Maintaining Gaussian Mixture Models of Data Streams Under Block Evolution

J.P. Patist, W. Kowalczyk, and E. Marchiori

Free University of Amsterdam, Department of Computer Science,
Amsterdam, The Netherlands
{jpp, wojtek, elena}@cs.vu.nl

Abstract. A new method for maintaining a Gaussian mixture model of a data stream that arrives in blocks is presented. The method constructs local Gaussian mixtures for each block of data and iteratively merges pairs of closest components. Time and space complexity analysis of the presented approach demonstrates that it is 1-2 orders of magnitude more efficient than the standard *EM* algorithm, both in terms of required memory and runtime.

1 Introduction

The emergence of new applications involving massive data sets such as customer click streams, telephone records, or electronic transactions, stimulated development of new algorithms for analysing massive streams of data, [2, 4].

In this paper we address the issue of maintenance of a Gaussian mixture model of a data stream under block evolution, see [5], where the modeled data set is updated periodically through insertion and deletion of sets of blocks of records. More specifically, we consider block evolution with a restricted window consisting of a fixed number of the most recently collected blocks of data. The window is updated one block at a time by inserting a new block and deleting the oldest one. Gaussian mixture models may be viewed as an attractive form of data clustering.

Recently, several algorithms have been proposed for clustering data streams, see e.g., [1], [6], or [8]. In our approach, we apply the classical *EM* algorithm, [3], to generate local mixture models for each block of data and a greedy merge procedure to combine these local models into a global one. This leads to a dramatic reduction of the required storage and runtime by 1-2 orders of magnitude.

2 Maintenance of Gaussian Mixture Models

A Gaussian mixture model with k components is a probability distribution on $\mathcal{R}^d$ that is given by a convex combination $p(x) = \sum_{s=1}^{k} \alpha_s p(x|s)$ of k Gaussian density functions:

$$p(x|s) = (2\pi)^{-d/2}|\Sigma_s|^{-1/2}\exp(-(x-\mu_s)^\top \Sigma_s^{-1}(x-\mu_s)/2), \;\; s = 1, 2, \ldots, k,$$

each of them being specified by its mean vector μ_s and the covariance matrix Σ_s.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 1071–1074, 2006.

Given a set $\{x_1, \ldots, x_n\}$ of points from $\mathcal{R}^d$, the learning task is to estimate the parameter vector $\theta = \{\alpha_s, \mu_s, \Sigma_s\}_{s=1}^k$ that maximizes the log-likelihood function $L(\theta) = \sum_{i=1}^n \log p(x_i; \theta)$. Maximization of the data log-likelihood $L(\theta)$ is usually achieved by running the *Expectation Maximization* (*EM*) algorithm [3]. For fixed values of d and k, the time and space complexity of this algorithm is linear in n.

Let us suppose that data points arrive in blocks of equal size and that we are interested in maintaining a mixture model for the most recent b blocks of data. An obvious, but expensive solution to this problem would be to re-run the *EM* algorithm after arrival of each block of data. Unfortunately, for huge data sets this method could be too slow.

In our approach, b local mixtures are maintained, one for each block. Mixtures are stored as lists of components. When a new block of data arrives, all components from the oldest block are removed and the *EM* procedure is applied to the latest block to find a local mixture model for this block. Finally, all bk local components are combined with help of a greedy merge procedure to form a global model with k components.

Two Gaussian components $(\mu_1, \Sigma_1, \alpha_1)$, $(\mu_2, \Sigma_2, \alpha_2)$ are merged into one component (μ, Σ, α) using the following formulas:

$$\mu = \frac{\alpha_1\mu_1 + \alpha_2\mu_2}{\alpha_1 + \alpha_2}, \Sigma = \frac{\alpha_1\Sigma_1 + \alpha_2\Sigma_2 + \alpha_1\mu_1\mu_1^T + \alpha_2\mu_2\mu_2^T}{\alpha_1 + \alpha_2} - \mu\mu^T, \alpha = \alpha_1 + \alpha_2.$$

The greedy merge procedure systematically searches for two closest components and merges them with help of the above formulas until there are exactly k components left.

The distance between components is measured with help of the Hotelling T^2 statistic, [7], which is used for testing whether the sample mean μ is equal to a given vector μ_0:

$$H^2(\mu, \alpha, \Sigma, \mu_0) = \alpha(\mu - \mu_0)^T\Sigma^{-1}(\mu - \mu_0).$$

The Hotelling distance between components C_1 and C_2 is then defined as follows:

$$dist_H(C_1, C_2) = (H^2(\mu_1, \alpha_1, \Sigma_1, \mu_2) + H^2(\mu_2, \alpha_2, \Sigma_2, \mu_1))/2.$$

Let us note that for a fixed value of d the merging process requires $O((bk)^2)$ steps and for large values of b and k it may be prohibitively slow. Therefore, we propose a more efficient method, called *k-means Greedy*, which is a combination of the k-means clustering and the greedy search. The combined algorithm reduces the initial number of components from bk to l, where $bk >> l > k$, with help the k-means clustering procedure and then the greedy search is used to reduce the number of components from l to k. A further speed-up can be achieved by using the Euclidean distance measure applied to $\mu's$ in the "k-means phase", rather than the expensive Hotelling statistic.

3 Space and Time Complexity Analysis

The main advantage of the proposed model construction and maintenance technique is the reduction of the required memory. Instead of storing the last n points, we store only the last block of data with up to n/b points and b local models. Now we will analyze the relation between the values of n (window size), d (data dimensionality), k (the number of components; the same for both local and the global model), b (the number

of blocks), and the memory reduction rate. We measure memory size in terms of the number of stored values.

When modeling data with multi-dimensional Gaussian distributions it is common to consider two models: a general one, with no restrictions on the covariance matrix Σ (other than being symmetric an positive definite), or a restricted one, where the covariance matrix is supposed to be diagonal. Let us first consider the case with full covariance matrices. Every component involves $1 + d + d(d+1)/2$ parameters (1 for prior, d for μ, and $d(d+1)/2$ for Σ), thus the total number of values that have to be stored, $M(b)$, is $bk(1 + d + d(d+1)/2) + dn/b$.

In order to find the optimal value of b that minimizes the above expression let us notice that the function $f(x) = \alpha x + \beta/x$, where $\alpha, \beta, x > 0$, reaches the minimum $2\sqrt{\alpha\beta}$ for $x = \sqrt{\beta/\alpha}$. Therefore, the optimal number of blocks, b_{opt}, is given by $b_{opt} = \sqrt{nd/k(1 + d + d(d+1)/2)}$, thus $M(b_{opt}) = 2\sqrt{ndk(1 + d + d(d+1)/2)}$. Hence, the optimal memory reduction rate, $R(b_{opt})$, satisfies:

$$R(b_{opt}) = nd/\sqrt{2ndk(1 + d + d(d+1))}.$$

In the case of diagonal covariance matrices similar reasoning gives:

$$R(b_{opt}) = nd/2\sqrt{ndk(1 + 2d)}.$$

To get a better insight into the relation between the compression rate and other parameters we produced two plots for a fixed $n = 50.000$, $d = 10$, and k ranging between 1 and 10, and $b = 1, 2 \ldots, 250$, see Figure 1.

Memory reduction rate is largest for small values of k: for $k = 2$ this rate is about 30-80 while for $k = 10$ it drops to 13-25, depending on d and the model type.

Finally, let us notice that the optimal number of blocks can be interpreted as the time speed-up factor. Indeed, for fixed values of d and k, the time complexity of the *EM*

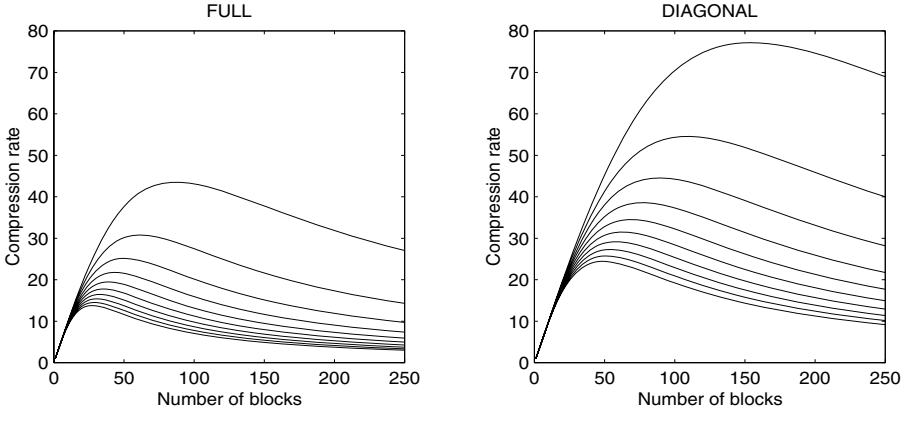

Fig. 1. The compression rate as a function of the number of blocks and the number of mixture components, $d = 10$. Each curve corresponds to another value of $k = 1, 2, \ldots, 10$. Lower curves correspond to larger values of k. The window size $n = 50.000$.

algorithm is linear in n. Therefore, the run time of this algorithm on a block of data of size n/b is b times smaller than when applied to the whole data set of n points. The additional time that is needed for updating the global model depends on the merging strategy and is $O((bk)^2)$ in case of greedy search, and $O(bk)$ when a combination of k-means and greedy search is applied (provided the value of parameter l is small enough, i.e., $l < \sqrt{bk}$). Taking into account that both k and b are relatively small compared to n, the influence of this factor on the overall run-time of the algorithm may be neglected. In practice, in case of relatively small (but realistic) values of k and d the speed-up factor ranges between 30-150 times.

4 Conclusions and Future Work

We presented a local approach for maintaining a Gaussian mixture model over a data stream under block evolution with restricted window. The proposed approach is 1-2 orders of magnitude more efficient than the standard *EM* algorithm, both in terms of required memory and runtime.

In our future research we would like to address three issues: 1) impact of our heuristic approach on the accuracy of models (some initial results are already reported in [9]), 2) dynamic identification of the optimal number of components (so far we assumed this number was known in advance and fixed), and 3) incremental modeling of mixtures of non-Gaussian distributions, e.g., mixtures of multinomials.

References

1. C. C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.
2. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM Press, 2002.
3. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.
4. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. *ACM SIGMOD Record*, 34(1), 2005.
5. V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining data streams under block evolution. *SIGKDD Explorations*, 3(2):1–10, 2002.
6. S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. Knowl. Data Eng.*, 15(3):515–528, 2003.
7. H. Hotelling. Multivariate quality control. In C. Eisenhart, M. W. Hastay, and W. A. Wallis, editors, *Techniques of Statistical Analysis*, pages 11–184. McGraw-Hill, New York, 1947.
8. S. Nassar, J. Sander, and C. Cheng. Incremental and effective data summarization for dynamic hierarchical clustering. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 467–478. ACM Press, 2004.
9. J. Patist, W. Kowalczyk, and E. Marchiori. Efficient Maintenance of Gaussian Mixture Models for Data Streams. *http://www.cs.vu.nl/∼jpp/GaussianMixtures.pdf*, Technical Report, Vrije Universiteit Amsterdam, 2005.