Memory Bounded Inference in Topic Models

Ryan Gomes GOMES@VISION.CALTECH.EDU
Dept. of Computation and Neural Systems, California Institute of Technology, Pasadena, CA 91125 USA

Max Welling WELLING@ICS.UCI.EDU
Bren School of Information and Computer Science, University of California at Irvine, Irvine, CA 92697 USA

Pietro Perona PERONA@VISION.CALTECH.EDU
Dept. of Computation and Neural Systems, California Institute of Techology, Pasadena, CA 91125 USA

Abstract

What type of algorithms and statistical techniques support learning from very large datasets over long stretches of time? We address this question through a memory bounded version of a variational EM algorithm that approximates inference in a topic model. The algorithm alternates two phases: "model building" and "model compression" in order to always satisfy a given memory constraint. The model building phase expands its internal representation (the number of topics) as more data arrives through Bayesian model selection. Compression is achieved by merging data-items in clumps and only caching their sufficient statistics. Empirically, the resulting algorithm is able to handle datasets that are orders of magnitude larger than the standard batch version.

1. Introduction

Consider a collection of surveillance cameras monitoring at an airport. The cameras learn a model of their environment without supervision. Moreover, they learn for many years without significant interruption. Gradually, as more data is captured, the cameras build a joint model of visual object categories.

This problem is akin to the way children learn to understand the world through the *continuous* process of mostly unsupervised learning. As children grow up they build an increasingly sophisticated internal representation of object categories that continuously restructures itself.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

In this paper we ask ourselves: What statistical techniques are suitable for this "*lifelong learning task*"? First, we need a class of models that can naturally expand as more data arrives, i.e. it's capacity should not be bounded a priori. Second, these models should allow efficient learning algorithms, both in terms of time and space. For instance, we should not have to store every single piece of information that has been captured. Our technique must produce a sequence of model estimates that reflect new information as it arrives, and the time required to produce each model update must scale modestly as more data is acquired. Finally, we require that the sequence of learned models are sufficiently similar to those that would be produced by a batch algorithm with access to the entire history of data observed at the time of each model update.

Nonparametric Bayesian techniques such as the Dirichlet Process (DP) (Ferguson, 1973) and the Hierarchical Dirichlet Process (HDP) (Teh et al., 2006) satisfy our first desideratum, in that they naturally increase their model complexity with the available data. However, most existing Nonparametric Bayesian approaches are batch algorithms: they require every single data-point to be stored and revisited during learning. A batch algorithm could be naively applied to the continuous learning scenario, but all data would need to be cached and a new batch learning process would be run on the entire dataset to produce each model update. This would violate our second criterion in that the time and space requirements would increase unacceptably as the system ages.

Here we propose a more flexible setup, where we impose a bound on the available memory but still allow the model order to increase with more data. We *compress* the data and the internal representation of the model without losing much in terms of model accuracy. The effect is that time and space requirements scale much more gradually over the lifetime of the system. The memory bound does impose a limit on the total capacity of the model, but this trade-off

is flexible and can be adjusted online, i.e. as the model is learned. Experiments with a memory bounded variational approximation to HDP show that this technique can handle datasets many times larger than the standard implementations and results in substantially shorter run-times.

2. A Memory Bounded Variational Topic Model

At a high level the idea is to use a variational approximation related to LDA (Blei et al., 2003) and HDP (Teh et al., 2006). Memory savings are achieved by "clumping" together data-cases. That is, we constrain groups of datapoints to have equal topic assignment variational distributions: $q(z_{ij}) = q(z_{i'j'}) = q(z_c)$ when points x_{ij} and $x_{i'j'}$ are members of the clump c. This allows us to achieve memory savings, because variational optimization performed under this constraint requires only the sufficient statistics of the data-cases in a clump, and the system can forget the exact identities of the summarized data points. Similarly, we will also clump entire documents (or images) by tying their variational distributions over topics: $q(\boldsymbol{\pi}_j) = q(\boldsymbol{\pi}_{j'}) = q(\boldsymbol{\pi}_s)$ if document j and j' belong to the same document group s. This tying of variational distributions guarantees that learning optimizes a lower bound to the exact free energy objective function, where the bound is increasingly loose with more tying. This idea was also leveraged in (Verbeek et al., 2003) and (Kurihara et al., 2006) to accelerate learning of Mixtures of Gaussians and DP Mixtures of Gaussians by using KD-trees.

In the following we will talk about documents, but we note that this refers to other structured objects such as images as well.

2.1. The Variational Topic Model

The following Bayesian topic model is our starting point,

$$p(\mathbf{x}, \mathbf{z}, \boldsymbol{\eta}, \boldsymbol{\pi}, \boldsymbol{\alpha}) = \prod_{ij} p(\mathbf{x}_{ij}|z_{ij}; \boldsymbol{\eta})\, \pi_{j,z_{ij}} \tag{1}$$
$$\left[\prod_k p(\boldsymbol{\eta}_k|\boldsymbol{\beta})\right] \left[\prod_j \mathcal{D}(\boldsymbol{\pi}_j; \boldsymbol{\alpha})\right] \left[\prod_k p(\alpha_k)\right]$$

where $\mathbf{x}_{ij}$ is word i in document j and z_{ij} denotes the topic that generated $\mathbf{x}_{ij}$. $\boldsymbol{\pi}_j$ denotes the mixture of topics that generated the words in document j, with $\sum_k \pi_{jk} = 1$. $\boldsymbol{\pi}_j$ are distributed according to a Dirichlet distribution with parameter $\boldsymbol{\alpha}$. Boldface symbols denote vector valued quantities. In this expression we will assume that $p(\mathbf{x}|z, \boldsymbol{\eta})$ is in the exponential family[1],

$$p(\mathbf{x}|z = k, \boldsymbol{\eta}) = \exp\left[\sum_l \eta_{kl}\, \phi_l(\mathbf{x}) - A_k(\boldsymbol{\eta}_k)\right] \tag{2}$$

and $p(\boldsymbol{\eta}|\beta)$ is conjugate to $p(\mathbf{x}|z, \boldsymbol{\eta})$,

$$p(\boldsymbol{\eta}_k|\boldsymbol{\beta}) = \exp\left[\sum_l \beta_l \eta_{kl} - \beta_0 A_k(\boldsymbol{\eta}_k) - B(\boldsymbol{\beta})\right] \tag{3}$$

The posterior distributions over $\boldsymbol{\pi}, \boldsymbol{\eta}, \mathbf{z}$ are approximated variationally as

$$q(\boldsymbol{\eta}) = \prod_k q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k) \tag{4}$$

$$q(\boldsymbol{\pi}) = \prod_j \mathcal{D}(\boldsymbol{\pi}_j; \boldsymbol{\zeta}_j) \tag{5}$$

$$q(\mathbf{z}) = \prod_{ij} q(z_{ij}) \tag{6}$$

where we have introduced variational parameters $\{\boldsymbol{\xi}_{kl}, \zeta_{kj}, q_{ijk}\}$, the latter subject to $\sum_k q_{ijk} = 1$. Furthermore, $\mathcal{D}$ denotes a Dirichlet distribution while $q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k)$ is also conjugate to $p(\mathbf{x}|z = k, \boldsymbol{\eta})$,

$$q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k) = \exp\left[\sum_l \xi_{kl}\eta_{kl} - \xi_{k0} A_k(\boldsymbol{\eta}_k) - B_k(\boldsymbol{\xi}_k)\right] \tag{7}$$

By writing down the variational free energy and minimizing it over $\boldsymbol{\xi}, \boldsymbol{\zeta}$ we find the following intuitive updates,

$$\xi_{kl} = F_{kl} + \beta_l; \qquad F_{kl} \triangleq \sum_{ij} q_{ijk}\, \phi_l(\mathbf{x}_{ij}) \tag{8}$$

$$\xi_{k0} = N_k + \beta_0; \qquad N_k \triangleq \sum_{ij} q_{ijk} \tag{9}$$

$$\zeta_{kj} = N_{kj} + \alpha_k; \qquad N_{kj} \triangleq \sum_i q_{ijk} \tag{10}$$

and

$$q_{ijk} \leftarrow \frac{1}{Z_{ij}} \frac{\exp\left[\sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\, \phi_{kl}(\mathbf{x}_{ij})\right]}{\exp\left[\mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}]\right]} \exp\left[\psi(\zeta_{kj})\right] \tag{11}$$

where Z_{ij} enforces the constraint $\sum_k q_{ijk} = 1$ and the expectations are over $q(\boldsymbol{\eta})$.

To learn the parameters $\{\alpha_k\}$ we first introduce gamma priors,

$$p(\boldsymbol{\alpha}) = \prod_k \mathcal{G}(\alpha_k; a, b) \tag{12}$$

[1] Strictly speaking, the exponential family includes additional multiplicative terms $h(\mathbf{x})$ in the expression for $p(\mathbf{x}|\boldsymbol{\eta})$ and $g(\boldsymbol{\eta})$ in the expression for $p(\boldsymbol{\eta}|\boldsymbol{\beta})$. We have left these terms out to simplify the derivation and because for most well known distributions they are simply 1. However, it is straightforward to include them.

Using the bounds in (Minka, 2000) we can derive the following updates if we first insert the updates for $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ into the free energy,

$$\alpha_k \leftarrow \frac{(a-1) + \alpha_k \sum_j [\psi(\zeta_{kj}) - \psi(\alpha_k)]}{b + \sum_j [\psi(\zeta_j) - \psi(\alpha)]} \tag{13}$$

with $\zeta_j = \sum_k \zeta_{kj}$ and $N_j = \sum_k N_{kj}$.

2.2. Optimizing the Number of Topics K

Our strategy to search for a good value of K is to *truncate* the topic distributions as $q(z_{ij} > K) = 0$ (see also (Teh et al., 2008)). This will have the effect that most terms in the free energy with $k > K$ will cancel, the exception being the prior terms $p(\alpha_k),\ k > K$. For these terms we know that the value for α_k minimizing the free energy is given by the MAP value of the gamma-prior $\alpha_k = \frac{a-1}{b},\ k > K$. Inserting this back into the free energy we accumulate $K_{\text{max}} - K$ terms

$$\Lambda = a \log b - \log \Gamma(a) + (a-1) \log \frac{a-1}{b} - (a-1) \tag{14}$$

where K_{max} is the maximum number of topics.

It is guaranteed that there exists a solution with lower free energy if we increase K. The reason is that we relax a self-imposed constraint on variational parameters (that $q(z_{ij} > K) = 0$). As K increases the relative improvement in free energy quickly attenuates. The final value for K is obtained by thresholding this relative improvement.

The *nesting* property (models with larger K are better) is the same for variational approximations to the DP in (Kurihara et al., 2006) and HDP (Teh et al., 2008). This raises the question if we can take the infinite limit for our model as well. The problem is that $(K_{\text{max}} - K)\Lambda \rightarrow \infty$ as $K_{\text{max}} \rightarrow \infty$. This can be traced back to the fact that we should have added a proper prior $p(K)$ which would have diminished the contribution at large K. Instead we choose an improper, constant prior to avoid the need to estimate likely values for K a priori. However, it is still possible to work with infinite free energies because we are only interested in the relative *change* in free energy after increasing K, which is a finite quantity.

In our experiments we chose $a = 1$ and $b = 0.5$, so that the MAP prior value of α_k is 0.

2.3. Clumping Data-Items and Documents

We will now tie some of the variational distributions $\{q_{ijk}\}$ across different data-items within and across documents (images) to a "clump distribution" q_{ck}. Similarly, we will tie some document specific distributions over topics $\{q(\boldsymbol{\pi}_j)\}$ into a document group $q(\boldsymbol{\pi}_s)$. Note that since we impose constraints on the variational distributions this has the effect of loosening the variational bound.

Define D_s to be the number of documents in a document group, N_c the number of data-items in a word clump, N_{cs} the number of words in document group s and word clump c and finally $\Phi^c_{kl} \triangleq \sum_{ij \in c} \phi_{kl}(\mathbf{x}_{ij})$. In terms of these we further define,

$$N_{ks} \triangleq \sum_c q_{ck} N_{cs} \tag{15}$$

$$N_k \triangleq \sum_c q_{ck} N_c \tag{16}$$

$$F_{kl} \triangleq \sum_c q_{ck}\ \Phi^c_{kl} \tag{17}$$

With these definitions we derive the following "clumped" update rules for the variational parameters ξ_{kl} and ζ_{ks},

$$\xi_{kl} = F_{kl} + \beta_l \tag{18}$$

$$\xi_{k0} = N_k + \beta_0 \tag{19}$$

$$\zeta_{ks} = \tfrac{N_{ks}}{D_s} + \alpha_k \tag{20}$$

and

$$q_{ck} \leftarrow \frac{1}{Z_c} \frac{\exp\left[\sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\ \frac{\Phi^c_{kl}}{N_c}\right]}{\exp\left[\mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}]\right]}\ \exp\left[\sum_s \frac{N_{sc}}{N_c} \psi(\zeta_{ks})\right] \tag{21}$$

The update for $\boldsymbol{\alpha}$ becomes

$$\alpha_k \leftarrow \frac{(a-1) + \alpha_k \sum_s D_s [\psi(\zeta_{ks}) - \psi(\alpha_k)]}{b + \sum_s D_s [\psi(\zeta_s) - \psi(\alpha)]} \tag{22}$$

An expression for the free energy, after inserting expressions 18, 19 and 20, is given by eq. 29 in the appendix.

3. Incremental Learning with a Memory Constraint

Our algorithm processes data in small groups composed of E documents, which we refer to as *epochs*. After the arrival of each epoch the algorithm proceeds in two stages: a model building phase during which a new model estimate is produced, and a compression phase in which decisions are made as to which words and documents to clump. The sufficient statistics of each clump are computed and data summarized by clumps are purged from memory. The assignment distributions $q(z)$ of purged data and topic distributions of merged documents $q(\boldsymbol{\pi})$ are discarded as well. The clump sufficient statistics are retained along with the current model estimate, which serves as a starting point for the next round of learning.

Model Building Phase (Algorithm 3.1)

Input: Previous model $\{\xi_{kl}, \zeta_{ks}, \alpha_k, \Phi^c_{kl}, N_{cs}, D_s\}$, and current epoch of E documents.
Initialize $\zeta_{jk} = \alpha_k$ for $j = |S|+1, \cdots, |S|+E$
Iterate eqs. 21, 18, 19, 20, and 22 until convergence
repeat
  Rank splits and merges according to criteria in (Ueda et al., 1999)
  for $i = 1$ **to** 10 **do**
    Split i-th ranked candidate topic along principal component
    Restricted iteration of eqs. 21, 18, 19, and 20 until convergence
    Evaluate change in eq. 29 resulting from split
  end for
  for $i = 1$ **to** 10 **do**
    Merge i-th ranked pair of topics
    Evaluate change in eq. 29 resulting from merge
  end for
  Select split or merge that yielded largest change in eq. 29
  Iterate eqs. 21, 18, 19, and 20 until convergence
until Change in eq. 29 is less than threshold

3.1. Model Building Phase

The model building phase optimizes the free energy under the parameter tying constraints induced by the choice of clumps in previous compression phases. We perform a split-merge procedure similar to (Ueda et al., 1999) to determine the number of topics, using the heuristics in that work to rank topic suitability for split or merge. In our experiments we use Gaussian topic distributions, so splits are proposed along the principal component of the topic. The split proposals are refined by restricted variational updates. That is: equations 21, 18, 19, 20, and 22 are iterated but only for data-points whose highest responsibility is to the split topic, and the points may be assigned only to the two descendent topics. Merges are carried out by instantiating a new topic with the data-points with highest responsibility to the merged topics. A total of 10 splits and 10 merges are proposed, and evaluated by the resultant change in free energy (eq. 29). The top ranked change is then used to initialize full variational updates (which involve all data points). The model building phase halts once the change in free energy divided by its previous value is below a threshold, which was chosen to be $1E-5$ in our experiments. The procedure is summarized in algorithm 3.1.

3.2. Compression Phase

The goal of the compression phase is to determine groups of data-points that are to be summarized by clumps, and to identify documents that are to be merged into document groups.

Clumps are identified using a greedy top down splitting procedure. Because datapoints summarized by clumps are ultimately discarded, the compression process is irreversible. Therefore it is of fundamental importance to predict the locations of future data when deciding which points to clump. In order to estimate this, we rank cluster splits according to a modified free energy (eq. 30) in which the data sample size is artificially increased by a factor $\frac{T_{pts}}{\sum_c N_c}$ and the number of documents is scaled by $\frac{T_{docs}}{\sum_s D_s}$, where T_{pts} and T_{docs} are the target number of data-points and documents expected during the lifetime of the system. This is equivalent to using the data empirical distribution as a predictive model of future data. If we determine clumps using the standard free energy, then the algorithm fails to split large groups of points that are likely to split once more data has arrived. Instead, it wastes memory by placing "stray" points in their own clumps.

We initialize the process by hard assigning each clump or data-point to the cluster with highest responsibility during the previous model building phase. We then proceed through each cluster and split it along the principal component, and refine this split by iterating restricted variational updates equations for the points in the cluster. The updates are modified by the data magnification factors:

$$\xi_{kl} = \left(\frac{T_{pts}}{\sum_c N_c}\right) F_{kl} + \beta_l \tag{23}$$

$$\xi_{k0} = \left(\frac{T_{pts}}{\sum_c N_c}\right) N_k + \beta_0 \tag{24}$$

$$\alpha_k \leftarrow \frac{(a-1) + \left(\frac{T_{docs}}{\sum_s D_s}\right) \alpha_k \sum_j \left[\psi(\zeta_{ks}) - \psi(\alpha_k)\right]}{b + \left(\frac{T_{docs}}{\sum_s D_s}\right) \sum_s \left[\psi(\zeta_s) - \psi(\alpha)\right]} \tag{25}$$

Updates for q_{ck} and ζ_{ks} are unchanged. After the clusters are refined, the data-points are then hard assigned to the sub-cluster with greatest responsibility, and the proposed split is ranked according to the resultant change in eq. 30. We then greedily split the cluster with highest rank. The process repeats itself, with new clusters ranked in the same way described above. We cache the results of each split evaluation to avoid redundant computation. After we have reached a given memory bound we extract the partitions resulting from this recursive splitting procedure as our new clumps.

Each clump must store sufficient statistics for full covariance Gaussian components which require $\frac{d^2+3d}{2}$ values, where d is the dimension of the feature space. In addition, $|S|$ (the number of document groups) values must be

Clump Compression (Algorithm 3.2)

Input: Output from model building phase: $\{q_{ck}, \Phi^c_{kl}, N_{cs}, D_s\}$, current epoch of E documents and memory bound M.
Hard partition clumps: $r_c = \arg\max_k q_{ck}$
while $MC < M$ (eq. 26) **do**
  for $i = 1$ **to** K **do**
    Split i-th cluster along principal component
    Iterate data magnified restricted updates until convergence
    Hard partition clumps into child clusters
    Evaluate change in eq. 30 resulting from split
  end for
  Select split that yielded largest change in eq. 30
  $K = K + 1$
end while

stored to represent the counts N_{cs} for each clump. Note that from this perspective, it only makes sense to create clumps within a cluster if it contains more than $\frac{d+3}{2} + \frac{1}{d}$ data-points. If not, then it is more efficient to store the individual data-points and we refer to them as "singlets". The total memory cost of summarizing the data is then

$$MC = \left(\frac{d^2 + 3d}{2}\right)|N_c > 1| + |S||N_c > 1| + d|N_c = 1|, \tag{26}$$

where $|N_c > 1|$ is the number of clumps with more than 1 data-item in them, and $|N_c = 1|$ is the number of singlets. The clump compression procedure is summarized in algorithm 3.2.

Document merging provides another way of controlling the memory cost, by reducing the number of image groups $|S|$. We use the following simple heuristic to rank the suitability of merging document groups s and s':

$$DM_{s,s'} = \frac{\sum_k \mathbb{E}[\pi_{sk}]\mathbb{E}[\pi_{s'k}]}{||\mathbb{E}[\boldsymbol{\pi}_s]||||\mathbb{E}[\boldsymbol{\pi}_{s'}]||} \tag{27}$$

Clumping and document merging enable a number of potential schemes for controlling space and time costs, depending on the application. We note that the time complexity per variational iteration scales as $O(K(|N_c > 1| + |N_c = 1|) + |S|K)$ and the space required to store $q(z_c)$ distributions is $O(K(|N_c > 1| + |N_c = 1|))$.

4. Experiments

We test our approach with two machine vision experiments. The first is an image segmentation task, and the second is an object recognition and retrieval task.

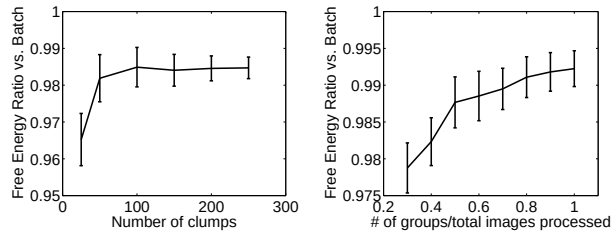

Figure 1. Image Segmentation experiment. Left: Free energy ratio as a function of the number of clumps permitted by the memory bound. Right: Free energy ratio versus the number of image groups relative to the total number of images processed.

4.1. Joint Image Segmentation

Our first experient is a joint image segmentation problem. The dataset is the Faces-Easy category of the Caltech 101 image dataset (Fei-Fei et al., 2004) consisting of 435 images. Each image contains a face centered in the image, but the lighting conditions and background vary. In terms of the vocabulary of the preceding sections, each image is a document and each pixel in the image is a word. Pixels are represented as five dimensional vectors of the following features: X and Y position relative to the center of the image, and three color coordinates in the CIELAB colorspace. The goal of our experiment is to find similar image regions across the multiple images, in an unsupervised way. We emphasize that our main objective is to study the efficiency of our algorithm, not to produce a state of the art image segmentation algorithm.

The images were scaled to be 200 by 160 pixels in size. Thus, the total size of the dataset is 32,000 pixels per image, times 435 images, times 5 features per pixel equals 69,600,000 real numbers. Each pixel requires an assignment distribution. Our baseline implementation (i.e. a batch algorithm that processes all images in memory at once and does not use pixel clumping or image merging) was only able to jointly segment 30 images simultaneously, before running out of memory. The majority of memory is used to store the assignment distributions of pixels, and this is problematic as the number of topics increases during learning, since the space requirements scale as $O(NK)$, where N is the total number of pixels and K is the number of topics.

We first compare the memory bounded approach to the baseline implementation on a joint segmentation task of 30 images in order to judge the impact of the pixel clumping approximation. We vary the upper limit on the number of clumps used to summarize the data during the compression phase, and compare the free energy bounds produced by the memory bounded algorithm to those produced by the baseline implementation. We define the free energy ratio

Figure 2. Top row: From left to right: an example segmentation produced by the baseline method, memory bounded algorithm with 30% of total images and 125 clumps, and the memory bounded algorithm with no images merged and 125 clumps. Row 2: Example clump distributions. Pixels of the same color are summarized in a single clump. Row 3: segmentations corresponding to clumps in row 2.

as $1 - \frac{FE_{batch} - FE_{mb}}{|FE_{batch}|}$. This process was repeated for different subsets of 30 images from the dataset. In the memory bounded approach, images were processed in epochs of five images at a time. Figure 1 summarizes the results. We find that performance tends to saturate beyond a certain number of clumps.

We also note a significant run time advantage of the memory bounded algorithm over the batch method. The average run time of the batch method was 3.09 hours versus 0.68 hours for the memory bounded approach.

Next we study the impact of image (document) merges on the relative performance of the memory bounded algorithm versus the baseline batch algorithm, while varying the maximum number of image (document) groups permitted. The results are shown in figure 1.

We find little qualitative difference between segmentations produced by the baseline and memory bounded algorithms. The possible exception is in the case when the memory bounded algorithm is run with a large number of image merges, in which case the algorithm seemed to discover fewer topics than the batch and memory bounded algorithm with only word clumping. Example image segmentations and clump distributions are shown in figure 2.

Finally, we demonstrate the memory bounded algorithm on the full dataset of 435 images, which is more than an order of magnitude larger than can be handled with the baseline algorithm. We process images in *epochs* of 10 images at

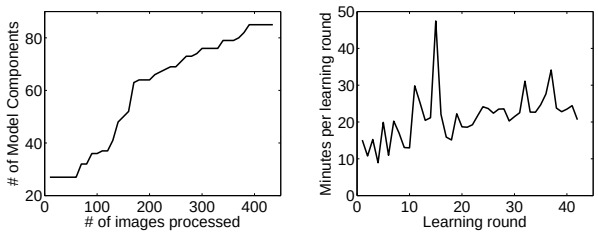

Figure 3. Joint segentation of 435 faces. The left plot shows the number of topics recovered as the system processes images. The right plot shows the run time for each learning round. This fluctuates with the number of new topics discovered during each round and tends to increase gradually with the total number of topics.

a time, for a total of 44 learning rounds. The upper limit on the number of clumps was set to 1000, which was likely many more than required since there were only 85 inferred topics. Because the number of documents was relatively small, we chose not to use document merges. The total run time of the algorithm was 15 hours. Figure 3 shows the number of topics as a function of the number of images processed, and the run time required during each image round. The run time is longer during learning rounds in which more new topics are discovered, because more split-merge operations are necessary. The memory required for the memory bounded algorithm was 22 MB to store the current image epoch and clumps, less than 1MB for the current model estimate, and 235 MB for assignment distributions, for a total of 257 MB. In contrast, the baseline batch implementation would have required 531 MB to store all 435 images, 8.8155 GB to store assignment distributions for each pixel assuming 85 topics, and less than 1 MB for the model, for a total of 9.3 GB. (All memory amounts assume double precision floating point.) The memory bounded implementation therefore achieved a memory savings factor of about 38 with very little loss in accuracy.

Figure 4 shows example joint segmentations produced by the memory bounded algorithm. These images were retrieved by first computing responsibilities for every image in the dataset, with respect to the final model estimate produced by the MB algorithm. Then, the images were sorted according to those that have the most pixels assigned to the largest topic. The largest topic indeed corresponds to a face, and is represented by the olive green segment in the figure. Other topics shared across images include hair and certain backgrounds.

4.2. Object Recognition and Retrieval

Our object recognition and retrieval experiment involves all 101 object categories in the Caltech 101 dataset. We randomly select 3000 training images and 1000 test images. We extract 128-dimensional SIFT (Lowe, 2004) local ap-

Figure 4. Examples of joint segmentation produced after processing all Caltech Face images. Pixels that are the same color have highest responsibility to the same topic. These images were retrieved by sorting images according to those that have the most pixels assigned to the largest topic, which is the olive green colored face segment in each image.

pearance descriptors from 500 randomly chosen locations in each image. The scale of each feature is also chosen randomly. In the language of topic models, each feature descriptor is a word, and the collection of feature descriptors in an image forms a document. This image representation is known as 'bag-of-features', because images are modeled as unordered collections of feature descriptors whose geometric positions are ignored. This dataset proved too large to compare directly to the batch algorithm

We train a single topic model on all training images, using epochs of 60 images at a time. Because hundreds of topics are discovered we use diagonal covariance Gaussians and adjust equation 26 accordingly. Given a test image $\tilde{\mathbf{x}}$, retrieval is performed by ranking each training image's similarity to the test image. To develop the similarity measure we begin with $\log \prod_i p(\tilde{\mathbf{x}}_{ij}|\mathbf{x})$, which is the log-probability that the detections in the test image were generated by training image j given the training set. Then we variationally lower bound this quantity to obtain a test free energy and drop all constant terms not involving the test image and index j. Finally we lower bound this quantity by assuming that detections in the test image are hard assigned to the topic with highest responsibility (this leads to an expression that is much faster to evaluate with neglible impact on retrieval performance.) The retrieval score is:

$$score(j) = \sum_i \max_k \Big\{ \sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\ \phi_{kl}(\tilde{\mathbf{x}}_{ij}) - \mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}] + \psi(\zeta_{kj}) - \psi(\sum_k \zeta_{kj}) \Big\} \quad (28)$$

where the expectations are with respect to $q(\boldsymbol{\eta})$ learned during training and ξ_{kl} and ζ_{kj} are from training as well. ζ_{kj}

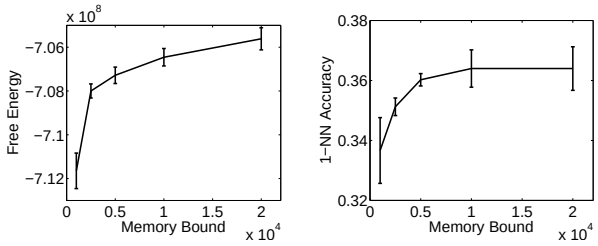

Figure 5. Object Recognition and Retrieval. Left: Training set free energy as a function of the memory bound. Right: 1-NN classification accuracy as a function of memory bound (measured as the equivalent number of data-points that could be stored in the same space).

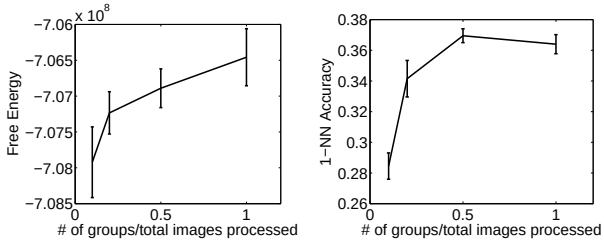

Figure 6. Object Recogniton and Retrieval. Left: Training set free energy versus the ratio of document groups to the total number of images processed. Right: 1-NN classification accuracy versus the ratio of document groups to total number of images processed.

are re-estimated for images that were merged into a document group during training. We compute nearest neighbor (1-NN) classification accuracy by classifying the test image to the class label of the highest scoring image in the training set.

Figure 5 shows the training set free energy and 1-NN classification accuracy as a function of the memory bound M (measured as the equivalent number of data points that could be stored in the same space.) Because we used diagonal covariance matrices, there were enough clumps even at low levels of memory to maintain comparable classification performance. We note that the training free energy increases with memory as expected, and that the 1-NN accuracy tends to saturate as memory increases.

Figure 6 shows the 1-NN accuracy and training free energy when the percentage of document groups relative to the number of total images processed is varied (the memory bound M is held fixed at 10000). We note that the classification performance suffers substantially when only small numbers of document groups are permitted. We use a heuristic for determining documents to merge (eq. 27). It is possible that a well motivated criterion (perhaps derived from the free energy) would give better performance.

5. Conclusion

Machine learning has largely focussed on algorithms that run for a relatively short period of time, fitting models of finite capacity on a data-set of fixed size. We believe that this scenario is unrealistic if we aim at building truly intelligent systems. We have identified nonparametric Bayesian models as promising candidates that expand their model complexity in response to new incoming data. The flip-side is that nonparametric Bayesian algorithms are "example-based" and as such require one to cache and process repeatedly every data-case ever seen. The objectives of infinite, adaptive model capacity on the one hand and efficiency, both in time and space on the other therefore seem to be fundamentally at odds with each other.

In this paper we have made a first step towards resolving this issue by introducing a class of models that can adapt their model complexity adaptively but are able to do so at a fraction of the memory requirements and processing times necessary for their batch counterparts. There is no magic of course: with a fixed memory budget there is a limit to how complex the model can be, but we have shown that one can learn much larger models reliably with much less memory than a naive implementation would allow. Moreover, our learning algorithms allow a flexible tradeoff between memory requirements and model complexity requirements that can be adapted online.

Intuitively, our method may be thought of as a two level clustering process. At the bottom level, data is clustered into clumps in order to limit time and space costs. At the top level, clumps are clustered to form topics in order to ensure good generalization performance.

Potential application areas of the techniques introduced here are manyfold. For instance, we can imagine learning topic models from very large text corpora or the world wide web to understand its structure and facilitate fast searching algorithms. Another exciting direction is to build a taxonomy of visual object categories from a continuous stream of video data captured by surveillance cameras.

5.1. Acknowledgements

We thank the anonymous reviewers for their helpful comments. This material is based on work supported by the National Science Foundation under grant numbers 0447903 and 0535278, the Office of Naval Research under grant numbers 00014-06-1-0734 and 00014-06-1-0795, and The National Institutes of Health Predoctoral Training in Integrative Neuroscience grant number T32 GM007737.

5.2. Appendix

The following expressions for the free energy are used in the main text. Note that they are only valid after the updates for $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ have been performed.

$$\begin{aligned}\mathcal{F} = & KB(\boldsymbol{\beta}) - \textstyle\sum_k B_k(\mathbf{F}_k + \boldsymbol{\beta}) \\ & + \textstyle\sum_{ks} D_s \log\Big(\Gamma(\alpha_k)/\Gamma(\alpha_k + \frac{N_{ks}}{D_s})\Big) - \sum_s D_s \log\Big(\Gamma(\alpha)/\Gamma(\alpha + \frac{N_s}{D_s})\Big) \\ & + \textstyle\sum_{ck} N_c q_{ck} \log q_{ck} \\ & - \textstyle\sum_k \Big((a-1)\sum_k \log(\alpha_k) - b\sum_k \alpha_k\Big) \\ & - (K_{\max} - K)\Big((a-1)\log\tfrac{a-1}{b} - (a-1)\Big) - K_{\max}(b\log(a) - \log\Gamma(a))\end{aligned} \tag{29}$$

$$\begin{aligned}\mathcal{F} = & KB(\boldsymbol{\beta}) - \textstyle\sum_k B_k\Big(\Big(\frac{T_{pts}}{\sum_c N_c}\Big)\mathbf{F}_k + \boldsymbol{\beta}\Big) \\ & + \Big(\tfrac{T_{docs}}{\sum_s D_s}\Big)\textstyle\sum_{ks} D_s \log\Big(\Gamma(\alpha_k)/\Gamma(\alpha_k + \frac{N_{ks}}{D_s})\Big) \\ & - \Big(\tfrac{T_{docs}}{\sum_s D_s}\Big)\textstyle\sum_s D_s \log\Big(\Gamma(\alpha)/\Gamma(\alpha + \frac{N_s}{D_s})\Big) \\ & + \Big(\tfrac{T_{pts}}{\sum_c N_c}\Big)\textstyle\sum_{ck} N_c q_{ck} \log q_{ck} \\ & - \textstyle\sum_k \Big((a-1)\sum_k \log(\alpha_k) - b\sum_k \alpha_k\Big) \\ & - (K_{\max} - K)\Big((a-1)\log\tfrac{a-1}{b} - (a-1)\Big) - K_{\max}(b\log(a) - \log\Gamma(a))\end{aligned} \tag{30}$$

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*.

Ferguson, T. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, *1*, 209–230.

Kurihara, K., Welling, M., & Vlassis, N. (2006). Accelerated variational dirichlet process mixtures. *NIPS*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Minka, T. (2000). *Estimating a dirichlet distribution* (Technical Report).

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet processes. *To appear in Journal of the American Statistical Association*.

Teh, Y. W., Kurihara, K., & Welling, M. (2008). Collapsed variational inference for HDP. *Advances in Neural Information Processing Systems*.

Ueda, N., Nakano, R., Gharamani, Z., & Hinton, G. (1999). Smem algorithm for mixture models.

Verbeek, J., Nunnink, J., & Vlassis, N. (2003). *Accelerated variants of the em algorithm for gaussian mixtures* (Technical Report). University of Amsterdam.