A clustering algorithm for multivariate big data with correlated components

Un algoritmo di clustering per big data multivariati con componenti correlate

Giacomo Aletti and Alessandra Micheletti

Abstract Common clustering algorithms require multiple scans of all the data to achieve convergence, and this is prohibitive when large databases, with millions of data, must be processed. Some algorithms to extend the popular K-means method to the analysis of big data are present in literature since 1998 [1], but they assume that the random vectors which are processed and grouped have uncorrelated components. Unfortunately this is not the case in many practical situations. We here propose an extension of the algorithm of Bradley, Fayyad and Reina to the processing of massive multivariate data, having correlated components.

Abstract *I comuni algoritmi di clustering richiedono di esaminare più volte tutti i dati per raggiungere la convergenza, e ciò risulta proibitivo quando devono essere analizzati database enormi, con milioni di dati. In letteratura sono presenti fin dal 1998 [1] alcuni algoritmi che estendono il popolare metodo K-medie all'analisi di big data, ma essi assumono che i vettori aleatori che vengono analizzati e raggruppati abbiano componenti non correlate. Purtroppo tale condizione non è soddisfatta in molti casi pratici. Qui proponiamo un'estensione dell'algoritmo di Bradley, Fayyad e Reina all'analisi di grandi moli di dati multivariati, con componenti correlate fra loro.*

Key words: big data, clustering, K-means, Mahalanobis distance

1 Introduction

Clustering is the division of a collection of data into groups, or *clusters*, such that points in the same cluster have a small distance from one another, while points in different clusters are at a large distance from one another. When the data are not very high dimensional, but are too many to fit in memory, because they are part of a huge dataset, or because they arrive in streams and must be processed immediately or they are lost, specific algorithms are needed to analyze progressively the data, store in memory only a small number of summary statistics, and then discard the already

G. Aletti, A.Micheletti
ADAMSS Center, Università degli Studi di Milano, Milano, Italy
e-mail: giacomo.aletti@unimi.it, alessandra.micheletti@unimi.it

processed data and free the memory. Situations like this, in which clustering plays a fundamental role, recur in many applications, like customer segmentation in e-commerce web sites, image analysis of video frames for objects recognition, recognition of human movements from data provided by sensors placed on the body or on a smartphone, etc. The key element in smart algorithms to treat such type of big data is to find methods by which the summary statistics that are retained in memory can be updated when each new observation, or group of observations, is processed. A first and widely recognized method to cluster big data is the Bradley-Fayyad-Reina (BFR) algorithm [1, 7], which is an extension of the classical K-means algorithm. The BFR algorithm responds to the following *data mining desiderata*: 1) Require one scan of the database and thus ability to operate on forward-only cursor; 2) Online anytime behavior: a "best" answer is always available, with status information on progress, expected remaining time, etc. provided; 3) Suspendable, stoppable, resumable; incremental progress can be saved in memory to resume a stopped job; 4) Ability to incrementally incorporate additional data with existing models efficiently; 5) Work within confines of a limited RAM buffer; 6)Utilize a variety of possible scan modes: sequential, index, and sampling scan, if available. The BRF Algorithm for clustering is based on the definition of three different sets of data: a) the *retained set* (RS): the set of data points which are not recognized to belong to any cluster, and need to be retained in the buffer; b) the *discard set* (DS): the set of data points which can be discarded after updating the sufficient statistics; c) the *compression set* (CS): the set of data points which form smaller clusters among themselves, far from the principal ones and can be represented with other sufficient statistics. Each data point is assigned to one of these sets on the basis of its distance from the center of each cluster. The main weakness of the BFR Algorithm resides in the assumption that the covariance matrix of each cluster is diagonal, which means that the components of the analyzed multivariate data should be uncorrelated. In this way at each step of the algorithm only the means and variances of each component of the cluster centers must be retained. In the following we will describe an extension of the BFR algorithm to the case of clusters having "full" covariance matrix. Since with our method also the covariance terms of the clusters centers must be retained, there is an increase in the computational costs, but such increase can be easily controlled and is affordable if the processed data are not extremely high dimensional.

2 An extension of the BFR clustering algorithm

We will use the same three sets of data a)-c) introduced in the BFR algorithm, but using different summary statistics to define the discard set and the compression set.

2.1 Data Compression

Like in the BFR algorithm, primary data compression determines items to be discarded (discard set DS), and updates the compression set CS with the sufficient summary statistics of the identified clusters. Secondary data-compression takes place over data points not compressed in primary phase. Data compression refers to representing groups of points by their sufficient statistics and purging these points from RAM. In the following we will always represent vectors as column vectors. Assume that data points $\mathbf{x}_1,\dots,\mathbf{x}_n \in \mathbb{R}^p$ must be compressed in the same cluster. We will retain only the sample mean $\bar{\mathbf{x}}_n = \sum_{i=1}^n \mathbf{x}_i$, and the unbiased sample covariance matrix $S_n = \frac{1}{n-1}\sum_{i=1}^n (\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^\top$. These two sufficient statistics can be easily computed by keeping in memory the following quantities:

$$n, \quad sumprod_{kl}(n) = \sum_{i=1}^{n} x_{ik}x_{il}, \quad sumprodcross_{kl}(n) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jl},$$

$$sumsq_k(n) = \sum_{j=1}^{n} x_{jk}^2, \quad sum_k(n) = \sum_{j=1}^{n} x_{jk}, \quad k,l = 1,\dots,p,\ k<l.$$

These sufficient statistics can be easily updated when a new data point $\mathbf{x}_{n+1}$ must be added to the cluster, without processing again the already compressed points. In fact, for $k,l = 1,\dots,n,\ k<l$, we have

$$\begin{aligned}
sumprod_{kl}(n+1) &= \sum_{i=1}^{n+1} x_{ik}x_{il} = sumprod_{kl}(n) + x_{(n+1)k}x_{(n+1)l} \\
sumprodcross_{kl}(n+1) &= \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} x_{ik}x_{jl} = sumprodcross_{kl}(n) + x_{(n+1)k}sum_l(n) \\
&\quad + x_{(n+1)l}sum_k(n) + x_{(n+1)k}x_{(n+1)l} \\
sumsq_k(n+1) &= \sum_{j=1}^{n+1} x_{jk}^2 = sumsq_k(n) + x_{(n+1)k}^2 \\
sum_k(n+1) &= \sum_{j=1}^{n+1} x_{jk} = sum_k(n) + x_{(n+1)k}
\end{aligned}$$

Thus at each step of the algorithm we have to retain in memory only p^2+p+1 sufficient statistics for each cluster, where p is the dimension of the data points. In addition, note that we should simply sum the corresponding statistics if we want to merge two clusters.

2.2 The covariance matrices of the clusters

Note that when a new cluster is formed, it contains too few data points to obtain a positive definite estimate of the covariance matrix, using the sample covariance matrix, at least until $n \le p$. This is a problem since we need to invert this matrix to

compute the Mahalanobis distance, that we will use to assign the observations to the clusters. Recent research methods in estimating covariance matrices include banding, tapering, penalization and shrinkage. We have focused on the Steinian shrinkage method since, as underlined in [8], it leads to covariance matrix estimators that are non-singular, well-conditioned, expressed in closed form and computationally cheap regardless of p. We use the diagonal matrix D_S of the sample covariance matrix S as "target matrix" of the shrinkage method, noting that D_S was the BRF estimate of the covariance of each cluster used in [1]. In other words, in presence of few data, our method coincides with that of [1], and we allow a progressive influence of correlation as the number of data increases. Summing up, we use a linear shrinkage estimator for the covariance matrix, like that proposed in [3, 4, 6, 8] of the form $\hat{S} = (1-\lambda)S + \lambda D_S$, where S is the sample covariance matrix, D_S is its diagonal matrix, and λ is a parameter in $[0,1]$, whose optimal value depends on the number n of data in the cluster. The parameter λ is initially settled to 1, and then its value is decreasing to 0 when $n \to \infty$. The theoretical optimal value λ^* of λ is found by minimizing the risk function relative to the quadratic loss $E[\|\hat{S} - \Sigma\|_F^2]$ (see, e.g., [8, 6]) and it is a ratio depending on the unknown Σ. When data are gaussian, the procedure proposed in [3] may be directly implemented to obtain unbiased estimators of numerator and denominator in the formula of λ^*. In non-gaussian setting, a bias due to the fourth moment is present in the numerator and it is corrected [6] with the use of further statistics, as the Q-statistics introduced in [4] (see also [2]). Unfortunately, it is not possible to compute the Q statistics on the basis of updatable sufficient statistics, as in our framework. To correct the bias, a new iterative procedure based on three updatable statistics for each cluster has been successfully developed.

2.3 Model update

Like in the BFR algorithm, the second step of our algorithm consists of performing K-means iterations over sufficient statistics of compressed, discarded and retained points. In order to assign a point to a cluster we use the Mahalanobis distance from its center (sample mean), i.e. we assign a new data point $\mathbf{x}$ to cluster h with center $\bar{\mathbf{x}}_h$ and estimated covariance matrix $\hat{S}_h$, if h is the index which minimizes $\Delta(\mathbf{x}, \bar{\mathbf{x}}_h) = (\mathbf{x} - \bar{\mathbf{x}}_h)^T (\hat{S}_h)^{-1} (\mathbf{x} - \bar{\mathbf{x}}_h)$, and if $\Delta(\mathbf{x}, \bar{\mathbf{x}}_h)$ is smaller than a fixed threshold δ. We also compare $\mathbf{x}$ with each point $\mathbf{x}_o$ in the retained set (RS), by computing $\Delta(\mathbf{x}, \mathbf{x}_o) = (\mathbf{x} - \mathbf{x}_o)^T (\hat{S}_P)^{-1} (\mathbf{x} - \mathbf{x}_o)$, where $\hat{S}_P$ matrix is the pooled covariance matrix based on all $\hat{S}_h$:

$$\hat{S}_P = \frac{(n_{h_1} - 1)\hat{S}_{h_1} + (n_{h_2} - 1)\hat{S}_{h_2} + \cdots (n_{h_M} - 1)\hat{S}_{h_M}}{n_{h_1} + n_{h_2} + \cdots + n_{h_M} - M}, \tag{1}$$

and where n_h is the number of points in cluster h. With $\hat{S}_P$, we emphasize the weighted importance of directions that are more significant for the clusters when we compute the distance between two "isolated" points. We then approximate locally the distribution of the clusters with a $p-$variate Gaussian and we build a confidence

regions around the centers of the clusters (see [5]). We then move $\bar{\mathbf{x}}_h$ in the farthest position from $\mathbf{x}$ in its confidence region, while we move the centers of the other clusters in the closest positions with respect to $\mathbf{x}$ and we check if the cluster center closer to $\mathbf{x}$ is still $\bar{\mathbf{x}}_h$. If yes, we assign $\mathbf{x}$ to cluster h, we update the corresponding sufficient statistics and we put $\mathbf{x}$ in the discard set; if the point is closer to a point $\mathbf{x}_o$ of the retained set than to any cluster, we form a new new secondary cluster (CS) with the two points and we put $\mathbf{x}$ and $\mathbf{x}_o$ in the discard set; otherwise, we put $\mathbf{x}$ in the retained set (RS).

2.4 Secondary data compression

The purpose of secondary data compression is to identify "tight" sub-clusters of points among the data that we can not discard in the primary phase. In [1], this is made in two phases. In the first one, a K-means algorithm tries to locate subclusters that are merged if they meet a "dense" condition. The candidate merging clusters are chosen sequentially based on a hierarchical agglomerative clustering build on the subclusters. In all this procedure, the euclidean metric was adopted. Finally, the number of clusters is initialized to K, and it can increase or decrease during the procedure. We adopt the same general idea, but we modify the procedure. First, we change the metric, by taking the pooled covariance $\hat{S}_P$ given in (1). As for isolated points, we think that this metric is more precise than the euclidean one for this stage. Then, a hierarchical clustering is performed using the Ward's method: the distance between two clusters h_1 and h_2 with n_{h_1}, n_{h_2} points and centroids $\bar{\mathbf{x}}_{h_1}$ and $\bar{\mathbf{x}}_{h_2}$, is given by

$$\Delta(A,B) = \frac{n_{h_1} n_{h_2}}{n_{h_1} + n_{h_2}} (\bar{\mathbf{x}}_{h_1} - \bar{\mathbf{x}}_{h_2})^\top \hat{S}_P (\bar{\mathbf{x}}_{h_1} - \bar{\mathbf{x}}_{h_2}).$$

Note that we sequentially merge two clusters only if a suitable dense condition is fulfilled. For example, the total variance (i.e., the trace of the sample covariance matrix) of the union of the two is required to be smaller than a suitable proportion of the sum of the total variances of the single groups.

3 Results on simulated data

Synthetic data were created for the cases of 5 and 20 clusters. Data were sampled from 5 or 20 independent p-variate Gaussians, with elements of their mean vectors (the true means) uniformly distributed on $[-5,5]$. The covariance matrices were generated by computing products of the type $\Sigma = UHU^T$, where H is a diagonal matrix with elements on the diagonal uniformly ditributed on $[0.7,1.5]$, and U is the orthonormal matrix obtained by the singular value decomposition of a symmetric matrix MM^T, where the elements of the $p \times p$ matrix M are uniformly distributed on $[-2,2]$. In either cases of 5 or 20 clusters, we generated 10.000 vectors for each cluster, having dimensions $p = 10, 20, 50$. This procedure guarantees that these clusters are fairly well-separated Gaussians, an ideal situation for K-Means. We applied

our procedure to these synthetic data, and we computed the secondary data compression after each bucket of 50 or 100 data points. The results are reported in Table 1. We note that the number of clusters is sometimes overestimated, in particular when the dimension p of the data points is small, which corresponds to the case where the clusters are less separated. In such cases, if the point clouds in different clusters are gathered in particularly "elongated" and rather close ellipsoids, then the correct detection of the clusters may be more difficult. We also note that in case of overestimation of the number of clusters, many of them are composed by 2 or 3 data points, which can then be revisited as small groups of outliers. The method seems to be almost unsensitive to the buckets size. We conclude that the method here proposed provides rather good results on synthetic data, even if some improvement could be considered for the secondary data compression. The method is also under testing on real data. An accurate comparison with the BFR algorithm will also be performed.

n. of true clusters	dimension p of data points	n. of data in each bucket	n. of estimated clusters	n. of small clusters	n. of retained points (outliers)
5	10	50	7	1	0
5	20	50	5	0	1
5	50	50	5	0	0
5	10	100	8	1	0
5	20	100	5	0	1
5	50	100	5	0	0
20	10	50	29	6	8
20	10	100	29	6	8

Table 1 Results of the application of the proposed algorithm to synthetic data. By small clusters we mean clusters containing less than 4 data points

References

1. Bradley, P.S., Fayyad, U.,Reina, C.: Scaling clustering to large databases, in KDD-98 Proceedings, pp 1-7, American Association for Artificial Intelligence, (1998) .
2. Chen, S. X., Zhang, L.-X., Zhong, P.-S.: Tests for high-dimensional covariance matrices. *J. Amer. Statist. Assoc.*, 105(490):pp. 810-819, (2010).
3. Fisher, T. J., Sun, X.: Improved Stein-type shrinkage estimators for the high-dimensional multivariate normal covariance matrix. *Comput. Statist. Data Anal.*, 55(5):pp. 1909-1918, (2011).
4. Himeno T., Yamada, T.: Estimations for some functions of covariance matrix in high dimension under non-normality and its applications. *J. Multivariate Anal.*, 130:27-44, (2014).
5. Hotelling, H.: The generalization of student's ratio. *Ann. Math. Statist.*, 2(3):360-378, 08 (1931).
6. Ikeda, Y., Kubokawa, T., Srivastava, M. S.: Comparison of linear shrinkage estimators of a large covariance matrix in normal and non-normal distributions. *Comput. Statist. Data Anal.*, 95:95-108, (2016).
7. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive datasets, Cambridge University Press (2014).
8. Touloumis, A.: Nonparametric Stein-type shrinkage covariance matrix estimators in high-dimensional settings. *Comput. Statist. Data Anal.*, 83:251-261, (2015).