FEMA: A Fast Expectation Maximization Algorithm based on Grid and PCA

Zhiwen Yu
Department of Computer Science
City University of Hong Kong
yuzhiwen@cs.cityu.edu.hk

Hau-san Wong
Department of Computer Science
City University of Hong Kong
cshswong@cityu.edu.hk

Abstract

EM algorithm is an important unsupervised clustering algorithm, but the algorithm has several limitations. In this paper, we propose a fast EM algorithm (FEMA) to address the limitations of EM and enhance its efficiency. FEMA achieves low running time by combining principal component analysis(PCA), a grid cell expansion algorithm(GCEA) and a hierarchical cluster tree. PCA and multi-dimensional grid are applied to find a set of "good" initial parameters for the EM algorithm, while the hierarchical cluster tree deals with the case where the cluster is concave by making use of a merging algorithm. The experiments indicate that FEMA outperforms EM by reducing 45% *of the CPU time.*

1. Introduction

EM algorithm is a fundamental unsupervised learning algorithm in pattern analysis, machine learning, data mining, database, multimedia and human motion analysis ([1], [2], [3], [4], [5], [6],[7]). Unfortunately, EM algorithm has several limitations: (i) the number of the clusters has to be pre-determined; (ii) the initial parameters of EM influences the performance of the algorithm; (iii) it does not work as well for the concave clusters. To solve case (i), EM algorithm first selects a large number of clusters. Then, it combines with an agglomerative clustering strategy to estimate the actual number of clusters based on measures such as the Minimum Description Length (MDL). To solve case (ii), subsampling, voting and two-stage clustering ([6]) are proposed to find "good" initial parameters of EM. Rough-set theory([7]) has also been applied to obtain the parameters of the mixture model. To solve case (iii), the minimal spanning tree(MST) is applied to merge the clusters according to the Mahalanobis distance. But few previous works consider the three limitations of EM algorithm at the same time. In this paper, we propose a fast EM algorithm (FEMA) which combines principal component analysis, a grid cell expansion algorithm(GCEA) and a hierarchical cluster tree to address the limitations of EM algorithm. PCA and GCEA are applied to find a set of "good" initial parameters for EM algorithm, while the hierarchical cluster tree deals with the case where the clusters are concave by applying a merging approach.

2. Overview of the algorithm

The objective of fast expectation maximization algorithm (FEMA) is to alleviate the above limitations of the EM algorithm. Figure 1 illustrates the framework of FEMA. FEMA first transforms the data points from high dimension to low dimension by PCA. Then, the data points are clustered by a grid cell expansion algorithm(GCEA) in an approximate way. In the third step, EM algorithm is applied to estimate the parameters of the mixture models according to the initial parameters obtained by GCEA. At the last stage, a hierarchical cluster tree is proposed to manage the clusters.

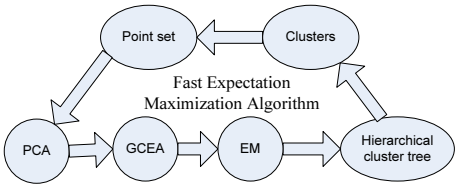


Figure 1. The framework of FEMA

2.1. Principal component analysis

The data points in a point set P are first transformed from a high dimensional space to low dimensional space by principle component method(PCA) as described in Figure 2 (where d is the original number of dimensions of the data points and l is the reduced number of dimensions. The output of the PCA algorithm is the input of the grid cell expansion algorithm.

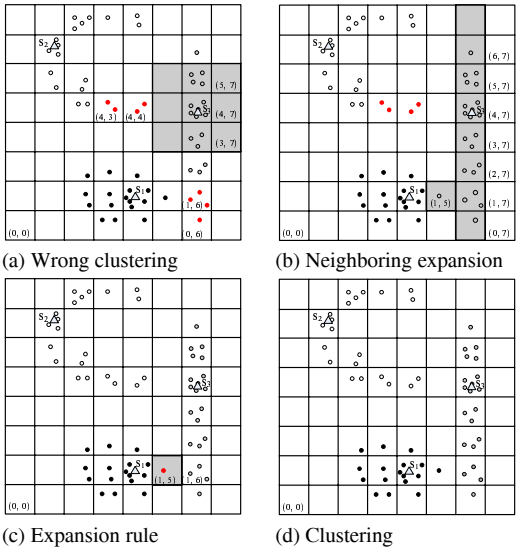
Algorithm **PCA**(Point set P, the dimensions l)
1. Compute the $d \times d$ covariance matrix of P;
2. Obtain the eigenvectors and eigenvalues of the covariance matrix;
3. Transform the data points to a new coordinate system which consists of the l eigenvectors with the maximum eigenvalues;

Figure 2. PCA algorithm

2.2. Grid cell expansion algorithm

In order to reduce the running time of EM algorithm, the initial parameters of the mixture model should be close to the actual parameters. A grid cell expansion algorithm (GCEA) is proposed to

obtain a set of "good" initial parameters. Specifically, GCEA obtains the initial parameters by approximately partitioning the data points into several clusters in an efficient way.



(a) Wrong clustering (b) Neighboring expansion

(c) Expansion rule (d) Clustering

Figure 3. Grid cell expansion algorithm

Figure 3 illustrates an example of applying the grid cell expansion algorithm. The algorithm first (i) determines the cells which correspond to the seeds (a seed is a grid cell which contains more points than its neighboring cells). There are three seeds $c(4,6), c(1,4), c(6,1)$ (The triangles in Figure 3(a)) in the example. Then, it considers these three seeds one by one. The first seed is the grid cell $c(4,6)$. The algorithm (i) initializes a heap H, (ii) performs the assignment $c(4,6).seed = s_3$(where $.seed$ stores the cluster which the grid cell belongs to, $c(4,6).seed = s_3$ denotes the cell $c(4,6)$ belongs to the cluster s_3), (iii) inserts the cell $c(4,6)$ with cardinality $|c(4,6)|$ into H. It then performs a de-heap operation recursively. The first de-heaped entry is the cell $c(4,6)$. The cell $c(4,6)$ determines which neighboring cells c_j (the blue cells in Figure 3(a)) should be expanded (The expansion implies that the neighboring cells now belong to the same cluster with the cell $c(4,6)$). There are constraints to restrict the expansion: (i) c_j should not be empty; (ii) $|c_j| \leq |c(4,6)|$; (iii) $|c_j.pre| < |c(4,6)|$ (For the case where every cell is expanded by several cells, $c_j.pre$ records the cell which has maximum cardinality among these cells by $c_j.pre$). Only the neighboring cells which satisfy the expansion conditions can be expanded. The cells $c(3,6)$ and $c(5,6)$ satisfy the conditions. As a result, $c(3,6).seed = s_3$, $c(5,6).seed = s_3$, $c(3,6).pre = c(4,6)$, $c(5,6).pre = c(4,6)$. The algorithm inserts the cells $c(3,6)$ and $c(5,6)$ into the heap H. The next de-heaped entry is the cell $c(5,6)$. The above operations are repeated in the case of the cell $c(4,6)$. The loop terminates when H is empty. When H is empty, the cluster s_3 includes the cells $c(6,6), c(5,6), c(4,6), c(3,6), c(2,6), c(1,6), c(1,5)$(The blue cells in Figure 3(b)). The next cell $c(1,4)$ with $c(1,4).seed = s_1$ is then expanded. Its neighboring cell $c(1,5)$ in the Figure 3(c) changes its owner from s_3 to s_1 since $|c(1,4)| > |c(1,5).pre| = |c(1,6)|$(In other words, the cell $c(1,5)$ satisfies the expansion conditions). After these three cells $c(4,6), c(1,4), c(6,1)$ are expanded, all the non-empty grid cells have their owners(seeds). The points in the grid cells are assigned to the owners that the grid cells belong to. At last, the points are divided into three clusters as shown in Figure 3(d).

Figure 4 provides an overview of the grid cell expansion algorithm. GCEA distributes the data points into several clusters in an approximate way, which becomes the input of EM algorithm. **Note that**, the user can control the final number of clusters by the seed number parameter k. If the user does not provide the value of k, the algorithm finds all possible clusters.

Algorithm **GCEA**(Point set P, seed number k)
1. Initialize an empty Max-heap H and a seed list L;
2. Hash all the points into the grid G;
3. Calculate the number of the points ($|c_j|$) in each grid cell c_j;
   ($1 \leq j \leq n_c$, n_c is the number of the grid cells)
4. Insert all the grid cells c_j with cardinality $|c_j|$ into the heap H;
5. **While** (H is not empty or $|L| < k$)
6. get the next entry c of H;
7. flag = true;
8. **For** each neighboring cell c_i of c
9. **If** ($|c_i| > |c|$) flag = false;
10. **If**(flag) add c into L;
11. **For** each grid cell c_i in L
12. Initialize an empty Max-heap H;
13. $c_i.seed = s_i$;
14. Insert the cell c_i into H;
15. **Repeat**
15. get the next entry c from H;
16. **For** each neighboring cell c_j of c;
17. **If** ($|c_j| \neq 0$ && $|c_j| \leq |c|$ && $|c_j.pre| < |c|$)
18. $c_j.seed = c.seed; c_j.pre = c$;
19. Insert c_j into H;
20. **Until** H is empty;
21. Assign the data points in the grid cells to the clusters which the grid cells belong to;

Figure 4. Grid cell expansion algorithm

2.3. Expectation Maximization algorithm

Table 1 illustrates the parameters of the EM algorithm. EM algorithm(Figure 5) first initializes the parameters of the mixture model according to the approximate clusters which is obtained through GCEA. In the first iteration(t=1), $\pi_j^{(1)}$ and $\mu_j^{(1)}$ are computed by Eqs. (1) and (2) respectively, while the entries of $\Sigma_j^{(1)}$ are evaluated based on Eq(3).

$$\pi_j^{(1)} = \frac{|c_j|}{\sum_{j=1}^{n_c}(|c_j|)} \tag{1}$$

$$\mu_j^{(1)} = \frac{\sum_{x \in c_j} x}{|c_j|} \tag{2}$$

$$Cov_j(k,l) = \frac{\sum_{x \in c_j}(x_k - \overline{x}_k)(x_l - \overline{x}_l)}{|c_j|} \tag{3}$$

Parameter	Meaning
x_i	The ith data point
$\|c_j\|$	The cardinality of the cluster c_j
n_c	The number of the clusters in the initial step
π_j	Mixing proportion of the cluster c_j
μ_j	the mean vector of the cluster c_j
Σ_j	covariance matrix of the cluster c_j
$Cov_j(k,l)$	covariance between k and l dimension of the cluster c_j
$\overline{x}_k$	the average value of the data points in the kth dimension
θ	the parameters to be estimated
t	the tth iteration
n	the cardinality of the point set P

Table 1. The parameter of the EM algorithms

After that, the algorithm performs the E-step. The probability that the data point x_i belongs to the cluster c_j in the tth iteration is computed by the Bayes rule as follows.

$$p^{(t)}(c_j|x_i) = \frac{p(x_i|c_j)}{\sum_{l=1}^{n_c} p(x_i|c_l)} \tag{4}$$

$$p^{(t)}(x_i|c_j) = \frac{1}{\sqrt{(2\pi)^d|\Sigma_i|}} e^{[-\frac{1}{2}(x_i-\mu_j)^T(\Sigma_j)^{-1}(x_i-\mu_j)]} \tag{5}$$

The sum of the probabilities is given by:

$$s_j^{(t)} = \sum_{x\in P} \frac{p(x_i|c_j)}{\sum_{l=1}^{n_c} p(x_i|c_l)} \tag{6}$$

During the M-step, the parameters of the mixture model are updated by

$$\pi_j^{(t+1)} = \frac{s_j^{(t)}}{n} \tag{7}$$

$$\mu_j^{(t+1)} = \frac{1}{s_j^{(t)}} \sum_{i=1}^{n} x_i \cdot p^{(t)}(c_j|x_i) \tag{8}$$

$$\Sigma_j^{(t+1)} = \frac{1}{s_j^{(t)}} \sum_{i=1}^{n} (p^{(t)}(c_j|x_i))(x_i - \mu_j^{(t+1)})(x_i - \mu_j^{(t+1)})^T \tag{9}$$

The algorithm performs the operations of E-step and M-step alternately until the change of log likelihood of the mixture model ($|L^{t+1}(\theta|P) - L^t(\theta|P)|$) is smaller than a threshold ϵ, where

$$L(\theta|P) = \sum_{x\in P} log(\sum_{j=1}^{n_c} \pi_j \cdot p(x|c_j)) \tag{10}$$

Finally, the algorithm assigns the data points to their corresponding clusters.

Algorithm **EM**(Data point set P, threshold ϵ)
1. Initialize the parameters of mixture models;
2. Repeat
3. E step: Calculate the likelihood;
4. M step: Update the parameter of the mixture model;
5. Until $|L^{t+1}(\theta|P) - L^t(\theta|P)| < \epsilon$
6. Assign the data points to the corresponding clusters;

Figure 5. EM algorithm

2.4. Hierarchical cluster tree

If the user does not provide a specific value k (k is the number of the clusters), EM generates a set of small clusters since the initial k value of EM is determined by GCEA. If the size of the grid cell is small, the number of clusters is large. To organize the clusters in a more orderly way, hierarchical cluster tree is proposed to index the clusters.

The hierarchical clustering tree is constructed through a merging approach. In the first step, the merging algorithm measures the similarity of the clusters by the evaluation function $sim(c_i, c_j)$:

$$sim(c_i, c_j) = \omega_1 \cdot d(c_i, c_j) + \omega_2 \cdot o(c_i, c_j) \quad (i \neq j) \tag{11}$$

$$d(c_i, c_j) = (\mu_i - \mu_j)^T(\Sigma_i + \Sigma_j)^{-1}(\mu_i - \mu_j) \tag{12}$$

$$o(c_i, c_j) = \sum_{l=0}^{d-1} (c_i^l.orientation - c_j^l.orientation)^2 \tag{13}$$

where μ_i, μ_j, Σ_i, Σ_j are the means and variances of the clusters c_i and c_j. $d(c_i, c_j)$ is the Mahalanobis distance between the clusters c_i and c_j, and $o(c_i, c_j)$ denotes the similarity of the orientation of the clusters. The orientation of the cluster is defined as the orientation with the maximum variance which is computed by Singular Value Decomposition(SVD). l denotes the l-th dimension.

The merging algorithm merges the two clusters (c_i^*, c_j^*) with the minimum value $sim(c_i, c_j)$ in the second step, i.e.

$$(c_i^*, c_j^*) = argmin_{c_i,c_j} \quad sim(c_i, c_j) \quad (i \neq j) \tag{14}$$

$$1 \leq i \leq n_c, 1 \leq j \leq n_c \tag{15}$$

where n_c is the number of the clusters. It performs the above operations repeatedly until there is only one cluster left. Figure 6 illustrates an example hierarchical cluster tree (here we assume k = 5). The circles denote the clusters, while the red dotted lines denote the process of merging. Cluster c_4 and cluster c_5 in Figure 6 are first merged by the algorithm based on their similarity to form the new cluster c_6. In a similar way, we obtain the following merged cluster pairs are $\{(c_2, c_3), (c_7, c_6), (c_1, c_8)\}$. The algorithm repeats this process until a single cluster c_9 is created which is the root note of the hierarchical cluster tree. **Note that**, every cluster in the hierarchy tree is only stored once.

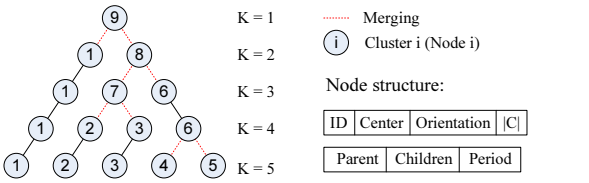


Figure 6. Hierarchical cluster tree

The node structure of the hierarchical cluster tree is shown in the lower right corner of Figure 6. The node stores the id, the center, the orientation, and the cardinality$|C|$ of the clusters. It also stores the pointer to the parent and the children respectively. The element Period stores the valid range of k values of the cluster. For example, the period of the cluster c_4 is $[5, 5]$, which means that if the user wants to obtain 5 clusters, c_4 will be one of the clusters.

Similarly, the period of the clusters c_1 and c_6 are $[2, 5]$ and $[3, 4]$ respectively. The user can retrieve arbitrary set of k clusters efficiently based on the hierarchical cluster tree ($1 \leq k \leq K_{max}$, K_{max} is 5 in Figure 6). The algorithm only traverses the hierarchical cluster tree once and returns all the clusters whose periods intersect with k value (e.g., if the user gives $k = 2$ in Figure 6, only the cluster c_1 with period $[2, 5]$ and c_8 with period $[2, 2]$ satisfy the requirement).

3. Experiment

All the experiments presented are executed with a Pentium 2.8 GHz CPU with 1 GByte memory. We generate two 3D data sets(10000 points) with arbitrary shapes of the distribution in the unit space $[0, 10000]^3$ by a Gaussian random variable generator (The centers and the standard deviations of the clusters are randomly selected). "Gaussian(10)" denotes the dataset with 10 Gaussian clusters with randomly selected centers and standard deviations. We apply FEMA and EM to cluster the color of the pixels for image segmentation. The images come from the bird(600 images) and butterfly(619 images) categories in the ponce group database [8]. The grid cell size is $400 \times 400 \times 400$ for the generated dataset and $5 \times 5 \times 5$ for image segmentation. The top row in Figure 7 shows a set of synthetic datasets generated by the gaussian generator and some images, while the bottom row illustrates partial segmentation results of the bird and butterfly dataset. It can be observed that the segmentation based on FEMA and EM are indistinguishable, while FEMA results in a significant reduction in computation time.

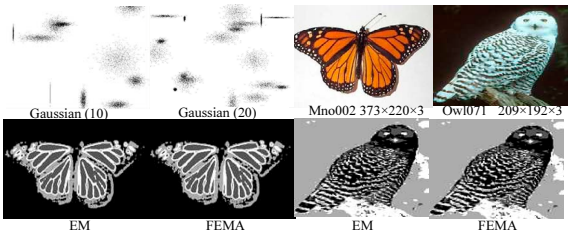


Figure 7. Dataset

We (i) perform FEMA and EM on the images, (ii) associate a hierarchical tree for each image (iii) select the best k value for each image by minimal description length(MDL). The average times for processing one image are $27.23(s)$ and $48.35(s)$ by FEMA and EM respectively (here we set the maximum k value to be equal to 20 for EM). It is obviously that FEMA outperforms its competitor for all the datasets by reducing 45% of CPU time. Because the initial parameters of mixture models in FEMA are close to the actual parameters, (i) the number of iterations decreases and (ii) the running time of the algorithm is reduced as well. Figure 8 shows the average results for the performance of FEMA and EM algorithm based on the synthetic datasets and the images in the categories of Butterfly_open, Zebra, Wood_duck, Owl in the ponce group database.

Figure 9 shows the average number of iterations for the different datasets. As expected, the number of iterations in FEMA is

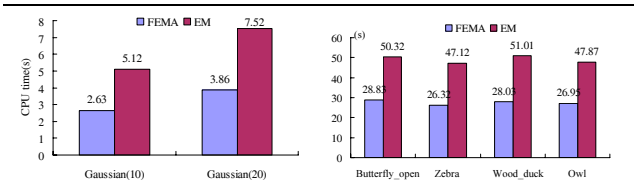


Figure 8. The running time

smaller than that in EM.

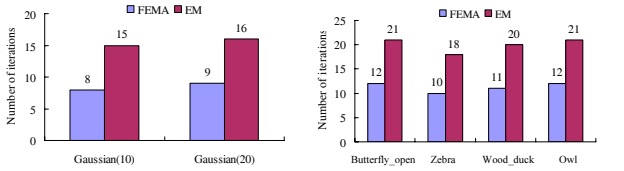


Figure 9. Iterations

The average classification error rates of FEMA and EM on the synthetic dataset are 5.67% and 5.68% respectively.

4. Conclusion and future work

This paper investigates the problem of reducing the running time of EM algorithm for data classifi cation. Our contribution is the introduction of a fast EM algorithm(FEMA) which achieves low running time by combining PCA, a grid cell expansion algorithm (GCEA) and a hierarchical cluster tree. The objective of FEMA is to reduce the number of iterations by fi nding initial parameters which are close to the actual parameters. In the future, we shall combine FEMA with other algorithms to classify the data in very large datasets.

Acknowledgments

The work described in this paper was partially supported by grants from the Research Grants Council of Hong Kong Special Administrative Region, China [Project No. CityU 1197/03E and CityU 121005] and grants from City University of Hong Kong [Project No. 7001596 and 9360091].

References

[1] Pernkopf, F., Bouchaffra, D.,Genetic-based EM algorithm for learning Gaussian mixture models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 27, Issue 8, Aug. 2005 Page(s):1344 - 1348.

[2] Govaert, G., Nadif, M.,An EM algorithm for the block mixture model,IEEE Transactions on Pattern Analysis and Machine Intelligence,Volume 27, Issue 4, April 2005 Page(s):643 - 647 .

[3] Anthony K. H. Tung, Xin Xu, Beng Chin Ooi. CURLER: Finding and Visualizing Nonlinear Correlated Clusters. ACM SIGMOD Conference. 2005.

[4] Yang Song; Goncalves, L., Perona, P., Unsupervised learning of human motion,IEEE Transactions on Pattern Analysis and Machine Intelligence,Volume 25, Issue 7, July 2003 Page(s):814 - 827

[5] Bin Luo, Hancock, E.R.,Structural graph matching using the EM algorithm and singular value decomposition , IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 23, Issue 10, Oct. 2001 Page(s):1120 - 1136

[6] M. Meila and D. Heckerman, An experimental comparison of several clustering and initialization methods, Microsoft, Redmond, WA, Microsoft Res. Tech. Rep. MSR-TR-98-06.

[7] Pal, S.K., Mitra, P.,Multispectral image segmentation using the rough-set-initialized EM algorithm , IEEE Transactions on Geoscience and Remote Sensing, Volume 40, Issue 11, Nov. 2002 Page(s):2495 - 2501

[8] $http://www-cvr.ai.uiuc.edu/ponce\_grp/data/$