Clustering of High Dimensional Data Streams

Sotiris K. Tasoulis[1], Dimirtis K. Tasoulis[2], and Vassilis P. Plagianakos[1]

[1] Department of Computer Science and Biomedical Informatics,
University of Central Greece,
Papassiopoulou 2–4, Lamia, 35100, Greece
{stas,vpp}@ucg.gr
[2] Winton Capital Management,
1–5 St Mary Abbot's Place, SW8 6LS, United Kingdom
d.tasoulis@wintoncapital.com

Abstract. Clustering of data streams has become a task of great interest in the recent years as such data formats is are becoming increasingly ambiguous. In many cases, these data are also high dimensional and in result more complex for clustering. As such there is a growing need for algorithms that can be applied on streaming data and the at same time can cope with high dimensionality. To this end, here we design a streaming clustering approach by extending a recently proposed high dimensional clustering algorithm.

Keywords: Clustering, Data Streams, Kernel Density Estimation, Incremental Principal Component Analysis.

1 Introduction

Recent technological advances have made the continues collection of data trivial, leading to very large databases that grow at an unlimited rate and usually it is either unnecessary or impractical to store them. These streaming data present new challenges to clustering algorithms.

A streaming clustering process aims to continuously track the clustering structure of the data. Since stream data by nature impose a one pass constraint on the design of the algorithms, this task becomes more difficult. In addition, in many cases, streaming data are also high dimensional and in result more complex to cluster, due to the effect that high dimensionality has on distance or similarity [10,1]. Recently in [11] a density based hierarchical clustering approach (dePDDP) has been proposed that can deal with high dimensional data by projecting them onto a lower dimensional subspace.

Most clustering methods cannot be used for streaming data, since they rely on the assumption that the data are available in a permanent memory structure, from which global information can be obtained at any time. However, in many cases a clustering algorithm can be extended to the concept of data streams. A k-means clustering model for data streams was proposed in [4] and more recently, in [3] the DENSTREAM algorithm was developed by extending the GDBSCAN

I. Maglogiannis, V. Plagianakos, and I. Vlahavas (Eds.): SETN 2012, LNAI 7297, pp. 223–230, 2012.

algorithm. In this paper, we extend the recently proposed dePDDP framework and propose a new method for high dimensional data stream clustering.

In the next Section, we present the background material. In Section 3, we summarize the proposed approach and in Section 4 we present the experimental analysis that demonstrate the method's efficiency. Finally, the paper ends with concluding remarks in Section 5.

2 The dePDDP Algorithm

The dePDDP algorithm [11] is an algorithm that can deal with ultra high dimensions and has the ability to automatically retrieve the number of the clusters in the dataset. dePDDP is a divisive hierarchical clustering algorithm, producing a nested sequence of partitions, with a single, all-inclusive cluster at the top. Starting from this all-inclusive cluster the nested sequence of partitions is constructed by iteratively splitting clusters, until a termination criterion is satisfied. The main characteristic of the dePDDP algorithm is that it incorporates information from the density of the projected data onto the first principal component. The dePDDP procedure suggests that by splitting the data based on the global minimizer of the estimated density of the projected data onto the first principal component, is the best we can do to avoid splitting coherent data clusters. The cluster selection criterion and the termination criterion are guided by the same idea.

To formally describe how the principal direction projection based algorithm operates, let us assume that the data at hand is represented by an $n \times a$ matrix D, in which each row represents a data sample $d_i, i = 1, \ldots, n$, and a denotes the dimensionality. If we define the vector b and matrix Σ to represent the mean vector and the covariance of the data respectively:

$$b = \frac{1}{n} \sum_{i=1}^{n} d_i, \quad \Sigma = \frac{1}{n} (D - be)^{\top} (D - be),$$

where e is a column vector of ones. The covariance matrix Σ is symmetric and positive semi-definite, so all its eigenvalues are real and non-negative. The eigenvectors u_j $j = 1, \ldots, k$ corresponding to the k largest eigenvalues are called the principal components or principal directions. The dePDDP algorithm uses the projections p_i:

$$p_i = u_1(d_i - b), \quad i = 1, \ldots, n,$$

onto the first principal component u_1 to initially separate the entire data set into two partitions P_1 and P_2 based on a global minimizer x^*:

Definition 1. *(***Global Minimizer***) A global minimizer x^* is a point of $\mathbb{R}$ such that $\hat{f}(x^*; h) = \min_x \mathcal{X}$, where $\mathcal{X} = \{x \in R : \exists \delta > 0, \hat{f}(x + \delta; h) > \hat{f}(x; h)$ and $\hat{f}(x - \delta; h) > \hat{f}(x; h)\}$.*

The kernel density estimation $\hat{f}(x;h)$ of the density of the projected data onto the first principal component is given by the equation:

$$\hat{f}(x;h) = n^{-1}h^{-1}\sum_{i=1}^{n} K\left((x-p_i)/h\right). \quad (1)$$

The kernel choice in this work is the standard multi-variate normal density given by $K(x) = (2\pi)^{-1/2}e^{-0.5|x|}$.

Thus, $\forall d_i \in D, if p_i \geqslant x^*$ then the i-th data point belongs to the first partition P_1 $P_1 = P_1 \cup d_i$; otherwise, it belongs to the second partition $P_2 = P_2 \cup d_i$. The algorithm proceeds by splitting the cluster with the lowest global minimizer value and stops the iteration if no minimizer exists for any of the remaining clusters.

3 The Proposed Clustering Algorithm

To extend the dePDDP approach to streaming data we need to online update the hierarchical structure of the algorithm at each data point arrival. Thus, the proposed streaming modification will use the standard dePDDP methodology to assign the data entry to an already defined cluster or to create a new one.

In more detail, at each time instant n for each new data point arrival d_n, the proposed algorithm appropriately updates the hierarchical clustering structure constructed up to that time point. Starting from the root node the data points will be first projected on the u_1 Principal Component (PC) of all the points that have already been assigned to that node. Consequently, they will be assigned to a sub-node as described in Section 2, based on the density estimate $\hat{f}(x;h)$ of all the points assigned to that node.

However, this would imply that all the points assigned to each node need to be kept in memory in order to calculate both the PC u_1 and $\hat{f}(x;h)$, which is unrealistic in the data stream scenario. Nevertheless, online methods have been developed that overcome this constraint for online adaptation of both u_1 and $\hat{f}(x;h)$ respectively. Here, for the principal component u_1 we use the candid covariance-free IPCA (CCIPCA) method [12], which is based on the works of Oja and Karhunen [6] and Sanger [8]. The CCIPCA method is described in Section 3.1. To efficiently calculate the density function over data stream, we employ the method introduced in [14] based on the M-kernels concept. The main characteristic of this methodology is that it only uses a fix-sized main memory, which is irrespective of the total number of data points in the stream and the time complexity is linear to the size of the data stream (see Section 3.2).

The complete algorithmic scheme of the new SPDC (Streaming Principal Direction Clustering) algorithm is presented at Algorithm 1. For each node of the hierarchical structure the algorithm keeps in memory the node identity ID, the PC u_1^{ID}, the Density Function $\hat{f}(x;h)^{ID}$, and the identity of the left and right kid $Rkid^{ID}$ and $Lkid^{ID}$, respectively. For each point arrival d_n, the PC is updated and d_n is projected onto the updated PC. Then the density function is

updated and d is assigned to the left or right sub-node based on the minimizer x^* as explained in Section 2. The iteration stops when there is not a minimizer and the algorithms returns the identity ID.

Algorithm 1. The SPDC algorithm summary

1: For each point arrival d_n
2: **Do**
3: $ID = RootNode$
4: Update u_1^{ID} and calculate $p_n^{ID} = u_1^{ID} d_n$
5: Update Density $\hat{f}(x;h)^{ID}$ with p_n^{ID}
6: **if** there is a minimizer $x*$ **then**
7: If $p_n^{ID} \geqslant x^*$ then $ID = Rkid^{ID}$ else $ID = Lkid^{ID}$
8: go to 4
9: **end if**

3.1 Incremental PCA

Let $d_1, d_2, \ldots$ be the sample vectors that are acquired sequentially. Each $d_n, n = 1, 2, \ldots$, is a a-dimensional vector. Without loss of generality, we can assume that d_n has a zero mean (the mean may be incrementally estimated and subtracted out). Then the nth step estimate u_1^n of u_1 is given by

$$u_1^n = \frac{n-1-l}{n} u_1^{n-1} + \frac{1+l}{n} d_n d_n^T \frac{u_1^{n-1}}{\|u_1^{n-1}\|},$$

where $(n-1/n)$ is the weight for the last estimate and $1/n$ is the weight of the new data. The positive parameter l is called the amnesic parameter. With the presence of l, larger weight is given to new samples and the effect of old samples will fade out gradually. In this work, we do not use an amnesic parameter and l is set to 0. Finally, to begin the iteration, we set $u_1^0 = d_1$ as the first direction of data spread. A mathematical proof of the convergence of CCIPCA can be found in [13].

3.2 Density Estimation over Data Streams

To calculate the density of a point at the time frame of the stream n, we need to calculate the sum of n elements as shown in equation (1). To find a minimizer $x*$, we need to calculate the density over n points. As expected when data comes in the form of data stream, the large volume and the endlessness of the data stream make it computationally impossible to keep them in memory.

To handle data stream efficiently, we can maintain a representative sample of points with appropriately weights in order to accurately approximate $\hat{f}(x;h)$, as described in [14].

Let m be the number of the sample points that we keep in memory. When a new point arrives at time $m+1$ in order to update the density function based on this point without increasing the computational complexity of the algorithm,

we merge two points based on a merging cost. As a merging cost we use the distance between two consecutive points. Let p_k and p_l be the two points with the smallest merging cost. Then p_k and p_l are substituted by $p_n = (p_k + p_l)/2$. The weight value of the kernel function that corresponds to p_n is the sum of the weight values of the kernel function of p_k and p_l respectively. Then at time point n, the density estimation can be written as

$$\hat{f}_n^*(x;h) = n^{-1}h^{-1}\sum_{i=1}^{m}\rho_i^* K\left((x-p_i)/h\right),$$

where $\sum_{i=1}^{m}\rho_i^* = n$. The bandwidth parameter is very important for the quality of the density estimation. Most well known bandwidth strategies [9] often assign a global bandwidth to all kernels. However, these strategies depend on the complete sample, which is not known in the concept of data streams. To overcome this problem, we use an approximate solution that complies with the processing requirements of data streams similar to the one used in [5]. We use the "normal reference rule" bandwidth strategy, which is the bandwidth that minimizes the Mean Integrated Squared Error (MISE). For a sample with n samples this is given by $h_{opt}^n = \sigma\left(\frac{4}{3n}\right)^{1/5}$, where σ is the standard deviation of the data. The standard deviation here is computed incrementally in constant time.

4 Experimental Analysis

In this Section, we perform an experimental evaluation of the proposed clustering method on streaming data. To achieve this we employ a series of simulated datasets. This gives the opportunity to pre-design and hence know beforehand the structure of the data that the clustering algorithm aims to recover. In particular we construct datasets by randomly drawing points from a finite mixture of k Gaussian distributions that represent the actual clusters in the data. 5000 points are drawn in total for each dataset. The mean of each Gaussian is randomly placed in $[100, 200]^a$ and the covariance matrix is also randomly generated by an appropriate procedure, so as to ensure that it is symmetric and positive definite.

To assess the quality of a data partition, we use the class labels which are not available to the clustering algorithms. We measure the degree of correspondence between the resulting clusters and the classes of each object. In detail, let $\mathcal{L}$ be the set of class labels $l_i \in \mathcal{L}$, for each point $d_i \in \mathcal{D}$, $i = 1, \ldots, n$, with l_i taking values in $\{1, \ldots, L\}$ we define the purity of a k-cluster partitioning as $\Pi = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. The purity of Π is defined by the following formula:

$$p(\Pi) = \frac{\sum_{j=1}^{k}\max\left\{|\{p_i \in \mathcal{C}_j : l_i = 1, \ldots, L\}|\right\}}{n},$$

so that $0 \leq p(\Pi) \leq 1$. High values indicate that the majority of vectors in each cluster come from the same class, so in essence the partitioning is "pure" with respect to class labels.

To address the question of whether all members of a given class are included in a single cluster we use the V-measure [7] criterion. The V-measure tries to capture cluster homogeneity and completeness, which summarizes a clustering solution's success in including every point of a single class and no others. Again, high values corresponds to better performance.

Tables 1 and 2 report the clustering performance with respect to the clustering purity, the V-measure and the number of the found clusters for several types of datasets, respectively. For each dataset, 50 experiments have been conducted and the mean values with the standard deviation (in the parenthesis) are presented. The clustering performance is always measured at the last 100 points of the stream. The performance of the proposed method is compared against the well known DENSTREAM algorithm [3]. For DENSTREAM, the ϵ parameter was set to 100. In the case of SPDC (Streaming Principal Direction Clustering), 100 M-kernels were used in all experiments. The bandwidth parameter for the density estimation was set as explained in Section 3.2. As shown the SPDC clustering results yield superior Purity and V-measure in most cases.

Table 1. Mean purity, V-measure and number of the found clusters for the artificial datasets

	Dimension 2			Dimension 10		
No. Of Cl.	Purity	V-measure	Clusters	Purity	V-measure	Clusters
	SPDC			SPDC		
2	1.00 (0.02)	0.68 (0.09)	6.02 (1.59)	0.99 (0.06)	0.81 (0.16)	4.00 (1.43)
5	0.96 (0.07)	0.92 (0.05)	6.90 (1.68)	0.96 (0.07)	0.95 (0.05)	5.64 (1.16)
10	0.72 (0.14)	0.82 (0.09)	8.14 (2.06)	0.82 (0.11)	0.89 (0.06)	10.08 (2.26)
	DENSTREAM			DENSTREAM		
2	0.51 (0.00)	0.00 (0.00)	1.00 (0.00)	0.85 (0.05)	0.70 (0.23)	1.70 (0.23)
5	0.22 (0.00)	0.00 (0.00)	1.00 (0.00)	0.56 (0.07)	0.60 (0.12)	2.70 (1.78)
10	0.12 (0.00)	0.00 (0.00)	1.00 (0.00)	0.23 (0.00)	0.35 (0.02)	2.00 (0.22)

Table 2. Mean purity, V-measure and number of the found clusters for the artificial datasets

	Dimension 50			Dimension 100		
No. Of Cl.	Purity	V-measure	Clusters	Purity	V-measure	Clusters
	SPDC			SPDC		
2	1.00 (0.00)	0.90 (0.10)	2.88 (0.82)	1.00 (0.00)	0.92 (0.08)	2.64 (0.62)
5	0.97 (0.07)	0.97 (0.05)	5.30 (0.78)	0.90 (0.13)	0.93 (0.08)	4.84 (0.86)
10	0.85 (0.11)	0.91 (0.06)	9.18 (1.73)	0.78 (0.13)	0.86 (0.10)	8.26 (1.81)
	DENSTREAM			DENSTREAM		
2	1.00 (0.00)	1.00 (0.00)	2.00 (0.00)	1.00 (0.00)	1.00 (0.00)	2.00 (0.00)
5	1.00 (0.00)	1.00 (0.00)	5.00 (0.00)	0.92 (0.01)	0.95 (0.00)	4.60 (0.26)
10	0.17 (0.00)	0.21 (0.05)	1.50 (0.27)	0.21 (0.00)	0.31 (0.04)	1.90 (0.54)

To examine the sensitivity of the number of M-kernels parameter used by SPDC algorithm we perform a series of experiments for the 50-dimensional 5-cluster case.

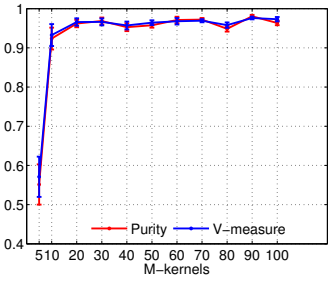

Fig. 1. Mean purity and V-measure with respect to the corresponding number of M-kernels

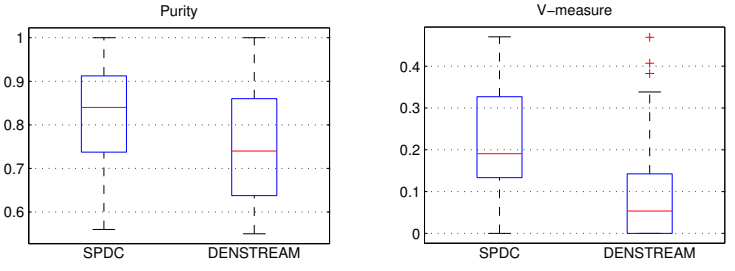

Fig. 2. Boxplots of Purity and V-measure for the Forest CoverType real world dataset

For each parameter value 50 experiments have been made. As shown in Figure 1 a parameter value higher than 10 is enough to achieve high quality results.

Finally, we test the efficiency of SPDC and DENSTREAM at the Forest CoverType real world dataset, obtained from the UCI machine learning repository [2]. This dataset is comprised of 581012 observations characterized in 54 attributes, where each observation is labelled in one of seven forest cover classes. Here we only use the 10 numerical attributes. In Figure 2 we can see boxplots of the Purity and the V-measure of the clustering result, obtained in the last 100 points for various time point of the data stream for SPDC and DENSTREAM, respectively. The ϵ parameter for DENSTREAM was set to 30 for better results. As shown the SPDC results are superior in both cases. Purity values are always high, but the V-measure obtain lower values since in most cases the algorithms tend to find more clusters than the actual.

5 Concluding Remarks

Although many data stream clustering algorithms have been proposed in the literature, very few of them can actually deal with high dimensionality. In this work, we present an algorithm that can effectively deal with high dimensional data streams. The proposed method shows promising results in synthetic and real data scenarios. In a future work we intent to extend this approach for clustering on evolving data streams.

Acknowledgments. This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II, Investing in knowledge society through the European Social Fund.

References

1. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When Is Nearest Neighbor Meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
2. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
3. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: 2006 SIAM Conference on Data Mining, pp. 328–339 (2006)
4. Domingos, P., Hulten, G., Edu, P.C.W., Edu, C.H.G.W.: A general method for scaling up machine learning algorithms and its application to clustering. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 106–113. Morgan Kaufmann (2001)
5. Heinz, C., Seeger, B.: Towards Kernel Density Estimation over Streaming Data. In: International Conference on Management of Data. Computer Society of India, COMAD 2006, Delhi, India (December 2006)
6. Oja, E., Karhunen, J.: On Stochastic Approximation of the Eigenvectors and Eigenvalues of the Expectation of a Random Matrix. Journal of Mathematical Analysis and Applications 106, 69–84 (1985)
7. Rosenberg, A., Hirschberg, J.: V-measure: A conditional entropy-based external cluster evaluation measure. In: 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 410–420 (2007)
8. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Networks 2(6), 459–473 (1989)
9. Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley Series in Probability and Statistics. Wiley (September 1992)
10. Steinbach, M., Ertöz, L., Kumar, V.: The challenges of clustering high dimensional data. New Vistas in Statistical Physics: Applications in Econophysics, Bioinformatics, and Pattern Recognition (2003)
11. Tasoulis, S., Tasoulis, D., Plagianakos, V.: Enhancing Principal Direction Divisive Clustering. Pattern Recognition 43, 3391–3411 (2010)
12. Weng, J., Zhang, Y., Hwang, W.: Candid covariance-free incremental principal component analysis (2003)
13. Zhang, Y., Weng, J.: Convergence analysis of complementary candid incremental principal component analysis (2001)
14. Zhou, A., Cai, Z., Wei, L., Qian, W.: M-kernel merging: Towards density estimation over data streams. In: International Conference on Database Systems for Advanced Applications, p. 285 (2003)