Incremental Learning of Nonparametric Bayesian Mixture Models

Anonymous CVPR submission

Paper ID 2926

Abstract

1. Introduction

2. Background

We start by reviewing the Dirichlet Process Mixture Model (DPMM) [2].

2.1. Dirichlet Process Mixture Model

In the traditional mixture model, a data point x_t is drawn i.i.d. from the distribution:

$$p(x_t) = \sum_{k=1}^{K} \pi_k p(x_t|\Phi_k), \tag{1}$$

where there are K components with associated parameters Φ_k, and a set of mixing weights π_k that sum to 1. An equivalent interpretation is that at each time t, first a component parameter ϕ_t is drawn from a discrete probability measure G and then x_t is drawn from the distribution $p(x_t|\phi_t)$:

$$p(x_t) = \int p(x_t|\phi_t) dG(\phi_t) \tag{2}$$

where $G(\cdot) = \sum_{k=1}^{K} \pi_k \delta_{\Phi_k}$, and $\sum_k \pi_k = 1$. In the DPMM, two extensions are introduced. The first is that $G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\Phi_k}$, meaning there are now a countably infinite number of distinct component parameter values (clusters) to choose from. The second extension is that the discrete measure G is itself treated as a random quantity drawn from a stochastic process $DP(\alpha, H)$ known as the Dirichlet Process [7]. H is a base probability distribution that serves as a prior over the component parameters Φ_c. $\alpha > 0$ is a concentration parameter that controls the relative sizes of the mixing weights π_c. When α is small, there tend to be a small number of large mixing weights (clusters), and when it is large there is a tendency to have many small weights. Random measures drawn from a Dirichlet Process are guaranteed to be discrete and $\sum_{c=1}^{\infty} \pi_c = 1$, as required to make a well-defined mixing distribution. To summarize, the DPMM generative model is:

- Draw random mixing measure $G \sim DP(\alpha, H)$
- For each x_t where $t = \{1 \dots N\}$
  1. Draw component parameter $\phi_t \sim G$
  2. Draw $x_t \sim p(x_t|\phi_t)$

It is convenient to introduce a set of auxiliary assignment variables $Z = \{z_1, \dots, z_n\}$, one for each data point x_t. $z_t \in \mathbb{N}$ designates the mixture component that generated data point x_t, equivalently $\phi_t = \Phi_{z_t}$. The assignment variables Z specify a clustering or partition of the data.

It is also useful to introduce an explicit construction for the mixture weights π_c in terms of *stick breaking* variables V_c [12]. The stick breaking variables V_c are assumed to be distributed independently according to $V_c \sim Beta(1, \alpha)$, where α is the concentration parameter of the Dirichlet Process. The mixing weights are then defined as

$$\pi_c = V_c \prod_{i=1}^{c-1} (1 - V_i). \tag{3}$$

In learning, we are interested in estimating the mixing distribution G given a sample of data $X = \{x_1 \dots x_n\}$. This entity is distributed according to the posterior stochastic process $G|X \sim DP(\cdot|X, \alpha H)$, which is itself a Dirichlet Process. It is sufficient to consider the posterior $p(Z, \Phi, V|X)$ over the assignment variables, the stick breaking variables, and the component parameters.

2.2. Exponential Family and Sufficient Statistics

We will restrict ourselves to component distributions that are members of the exponential family:

$$p(x|\phi) = f(x) \exp\{\phi^T F(x) + a(\phi)\}, \tag{4}$$

where $F(x)$ is a fixed length vector sufficient statistic. An additional restriction is that the base measure of the Dirichlet process must be conjugate to the component distributions. It must be of the form:

$$H = p(\phi|\nu, \eta) = h(\eta, \nu) \exp\{\phi^T \nu + \eta a(\phi)\}. \tag{5}$$

η and ν are the natural parameters for the conjugate prior distribution. The exponential family includes the Gaussian, Multinomial, Beta, Gamma, and other distributions.

The following fact is fundamental to our approach: If a set of observations X are all assigned to the same mixture component ($z_i = c$ for all i such that $x_i \in X$), then the component parameter ϕ_c is a function of the quantity

$$S = \sum_{x_i \in X} F(x_i), \tag{6}$$

which is the sum of the sufficient statistic vectors of each observation $x_i \in X$. The significance of this fact is that if a set of assignment variables are constrained to be equal (i.e. their corresponding observations are assumed to be generated by the same mixture component). Their inferential impact can be fully summarized by S, a vector whose length does not increase with the number of observations.

3. Existing Approaches

We briefly review existing approaches for learning Bayesian Mixture Models in an online fashion. Existing approaches have in common that they explicitly consider a number of alternative clusterings or mixture models in parallel, and update each of these hypotheses independently as new data arrives. We find that this approach becomes very expensive for learning Dirichlet Process Mixture models, as the number of alternative clusterings that must be modeled becomes very large as the amount of data increases.

3.1. Online Variational Bayes

Sato [1] derives recursive update rules for Variational Bayesian learning of mixture models. Sato's technique produces an update to the mixture models after every data point arrival. Only the alternative models are stored in memory, and each data point is discarded after it is used to update each parallel hypothesis. A "forgetting factor" is used in order to decay the contribution of "old" data points, since they are likely to be incorrectly assigned to components. Empirically, the forgetting factor means that much more data is needed to learn a model when compared with a batch technique. This makes the forgetting factor undesirable from the standpoint of our requirement to have an incremental algorithm that outputs results substantially close to the results that a batch algorithm would output given the total data seen. Also, choosing the forgetting factor is non-trivial and the choice may be dependent on the nature of the data. Finally, model parameters must be stored for each alternative hypotheses, and this becomes prohibitively expensive as the number of models increases. Indeed, [1] demonstrates the technique on a relatively simple task with just two alternative mixture models.

3.2. Particle Filters

Fearnhead [6] developed a particle filter learning algorithm for the Dirichlet Process. The technique operates only with the assignment variables Z; all other parameters are integrated away. This approach approximates $p(Z^{T-1}|X^{T-1})$ with

$$\tilde{p}(Z^{T-1}|X^{T-1}) = \sum_{i=1}^{M} w^{(i)} \delta_{Z^{T-1,(i)}}, \tag{7}$$

where $Z^{T-1,(i)}$ are a set of M distinct particles (clustering hypotheses) with weights $w^{(i)}$. Upon arrival of x_T, the M particles are extended to include a new assignment z_T and none of the assignments for the previous observations change. Every possible extension of each particle is constructed, for a total of $\sum_{i=1}^{M}(k_i+1)$, where k_i is the number of clusters inherent in the i-th particle. This is because x_T may be assigned to one of the particle's existing clusters, or to a new cluster. These new particles are assigned initial weights according to

$$w^{(i,j)} \propto p(x_T, z_T = j|X^{T-1}, Z^{T-1,(i)})w^{(i)}, \tag{8}$$

where $w^{(i,j)}$ is the j-th extension of the i-th particle from time $T-1$. In order to prevent combinatorial explosion over time, only M of these descendant particles are retained. Fearnhead derives a threshold on the weights $w^{(i,j)}$, and keeps all particles above this threshold. A subset of those that fall below are selected using a stratified resampling scheme and given a new weight equal to the threshold. Fearnhead's approach is similar to a greedy beam search, in that mainly the top ranked clustering hypotheses are retained after each update.

In our experiments, this approach behaves poorly for large datasets. The relative rankings of clusterings Z^T given X^T (the data seen up until time T) can be drastically different from the relative rankings given the set of observations X^n available at a later time. In other words, unseen observations can have a drastic effect on the rankings of the assignments Z^T. The algorithm greedily keeps only the top ranked assignments, and those that it discards can never be considered in the future. The allowance of multiple particles should somewhat counteract this tendency, however it is observed that these particles tend to concentrate themselves in a single clustering mode.

4. Our Approach

We observe that the chief difficulty with existing approaches is that they must explicitly enumerate and update a huge number of alternative clusterings in order to produce accurate results. We wish to avoid this explicit enumeration, while at the same time keeping a large number of alternatives alive for consideration. Our approach must also

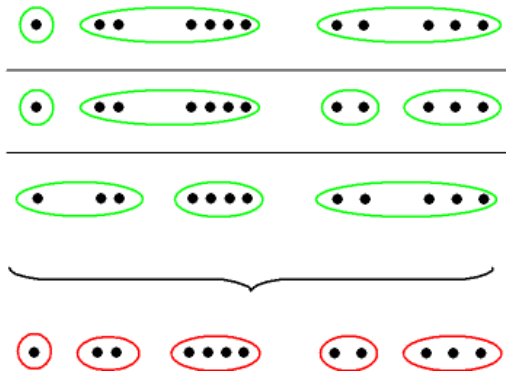

Figure 1. A schematic depiction of a one-dimensional clustering problem. Alternative solutions are displayed in green. The set of clump constraints consistent with all solutions are shown in red.

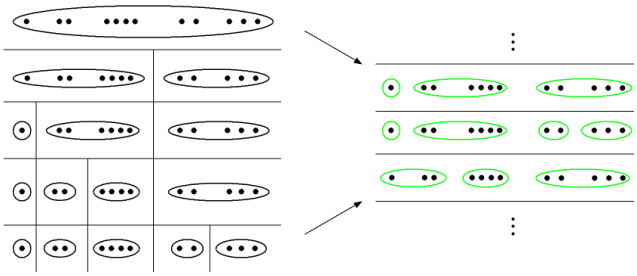

Figure 2. On the left, the problem is decomposed into (approximately) independent subproblems. The bottom level represents the set of constraints consistent with all identifed groups (these form "clumps" of points that are constrained to have equal $q(z_s)$.) On the right, the implicit ensemble of solutions that can be hierarchically composed using the clumps.

require bounded time and space requirements to produce an update given new data: the computational requirements must not scale with the total number of data seen.

Figure 1 shows how multiple clustering hypotheses can be combined into a single set of assignment constraints. Rather than explicitly fixing the assignments in each parallel branch, the constraints now take the form of points that are grouped together in every alternative. We will call these groups of points "clumps." We define sets of indices C_s such that if $i \in C_s$ and $j \in C_s$ for some s, then data points x_i and x_j are assigned to the same component in all of the alternative clusterings. The sets C_s are disjoint, meaning that no data point can exist in more than one clump. A single optimization procedure done under the clump constraints will yield the best clustering mode (modulo local minima issues) that is compatible with the *implicit* ensemble of alternatives inherent in the constraints. The implicit ensemble of alternatives is huge; it is composed of every possible clustering of the clumps, and is much larger than could be explicitly modeled.

This raises the question: How can these clump constraints be computed without first explicitly computing a number of plausible solutions? We observe that alternative models, while distinct, have considerable redundancy. The reason is that the clustering of data points in one region of space has little impact on the clustering assignments of data in a distant part of space. Any two alternatives will tend to vary from one another only for a subset of data points. Our approach is to partition the clustering problem into a series of independent subproblems. This is carried out in a top down fashion, as illustrated in fig. 2. This forms a tree of possible groupings of data, and the bottom level of this tree defines our clump constraints. Variational Bayes techniques provide a convenient framework for carrying out this procedure.

Our algorithm receives data in epochs of E data points. After the arrival of each data epoch, we first compute an estimate of the current best estimate mixture model. Then we carry out a compression phase in which clump constraints are computed in a top down recursive fashion, and this phase halts when a stopping criterion is met. Data points that belong to the same clump are summarized by their average sufficient statistics (see sec. 2.2), and the individual data points from the epoch are discarded. The clumps now summarize the data in a compressed form, and these clumps are retained in the next data epoch. The clumps are each given an assignment variable z_s and can be treated in the same fashion as data points during the next round of model learning. We bound the computational time and space requirements in each data epoch by controlling the number of clump constraints discovered during the compression phase. Because of this property, we refer to our algorithm as Complexity Bounded Variational Dirichlet Process (CB-VDP).

4.1. Model Building Phase

Data epochs begin by computing a current best estimate mixture model using Variational Bayes (VB) [3] [8] [16]. In the Variational Bayes approach, intractable posterior distributions are approximated with simpler proxy distributions that are chosen so that they are tractable to compute. Blei and Jordan [4] extended this technique to the DPMM.

Given the observed data X^T, the batch VB algorithm optimizes the variational Free Energy functional:

$$\mathcal{F}(X^T; q) = \int q(V)q(\Phi)q(Z^T) \log \frac{p(V, \Phi, Z^T, X^T|\eta, \nu, \alpha)}{q(V)q(\Phi)q(Z^T)} dW, \quad (9)$$

which is a lower bound on the log-evidence

$\log p(X^T|\eta,\nu,\alpha)$. The proxy distributions

$$q(V) = \prod_{k=1}^{K} q(V_k;\xi_{k,1},\xi_{k,2}), \quad (10)$$
$$q(\Phi) = \prod_{k=1}^{K} q(\phi_k;\zeta_{k,1},\zeta_{k,2}),$$
$$q(Z^T) = \prod_{t=1}^{T} q(z_t)$$

are products of Beta distributions for the stick breaking variables (with hyperparameters ξ), component parameters (with hyperparameters ζ), and assignment variables, respectively. Update equations for each proxy distribution can be cycled in an iterative coordinate ascent and are guaranteed to converge to a local maximum of the free energy. The approach may be interpreted as a generalization of the EM algorithm. The true DPMM posterior allows for an infinite number of clusters, but the proxy posterior limits itself to K components. Kurihara *et al.* [10] showed that K can be determined by starting with a single component, and repeatedly splitting components as long as the free energy bound $\mathcal{F}(X^T;q)$ improves. Splitting and merging clusters [13] and then re-running update rules is effective for escaping undesirable local maxima.

Like the batch approach, our algorithm optimizes $\mathcal{F}(X^T;q)$ during model building, but this optimization is carried out under the clump constraints discovered during previous data epochs. The resulting Free Energy bound is a lower bound on the optimal batch solution. (In practice, the batch process itself may not achieve the optimal bound because of local optima issues.) Formally this can be expressed as:

Proposition 1 *The CB-VDP model-building phase optimizes $\mathcal{F}(X^T;q)$ subject to the constraints that $q(z_i) = q(z_j)$ for all $i \in C_s$ and $j \in C_s$ and all clump constraints C_s. The resulting solution lower bounds the optimal batch solution: $\mathcal{F}_{CB}(X^T;q) \leq \max_q \mathcal{F}(X^T;q)$.*

The bound follows because solutions to the constrained problem are in the space of feasible solutions of the unconstrained optimization problem.

Hyperparameter update equations that optimize the Free Energy under the clump constraints can be derived that depend only on the sufficient statistics of the points in each clump:

$$\xi_{k,1} = 1 + \sum_s n_s q(z_s = k) \quad (11)$$
$$\xi_{k,2} = \alpha + \sum_s n_s \sum_{j=k+1}^{K} q(z_s = k)$$
$$\zeta_{k,1} = \eta + \sum_s n_s q(z_s = k)\langle F(x)\rangle_s$$
$$\zeta_{k,2} = \nu + \sum_s n_s q(z_s = k)$$
$$q(z_s = k) \sim \exp(S_{sk})$$
$$S_{sk} = E_{q(V,\phi_k)} \log\{p(z_s = k|V)p(\langle F(x)\rangle_s|\phi_k)\}$$

$\langle F(x)\rangle_s$ are the average sufficient statistics of the points in clump s, and n_s are the number of data points in clump s. The free energy has the form:

$$\mathcal{F}(X^T;q) = -\sum_{k=1}^{K} KL(q(v_k)||p(v_k|\alpha)) \quad (12)$$
$$-\sum_{k=1}^{K} KL(q(\phi_k)||p(\phi_k|\lambda))$$
$$+\sum_s n_s \log \sum_{k=1}^{K} \exp(S_{sk})$$

Update equations 13 were used in [10] to augment DPMM learning with a kd-tree in order to speed up inference (also [15] for EM learning). Sufficient statistics of data points contained in child nodes were cached at nodes of the kd-tree and used to perform approximate inference. Our approach differs from these algorithms in several ways. [10] and [15] build a kd-tree upfront with the complete batch of data, whereas we process data in sequential epochs and recompute clump constraints after each epoch. We irreversibly discard individual data points that are summarized by clump statistics in order to save storage cost, whereas [10] and [15] always have the option of working with individual data points if it leads to improvement in a Free Energy bound. We do not use a kd-tree to compute clump constraints but instead build a tree by greedily splitting collections of data points according to a Free Energy based cost function, as discussed in the next section.

4.2. Compression Phase

The goal of the compression phase is to identify groups of data points that are very likely to belong to the same mixture component, no matter what the exact clustering behavior of the rest of the data is. These groups can then be summarized with sufficient statistics and treated as data in subsequent learning rounds.

CVPR #2926

4.2.1 Bootstap Free Energy

We must find collections of points that are not only likely to be assigned to the same component given the first T data points, but also are likely to remain together at some target time N, with $N >> T$. The clustering behavior after N data points is governed by the Free Energy $\mathcal{F}(X^N; q)$. We therefore estimate this quantity with

$$E_{p(X^{T+1:N})}\mathcal{F}(X^N; q(V), q(\Phi), q(Z^N)), \qquad (13)$$

where $p(X^{T+1:N})$ is the probability distribution over the unseen future data $X^{T+1:N}$. We do not know how future data is distributed, so we estimate $p(X^{T+1:N})$ as a series of i.i.d. samples from the empirical distribution of the first T data points X^T:

$$p(X^{T+1:N}) = \prod_{n=T+1}^{N} \frac{1}{T}\sum_{t=1}^{T}\delta(x_n - x_t). \qquad (14)$$

This is equivalent to the bootstrap procedure [5] from statistics, so we refer to eq. 15 as the *bootstrap Free Energy*. However, we need not draw random bootstrap samples from $p(X^{T+1:N})$ or carry out multiple Free Energy optimizations in parallel. Instead we derive closed form coordinate ascent equations that optimize eq. 15 directly.

Proposition 2 *The bootstrap free energy may be optimized with the following coordinate ascent update rules:*

$$\xi_{k,1} = 1 + \frac{N}{T}\sum_s n_s q(z_s = k) \qquad (15)$$

$$\xi_{k,2} = \alpha + \frac{N}{T}\sum_s n_s \sum_{j=k+1}^{K} q(z_s = k)$$

$$\zeta_{k,1} = \eta + \frac{N}{T}\sum_s n_s q(z_s = k)\langle F(x)\rangle_s$$

$$\zeta_{k,2} = \nu + \frac{N}{T}\sum_s n_s q(z_s = k)$$

$$q(z_s = k) \sim \exp(S_{sk})$$

$$S_{sk} = E_{q(V,\phi_k)} \log\{p(z_s = k|V)p(\langle F(x)\rangle_s|\phi_k)\}$$

The bootstrap free energy is:

$$E_{p(X^{T+1:N})}\mathcal{F}(X^N; q(V), q(\Phi), q(Z^N)) \qquad (16)$$
$$= -\sum_{k=1}^{K} KL(q(v_k)||p(v_k|\alpha))$$
$$-\sum_{k=1}^{K} KL(q(\phi_k)||p(\phi_k|\lambda))$$
$$+\frac{N}{T}\sum_s n_s \log\sum_{k=1}^{K}\exp(S_{sk})$$

The proof is omitted because of space constraints. Note the similarity to eqs. 13 and 14. The equations for the responsibilities $q(z_s)$ are unchanged, and we need not compute responsibilities for unseen future data. The remaining update equations are modified with a data magnification factor $\frac{N}{T}$. This would lead to overfitting if used during the model building phase, however it is not a concern in the compression phase. The compression phase's goal is to identify points that are assigned together now and in the future, not with producing models that generalize well. If the free energy after T data points (eqs. 13 and 14) are used instead, the compression phase does not split large groups of data aggressively enough, and the algorithm may lock in to mixture model solutions prematurely.

4.2.2 Greedy Top Down Optimization

As indicated in fig. 2, we compute clump constraints in a top down fashion. We start the process with the clustering estimate determined during the preceding model building phase, which is given by the assignment distributions $q(z_s = k)$. For simplicity's sake, we then hard assign each clump or data point to the partition with the highest responsibility:

$$r_s = \arg\max_k q(z_s = k). \qquad (17)$$

The variables r_s indicate which partition the clump (or data point) s belongs to in the compression process. We then proceed through each partition and split it along the principal component defined by the clumps in the partition. We iterate the bootstrap Free Energy update eqs. 17 for the points in the partition in order to refine this split. Note that these updates involve only the clumps in the partition, and they may be assigned only to the two subpartitions. After this update process converges, the clumps are then hard assigned to one of the candidate subpartitions.

Each potential partition split is then ranked according to the resulting change in the bootstrap free energy (eq. 18). Because of the hard assignment to partitions, this difference involves only quantities related to clumps in the candidate partition. We then greedily choose the split that results in the largest change in bootstrap free energy. The process then repeats itself, with the new partitions ranked in the same way described above. We cache the results of each split evaluation in order to prevent redundant computation. When the process halts (see below) the partitions become the new clumps: data and clumps within the partition are replaced with their sufficient statistics.

We must restrict the number of clumps that are retained in order to ensure that the time and space complexity is bounded in the next round of learning. A stopping criterion determines when to halt the top down splitting process. A number of possible criteria are possible, depending on the situation.

When learning DPMM's with full-covariance Gaussian components, each clump requires $\frac{d^2+3d}{2}+1$ values to store sufficient statistics (mean, symmetric covariance matrix, and number of data points summarized). It is convenient to express the stopping criterion as a limit on the amount of memory required to store the clumps. From this perspective, it makes sense to replace a clump with its sufficient statistics if it summarizes more than $\frac{d+3}{2}$ data points. If a clump summarizes fewer points, then the individual data points are retained instead. We refer to these individual retained data points as singlets. The clump memory cost for mixture models with full covariance matrices is therefore

$$MC = \left(\frac{d^2+3d}{2}+1\right) N_c + dN_s, \qquad (18)$$

where N_c is the number of clumps (partitions with greater than $\frac{d+3}{2}$ points) and N_s is the number of singlets (the number of data points that fall in partitions with fewer than $\frac{d+3}{2}$ data points). An upper limit on clump memory cost M is defined, and the compression phase halts when $MC \geq M$.

Algorithm 1 Complexity Bounded Variational Dirichlet Process

while There is more data to collect **do**
  Collect E data points from the world.
  Perform model building phase using new data and clumps from previous rounds following [10] and eqs. 13.
  Initialize compression phase according to eq. 19.
  while $MC < M$ (eq. 20) **do**
    for $k = 1$ to K **do**
      {Loop over partitions.}
      if $evaluated(k)$=FALSE **then**
        Split partition k and refine according to eqs. 17
        $\Delta bfe(k)$ = change in bootstrap free energy (eq. 18)
        $evaluated(k)$ =TRUE
      end if
    end for
  end while
  Split partition $\arg\max_k \Delta bfe(k)$
  $K = K + 1$
  Retain clumps (suffcient statistics of partitions) as well as singlets into next round.
end while

The complete CB-VDP algorithm is summarized in algorithm 1. We will show empirically that this algorithm is able to incrementally learn large data sets. The time required for the algorithm to learn the entire data set is typically less than the batch variational DPMM approach outlined in [10]. This is because full variational updates in the batch procedure require $O(KN)$, where K is the number of clusters and N is the number of data points. The CB-NB algorithm requires only $O(K(N_c + N_s))$ for an iteration during the model building phase. The time required during the compression phase is quite modest when compared to the model building phase, because the compression phase only entails restricted variational updates that involve subsets of the data. Our approach requires much less memory and will produce a best estimate after each epoch of data. The clustering estimates are quite similar to those produced by the batch procedure according to both qualitative and quantitative measures.

Vasconcelos and Lippman [14] proposed learning a hierarchy of EM mixture model solutions using a bottom up procedure (although they did not investigate this approach in the context of incremental learning). Could this approach be applied here in order to learn the clumps constraints more directly? Our experience has been that this is difficult. Variational updates for the DPMM are very sensitive to initial conditions, and multiple initializations would likely be necessary, from which the best solution would be chosen. In contrast, our top down method sidesteps this initialization problem.

5. Experimental Results

The first set of experiments compares the performance of our method with that of Fearnhead's particle filter. The data set consists of four categories from Caltech 256 (Airplanes, Motorbikes, Faces, and T-Shirts) that are projected to a 20 dimensional feature space using Kernel PCA with Lazebnik's Spatial Pyramid Match Kernel [11]. There are 1400 data points (images) in total. α (the concentration parameter for the Dirichlet Process) is set to 0.05, in order to obtain a relatively small number of components. The hyperparameters for Normal Inverse Wishart prior on cluster parameters (H) were chosen by hand, based on prior knowledge about the scale of the data. The batch algorithm tends to find 12 to 15 clusters in this setting. The clusters discovered respect the categories, that is, very few objects from different classes are clustered together. However, the algorithm divides each category into sub-categories according to perceptually relevant differences.

The algorithms were judged quantitatively according to predictive likelihood. 1300 of the 1400 images were chosen at random as a training set, and the algorithm is trained on a complete pass through this data. The data are also shuffled in to a random order. The remaining 100 images form a test set, and the average likelihood of each of these test points according to the learned model is computed as a measure for how well the algorithm is able to generalize to unseen data. Higher values are associated with better generalization. The particle filter was tested at different numbers of particles. The amount of memory was varied for our algorithm. In

Particles	Ave Predictive Log-Likelihood	Runtime
100	4.99 ± 0.34	9.9 min
1000	5.43 ± 0.28	47.6 min
10000	5.80 ± 0.22	6.9 hr

Table 1. Particle filtering predictive performance and runtime.

Memory	Ave Predictive Log-Likelihood	Runtime
200	6.37 ± 0.32	73.3 s
400	6.93 ± 0.32	57.08 s
600	6.99 ± 0.31	57.76 s

Table 2. CB-VDP predictive performance and runtime. Batch performance was 7.04 ± 0.28 with runtime $71.4s$ on 1300 data points.

our algorithm, the memory value represents the memory required to store both the clumps from earlier rounds of memory and the current epoch of points. In all cases, the epoch sizes are chosen to be one-half of the memory size, so for an effective memory of 200, the algorithm progesses through the data in epochs of 100 points. Note that at a memory size of 200 only the equivalent of 100 data points are available to store the clumps and therefore there is not enough space to store the full 12 to 15 components inherent in the data. The algorithm therefore produces a model with fewer clusters in this case. At memory 400 and 600, our algorithm discovers close to the same number of clusters that the batch algorithm discovers, with small differences depending on the ordering of the data.

Table 1 shows the performance of the particle filter, and table 2 shows the performance of our algorithm. Our algorithm beats the particle filter in terms of generalization accuracy at all parameter values. Our algorithm also produces generalization results that are close to the performance of the batch algorithm that trains on all 1300 points in the training set simultaneously. Error bars are calculated as the standard deviation of the mean.

The runtime advantage of our approach is very significant over that of the particle filter. Much of the advantage is due to the fact that our approach ammortizes expensive calculations over an entire epoch of data points instead of one at a time for the particle filter. Our approach is also better able to take advantage of vectorization.

The next experiment involves a much larger data set and so the particle filter is unsuitable. In this case, our approach is compared directly against the batch algorithm of [10]. The 60000 hand-written digits from the MNIST set were reduced to 50 dimensions using PCA in a preprocessing step. The resulting vectors were then normalized to have unit length. Our algorithm was set to have a memory size equivalent to 6000 data points, which is an order of magnitude smaller than the size of the data set. Our algorithm processes data in epochs of 3000.

Figure 3 shows the cluster means discovered by our algorithm as it passes through more data. Since the DPMM is

Figure 3. Cluster means discovered by incremental algorithm after 6000, 30000, and 60000 digits encountered. As expected, the model complexity increases as data arrives. However, the algorithm's memory usage is constant.

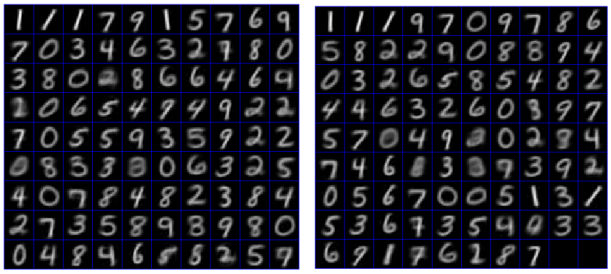

Figure 4. Left: Cluster centers produced by incremental algorithm after visiting all 60000 digits, with effective memory size of 6000 digits. Right: Cluster centers produced by batch algorithm. Clusters are ordered according to size, from top left to bottom right.

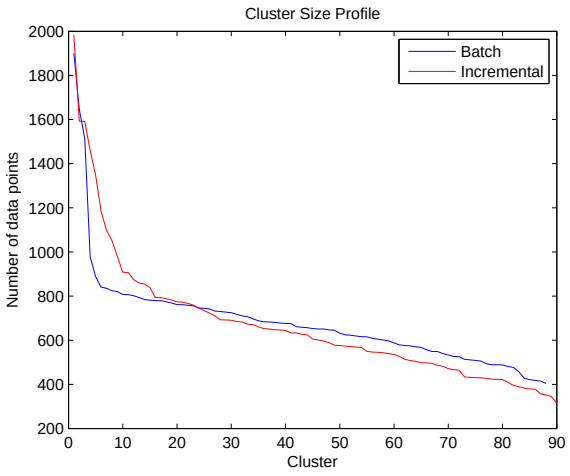

Figure 5. Size of clusters for batch and incremental (after full pass through dataset).

nonparametric, we expect the model complexity to increase as more data is seen. We find that this is the case with our algorithm. Figure 4 shows the cluster centers discovered by our approach after processing the entire data set compared to those produced by the batch algorithm. The clusters are qualitatively quite similar, and the two algorithms discover a comparable number of clusters (88 for the batch approach, 90 for our algorithm). In all figures the clusters are ordered according to size from the top left to bottom right.

Figure 5 shows the difference in sizes of the clusters for the two approaches. The run time for the batch algorithm was 31.5 hours, while for our approach it was 20 hours for a complete pass through. Our approach is faster, but its chief advantage is that a series of model estimates are available after each epoch, which can be put to use before the algorithm has seen the entire dataset. As in [10], we compare the free energy bounds produced by the two approaches. The ratio of these two values is 1.025 meaning that our incremental algorithm produces a slightly worse lower bound on the likelihood, with lower values being better. By comparison, the kd-tree accelerated algorithm in [10] produced a free energy ratio of 1.044.

References

[1] M. aki Sato. Online model selection based on the variational bayes. *Neural Computation*, 13(7):1649–1681, 2001. 2

[2] C. Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics- Institute of Mathematical Statistics*, 1974. 1

[3] H. Attias. A variational baysian framework for graphical models. In *NIPS*, pages 209–215, 1999. 4

[4] D. M. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Journal of Bayesian Analysis*, 1(1):121–144, 2005. 4

[5] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, London, 1993. 5

[6] P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Journal of Statistics and Computing*, 14:11–21, 2004. 2

[7] T. S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1973. 1

[8] Z. Ghahramani and M. Beal. Graphical models and variational methods, 2001. 4

[9] A. Honkela and H. Valpola. On-line variational bayesian learning. 2

[10] K. Kurihara, M. Welling, and N. Vlassis. Accelerated variational dirichlet process mixtures. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007. 4, 6, 7, 8

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR (2006)*, pages 2169–2178, 2006. 6

[12] J. Sethuraman. A constructive definition of dirichlet priors. *Statist. Sinica*, 4:639–650, 1994. 1

[13] N. Ueda, R. Nakano, Z. Gharamani, and G. Hinton. Smem algorithm for mixture models, 1999. 4

[14] N. Vasconcelos and A. Lippman. Learning mixture hierarchies. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 606–612, Cambridge, MA, USA, 1999. MIT Press. 6

[15] J. J. Verbeek, J. Nunnink, and N. A. Vlassis. Accelerated em-based clustering of large data sets. *Data Min. Knowl. Discov.*, 13(3):291–307, 2006. 4

[16] M. Wainwright and M. Jordan. A variational principle for graphical models. In *chapter 11 in: New Directions in Statistical Signal Processing*. MIT Press, 2005. 4