Bayesian K-Means as a "Maximization-Expectation" Algorithm

October 18, 2007

Abstract

We introduce a new class of "maximization expectation" (ME) algorithms where we maximize over hidden variables but marginalize over random parameters. This reverses the roles of expectation and maximization in the classical EM algorithm. In the context of clustering, we argue that these hard assignments open the door to very fast implementations based on data-structures such as kd-trees and conga-lines. The marginalization over parameters ensures that we retain the ability to infer model structure (i.e. number of clusters). As an important example we discuss a top-down "Bayesian k-means" algorithm and a bottom-up agglomerative clustering algorithm. In experiments we compare these algorithms against a number of alternative algorithms that have recently appeared in the literature.

1 Introduction

K-means is undoubtedly one of the workhorses of machine learning. Faced with the exponential growth of data researchers have recently started to study strategies to speed up k-means and related clustering algorithms. Most notably, methods based on kd-trees [Pelleg and Moore, 1999, Moore, 1998, Zhang et al., 1996, Verbeek et al., 2003] and

methods based on the triangle inequality [Moore, 2000, Elkan, 2003] have been very successful in achieving orders of magnitude improved efficiency.

Another topic of intense study has been to devise methods that automatically determine the number of clusters from the data [Pelleg and Moore, 1999, Hamerly and Elkan, 2003]. Some of the most promising algorithms in this respect are based on the "variational Bayesian" (VB) paradigm [Attias, 2000, Ghahramani and Beal, 2000]. Here distributions over cluster assignments and distributions over stochastic parameters are alternatingly estimated in an EM-like fashion. Unfortunately, it is difficult to apply the speed-up tricks mentioned above to these algorithms, at least without introducing approximations. One of the main goals in this paper is to propose a modification of VB-clustering that combines model selection with the possibility for fast implementations.

The technique we propose is an instance of a new class of "ME algorithms" that reverses the roles of expectation and maximization in the EM algorithm. Alternatively, it can be viewed as a special case of the VB framework where expectation over hidden variables is replaced with maximization. We show that convergence is guaranteed by deriving the objective that is optimized by these iterations.

In the context of clustering we discuss an ME algorithm that is very similar to k-means but uses a full covariance and an upgraded "distance" to penalize overly complex models. We also derive an alternative agglomerative clustering algorithm. Both algorithms can be implemented efficiently using kd-trees and conga-lines respectively. Experimentally, we see no performance drops relative to VB but at the same time we demonstrate significant speedup factors. Our software is publicly available at *http://mi.cs.titech.ac.jp/kurihara/bkm.html*

2 Maximization-Expectation Algorithms

Consider a probabilistic model, $P(x, z, \theta)$, with observed random variables (RVs), x, hidden RVs , z, and parameters, θ (which are assumed random as well). We adopt the convention that RVs are called "hidden" if there is a hidden-RV for each data-case (e.g.

cluster assignment variables), while RVs are called parameters if their number is fixed or needs to be inferred from the data (e.g. cluster centroids).

Given a dataset $\mathcal{D}$, a typical task in machine learning is to compute posterior or marginal probabilities such as $p(z|\mathcal{D})$, $p(\theta|\mathcal{D})$ or $p(\mathcal{D})$. The former can be used to determine for instance a clustering of the data, when z represents cluster assignment variables. It is not untypical that exact expressions for these quantities can not be derived and approximate methods become necessary. We will now review a number of approaches based on alternating model estimation.

A standard approach is to represent the distribution using samples. The canonical example being a Gibbs sampler which alternatingly samples,

$$\theta \sim p(\theta|z, \mathcal{D}) \leftrightarrow z \sim p(z|\theta, \mathcal{D}). \tag{1}$$

Instead of sampling, one can fit factorized variational distributions to the exact distribution, i.e. $p(\theta, z|\mathcal{D}) \approx q(\theta)q(z)$. This "variational Bayesian" (VB) approximation [Attias, 2000, Ghahramani and Beal, 2000] alternatingly estimates these distributions by minimizing the KL-divergence, $KL[q(\theta)q(z)||p(\theta, z|\mathcal{D})]$ between the approximation and the exact distribution. This results in the updates,

$$q(\theta) \propto \exp(\mathbb{E}[\log p(\theta, z, \mathcal{D})]_{q(z)}) \leftrightarrow q(z) \propto \exp(\mathbb{E}[\log p(\theta, z, \mathcal{D})]_{q(\theta)}). \tag{2}$$

Instead of maintaining a distribution over parameters, one could decide to estimate a MAP value for θ. To find this MAP value, θ^*, one can use the expectation-maximization algorithm (EM) to alternatingly compute,

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ \mathbb{E}[\log(p(\theta, z, \mathcal{D})]_{q(z)} \leftrightarrow q(z) = p(z|\theta^*, \mathcal{D}). \tag{3}$$

This turns out to be a special case of the VB formalism by approximating the posterior as $q(\theta) = \delta(\theta, \theta^*)$, where $\delta(\cdot)$ is a delta function. Finally, one can also choose to use point-estimates for both θ and z, which is known as "iterative conditional modes"

(ICM). In this case we iteratively maximize the posterior distributions, $p(\theta|z,\mathcal{D})$ and $p(z|\theta,\mathcal{D})$ which is equivalent to,

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ p(z^*,\theta,\mathcal{D}) \leftrightarrow z^* = \underset{z}{\operatorname{argmax}}\ p(z,\theta^*,\mathcal{D}). \tag{4}$$

In figure 1 we have depicted the 4 algorithms discussed so far, Gibbs sampling, VB, EM, and ICM, together with some canonical examples. A particular choice among these methods often represents a trade-off between computational efficiency and accuracy, where ICM is the most efficient and VB and Gibbs sampling the least efficient of these methods. On the other hand, VB has the potential of model selection because it maintains distributions over its parameters. All methods except Gibbs sampling are asymptotically biased, but, unlike Gibbs sampling, do not suffer from variance in the estimates.

Viewing VB as an extension of EM, we could classify VB as an "expectation-expectation" (EE) algorithm while ICM should be interpreted as a "maximization-maximization" (MM) algorithm. This paper is about a new class of algorithms which we call "maximization-expectation" algorithms where we maximize over hidden variables but take expectations over parameters,

$$q(\theta) = p(\theta|z^*,\mathcal{D}) \leftrightarrow z^* = \underset{z}{\operatorname{argmax}}\ \mathbb{E}[\log(p(\theta,z,\mathcal{D})]_{q(\theta)}. \tag{5}$$

In analogy to the relation between EM and VB, we can interpret the ME algorithm as an approximation of the VB formalism where we use $q(z) = \delta(z,z^*)$. The main motivation behind this idea is to propose a class of algorithms that is computationally efficient on the one hand but has the necessary ingredients for model selection on the other. In particular for the clustering example under consideration, we will show that efficient data-structures such as kd-trees can be employed with relative ease. Although this paper focuses on the "Bayesian K-Means" algorithm we like to stress that our ideas are rather more general than this.

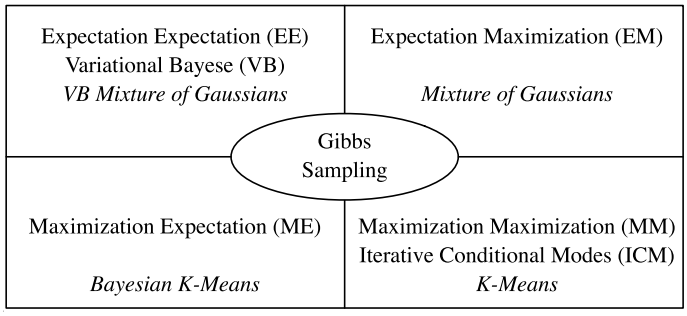

Figure 1: Four faces of alternating model learning: EE, EM, ME and MM algorithms

3 "Bits-Back" Coding

An elegant interpretation for the approximations discussed in the previous section is provided by the "bits-back" argument [Hinton and van Camp, 1993]. We imagine that we want to communicate the data to some receiver as compactly as we can. To that end, we first learn a probabilistic model of the data (say a "mixture of Gaussians" (MoG) model). We use the internal representation of the hidden (or latent, unobserved) variables to encode the data-vectors. For instance, the cluster assignments in a MoG can be used to vector-quantize the data-vectors. To communicate the data, we first send the parameters of the model that we learned. Next, we send the hidden representation for each data-vector. The receiver will predict the data vectors from this information, but since the code we send is "lossy" we still need to send (hopefully small) error corrections. Note that the sender and receiver need to agree on a quantization level (or resolution) with which they measure parameters, code vectors and data vectors. Given such a quantization level, it takes less bits to encode small error terms than to encode the entire data vectors themselves resulting in the desired compression.

The reason that the optimal compression results in model selection follows from the fact that the number of parameters that are sent is constant (they have to be send only once) while the hidden representation and the error terms scale with the number of data-vectors. Hence, when we have a few data vectors to communicate it doesn't pay back to train a complicated model: the bits needed to send the model parameters

may be more expensive than sending the data vectors directly. On the other hand, using a complicated model to encode the data is warranted if we have to communicate lots of data vectors (assuming there is structure to be found in the data). It is this delicate interplay that determines the complexity of the model that results in optimal compression.

According to Shannon [1948] we need $\log_2 p(\theta|M)$ bits to encode the parameters, $\log_2 p(z|\theta, M)$ bits to encode the hidden representation z (e.g. cluster assignments) and $\log_2 p(x|z, \theta, M)$ to encode the error terms for a model "M". Although one is inclined to think that the optimal model and encoding results from minimizing the sum of the 3 terms over z and θ (called energy), one can in fact do better. Consider for instance the case that two choices for z both minimize the energy. Since both options are optimal, one can use this additional freedom to encode some side information (or future data vectors). In effect, we got some bits refunded. More generally, we can use a distribution $q(z, \theta)$ to choose which z and θ we use for the encoding. This will result in an an average energy term of,

$$E[q] = \int \mathrm{d}\theta \sum_z q(z, \theta)(\log_2 P(x|z, \theta, M) + \log_2 P(z|\theta, M) + \log_2 P(\theta|M)).$$

However, the amount of information contained in our stochastic choices of the code and model parameters is,

$$H[q] = -\int \mathrm{d}\theta \sum_z q(z, \theta) \log_2 q(z, \theta). \tag{6}$$

Hence, the total amount of bits needed to communicate the data becomes $F[q] = E[q] - H[q]$ which is equal to the negative of the logarithm of the marginal likelihood (or evidence).

We are now ready to interpret the approximations made by the 4 algorithms in figure 1. The EM algorithm uses stochastic choices for the code vectors z but maximizes over the parameters θ declining the refund possible from random choices. The VB

formalism uses stochastic choices for both z and θ but uses a factorized distribution $q(z, \theta) = q(z)q(\theta)$ to choose them which is also suboptimal[1]. ICM will not receive any bits back from either z or θ because it maximizes over both of them. Finally, in the ME formalism we receive bits back from the parameters θ but since we optimize over the encoding z we decline a potential refund which we would have obtained through stochastic choices. The argument we are making in this paper is that this sacrifice will help us develop computationally efficient algorithms.

4 The Large N Limit

In the previous sections we have viewed the ME procedure as either an approximation to Bayesian model selection or as an approximation to the bits-back coding principle. In this section we will derive the result that for large N we recover the well known Bayesian information criterion (BIC), or equivalently the maximum description length (MDL) penalty.

The variational bound on the negative marginal log-likelihood is given by the following expression in case of the ME approximation,

$$\mathcal{F}(K) = \min_z \mathcal{F}(z, K) \tag{7}$$

$$\begin{aligned} \mathcal{F}(z, K) &= -\mathbb{E}[\log p(\mathcal{D}, \theta, z')]_{q(\theta)\delta(z', z)} - H[q(\theta)] \\ &= -\log p(\mathcal{D}) + KL[q(\theta)\delta(z', z)||p(\theta, z'|\mathcal{D})] \\ &\geq -\log p(\mathcal{D}) \quad \forall z, \quad \Rightarrow \mathcal{F}(K) \geq -\log(\mathcal{D}). \end{aligned} \tag{8}$$

We can rewrite $\mathcal{F}(z, K)$ as follows,

$$\mathcal{F}(z, K) = \int \mathrm{d}\theta \left[-q(\theta) \log p(\mathcal{D}, z|\theta) - q(\theta) \log p(\theta) + q(\theta) \log q(\theta)\right]. \tag{9}$$

Since we fixed z, we can treat it as observed data alongside $\mathcal{D}$. Then, as N grows sufficiently large, $p(\mathcal{D}, z|\theta)$ will concentrate on the maximum likelihood solution for the

[1]Note that one should in fact expect strong dependencies between z and θ.

extended observations $\{\mathcal{D}, z\}$. Since this is not the maximum likelihood solution, θ^{ML}, for the original model with z integrated out, $p(\mathcal{D}|\theta)$, we will denote this as θ^*. We also know that asymptotically the posterior distribution over parameters (given z) converges to a Gaussian distribution with a covariance that scales as $\mathcal{O}(\frac{1}{N})$.

Expanding $\log p(\mathcal{D}, z|\theta)$ up to second order around its maximum at θ^*, we find,

$$\log p(\mathcal{D}, z|\theta) \approx \log p(\mathcal{D}, z|\theta^*) + \frac{N}{2}(\theta - \theta^*)^T I_F(\theta - \theta^*) \tag{10}$$

with I_F the Fisher information. Using that the Fisher information scales as $\mathcal{O}(\frac{1}{N})$ and only retaining terms that grow with N, we find for the first term of Eqn.9,

$$\log p(\mathcal{D}, z|\theta) \approx \log p(\mathcal{D}, z|\theta^*). \tag{11}$$

Using a similar approximation for the second term in Eqn.9 but observing that the Hessian for $\log p(\theta)$ does not grow with N we find that it can be ignored. The third term represents the entropy of $q(\theta)$ which asymptotically becomes

$$H(\theta) \approx -\frac{M}{2}\log N \tag{12}$$

where M is the number of parameters in the model. Putting everything together we find,

$$\mathcal{F}(z, K) \approx \log p(\mathcal{D}, z|\theta^*) - \frac{M}{2}\log N \tag{13}$$

for every value of z. In particular, we can choose the value of z that minimizes $\mathcal{F}(z, K)$ so we also have,

$$\mathcal{F}(K) \approx \log p(\mathcal{D}, z^*|\theta^*) - \frac{M}{2}\log N \tag{14}$$

This limit thus reveals a BIC/MDL penalty term in addition to a likelihood term where marginalization is replaced with minimization. We do *not* claim that asymptotically this ME-free energy converges to the true negative log-marginal likelihood since the

first (dominant) term will typically not converge to the true likelihood. The reason is that even for very large N the posterior distributions $p(z|\theta^{\mathrm{ML}}, \mathcal{D})$ for data-cases at the intersection of overlapping clusters will be "soft", and not deterministic as in the ME approximation. So $\mathcal{F}(K)$ should be interpreted as a computationally efficient proxy for the true negative log-marginal likelihood, even for very large N. Note that for small N the ME approximation is in fact different from the BIC/MDL penalty because it retains all orders in N.

5 The Hard Assignment Approximation for Bayesian Clustering

In this section we will consider a specific example of an ME algorithm in the context of clustering.

Let $\mathbf{x}_n$, $\{n = 1, ..., N\}$ be a vector of IID continuous-valued observations in D dimensions. For each data-case we define a cluster assignment variable, $z_n \in [1, ..., K]$. We assume that the data has been generated following a mixture of Gaussians and we have placed the standard conjugate priors on the parameters,

$$p(\mathbf{x}_n, z_n, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_n|z_n, \boldsymbol{\mu}, \Omega)\ \mathcal{M}(z_n|\boldsymbol{\alpha})\ \mathcal{D}(\boldsymbol{\alpha}|\phi_0)\ \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_0, \xi_0\Omega)\ \mathcal{W}(\Omega|\eta_0, B_0). \quad (15)$$

where $\boldsymbol{\mu}$ are the cluster means, Ω the inverse cluster covariances and $\boldsymbol{\alpha}$ the mixing coefficients. $\mathcal{N}(\cdot)$ represents a normal distribution, $\mathcal{M}(\cdot)$ a multinomial (or discrete) distribution, $\mathcal{D}(\cdot)$ a Dirichlet distribution and $\mathcal{W}(\cdot)$ a Wishart distribution. The priors depend on some (non-random) hyper-parameters, $\mathbf{m}_0, \phi_0, \xi_0, \eta_0, B_0$. Uninformative priors are covered in appendix A.

For clustering, we are interested in the posterior distribution $p(z|\mathcal{D})$, where $\mathcal{D}$ denotes the data-set and z are binary assignment variables for each data-case. In the approximation we propose we replace the full posterior with the following factorized form,

$$p(z, \boldsymbol{\theta}|\mathcal{D}) \approx q(\boldsymbol{\theta})\ \delta(z, z^*) \tag{16}$$

that is, we maintain a distribution over parameters, but settle for a point-estimate of the assignment variables. Minimizing the KL-divergence between this approximate posterior and the exact posterior, $KL[q(\boldsymbol{\theta})\delta(z, z^*)||p(z, \boldsymbol{\theta}|\mathcal{D})]$, over $q(\boldsymbol{\theta})$ we find,

$$q(\boldsymbol{\mu}, \Omega, \boldsymbol{\alpha}) = \left[\prod_{c=1}^{K} \mathcal{N}(\boldsymbol{\mu}_c|\mathbf{m}_c, \xi_c\Omega_c)\ \mathcal{W}(\Omega_c|\eta_c, B_c)\right] \mathcal{D}(\alpha|\{\phi_c\}) \tag{17}$$

where

$$\begin{aligned}
&\xi_c = \xi_0 + N_c \qquad\qquad \mathbf{m}_c = \frac{N_c\bar{\mathbf{x}}_c + \xi_0\mathbf{m}_0}{\xi_c} \qquad\qquad \eta_c = \eta_0 + N_c \\
&B_c = B_0 + N_cS_c + \frac{N_c\xi_0}{\xi_c}(\bar{\mathbf{x}}_c - \mathbf{m}_0)(\bar{\mathbf{x}}_c - \mathbf{m}_0)^T \\
&\phi = \{\phi_1, ..., \phi_K\} \qquad\qquad \phi_c = \phi_0 + N_c
\end{aligned} \tag{18}$$

and where N_c denotes the number of data-cases in cluster c, $\bar{\mathbf{x}}_c$ is the sample mean of cluster c, and S_c is its sample covariance.

The main simplification that has resulted from the approximation in Eqn.16 is that this distribution now factorizes over the clusters. Using this and Eqn.7 we derive the following bound on the log evidence (or "free energy"),

$$\begin{aligned}
\mathcal{F}(z, K) = \sum_{c=1}^{K} \Bigg[&\frac{DN_c}{2}\log\pi + \frac{D}{2}\log\frac{\xi_c}{\xi_0} + \frac{\eta_c}{2}\log\det(B_c) - \frac{\eta_0}{2}\log\det(B_0) \\
&- \log\frac{\Gamma_D(\frac{\eta_c}{2})}{\Gamma_D(\frac{\eta_0}{2})} + \frac{1}{K}\log\frac{\Gamma(N + K\phi_0)}{\Gamma(K\phi_0)} - \log\frac{\Gamma(\phi_c)}{\Gamma(\phi_0)}\Bigg]
\end{aligned} \tag{19}$$

where $\Gamma_D(x) = \pi^{\frac{D(D-1)}{4}} \prod_{i=1}^{D} \Gamma(x + \frac{1-i}{2})$ and $\Gamma(\cdot)$ the Gamma function. We observe that this objective also nicely factorizes into a sum of terms, one for each cluster. Unlike the maximum likelihood criterion, this objective automatically penalizes overly complex models. Our task is to minimize Eqn.19 jointly over assignment variables $z_n,\ n = 1..N$ and K, the number of clusters.

6 The ME-Algorithm for Clustering: "Bayesian K-Means"

In this paper we study two approaches to minimizing the cost Eqn.19. The one which we discuss in this section is a Bayesian variant of k-means (BKM), which is an instance of the more general class of "ME-algorithms". In section 7 we will discuss an agglomerative clustering algorithm.

From Eqn.5 we see that the ME algorithm iteratively maximizes $\mathbb{E}[\log(p(\boldsymbol{\theta}, z, \mathcal{D})]_{q(\boldsymbol{\theta})}$, where $p(\boldsymbol{\theta}, z, \mathcal{D})$ can be derived from Eqn.15 and $q(\boldsymbol{\theta})$ is given by Eqn.17. This is alternated with resetting $q(\boldsymbol{\theta})$ to the distribution $p(\theta|z^*, \mathcal{D})$ (Eqn.17) with z^* the new values for z computed in the previous phase. This leads (after some algebra) to the following iterative "labeling cost",

$$\mathcal{C}_{\text{BKM}} = \sum_n d_{z_n}(\mathbf{x}_n) \qquad \text{with} \tag{20}$$

$$\begin{aligned} d_{z_n}(\mathbf{x}_n) =& \frac{\eta_{z_n}}{2}(\mathbf{x}_n - \mathbf{m}_{z_n})^T B_{z_n}^{-1}(\mathbf{x}_n - \mathbf{m}_{z_n}) \\ &+ \frac{1}{2}\log\det(B_{z_n}) + \frac{D}{2\xi_{z_n}} - \frac{1}{2}\sum_{d=1}^{D}\Psi\left(\frac{\eta_{z_n}+1-d}{2}\right) - \Psi(\phi_{z_n}). \end{aligned}$$

where $\Psi(x) = \frac{\partial \log\Gamma(x)}{\partial x}$ is the "digamma" function. We recognize $d_{z_n}(\mathbf{x}_n)$ as the Mahalanobis distance plus some constant correction term. Due to this extra constant $d_{z_n}(\mathbf{x}_n)$ can not be interpreted as a proper distance.

The ME algorithm thus alternates updating assignment variables z_n to minimize $\mathcal{C}_{\text{BKM}}$ and recalculating the quantities $\{B_c, \xi_c, \eta_c, \phi_c, \mathbf{m}_c\}$ which are in turn simple functions of the sufficient statistics $\{N_c, \bar{\mathbf{x}}_c, S_c\}$ for each cluster, see Eqn.18. This algorithm is very similar to the classical k-means algorithm. The change in the labeling cost however has an essential function in terms of model selection, because unlike the classical k-means algorithm, it penalizes overly complex models. To search over different numbers of clusters we need to introduce cluster split and merge operations. We have followed

the approach taken in the SMEM algorithm [Ueda et al., 2000], but note that in our case we don't have to balance the number of splits and merges to retain the same number of clusters. When we split, we initialize the new clusters with the heuristic used by Hamerly and Elkan [2003].

The performance of the algorithm depends on the settings of the hyper-parameters $\{B_0, \xi_0, \eta_0, \phi_0, \mathbf{m}_0\}$ which express our prior expectations. We have used the following data-informed settings for the hyper-parameters throughout all our experiments involving BKM,

$$\begin{aligned} \xi_0 &= 0.1 \qquad & \mathbf{m}_0 &= \bar{\mathbf{x}} \qquad & \eta_0 &= D \\ \phi_0 &= 2 \qquad & B_0 &= d_{\text{small}}^2 \; D \; S/\text{trace}(S) \end{aligned} \tag{21}$$

where $\bar{\mathbf{x}}$ is the sample mean of the data, D the dimensionality of the problem, S the sample covariance of the data and d_{small} is computed by determining the closest neighbor for 10% of the data-cases and averaging over the 3 closest pairs.

7 Agglomerative Bayesian Clustering

If one is interested in the hierarchical structure of the data, bottom-up agglomerative approaches to clustering become interesting. Since we have an explicit objective function in the form of the free energy in Eqn.19, implementing a naive bottom-up clustering algorithm is trivial. One starts with assigning a separate cluster to every data-case. At each iteration we search for the best pair of clusters to merge in the sense that they maximally decrease the objective. At some point the objective may start to increase at which point we may continue and choose merges that minimally increase the objective. The point where the objective is minimal has a special status in the sense that it indicates optimal model complexity.

Unfortunately, the settings of the hyper-parameters given in Eqn. 21 did not perform satisfactorily in obtaining good solutions and good model complexity. The reason is

Top-Down Bayesian K-Means

1 *Initialize:*

1i Set hyper-parameters $\{B_0, \xi_0, \eta_0, \phi_0, \mathbf{m}_0\}$ using Eqn.21.

1ii Assign all data-cases to a single cluster $\{z_n\} = 1$.

2 *Repeat split operations until no more improvement is possible:*

2i Use heuristic Eqn.47 to rank clusters for splitting.

2ii Split highest ranked cluster using procedure in appendix D.3.

2iii *Run Bayesian k-means updates until convergence:*

2iiia Update quantities $\{B_c, \xi_c, \eta_c, \phi_c, \mathbf{m}_c\}$ using Eqn.18.

2iiib Update assignment variables using labeling cost Eqn.20.

2iv *Accept or Reject Split*

2iva If free energy in Eqn.19 has decreased: accept split and goto 2i.

2ivb If free energy has increased: reject split, remove candidate from list and goto 2ii.

3 *Merge two clusters into one cluster:*

3i Use heuristic Eqn.50 to rank pairs of clusters for merging.

3ii Merge highest ranked cluster.

3iii Run Bayesian k-means updates until convergence (see 2iiia & 2iiib above)

3iv *Accept or Reject Merge*

3iva If free energy in Eqn.19 has decreased: accept merge and goto 2.

3ivb If free energy has increased: reject merge, remove candidate from list and goto 3ii.

presumably that the agglomerative algorithm is too greedy in trying to optimize the free energy. In particular, in the initial stages of optimization when very few data-cases are in each cluster, the priors are overpowering the likelihood term resulting in suboptimal clusterings which can not be "undone" in later stages. To diminish the influence of the prior we have used the following hyper-parameter settings in all our agglomerative clustering experiments,

$$\begin{aligned} &\xi_0 = 0.01 \qquad &&\mathbf{m}_0 = \bar{\mathbf{x}} \qquad &&\eta_0 = D \\ &\phi_0 = 2 \qquad &&B_0 = 0.01 \; d^2_{\text{small}} \; I_D \end{aligned} \tag{22}$$

where I_D is a D by D identity matrix.

We emphasize that compensating an over-greedy algorithm with a change in the prior in order to guide the bottom-up optimization to better optima is somewhat of an unprincipled "hack". It would for instance be preferable to use look-ahead schemes to obtain better solutions, but this was beyond the scope of what we intended to study for this paper. As a result, we cannot claim this approach to be a principled method to estimate model complexity. It is still useful because unlike BKM it returns a full clustering hierarchy instead of a single optimal solution.

Agglomerative Bayesian Clustering

1 *Initialize:*

1i Set hyper-parameters $\{B_0, \xi_0, \eta_0, \phi_0, \mathbf{m}_0\}$ using Eqn.22.

1ii Assign all data-cases to a separate cluster $\{z_n = n\}$.

2 *Repeat merge operations until no more improvement is possible:*

2i Update quantities $\{B_c, \xi_c, \eta_c, \phi_c, \mathbf{m}_c\}$ using Eqn.18.

2ii Find the pair of clusters that generates the largest decrease in the free energy in Eqn.19.

2iii Merge closest pair by assigning their data-cases to a single cluster and goto 2i.

8 Clustering for Infinite Mixtures

Dirichlet process mixture models have gained considerable attention in recent years. In this section we briefly comment on the infinite limit for our model.

In taking the infinite limit we need to switch from assignment vectors to partitions (see e.g. Griffiths and Ghahramani [2006]). For example, assignments $(z_1, z_2, z_3) = (1, 1, 2)$ mean $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are assigned to cluster 1, 1 and 2 respectively. If 1 and 2 are just labels, $(1, 1, 2)$ would be equal to $(2, 2, 1)$. A partition denotes a set of equivalent assignments. In this case, partition $[z]$ is equal to the set $\{(1, 1, 2), (2, 2, 1)\}$, i.e. it collapses the space to the much smaller space of "assignments" equivalent under relabeling the clusters.

Setting ϕ_0 to ϕ_0/K, adding the appropriate counting factors necessary in switching from assignments to partitions (see Griffiths and Ghahramani [2006]) and taking the limit $K \to \infty$ we obtain the following free energy,

$$\begin{aligned}\mathcal{F}([z], K_+) = &\sum_{c=1}^{K_+} \left[\frac{DN_c}{2} \log \pi + \frac{D}{2} \log \frac{\xi_c}{\xi_0} + \frac{\eta_c}{2} \log \det(B_c) - \frac{\eta_0}{2} \log \det(B_0) \right. \\ &\left. - \log \frac{\Gamma_D(\frac{\eta_c}{2})}{\Gamma_D(\frac{\eta_0}{2})} + \frac{1}{K_+} \log \frac{\Gamma(N + \phi_0)}{\Gamma(\phi_0)} - \log \Gamma(N_c) - \log \phi_0 \right], \end{aligned} \tag{23}$$

where K_+ is the number of clusters which have at least one data-case.

Similarly, the labeling cost becomes

$$\mathcal{C}_{\text{BKM}} = \sum_n d_{z_n}(\mathbf{x}_n) \qquad \text{with} \tag{24}$$

$$\begin{aligned} d_{z_n}(\mathbf{x}_n) = &\frac{\eta_{z_n}}{2} (\mathbf{x}_n - \mathbf{m}_{z_n})^T B_{z_n}^{-1} (\mathbf{x}_n - \mathbf{m}_{z_n}) \\ &+ \frac{1}{2} \log \det(B_{z_n}) + \frac{D}{2\xi_{z_n}} - \frac{1}{2} \sum_{d=1}^{D} \Psi\left(\frac{\eta_{z_n} + 1 - d}{2} \right) - \Psi(N_c). \end{aligned}$$

Eqn 23 is in fact equivalent to $-\log P(\mathcal{D}, [z])$ for a DP mixture. Note however that instead of sampling $\{z_n\}$ sequentially we jointly maximize over $\{z_n\}$. However, compared to Eqn.19 not much has changed and we do not expect the infinite limit to behave radically different than the finite Bayesian K-means algorithm.

9 Efficient Implementations

One of the main motivations behind our hard assignment Bayesian clustering algorithms is to combine model selection with efficiency. The most impressive speedups for k-means (at least in low dimensions) are based on special data-structures such as kd-trees to avoid redundant distance calculations [Pelleg and Moore, 1999, Moore, 1998, Zhang et al., 1996, Verbeek et al., 2003]. Similarly, efficient data-structures such has the "Conga-Line" were proposed by Eppstein [1998] to reduce time complexity of agglomerative clustering methods. In the following we report on implementing and adapting these techniques to the algorithms under study.

9.1 KD-Trees

In our implementation we used kd-trees in a similar fashion as reported in Pelleg and Moore [1999], but the details are slightly different due to the fact that our "distance" function in Eqn.20 is different[2].

A kd-tree is a data-structure where every node is associated with a hyper-rectangle h containing a subset of data-cases as shown in Fig.2. Each node stores the sufficient statistics $N_h, \boldsymbol{t}_h, M_h$ of the data-cases in its corresponding hyper-rectangle, where

$$N_h = \#(\text{data-cases in } h), \qquad \boldsymbol{t}_h = \sum_{\mathbf{x}_n \in h} \mathbf{x}_n, \qquad M_h = \sum_{\mathbf{x}_n \in h} \mathbf{x}_n \mathbf{x}_n^T. \tag{25}$$

The tree is constructed recursively by dividing the hyper-rectangles in two smaller hyper-rectangles and associating them (and their sufficient statistics) with the child nodes of the kd-tree. The construction is done only once at the beginning of the Bayesian k-means algorithm.

To see how we can improve efficiency, we first note that the updates for the Bayesian k-means algorithm in Eqn.18 only depend on the sufficient statistics of the data, $\{N_c, \bar{\mathbf{x}}_c, S_c\}$ for each cluster c. Instead of recomputing these sufficient statistics after every update by

[2]Note that Eqn.20 does not in fact define a distance but that this fact is irrelevant for the workings of the algorithm.

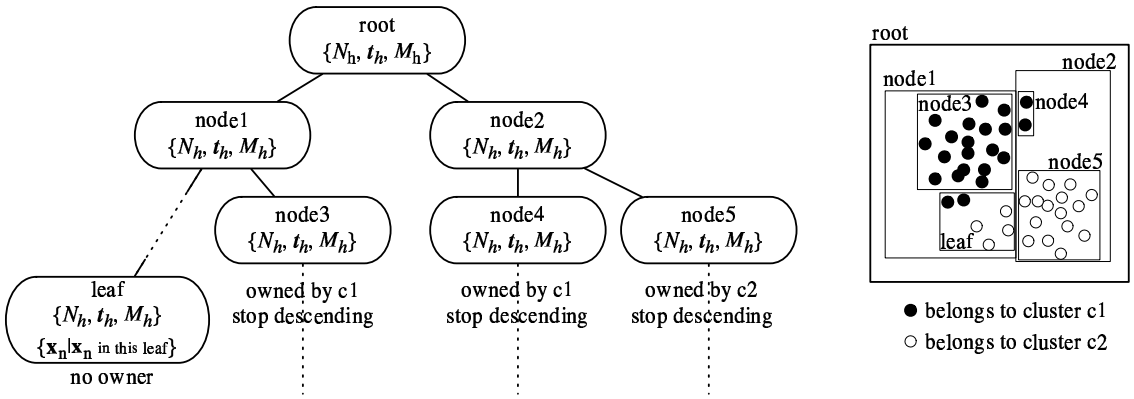

Figure 2: An example of a kd-tree and data

considering each data-case separately, we can obtain them using the sufficient statistics stored at the nodes of the kd-tree,

$$N_c = \sum_{h \in \mathcal{H}_c} N_h, \qquad \bar{\mathbf{x}}_c = \frac{1}{N_c} \sum_{h \in \mathcal{H}_c} \boldsymbol{t}_h, \qquad S_c = \frac{1}{N_c} \left(\sum_{h \in \mathcal{H}_c} M_h \right) - \bar{\mathbf{x}}_c \bar{\mathbf{x}}_c^T \tag{26}$$

where $\mathcal{H}_c$ is a set of hyper-rectangles whose data-cases are all assigned to cluster c. Eqn.26 provides an efficient computation of the sufficient statistics. For example, the computational complexity of $\bar{\mathbf{x}}_c$ in Eqn.26 is $O(|\mathcal{H}_c|)$ whereas without the kd-tree $\bar{\mathbf{x}}_c$ ($\equiv \sum_{n=1}^{N} \delta(z_n, c)\mathbf{x}_n$) requires $O(N_c)$.

To utilize Eqn.26, a hyper-rectangle needs to be associated with a single cluster, i.e. the cluster must "own" all the data-cases in that hyper-rectangle. To associate hyper-rectangles to clusters we start at the root and descend down the tree until we find an owner, see Fig.2. Clearly, the deeper we have to descend, the more inefficient the algorithm will be. The decision problem of "*cluster c is the owner of hyper-rectangle h*" can be defined as follows (where $d_c(x)$ is defined by the labeling cost in Eqn.20),

$$h \text{ is owned by } c \quad \stackrel{def}{\Leftrightarrow} \quad d_c(\mathbf{x}) < d_{c'}(\mathbf{x}) \quad \text{for all } \mathbf{x} \in h, c' \neq c \tag{27}$$

$$\Leftarrow \quad d_c^{\max}(h) < d_{c'}^{\min}(h) \quad \text{for all } c' \neq c \tag{28}$$

$$\Leftarrow \quad \tilde{d}_c^{\max}(h) < \tilde{d}_{c'}^{\min}(h) \quad \text{for all } c' \neq c. \tag{29}$$

and

$$d_c^{\max}(h) \equiv \max_{\mathbf{x}\in h} d_c(\mathbf{x}), \qquad d_c^{\min}(h) \equiv \min_{\mathbf{x}\in h} d_c(\mathbf{x})$$
$$\tilde{d}_c^{\max}(h) \equiv \max_{\mathbf{x}\in h} d_c^{+}(\mathbf{x}), \qquad \tilde{d}_c^{\min}(h) \equiv \min_{\mathbf{x}\in h} d_c^{-}(\mathbf{x}).$$

d_c^+ and d_c^- are the upper and lower bounds of d_c respectively. Note that the use of upper and lower bounds still leads to exact speed-ups. The only impact will be that we sometimes have to descend a bit deeper in the kd-tree to find nodes that are owned by clusters. This slight inefficiency is traded-off against the improved efficiency of maximizing and minimizing the bounds instead of computing the exact values for $d_c^{\max}$ and $d_c^{\min}$.

First, we compute $\tilde{d}_c^{\min}(h)$. d_c^- is derived from Eqn.20 by replacing $\frac{\eta_c}{2}B_c^{-1}$ with $\lambda_c^{\min} I$ where $\lambda_c^{\min}$ is the smallest eigenvalue of $\frac{\eta_c}{2}B_c^{-1}$,

$$d_c^{-}(\mathbf{x}) = \lambda_c^{\min}(\mathbf{x}-\mathbf{m}_c)^T(\mathbf{x}-\mathbf{m}_c) + a_c$$

where $a_c = \frac{1}{2}\log\det(B_{z_n}) + \frac{D}{2\xi_{z_n}} - \frac{1}{2}\sum_{d=1}^{D}\Psi\left(\frac{\eta_{z_n}+1-d}{2}\right) - \Psi(\phi_{z_n})$. Although $\min_{\mathbf{x}\in h} d_c(\mathbf{x})$ is originally a constrained convex quadratic program, it is easier to solve $\tilde{d}_c^{\min}(h) = \min_{\mathbf{x}\in h} d_c^-(\mathbf{x})$ instead. We can in fact find that $\tilde{d}_c^{\min}(h)$ has the following solutions,

$$\tilde{d}_c^{\min}(h) = \begin{cases} a_c & \text{if } \mathbf{m}_c \in h \\ \lambda_c^{\min}\sum_{d=1}^{D}\min\left((\mathbf{u}_h^{(d)} - \mathbf{m}_c^{(d)})^2, (\mathbf{l}_h^{(d)} - \mathbf{m}_c^{(d)})^2\right) + a_c & \text{otherwise} \end{cases}$$

where $\mathbf{u}_h$ and $\mathbf{l}_h$ are the upper and lower boundaries of the hyper-rectangle h respectively, and superscript (d) denotes the d-th dimension of a vector.

Next, we give a solution of $d_c^{\max}(h) = \max_{\mathbf{x}\in h} d_c(\mathbf{x})$. The solution to the maximization problem can be shown to lie on one of the corners of the hyper-rectangle. Checking all corners has complexity 2^D which is manageable until $D = 8$. After that we solve $\tilde{d}_c^{\max}(h) = \max_{\mathbf{x}\in h} d_c^+(\mathbf{x})$ instead. $d_c^+(\mathbf{x})$ is defined in a similar way to the minimization problem. We replace $\frac{\eta_c}{2}B_c^{-1}$ in Eqn.20 with $\lambda_c^{\max} I$ where $\lambda_c^{\max}$ is the largest eigenvalue of $\frac{\eta_c}{2}B_c^{-1}$. The solution can be found as,

$$\tilde{d}_c^{\max}(h) = \lambda_c^{\max}\sum_{d=1}^{D}\max\left((\mathbf{u}_h^{(d)} - \mathbf{m}_c^{(d)})^2, (\mathbf{l}_h^{(d)} - \mathbf{m}_c^{(d)})^2\right) + a_c.$$

Using these solutions of Eqn.29, the decision problem of finding cluster c that is the owner of hyper-rectangle h can deterministically be solved with computational complexity $O(DK)$ given hyper-rectangle h and cluster c.

To expedite the process of finding owners we implemented the "blacklisting algorithm" [Pelleg and Moore, 1999]. We maintain a list of clusters that could *not* own the data-cases in each hyper-rectangle while descending the kd-tree. As we descend deeper more and more clusters get "blacklisted". This is continued until we find a hyper-rectangle small enough so that all its data-cases are owned by a single cluster. Deeper hyper-rectangles are ignored at that point, see Fig.2. At hyper-rectangle h, we blacklist[3] any clusters c', s.t. $\tilde{d}_c^{\max}(h) < \tilde{d}_{c'}^{\min}(h)$ where $c = \arg\min_{c \notin blacklist} d_c^{\min}(h)$. These calculations are repeated (for non-blacklisted clusters) while descending down the tree until all hyper-rectangles are owned by a single cluster or we may stop at any pre-specified level (defined as a leaf) where we compute the sufficient statistics for every cluster c separately.

We have summarized the calculation of the sufficient statistics in pseudo-code above. $\boldsymbol{t}_c$ and M_c are equal to $\sum_{h \in \mathcal{H}_c} \boldsymbol{t}_h$ and $\sum_{h \in \mathcal{H}_c} M_h$ in Eqn.26, respectively. By calling "descend(*root-node*, empty-blacklist)", the sufficient statistics are thus collected. Note that "descend(*root-node*, empty-blacklist)" has to be repeated for every hyper-parameter update.

9.2 Conga Lines

Conga line data-structures were proposed by Eppstein [1998] to speed up agglomerative clustering algorithm from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2 \log(N))$. The basic idea is to maintain a changing partition of the data-set, where each partition is associated with a directed graph consisting of a collection of paths. There are $\mathcal{O}(N)$ edges and the closest pair will always be represented by some edge in this collection. The data-structure gets updated,

[3]We do not need to fix c as $c = \arg\min_{c \notin blacklist} d_c^{\min}(h)$. In other words, we can blacklist any c' s.t. $\tilde{d}_c^{\max}(h) < \tilde{d}_{c'}^{\min}(h)$ for "any" c. But, this requires $O(K^2)$ in the worst case. We just fix c for efficiency.

Calculation of Sufficient Statistics using a kd-tree

$\{N_c, \boldsymbol{t}_c, M_c\}_{c=1}^K$ = descend(*root-node*, empty-blacklist)

function $\{N_c^h, \boldsymbol{t}_c^h, M_c^h\}_{c=1}^K$ = descend(node h, black-list $list$)
  blacklist any cluster c' to $list$ s.t. $\tilde{d}_c^{\max}(h) < \tilde{d}_{c'}^{\min}(h)$
    where $c = \arg\min_{c \notin list} d_c^{\min}(h)$
  set $\{N_c^h, \boldsymbol{t}_c^h, M_c^h\}_{c=1}^K$ to zero.
  if $|\{c|c \notin list\}| = 1$ (i.e. h has an owner cluster c)
    $N_c^h = h.N_h, \quad \boldsymbol{t}_c^h = h.\boldsymbol{t}_h, \quad M_c^h = h.M_h.$
  else if *node* is a leaf
    calculate $N_c^h, \boldsymbol{t}_c^h, M_c^h$ for every c according to assignments given by Eqn.20.
  else
    $\{N_c^h, \boldsymbol{t}_c^h, M_c^h\}_{c=1}^K$ = descend($h.child_1$, $list$) + descend($h.child_2$, $list$).
  end if
end function

i.e. new partitions and graphs are formed if we remove two clusters and insert the newly merged cluster. It can be shown that these basic operations take $\mathcal{O}(N \log(N))$ time instead of the usual $\mathcal{O}(N^2)$ for naive implementations. Experimental results on speedup factors are reported in section 11.

10 Other Instances of the ME Algorithm

So far, we have described the Bayesian k-means algorithm and agglomerative Bayesian clustering. However, it is straightforward to extend the ideas to clustering with discrete attributes. In section 11 we report on some experiments with multinomial distributions and product of Bernoullis distributions in addition to Gaussian distributions.

The derivation of the relevant quantities are detailed in appendix B and C. The top-down and agglomerative algorithms of these models are the same as the algorithms described in section 6 and 7 with the free energy and the labelling costs replaced by

the ones for these models. Although the labelling costs, Eqn.40 and 46, are in fact different than Eqn.20, it is still possible to apply kd-tree data-structures to speed up the algorithm. Empirical speedup factors are shown in section 11.

11 Experimental Results

In this section, we will show experimental results on a number of synthetic and real datasets. We will also show the speedup between naive ME algorithms and ME algorithms using kd-trees and conga-lines.

11.1 Synthetic Data

In our first experiment we compared Bayesian k-means (BKM) with G-means (software obtained from Hamerly and Elkan [2003]), k-means+BIC [see also Pelleg and Moore, 2000], mixture of Gaussians (MoG) + BIC, variational Bayesian learning for mixtures of Gaussians (VBMG) [Attias, 2000, Ghahramani and Beal, 2000] and a collapsed Gibbs sampler for Dirichlet process Gaussian mixtures[MacEachern and Müller, 1998][4]. K-means+BIC, mixtures of Gaussians+BIC and VBMG use the same split and merge strategy as BKM, but different penalty terms to control the number of clusters. These algorithms were tested on a number of synthetic datasets that were generated similarly to Hamerly and Elkan [2003]. For each synthetic dataset, we sampled K centroids and stds. such that no two clusters are closer than $\tau \times$ (the sum of two stds.)/2. After we generated 10 datasets of size N, we multiplied the data within each cluster with a random matrix to give them full covariance structure. Figure 3 shows some typical results for BKM on three random datasets with $\tau = 0.1, 0.5$ and 2.

Table 1 shows the results of this experiment. We set K to 10. Therefore, the closer to 10 the better a result is[5]. It should come as no surprise that BKM, VBMG,

[4]The model of VBMG and the collapsed Gibbs sampler is exactly the same as BKM (see Eqn.15).

[5]Note that since the data was sampled from the (Gaussian mixture) model we should expect a consistent estimator for K to return $K = 10$ if N is sufficiently large.

MoG+BIC and the collapsed sampler outperform G-means and k-means+BIC since the latter assume isotropic covariance. Between BKM, VBMG and MoG+BIC it seems hard to discern significant difference in performance in these experiments. The collapsed Gibbs sampler sometimes performed worse than BKM, VBMG and MoG+BIC. This is because the collapsed Gibbs sampler does not utilize split and merge operations. Thus, it was often trapped at local optima.

We note however that the speedup for BKM is exact while speedup for MoG+BIC is approximate [Moore, 1998] and no speedup is currently available for VBMG. The performance of the algorithms VBMG and BKM also depend on the setting of the hyper-parameters. The freedom to choose the hyper-parameters is an additional source of flexibility. Certainly, when good priors can be specified this should be helpful. In the experiments reported here we did not make an attempt to improve results by specifying priors on a case by case basis, but instead used the empirical Bayesian "rule of thumb" (Eqns.21 and 22) for all experiments. Note that BIC can be derived as the "large-N" limit of both VBMG and BKM (see section 4).

11.2 Real Data

Next, we evaluated agglomerative Bayesian clustering (ABC) against the following traditional agglomerative clustering methods: single, complete and average linkages. We ran these algorithms on real datasets: Pendigits, CEDAR dataset, MNIST dataset, UCI spambase dataset and 20 newsgroups dataset. Pendigits, CEDAR and MNIST datasets are handwritten digits (0-9) data, and they have 16, 64 and 784 dimensions respectively. Following Hamerly and Elkan [2003], we applied a random projection to the MNIST dataset to reduce the dimension to 50. From the 20 newsgroup dataset, we used only rec.autos, rec.sport.baseball, rec.sport.hockey and sci.space categories and chose 50 features using mutual information. The spambase and the 20 newsgroups datasets were binarized. ABC used a MoG on the handwritten digits and a product of Bernoullis on the spambase and 20 newsgroup datasets.

We evaluate results with the *dendrogram purity*[6]. Table 2 shows the results. For each dataset, we created 10 random subsets. Each of the handwritten digits datasets contains 100 data-cases. Spambase and 20 newsgroup contain 200 and 240 data-cases respectively. On each dataset ABC built dendrograms which have the highest *dendrogram purity*.

Figure 4 is an illustrative example of a dendrogram for a small subset of spambase. We used 50 data-cases only (25 spam cases, "N", and 25 non-spam cases, "S") so that every label can be plotted on the x-axis. The purity was 0.816 for this example. The height of each linkage represents the negative free energy. Since a valid dendrogram accepts positive heights, we added 11,000 to the negative free energy.

11.3 Efficient implementations

Finally, we compared naive ME algorithms with efficient implementations using kd-trees and conga-lines. For the BKM + kd-trees experiment we varied one parameter at a time and used default values of $N = 20000$, $D = 2$, $\tau = 3$ and $K = 5$ to sample data. A node in a kd-tree is declared a leaf if the number of data-cases in the node is less than 1000 (the remaining points are treated individually). In the experiments with ABC + conga-lines, we vary N and use $D = 2$, $\tau = 3$ and $K = 10$. For the multinomial ME (MME) + kd-trees, we set default values, $N = 10000$ and $D = 8$, and varied N and D.

Figure 5 shows the speedup factors. For BKM, speedup increases very fast with N (factor of 67 at $N = 80000$), moderately fast with τ (less overlap helps to blacklist competing clusters) and decreases surprisingly slow with dimension (factor of 3.6 in 256 dimensions at $N = 20000$). ABC using the conga-lines is only slightly faster than the naive implementation due to large overhead but this clearly becomes more significant with growing N. MME also achieved speedup using kd-trees, but not as significant as

[6]Dendrogram purity is defined as follows: pick a leaf node i and another leaf node j assigned to the same class randomly. Measure the fraction of leaf nodes in the smallest subtree containing both i and j that are assigned to the same class as i and j. The expected value of this fraction is the dendrogram purity. The overall tree purity is 1 if and only if all leaves in each class are contained within some pure subtree. This quantity can be efficiently computed in a single bottom-up pass through the tree.

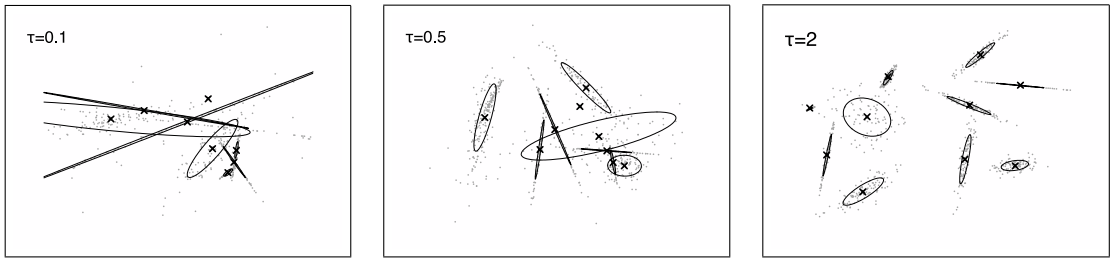

Figure 3: Typical results for BKM for various values of $\tau = 0.1, 0.5, 2$. The true K is equal to ten. In these plots, BKM found eight, nine and ten clusters respectively. Some ellipses are too small or too large to be visible.

BKM.

12 Conclusion

The contributions of this paper are twofold. Firstly a new class of algorithms is introduced that reverses the roles of "expectation" and "maximization" in the traditional EM algorithm. This combines the potential for model selection (because we integrate over the parameters) with fast implementations (because hard assignments are well suited for efficient data-structures such as kd-trees). Secondly, we have implemented and studied one possible application of this idea in the context of clustering. Explicit algorithms were derived and experimentally tested for bottom-up Bayesian agglomerative clustering and top-down Bayesian k-means clustering.

The experimental results have verified that for data sampled from a Gaussian mixture the BKM algorithm is competitive with alternative approaches in finding the correct number of clusters. We have also verified that the agglomerative Bayesian clustering algorithm scored high on dendrogram purity relative to some standard competitors in the literature. Finally, and perhaps most significantly, we found strong speed-up results for the Gaussian model as we increase the number of data-cases. We also observed a surprisingly graceful decrease of speed-up factors as we increased the dimensionality of the data.

τ	N	D	G-means	k-means+BIC	MoG+BIC	BKM	VBMG	CGibbs
0.1	100	2	4.50±2.42	2.00±1.41	3.80±2.04	2.00±0.82	4.30±2.16	6.32±1.39
		32	7.40±3.86	1.00±0.00	1.70±0.48	2.50±4.74	1.00±0.00	2.60±0.84
		64	13.10±5.47	1.00±0.00	1.00±0.00	13.30±1.06	1.30±0.48	1.70±0.48
	1000	2	32.10±12.33	3.50±1.90	7.80±1.32	7.40±1.96	8.80±0.79	15.11±1.84
		32	39.10±10.07	1.10±0.32	5.60±1.65	7.70±1.16	9.90±0.32	7.20±1.14
		64	49.40±22.51	1.00±0.00	3.60±1.43	5.70±4.79	2.90±0.74	6.50±0.97
	5000	2	108.90±15.07	3.80±1.62	9.20±2.70	8.10±4.72	9.90±0.32	15.23±2.11
		32	130.90±46.17	2.70±1.49	3.90±2.64	10.00±0.00	9.80±0.42	7.90±0.57
		64	104.30±51.93	1.30±0.67	2.10±0.32	10.00±0.00	10.00±0.00	7.20±1.55
0.5	100	2	4.80±2.49	2.50±1.72	5.20±2.35	4.10±3.60	4.40±2.12	6.73±1.48
		32	7.40±4.06	1.00±0.00	2.00±0.00	14.80±5.35	1.00±0.00	3.07±0.92
		64	10.60±6.55	1.00±0.00	1.00±0.00	13.80±0.92	1.40±0.52	1.70±0.48
	1000	2	21.30±4.55	4.20±1.87	9.20±1.32	4.30±3.65	9.40±0.84	14.52±2.45
		32	25.60±10.31	1.20±0.42	9.20±0.63	9.20±2.10	9.90±0.31	7.20±0.79
		64	37.90±9.72	1.60±1.08	3.80±0.84	11.40±6.15	3.80±1.03	6.10±1.37
	5000	2	82.00±14.91	4.20±1.32	9.10±2.76	8.10±3.57	9.80±0.42	12.00±1.76
		32	74.00±32.40	2.60±0.97	8.90±1.45	9.90±0.32	10.00±0.00	7.40±1.26
		64	74.30±31.72	2.50±1.72	9.60±0.52	9.70±0.48	9.80±0.42	7.20±1.69
2	100	2	6.80±3.26	2.10±1.45	6.50±1.96	8.30±1.89	7.20±2.15	5.96±0.61
		32	11.90±1.20	3.30±2.21	2.40±0.00	10.70±5.17	1.00±0.00	4.75±0.97
		64	9.90±3.87	2.20±2.20	1.00±0.00	13.80±0.63	1.00±0.00	2.96±0.68
	1000	2	18.30±5.29	4.30±1.57	9.50±1.65	9.60±1.26	9.80±0.42	8.26±1.63
		32	13.90±2.08	6.80±2.94	7.40±1.90	9.40±1.26	9.00±0.94	7.50±0.53
		64	13.50±3.66	8.60±2.41	3.00±0.67	12.60±3.31	4.00±1.03	6.40±0.97
	5000	2	48.70±14.22	5.30±2.50	9.60±1.58	10.00±0.00	9.40±1.43	9.33±1.12
		32	15.00±5.79	9.50±1.84	7.90±1.10	10.00±0.00	10.00±0.00	7.50±0.85
		64	11.60±2.07	9.90±2.12	8.40±0.97	10.00±0.00	10.00±0.00	6.60±1.58

Table 1: The number of clusters estimated on synthetic data by G-means, K-means+BIC, the mixture of Gaussians+BIC, Bayesian k-means, variational Bayesian learning for the mixture of Gaussians and a collapsed Gibbs sampler for Dirichlet process Gaussian mixtures. The true number of clusters is ten for each dataset. Results are averaged over ten runs of the algorithm.

Data Set	ABC	Single Linkage	Complete Linkage	Average Linkage
Pendigits	0.730±0.059	0.691±0.055	0.653±0.055	0.707±0.054
CEDAR	0.465±0.077	0.297±0.051	0.402±0.055	0.403±0.039
MNIST	0.410±0.078	0.316±0.056	0.358±0.040	0.389±0.048
spambase	0.742±0.057	0.632±0.026	0.703±0.036	0.686±0.028
20 newsgroup	0.448±0.033	0.300±0.008	0.433±0.035	0.356±0.011

Table 2: Dendrogram purities on various real datasets for the following agglomerative clustering algorithms: agglomerative Bayesian clustering (ABC), single linkage, complete linkage and average linkage. ABC was modeled with a mixture of Gaussians on handwritten digits, Pendigits, CEDAR and MNIST, and with a product of Bernoullis on spambase and 20 newsgroup

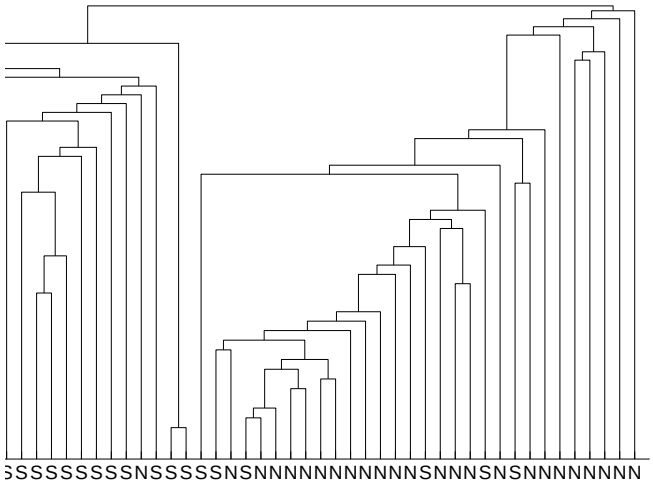

Figure 4: An illustrative example of a dendrogram built by ABC on the spambase dataset using a product of Bernoullis distribution. The purity was 0.816. Labels "N" and "S" denote non-spam and spam. The number of data-cases is 50 where 25 data-cases are spam and 25 data-cases are non-spam. Heights represents the negative free energy. Since a valid dendrogram accepts positive heights, we added 11,000 to the negative free energy.

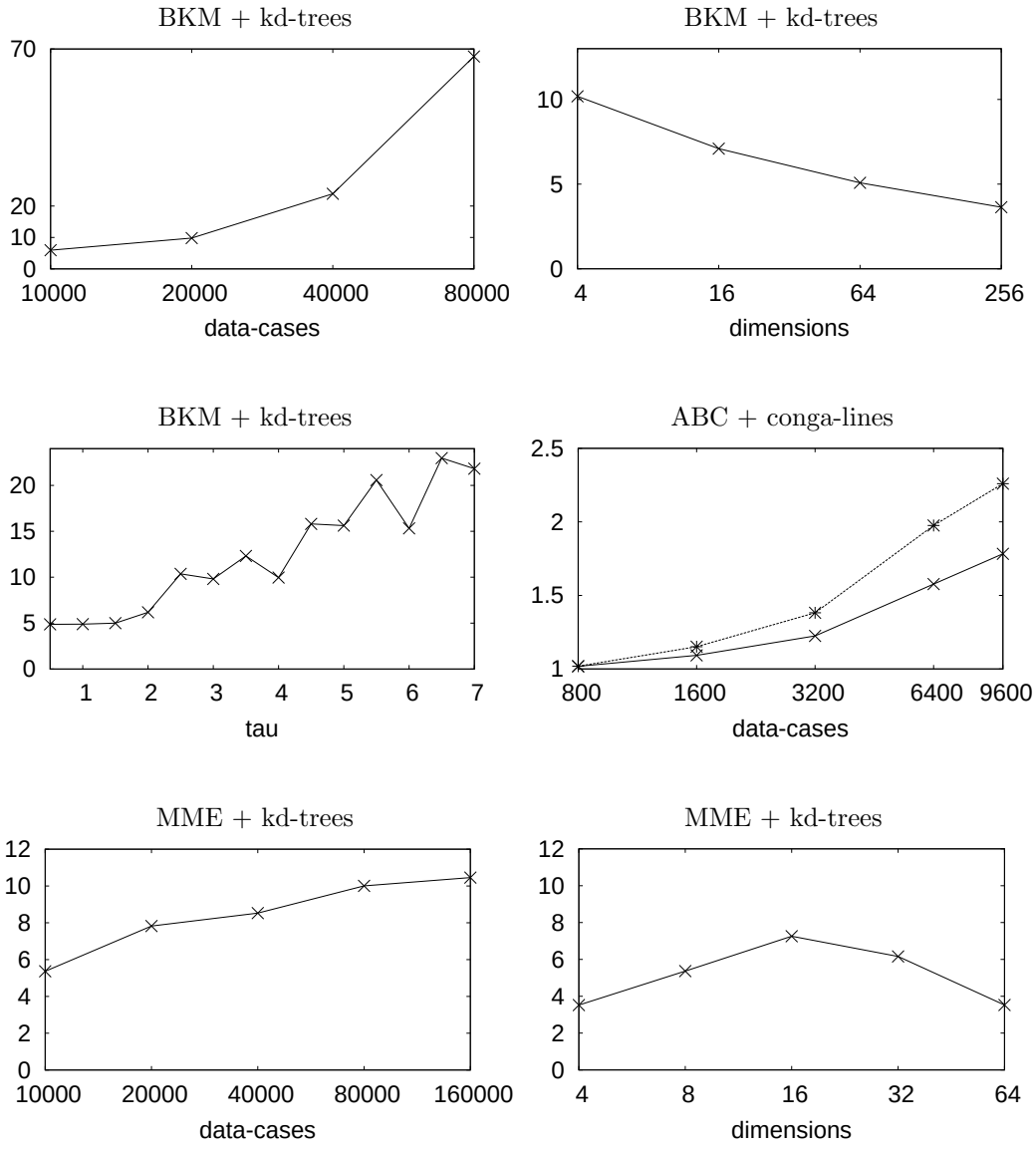

Figure 5: Speedup factors using fast data-structures, kd-trees and conga-lines. For Bayesian k-means (BKM) + kd-trees, the number of data-cases, dimensions and τ were varied. The result of agglomerative Bayesian clustering (ABC) + conga-lines shows two plots; $\times$ and $*$ denote overall speedup factor and speedup factor per iteration respectively. For the multinomial ME (MME) + kd-trees, the number of data-cases and dimensions were varied.

It is well known that kd-trees break down in high dimensions. Therefore, for high dimensional data we recommend running the proposed algorithms on random (or informed) projections of the data. Strong theoretical and empirical results exist that show that surprisingly little information about the cluster structure is lost in these projections Dasgupta [2000].

Although we have explored these ideas in the simplest possible context, namely clustering, the proposed techniques seem to readily generalize to more complex models such as HMMs. Whether the efficiency gains can also be achieved in this setting remains to be investigated.

Acknowledgment

We are grateful to Moore [1998] and Hamerly and Elkan [2003] for sharing their code and to Heller for datasets. Also, special thanks to Y.W. Teh for suggesting infinite mixtures. Finally, we acknowledge Andrew Moore for answering questions about data-structures for fast implementations.

A Bayesian K-Means with Uninformative Priors

In this section, we derive the free energy of Bayesian k-means with uninformative priors. First of all, we set uninformative priors on the parameters,

$$p(\boldsymbol{\mu}_c) = \prod_{d=1}^{D} \frac{1}{(\mathbf{x}_{\max}^{(d)} - \mathbf{x}_{\min}^{(d)})}, \qquad p(\Omega_c) = \det(\Omega_c)^{-(D+1)/2}, \qquad p(\alpha) = \Gamma(K), \tag{30}$$

where $\mathbf{x}_{\max}^{(d)} = \max_n \mathbf{x}_n^{(d)}$, $\mathbf{x}_{\min}^{(d)} = \min_n \mathbf{x}_n^{(d)}$ and $\mathbf{x}_n^{(d)}$ denotes the d-th dimension value of $\mathbf{x}_n$.

As we see in section 5, we find an approximate posterior,

$$q(\boldsymbol{\mu}, \Omega, \alpha) = \left[\prod_{c=1}^{K} \mathcal{N}(\boldsymbol{\mu}_c | \bar{\mathbf{x}}_c, N_c \Omega_c) \mathcal{W}(\Omega_c | N_c - 1, N_c S_c) \right] \mathcal{D}(\alpha | N_c + 1). \tag{31}$$

From this posterior and Eqn.7, we derive the free energy,

$$\begin{aligned} \mathcal{F}(z, K) = & \sum_{c=1}^{K} \left[\frac{D(N_c - 1)}{2} \log \pi + \frac{1}{2} \log N_c + \frac{N_c - 1}{2} \log \det(N_c S_c) - \log \Gamma_D \left(\frac{N_c - 1}{2} \right) \right. \\ & \left. + \frac{1}{K} \log \frac{\Gamma(N + K)}{\Gamma(K)} - \log \Gamma(N_c + 1) + \sum_{d=1}^{D} \log(\mathbf{x}_{\max}^{(d)} - \mathbf{x}_{\min}^{(d)}) \right]. \end{aligned} \tag{32}$$

We notice that this free energy is singular if there exists a cluster c such that $Nc \leq (D-1)/2$. This singularity leads to problems in agglomerative clustering because Nc is equal to 1 for each class c in the first step of agglomerative clustering.

The "labeling cost" is given by,

$$\mathcal{C}_{\mathrm{BKM}} = \sum_{n} d_{z_n}(\mathbf{x}_n) \qquad \text{with} \tag{33}$$

$$\begin{aligned} d_{z_n}(\mathbf{x}_n) = & \frac{N_{z_n}(N_{z_n} - 1)}{2} (\mathbf{x}_n - \bar{\mathbf{x}}_{z_n})^T S_{z_n}^{-1} (\mathbf{x}_n - \bar{\mathbf{x}}_{z_n}) \\ & + \frac{1}{2} \log \det(N_{z_n} S_{z_n}) + \frac{D}{2N_{z_n}} - \frac{1}{2} \sum_{d=1}^{D} \Psi \left(\frac{N_{z_n} - d}{2} \right) - \Psi(N_{z_n} - 1). \end{aligned} \tag{34}$$

This labelling cost shows that we can apply kd-trees in the same way as section 9.1.

B Multinomial ME Algorithm

In this section, we derive a ME algorithm modeled with multinomial. Let $\mathbf{x}_n$ be a vector, $(\mathbf{x}_{n1}, ..., \mathbf{x}_{nD})$, where $\mathbf{x}_{nd}$ denotes the number of occurrences of event d and $\sum_{d=1}^{D} \mathbf{x}_{nd} = W$.

The joint probability of $\mathbf{x}_n$ and z_n is described as,

$$p(\mathbf{x}_n, z_n | \boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{M}(z_n | \boldsymbol{\alpha}) \; \mathcal{M}(\mathbf{x}_n | \boldsymbol{\beta}_{z_n}) \tag{35}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the parameters of multinomials. We place conjugate priors on the parameters, then the marginal probability is given as,

$$p(\mathbf{x}_n, z_n, \boldsymbol{\theta}) = \mathcal{M}(z_n|\boldsymbol{\alpha})\ \mathcal{M}(\mathbf{x}_n|\boldsymbol{\beta}_{z_n})\ \mathcal{D}(\boldsymbol{\alpha}|\phi_0)\ \mathcal{D}(\boldsymbol{\beta}_{z_n}|\psi_0) \tag{36}$$

where ϕ_0 and ψ_0 are hyper-parameters. In our experiments, we set ϕ_0 and ψ_0 to 1 and 0.1 respectively.

Similarly to Eqn.17, we derive an approximate posterior,

$$q(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{D}(\boldsymbol{\alpha}|\phi)\ \left[\prod_{c=1}^{K} \mathcal{D}(\boldsymbol{\beta}_c|\psi_c)\right] \tag{37}$$

where

$$\begin{aligned} \phi &= \{\phi_1, ..., \phi_K\} & \phi_c &= \phi_0 + N_c \\ \psi_c &= \{\psi_{c1}, ..., \psi_{cD}\} & \psi_{cd} &= \psi_0 + \sum_{n_c=1}^{Nc} x_{n_c d}. \end{aligned} \tag{38}$$

Using Eqn.7 we derive the free energy,

$$\begin{aligned} \mathcal{F}(z, K) = &\sum_{c=1}^{K} \left[\log \frac{\Gamma(\phi_0)}{\Gamma(\phi_c)} + \log \frac{\Gamma\left(\sum_{d=1}^{D} \psi_{cd}\right)}{\Gamma(D\psi_0)} + \sum_{d=1}^{D} \log \frac{\Gamma(\psi_0)}{\Gamma(\psi_{cd})} + \frac{1}{K} \log \frac{\Gamma(K\phi_0 + N)}{\Gamma(K\phi_0)}\right] \\ &+ \sum_{n=1}^{N} \log \frac{\prod_{d=1}^{D} \Gamma(x_{nd} + 1)}{\Gamma(W + 1)}. \end{aligned} \tag{39}$$

The "labeling cost" takes the same general form as in section 6,

$$\mathcal{C}_{\text{MME}} = \sum_{n} d_{z_n}(\mathbf{x}_n) \tag{40}$$

where

$$d_{z_n}(\mathbf{x}_n) = \mathbf{a}_{z_n}^T \mathbf{x}_n + b_{z_n}$$

$$\mathbf{a}_{z_n} = \{a_{z_n 1}, ..., a_{z_n D}\}^T \qquad a_{z_n d} = -\Psi(\psi_{z_n d}) + \Psi\left(\sum_{d=1}^{D} \psi_{z_n d}\right)$$

$$b_{z_n} = -\Psi(\phi_{z_n}).$$

Note that this labeling cost is linear in $\mathbf{x}_n$ whereas BKM's labeling cost is quadratic. Therefore, it is in fact easier to apply kd-trees to BKM with multinomial distributions than with Gaussian distributions.

C Product of Bernoullis ME Algorithm

In this section, we apply the ME algorithm to a product of Bernoullis model. Let $\mathbf{x}_n$ be a vector, $(\mathbf{x}_{n1}, ..., \mathbf{x}_{nD})$, where $\mathbf{x}_{nd} \in 0, 1$.

The joint probability of $\mathbf{x}_n$ and z_n is described as

$$p(\mathbf{x}_n, z_n|\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \mathcal{M}(z_n|\boldsymbol{\alpha}) \prod_{d=1}^{D} \mathcal{B}(\mathbf{x}_{nd}|\boldsymbol{\lambda}_{z_n d}) \tag{41}$$

where $\mathcal{B}(\cdot)$ denotes a Bernoulli distribution, $\mathcal{B}(\mathbf{x}_{nd} = 1|\boldsymbol{\lambda}_{z_n d}) = \boldsymbol{\lambda}_{z_n d}$, $\mathcal{B}(\mathbf{x}_{nd} = 0|\boldsymbol{\lambda}_{z_n d}) = 1 - \boldsymbol{\lambda}_{z_n d}$. Using conjugate priors, the marginal probability is given as,

$$p(\mathbf{x}_n, z_n, \boldsymbol{\theta}) = \mathcal{M}(z_n|\boldsymbol{\alpha})\ \mathcal{D}(\boldsymbol{\alpha}|\phi_0)\ \left[\prod_{d=1}^{D} \mathcal{B}(\mathbf{x}_{nd}|\boldsymbol{\lambda}_{z_n d}) \mathcal{D}(\boldsymbol{\lambda}_{z_n d}|\omega_0)\right] \tag{42}$$

where ϕ_0 and ω_0 are hyper-parameters. In our experiments, we set them to 1 and 0.1 respectively.

Similarly to the preceding section, we derive the approximate posterior,

$$q(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \mathcal{D}(\boldsymbol{\alpha}|\phi)\ \left[\prod_{c=1}^{K} \prod_{d=1}^{D} \mathcal{D}(\boldsymbol{\lambda}_{cd}|\omega_{cd})\right] \tag{43}$$

where

$$\phi = \{\phi_1, ..., \phi_K\} \qquad \phi_c = \phi_0 + N_c$$
$$\omega_{cd} = \{\omega_{cd0}, \omega_{cd1}\} \qquad \omega_{cdb} = \omega_0 + \sum_{n_c=1}^{Nc} \delta(\mathbf{x}_{n_c d}, b) \quad b \in \{0, 1\}. \tag{44}$$

By substituting this approximate posterior into Eqn.7, we find the free energy,

$$\mathcal{F}(z, K) = \sum_{c=1}^{K} \left[\sum_{d=1}^{D} \left\{ \log \frac{\Gamma(\omega_{cd0} + \omega_{cd1})}{\Gamma(2\omega_0)} - \log \frac{\Gamma(\omega_{cd0})}{\Gamma(\omega_0)} - \log \frac{\Gamma(\omega_{cd1})}{\Gamma(\omega_0)} \right\} + \frac{1}{K} \log \frac{\Gamma(K\phi_0 + N)}{\Gamma(K\phi_0)} - \log \frac{\Gamma(\phi_c)}{\Gamma(\phi_0)} \right]. \tag{45}$$

The "labeling cost" is given by,

$$\mathcal{C}_{\mathrm{BME}} = \sum_n d_{z_n}(\mathbf{x}_n) \tag{46}$$

where

$$d_{z_n}(\mathbf{x}_n) = \mathbf{a}_{z_n}^T \mathbf{x}_n + b_{z_n}$$
$$\mathbf{a}_{z_n} = \{a_{z_n 1}, ..., a_{z_n D}\}^T \qquad a_{z_n d} = -\Psi(\omega_{z_n d1}) + \Psi(\omega_{z_n d0})$$
$$b_{z_n} = \sum_{d=1}^{D} \{-\Psi(\omega_{z_n d0}) + \Psi(\omega_{z_n d0} + \omega_{z_n d1})\} - \Psi(\phi_{z_n}).$$

Eqn.46 is also linear in $\mathbf{x}_n$ facilitating the application of kd-trees.

D Split and Merge Procedure

We have followed the split and merge procedure in the SMEM algorithm [Ueda et al., 2000]. In this section, we describe split and merge criteria, J_{split} and J_{merge}, and the initialization of newly created clusters.

D.1 Split Criterion

The split criterion J_{split} is defined as

$$J_{split}(c) = \int \mathrm{d}x \; f_c(x|\bar{\theta}) \log \frac{f_c(x|\bar{\theta})}{p(x|c,\bar{\theta})}, \tag{47}$$

where

$$f_c(x|\bar{\theta}) = \frac{\sum_{n=1}^{N} \delta(x - x_n) p(c|x_n,\bar{\theta})}{\sum_{n=1}^{N} p(c|x_n,\bar{\theta})}, \tag{48}$$

$$\text{with} \qquad \bar{\theta} = \int \mathrm{d}\theta \; \theta \; q(\theta). \tag{49}$$

J_{split} is a KL-divergence between the local empirical density $f_c(x)$ around cluster c, where each data-cases is weighted according to its responsibility under a mixture of Gaussians model with average parameters $\bar{\theta}$. A cluster which has the largest value for J_{split} should be split because the large J_{split} means that the cluster has a poor estimate of its local density.

D.2 Merge Criterion

The merge criterion J_{merge} is a cosine between N-dimensional vectors $\mathbf{P}_{c1}$ and $\mathbf{P}_{c2}$,

$$J_{merge}(c1, c2|\bar{\theta}) = \frac{\mathbf{P}_{c1}^T \mathbf{P}_{c2}}{||\mathbf{P}_{c1}|| \; ||\mathbf{P}_{c2}||} \tag{50}$$

where

$$\mathbf{P}_c = (p(c|x_1,\bar{\theta}), \cdots, p(c|x_N,\bar{\theta}))^T. \tag{51}$$

represent the responsibilities. When for two clusters these responsibility vectors are very similar, i.e. their J_{merge} is large, then the clusters are good candidates for a merge.

D.3 Initialization of New Clusters

Clusters created by split or merge are initialized as follows. When we split a cluster c, we run one iteration of the usual K-means algorithm with centroids $\mathbf{m} \pm \mathbf{d}$ where $\mathbf{m}$ is

the centroid of cluster c and $\mathbf{d} = \mathbf{s}\sqrt{\lambda}$ with $\mathbf{s}$ is the principal eigenvector of cluster c and λ is its eigenvalue.

After we merge clusters we compute new hyper-parameters using Eqn.18.

References

H. Attias. A variational bayesian framework for graphical models. In *NIPS*, volume 12, 2000.

Sanjoy Dasgupta. Experiments with random projection. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence*, pages 143–1, 2000.

C. Elkan. Using the triangle inequality to accelerate k-means. In *Proc. 20 International Conf. on Machine Learning*, pages 147–153, 2003.

David Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1998. URL `citeseer.ist.psu.edu/eppstein98fast.html`.

Z. Ghahramani and M. J. Beal. Variational inference for Bayesian mixtures of factor analysers. In *NIPS*, volume 12, 2000.

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. In *Advances in Neural Information Processing Systems 18*, pages 475–482, 2006.

G. Hamerly and C. Elkan. Learning the k in k-means. In *Neural Information Processing Systems*, volume 17, 2003.

U. Hinton and G. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proc. 6th Annual Workshop on Comput. Learning Theory*, pages 5–13. ACM Press, New York, 1993.

S.N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Communications in Statistics*, 7:223–238, 1998.

A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *NIPS*, volume 10, 1998.

A. Moore. The anchors hierarchy: Using the triangle inequality to survive high-dimensional data. In *Proc. of the 12th Conf. on Uncertainty in Artificial Intelligence*, pages 397–405, 2000.

D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proc. of the 5th Int'l Conf. on Knowledge Discovery in Databases*, pages 277–281, 1999.

D. Pelleg and A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *ICML*, volume 17, pages 727–734, 2000.

C.E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. Journal*, 27: 379–423, 623–656, 1948.

N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

J. Verbeek, J. Nunnink, and N. Vlassis. Accelerated variants of the em algorithm for gaussian mixtures. Technical report, University of Amsterdam, 2003.

T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *ACM-SIGMOD Int. Conf. Management of Data*, pages 103–114, 1996.