer Science*, 2130:322–330, 2001.

[17] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, 2002.

[18] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis.* Cambridge University Press, Cambridge, UK, 2004.

[19] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines, 1999.

[20] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.