Bag-of-Components: An Online Algorithm for Batch Learning of Mixture Models

Olivier Schwander[1,2(✉)] and Frank Nielsen[3]

[1] Laboratoire des Signaux et Systèmes (L2S, UMR CNRS 8506), CentraleSupélec-CNRS-Université Paris-Sud, Orsay, France
olivier.schwander@ens-lyon.org
[2] Viper Group, Computer Vision and Multimedia Laboratory, University of Geneva, Geneva, Switzerland
[3] Laboratoire D'Informatique (LIX, UMR 7161), École Polytechnique, Palaiseau, France

Abstract. Practical estimation of mixture models may be problematic when a large number of observations are involved: for such cases, online versions of Expectation-Maximization may be preferred, avoiding the need to store all the observations before running the algorithms. We introduce a new online method well-suited when both the number of observations is large and lots of mixture models need to be learned from different sets of points. Inspired by dictionary methods, our algorithm begins with a training step which is used to build a dictionary of components. The next step, which can be done online, amounts to populating the weights of the components given each arriving observation. The usage of the dictionary of components shows all its interest when lots of mixtures need to be learned using the same dictionary in order to maximize the return on investment of the training step. We evaluate the proposed method on an artificial dataset built from random Gaussian mixture models.

1 Introduction and Motivation

The problem of estimating the probability density function of an unknown probability law is old and well-studied and among all the techniques used mixture models are particularly widespread in practical applications. A lot of work is thus devoted to the improvement of the speed of the algorithms for mixture parameters estimation, which is of particular interest in real-time applications such as object tracking in videos [4,9,10].

The most common axes of research for mixture models can be divided into three main categories. First, the goal may be to reduce the computational burden of the algorithm itself: for example k-MLE [7] and cEM [3] are fast variants of EM where the slow step of soft assignment is replaced by a fast step of hard assignment. Second, a work on the input data may be done: in [5], coresets are used to reduce the number of points needed to build the model. Third, online algorithms can be designed to deal more easily with large datasets [2,10], avoiding the need to store all the content of the dataset.

F. Nielsen and F. Barbaresco (Eds.): GSI 2015, LNCS 9389, pp. 387–395, 2015.
DOI: 10.1007/978-3-319-25040-3_42

We take here a slightly different point of view: we address both the massive data problem and the online constraint in the case where a large number of different mixtures from quite similar sets of points are needed. As such, our new algorithm is divided into two steps: a first training step (which can be slow but it does not really matter since it will be done only once) is used to build a dictionary of components (where atoms are the parameters of the distributions), and a second step uses a nearest-neighbor search to associate each incoming observation to the most probable component, thus incrementally populating the vectors of weights of the mixture. This learning step is obviously online, since the processing can be done observation by observation, and is faster than Expectation-Maximization (EM), since the nearest-neighbor search is rather simple compared to a full-blown EM.

We believe that the separation between a training step and learning step for mixture model can be very useful in numerous applications. For example, in a video analysis application (on a MPEG compressed stream), the dictionary can be built on a key-frame and inter-frames can be modeled using the dictionary of the corresponding key-frame: the dictionary learned on a key-frame will be well suited for the following images but a new one will be needed if a too different scene appears.

Our contributions are the following: first we define the co-mixture concept and present an Expectation-Maximization based algorithm, called *co-EM*, to estimate the parameters; then we introduce an online algorithm, called *Bag of Components* , which relies on a co-mixture to learn a dictionary of components and uses a nearest-neighbor search to estimate a new mixture from observations arriving one by one.

This article is organized as follows: the first part describes the co-mixtures and the algorithm co-EM; the second part introduces the Bag of Components and the online algorithm; the next part discusses some improvements over the basic algorithm; and the last part shows some experimental results.

2 Co-mixture Models

We formally define a co-mixture of exponential families as a set of S mixture models sharing the same parameters for theirs components:

$$\begin{cases} m_1(x;\omega_1^{(1)}\dots\omega_K^{(1)}) = \sum_{i=1}^K \omega_i^{(1)} p_F(x;\theta_i) \\ m_2(x;\omega_1^{(2)}\dots\omega_K^{(2)}) = \sum_{i=1}^K \omega_i^{(2)} p_F(x;\theta_i) \\ \dots \\ m_S(x;\omega_1^{(S)}\dots\omega_K^{(S)}) = \sum_{i=1}^K \omega_i^{(S)} p_F(x;\theta_i) \end{cases} \tag{1}$$

where p_F is the exponential family with log-normalizer F and $\theta_1 \dots \theta_K$ are the parameters of the components and are shared between all the individual mixtures of the co-mixture; the S vectors $\omega_1^{(s)} \dots \omega_K^{(s)}$ are the vectors of weights (thus positive and normalized to 1).

In the previous expressions, all the mixtures have the same number of components but since the weight associated to a component may be zero, it is not a limitation.

In order to build such a set of mixtures from a dataset made of S sets of point $\mathcal{X}^{(l)} = \{x_1^{(l)}, \ldots, x_{n_l}^{(l)}\}$ (where n_l is the number of observations in the set of points $\mathcal{X}^{(l)}$), we design an EM-based iterative algorithm, called co-EM. For clarity, we write a generic version working for any exponential family: it is a variant of Bregman Soft Clustering [1] for which the maximization is simply an arithmetic mean in the expectation parameters space (which is in bijection with the usual parameters). It can be described by three main steps:

- Expectation step,
- Maximization step (set of points by set of points),
- Maximization step (aggregation).

Expectation Step. We compute S responsibility matrices $p^{(1)}, \ldots, p^{(S)}$: the coefficient $p^{(l)}(i,j)$ measures the likelihood for the observation $x_i^{(l)}$ from the set of points $\mathcal{X}^{(l)}$ to come from the j-th component of the mixture m_l given the current estimate of the parameters $\eta_1, \ldots, \eta_k$ and of the weights for the l-th mixture $\omega_1^{(l)}, \ldots, \omega_k^{(l)}$. In short, we have:

$$p^{(l)}(i,j) = \frac{\omega_j^{(l)} p_F(x_i^{(l)}, \eta_j)}{m(x_i^{(l)} | \omega^{(l)}, \eta)} \tag{2}$$

Maximization Step (Set of Points by Set of Points). In the first part of the maximization step S partial estimates $(\eta_1^{(1)}, \ldots, \eta_K^{(1)}), \ldots, (\eta_1^{(S)}, \ldots, \eta_k^{(S)})$ are made, one for each individual mixture of the comixture.

The new estimates $(\eta_1^{(l)}, \ldots, \eta_K^{(l)})$ for the l-th set of points are computed using the observations for $\mathcal{X}^{(l)}$ and the l-th responsibility matrix:

$$\eta_j^{(l)} = \sum_i \frac{p^{(l)}(i,j)}{\sum_u p^{(l)}(u,j)} \, t(x_i^{(l)}) \tag{3}$$

And the weights of each individual mixtures are updated with:

$$\omega_j^{(l)} = \frac{1}{n_l} \sum_{i=1}^{n_l} p^{(l)}(i,j) \tag{4}$$

Maximization Step (Aggregation). All these partial estimates are then aggregated into the new estimate of the parameters $\eta_1, \ldots, \eta_K$.

For the component j, the new estimate of η_j is computed with a Bregman barycenter of all the $\eta_j^{(1)}, \ldots, \eta_j^{(S)}$:

$$\eta_j = \frac{1}{S} \sum_{l=1}^{S} \eta_j^{(l)} \tag{5}$$

This aggregation step gives the same weight to all the set of points, no matter the number of components inside, allowing to remove the influence of various set of points sizes.

Fig. 1. Segmentation with regular EM and co-EM using a 5D RGBxy description of the images.

The algorithm co-EM converges to the average of the log-likelihoods on all the individual mixtures of the co-mixture and can be used independently of the Bag of Components: Fig. 1 shows an image segmentation application.

3 Bag of Components

This online algorithm is inspired by dictionary methods. As such, the training step amounts to building a dictionary of components (the atoms of the dictionary) and the learning step amounts to computing the activation of each atom given the observations.

The dictionary can be directly extracted from the output of co-EM (or from the output of any algorithm building a co-mixture). Given a co-mixture, the

dictionary is the set of parameters:

$$\mathcal{D} = \{\theta_1, \ldots, \theta_K\} \tag{6}$$

Due to the need to build a co-mixture, the training step is potentially expensive but this cost is counterbalanced by two points. First it is made only once and the results are reused during the learning step. Second, there is no overload if the set used to build the co-mixtures is a subset of the interesting sets of points: in this case, since it is not more costly to build a co-mixture of size S with co-EM than to learn S mixtures with EM, the global cost of the training and learning steps is still smaller than the cost of doing an EM on all the dataset.

The learning step can be done online: we do not need to work on the entire input points but we can rather update the model parameters each time we see a new observation. This step amounts to a hard-assignment step: given a new observation, we search the most probable component among the atoms of the dictionary (using a naive linear search):

$$\hat{\imath} = \arg\max_{\theta \in \mathcal{D}} p_F(x_i, \theta) \tag{7}$$

We then increment the value in the bin $\hat{\imath}$ of the histogram which counts how many observations have been associated to each atom. At the end of the processing, it is straightforward to go from the histogram to a real vector of weights by dividing by the total number of observations.

4 Improvements

The previous maximization problem can be rewritten as a nearest-neighbor search using the bijection between exponential families and Bregman divergences [1]:

$$\hat{\imath} = \arg\min_{\theta \in \mathcal{D}} B_{F^*}\left(t(x_i) \| \eta(\theta)\right) \tag{8}$$

where F^* is the Legendre dual of the log-normalizer F of the exponential family and $\eta(\theta)$ is the transformation of the natural parameter θ into the space of expectation parameters.

As such, it is possible to improve the linear time search described previously by using appropriate nearest-neighbor techniques and data structures such as Bregman ball tree [8] and to go below the linear time search.

Another possible variant is to enforce the sparsity of the weights: after the computation of the vector of weights, we are likely to have some components with a very low weight and thus carrying nearly no information. We assume we can remove these components by thresholding and renormalizing the weights. Another choice may be to clusterize the mixture using the k-medoids [6] algorithm to concentrate weights on most important components.

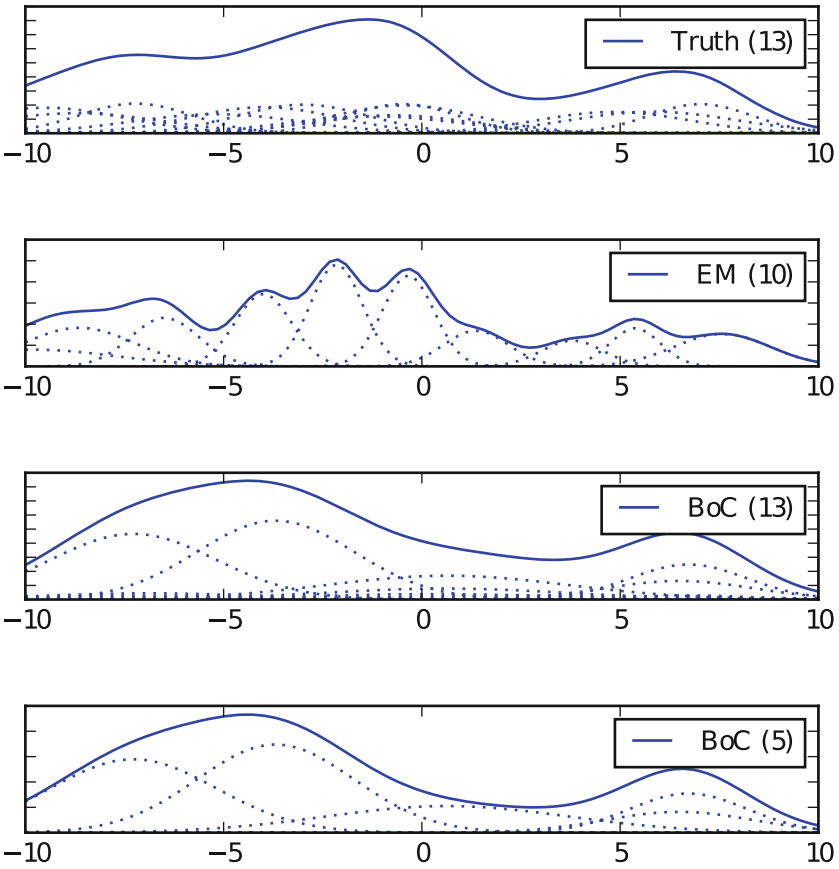

Fig. 2. From top to bottom: original mixture, Expectation-Maximization, raw Bag of Components, Bag of Components with weights thresholded. Between parentheses is the number of components with non-zero weights.

5 Experiments

We evaluate the Bag of Components algorithm on artificially generated mixture models. In order to generate mixture models sufficiently similar to use with a dictionary-based method , we first generate a dictionary of multivariate Gaussian distributions (the covariance matrices are generated from a LDL^T decomposition, where L is a triangular unit matrix and D a diagonal matrix with positive coefficients). We then generate mixtures by randomly drawing the weights, imposing that only a small fraction of the components has a non-zero weight (to enforce some diversity between the random mixtures).

In all the following experiments, the random mixtures are generated from a dictionary of size 30 with only 30 % of non-zero weights. co-EM builds a 30

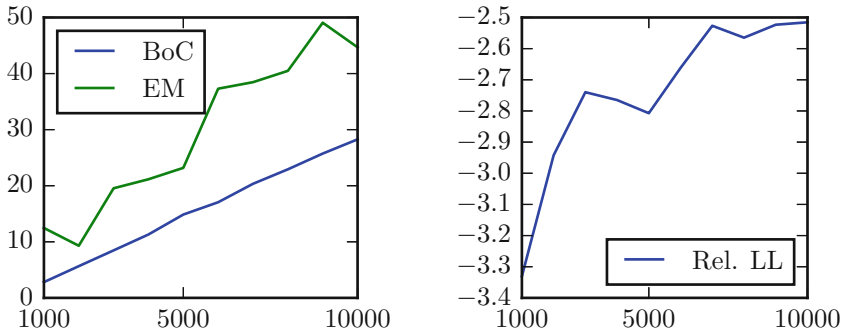

Fig. 3. Computation time for EM and BoC (left) and relative log-likelihood (in percent, right) with respect to the number of observations during the learning step (from 1000 to 10000, in dimension 5).

components co-mixture from 10 sets of 1000 points. The components of this co-mixture are used as a dictionary for Bag of Components.

The goal of the first experiment is to visually check the quality of a 1D mixture built with Bag of Components (from 1000 observations). Figure 2 compares the original mixture (first curve) with the output of EM (second curve, with 10 components) and the output of Bag of Components (third curve). On the third curve, some components have clearly a low weight compared to the most prominent Gaussians so in the fourth curve weights are thresholded under 0.06 in order to keep only 5 components: in this particular case, most of the information seem to be preserved.

A second experiment in Fig. 3 compares the execution time (left) and the quality (right) of the output of Bag of Components and of an EM (10 components) with respect to the number of points in the input set (from 1000 to 10000 points, in dimension 5). Given a dictionary built with co-EM during a preprocessing offline step, we build a mixture with the Bag of Components method from a new input set of points (not present in the dataset used for the dictionary learning step) and compare the output mixture to the result of a classical EM.

The quality of the mixtures from the two algorithms is compared using the relative log-likelihood $\frac{\mathrm{ll}_{\mathrm{BoC}} - \mathrm{ll}_{\mathrm{EM}}}{\mathrm{ll}_{\mathrm{BoC}}}$ so a negative value means Bag of Components produces worse mixtures than EM: on the explored range, the two algorithms produce mixtures of similar quality, with roughly between –4 % and –2 % of relative difference.

The left part of Fig. 3 measures the execution time of Bag of Components (without the dictionary building step, since this step is made offline): not surprisingly, it is perfectly linear with a speed-up between 1.2 and 4 compared to EM (which has a very irregular execution time).

The speed-up from EM to Bag of Components is real but not so high. Indeed, even if the learning step of Bag of Components is made in $O(nK)$ (where n is the number of observations and K the number of atoms of the dictionary) and

the EM is made in $O(nki)$ (where k is the number of components and i the number of iterations), the number of atoms K is higher than the number of components k (three times in the experiments). The execution time of Bag of Components is thus of the same order of magnitude than all the iterations of EM, giving an execution time which can be nearly the same when EM converges in few iterations. We may increase the speed of Bag of Components by using a Bregman Ball Tree which would allow a sub-linear nearest-neighbor search. Moreover, independently of the time, Bag of Components has the big advantage of being an online algorithm (so in the curves on Fig. 3, each point for Bag of Components is not a new mixture built from scratch, but only an improvement of the previous one).

6 Conclusion

We described the notion of co-mixtures along with the algorithm co-EM. It is used as a basis to design a new algorithm for mixture model learning, called Bag of Components: this new algorithm works online and allows to build a mixture faster than Expectation-Maximization. It is well suited when a lot of mixtures from related or similar sets of points are needed: in such a case, it is worth building a dictionary on a subset of the sets of points and apply Bag of Components on the remaining sets of points. It is also interesting when only a few sets of points are available at a time: the available sets can be used to learn the dictionary of components and new mixtures can be built on new sets of points at soon as they become available.

There is room for lots of improvements both on the speed, by using efficient nearest-neighbor or approximate nearest-neighbor techniques, and for the sparsity of the weights, by evaluating the need and the interest of removing low weight components. Furthermore, we leave for future work to validate co-EM and Bag of Components on a real application instead of artificial mixtures.

References

1. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with Bregman divergences. J. Mach. Learn. Res. **6**, 1705–1749 (2005)
2. Cappé, O., Moulines, E.: On-line expectation-maximization algorithm for latent data models. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **71**(3), 593–613 (2009)
3. Celeux, G., Govaert, G.: A classification EM algorithm for clustering and two stochastic versions. Comput. Stat. Data Anal. **14**(3), 315–332 (1992). doi:10.1016/0167-9473(92)90042-E
4. Chen, G., Yu, Z., Wen, Q., Yu, Y.: Improved Gaussian mixture model for moving object detection. In: Deng, H., Miao, D., Lei, J., Wang, F.L. (eds.) AICI 2011, Part I. LNCS, vol. 7002, pp. 179–186. Springer, Heidelberg (2011)
5. Feldman, D., Faulkner, M., Krause, A.: Scalable training of mixture models via coresets. In: Advances in Neural Information Processing Systems, pp. 2142–2150 (2011)

6. Kaufman, L., Rousseeuw, P.: Clustering by means of medoids. North-Holland, Amsterdam (1987)
7. Nielsen, F.: k-MLE: a fast algorithm for learning statistical mixture models. CoRR (2012)
8. Nielsen, F., Piro, P., Barlaud, M.: Tailored bregman ball trees for effective nearest neighbors. In: Proceedings of the 25th European Workshop on Computational Geometry (EuroCG), pp. 29–32 (2009)
9. Sicre, R., Nicolas, H.: Improved Gaussian mixture model for the task of object tracking. In: Real, P., Diaz-Pernil, D., Molina-Abril, H., Berciano, A., Kropatsch, W. (eds.) CAIP 2011, Part II. LNCS, vol. 6855, pp. 389–396. Springer, Heidelberg (2011). http://dx.doi.org/10.1007/978-3-642-23678-5_46
10. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1999, vol. 2, IEEE (1999)