Nonparametric Bayesian Methods for Supervised and Unsupervised Learning

by

Vikash Kumar Mansinghka

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Department of Electrical Engineering and Computer Science
April 8, 2009

Certified by .....
Joshua B. Tenenbaum
Paul E. Newton Career Development Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Nonparametric Bayesian Methods for Supervised and Unsupervised Learning

by

Vikash Kumar Mansinghka

Submitted to the Department of Electrical Engineering and Computer Science
on April 8, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

I introduce two nonparametric Bayesian methods for solving problems of supervised and unsupervised learning. The first method simultaneously learns causal networks and causal theories from data. For example, given synthetic co-occurrence data from a simple causal model for the medical domain, it can learn relationships like "having a flu causes coughing", while also learning that observable quantities can be usefully grouped into categories like diseases and symptoms, and that diseases tend to cause symptoms, not the other way around. The second method is an online algorithm for learning a prototype-based model for categorial concepts, and can be used to solve problems of multiclass classification with missing features. I apply it to problems of categorizing newsgroup posts and recognizing handwritten digits.

These approaches were inspired by a striking capacity of human learning, which should also be a desideratum for any intelligent system: the ability to learn certain kinds of "simple" or "natural" structures very quickly, while still being able to learn arbitrary — and arbitrarily complex — structures given enough data. In each case, I show how nonparametric Bayesian modeling and inference based on stochastic simulation give us some of the tools we need to achieve this goal.

Thesis Supervisor: Joshua B. Tenenbaum
Title: Paul E. Newton Career Development Professor

Acknowledgments

These acknowledgements — and, it turns out, this thesis document — are being intentionally cut short. See my dissertation for a more comprehensive treatment.

Chapter 2 is based on a paper published in UAI 2005 co-authored with Charles Kemp, Tom Griffiths and Josh Tenenbaum. Chapter 3 is based on a paper published in AISTATS 2007 co-authored with Daniel Roy, Ryan Rifkin and Josh Tenenbaum. This work was partially supported by gifts from NTT Communication Science Laboratories and Eli Lilly and company. Tom Minka generously provided logistic regression code, and Keith Bonawitz gave helpful feedback.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

General purpose inductive learning requires a delicate balance of conceptual flexibility and inductive constraint. Conceptual flexibility is necessary so that a learner can entertain all of the hypotheses that could possibly account for the data. On the other hand, inductive constraints that exclude hypotheses from consideration or downweight them in probabilistic learning are necessary so that a learner can generalize confidently from small numbers of examples.

One powerful approach to achieving this balance of constraint and flexibility comes from nonparametric Bayesian statistics. In nonparametric Bayes, one writes down a probability distribution over a space of hypotheses that is infinitely expressive in principle, but whose effective support grows in sync with the data it is supposed to explain. For example, there is only 1 way to partition a set of size 1 into groups, but 2 ways to partition a set of size 2, and 4 ways to partition a set of size 3. To represent the full space of logically possible partitionings in terms of labels, we need as many labels as there are datapoints, but no more; extra labels constitute wasted representational capacity. Nonparametric Bayesian modeling of partitions involves a potentially infinite pool of labels, so that an arbitrarily large dataset can be arbitrarily partitioned, but introduces them sparingly, so that for any finite amount of data only finitely many labels will be used.

In this thesis, I introduce one unsupervised learning method and one supervised learning method that use nonparametric Bayesian methods to manage this tradeoff. The first method simultaneously learns causal networks and causal theories from data. For example, given synthetic co-occurrence data from a simple causal model for the medical domain, it can learn relationships like "having a flu causes coughing", while also learning that observ-

able quantities can be usefully grouped into categories like diseases and symptoms, and that diseases tend to cause symptoms, not the other way around. The basic approach is to formalize abstract knowledge about typical causal networks - for example, the idea that events can be grouped into categories, and that category membership predicts causal interactions - as a nonparametric prior on Bayesian networks. The nonparametric approach allows the method to learn any causal network given enough data, but favors networks that can be explained in terms of a small number of causally predictive categories. It can sometimes make striking inductive leaps, converging on the rough causal structure at play from a number of observations equal to the number of variables.

The second method is an online algorithm for learning a prototype-based model for categorial concepts, and can be used to solve problems of multiclass classification with missing features. Each category is modeled via a Chinese restaurant process mixture model, a nonparametric Bayesian model that can asymptotically duplicate any probability distribution (on mixed discrete and continuous data) but which favors distributions that can be explained in terms of a small number of "prototypical" examples. It provides a probabilistic alternative to methods like support vector machines, producing a sparse representation of each class-conditional density and the classification decision boundary. However, since it is based on a generative model, it can manage missing features, and can be naturally extended to missing labels and the semi-supervised case. It leverages sequential Monte Carlo methods for online inference, showing that propagation of a very small number of guesses may be sufficient for accurate probabilistic inference. I apply it to problems of categorizing newsgroup posts and recognizing handwritten digits.

Both of these approaches were inspired by a striking capacity of human learning, which should also be a desideratum for any intelligent system: the ability to learn certain kinds of "simple" or "natural" structures very quickly, while still being able to learn arbitrary — and arbitrarily complex — structures given enough data. In each case, I show how nonparametric Bayesian modeling and inference based on stochastic simulation give us some of the tools we need to achieve this goal.

Chapter 2

Structured Priors for Structure Learning

Unsupervised discovery of structured, predictive models from sparse data is a central problem in artificial intelligence. Bayesian networks provide a useful language for describing a large class of predictive models, and much work in unsupervised learning has focused on discovering their structure from data. Most approaches to Bayes net structure learning assume generic priors on graph structure, sometimes encoding a sparseness bias but otherwise expecting no regularities in the learned graphs. However, in many cases, we expect more systematicity: variables in real-world systems often play characteristic roles, and can be usefully grouped into classes that predict the kinds of probabilistic dependencies they participate in. This systematicity provides important constraints on many aspects of learning and inference.

Consider the domain of medical learning and reasoning. Knowledge engineers in this area have historically imposed strong structural constraints; the QMR-DT network [24], for example, segregates nodes into *diseases* and *symptoms*, and only permits edges from the former to the latter. Recent attempts at medical knowledge engineering have continued in this tradition; for example, [11] explicitly advocate organizing Bayes nets for diagnosis into three layers, with influence flowing only from *context* through *fault* to *influence* nodes. Similar divisions into classes pervade the literature on probabilistic graphical models for biological interactions; [17], for example, learn gene networks by discovering a salient example, structuring the problem of learning gene networks around the discovery of a small set of *regulators* that are responsible for controlling the activation of all other genes and may also influence each other.

Knowledge about relationships between classes provide inductive constraints that allow Bayesian networks to be learned from much less data than would otherwise be possible. For example, consider a structure learner faced with a database of medical facts, including a list of patients for which a series of otherwise undifferentiated conditions have been provided. If the learner knew that, say, the first ten conditions were *diseases* and the rest *symptoms*, with influence only possible from diseases to symptoms, the learner would need to consider a dramatically reduced hypothesis space of structures. Various forms of less specific knowledge could also be quite useful, such as knowing that some variables played a similar causal role, even without the precise knowledge of what that role entailed. Causal roles also support transfer: a medical expert system knowledgeable about lung conditions faced with data about liver conditions defined over entirely new variables would not be able to transfer any specific probabilistic dependencies but could potentially transfer abstract structural patterns (e.g., that diseases cause symptoms). Finally, abstract structural knowledge is often interesting in its own right, aside from any inductive bias it contributes. For example, a biologist might be interested to learn that certain genes are regulators, over and above learning about specific predictive or causal links between particular genes.

We present a hierarchical Bayesian approach to structure learning that captures this sort of abstract structural knowledge using nonparametric block-structured priors over Bayes net graph structures. Given observations over a set of variables, we simultaneously infer both a specific Bayes net over those variables and the abstract structural features of these dependencies. We assume that variables come in one or more classes and that the prior probability of an edge existing between two variables is a function only of their classes. In general, we do not assume knowledge of the number of classes, the class assignments of variables, or the probabilities of edges existing between any given classes; these aspects of the prior must be inferred or estimated at the same time that we learn the structure and parameters of the Bayes net. We derive an approach for simultaneous inference of all these features from data: all real-valued parameters are integrated out analytically, and MCMC is used to infer the number of classes, the class assignments of variables, and the Bayes net structure over those variables. We show that the bias towards systematicity of connections provided by our model can yield more accurate recovery of network structures than a uniform prior approach, when the underlying structure exhibits some systematicity. We also demonstrate that on randomly generated sparse DAGs, the nonparametric form of our inductive bias can protect us from incurring a substantial penalty due to prior mismatch.

A number of previous approaches to learning and inference have exploited structural constraints at more abstract levels than a specific Bayesian network. These approaches typically show a tradeoff between the representational expressiveness and flexibility of the abstract knowledge, and the ease with which that knowledge can be learned. At one end of this tradeoff, approaches like [23] assume that the Bayes net to be learned must respect very strict constraints — in this case, that variables can be divided into modules, where all variables in a module share the same parents and the same conditional probability table (CPT). This method yields a powerful inductive bias for learning both network structure and module assignments from data, especially appropriate for the study of gene regulatory networks, but it is not as appropriate for modeling many other domains with less regular structure. The QMR-DT [24] or Hepar II [11] networks, for example, contain highly regular but largely nonmodular structure, and we would like to be able to discover both this sort of network and a good characterization of this sort of regularity. At the other end of the flexibility-learnability tradeoff, frameworks such as probabilistic relational models (PRMs) [6] provide a far more expressive language for abstract relational knowledge that can constrain the space of Bayesian networks appropriate for a given domain, but it is a significant challenge to learn the PRM itself — including the class structure — from raw, undifferentiated event data. Our work aims at a valuable intermediate point on the flexibility-learnability spectrum. We consider hypotheses about abstract structure that are less expressive than those of PRMs, but which are simultaneously learnable from raw data along with specific networks of probabilistic dependencies. Our nonparametric models also yield a stronger inductive bias for structure learning than a uniform prior but a weaker and more flexible one than module networks.

2.1 Block-Structured Priors

The intuition that nodes in a Bayes net have predictive classes can be formalized in several ways. In this paper, we focus on two structure priors obtained from similar formalizations: the *ordered blockmodel* and the *blockmodel*. The joint distributions we work in (including the uniform baseline) can be described by a meta-model consisting of three major pieces (see Figure 2-1): a prior over the structure of Bayes nets, a prior on the parameterizations of Bayes nets given their structure, and the data likelihood induced by that parameterization. For simplicity, we work with discrete-state Bayesian networks with known domains, and use

the standard conjugate Dirichlet-Multinomial model for their conditional probability tables [2]. Note that the variable G in Figure 2-1 refers to a graph structure, with B its elaboration into a full Bayesian network with conditional probability tables. Inference in the model of Figure 2-1 and its specializations characterizes the Bayes net structure learning problem.

We are interested in discovering node classes that explain patterns of incoming and outgoing edges, as a step towards representation and learning of causal roles. Our starting point is the *infinite blockmodel* of [10], a nonparametric generative model for directed graphs, which we modify in two ways to only produce acyclic graphs. We first describe the generative process for the ordered blockmodel:

1. Generate a class-assignment vector $\vec{z}$ containing a partition of the N nodes of the graph via the Chinese restaurant process or CRP [18] with hyperparameter α. The CRP represents a partition in terms of a restaurant with a countably infinite number of tables, where each table corresponds to a group in the partition (see Figure 2-1b) and the seating assignment of person i is the class of the ith object. People are seated at each existing table k (from 1 up to K^+) with probability proportional to m_k, the number of previous occupants of the table. New tables are created with probability proportional to the hyperparameter α. As the CRP is exchangeable, the distribution on partitions (represented by $\vec{z}$) that it induces is invariant to the order of entry of people into the restaurant. Specifically, we have:

$$P(\vec{z}|\alpha) = \alpha^{K^+} \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)} \prod_{k=1}^{K^+} (m_k - 1)! \tag{2.1}$$

2. Generate an ordering $\vec{o}$ of the K^+ classes in the partition $\vec{z}$ uniformly at random:

$$P(\vec{o}|\vec{z}) = \frac{1}{K^+!} \tag{2.2}$$

   Then o_a contains the order of class a.

3. Generate a square graph template matrix η, of size equal to the number of classes in the generated partition, where η_{o_a,o_b} represents the probability of an edge between a node of class a and a node of class b. To ensure acyclicity, fix all entries to be 0 except those strictly above the diagonal; these possibly nonzero entries η_{o_a,o_b} with $o_b > o_a$ are

drawn from a $Beta(\vec{\beta})$ distribution (with $B(\cdot,\cdot)$ its normalizing constant):

$$P(\eta_{o_a,o_b}|\vec{z}) = \frac{1}{B(\beta_1,\beta_2)}\eta_{o_a,o_b}^{\beta_1-1}(1-\eta_{o_a,o_b})^{\beta_2-1} \tag{2.3}$$

4. Generate a graph G by drawing each edge $G_{i,j}$ from a Bernoulli distribution with parameter $\eta_{o_{z_i},o_{z_j}}$.

$$P(G|\vec{z},\vec{o},\eta) = \prod_{i=1}^{N}\prod_{j=1}^{N}\eta_{o_{z_i},o_{z_j}}^{G_{i,j}}(1-\eta_{o_{z_i},o_{z_j}})^{1-G_{i,j}}$$

Note that all graphs are possible under this process, though the ones lacking salient block structure typically require more classes. Also note that the hyperparameters in this process are intuitive and interpretable. α controls the prior on partitions of nodes into classes; smaller values favor fewer groups *a priori*. $\vec{\beta}$ controls the η matrix, with values such as (0.5, 0.5) favoring clean blocks (either all edges absent or all present) and increases in β_1 or β_2 yielding biases towards sparseness or denseness respectively. The ordering $\vec{o}$ encodes the "causal depth" of nodes based on their classes. Because we believe specific, directed dependencies are important outputs of structure discovery for causal systems, we do not constrain our priors to assign equal mass to members of a particular Markov equivalence class.

The blockmodel prior does not include the ordering $\vec{o}$. Graph generation is essentially as above, with $\eta_{a,b}$ being the probability of an edge between a node of class a and a node of class b, and all entries permitted to be nonzero. Cyclic graphs are rejected afterwards by renormalization. More formally, for the blockmodel version, let $\Delta(G)$ be 1 if G is acyclic and 0 otherwise. Then we have

$$P(G|\vec{z},\eta) \propto \Delta(G)\prod_{i=1}^{N}\prod_{j=1}^{N}\eta_{z_i,z_j}^{G_{i,j}}(1-\eta_{z_i,z_j})^{1-G_{i,j}} \tag{2.4}$$

Both priors may be appropriate depending on the situation. If we are interested in classes that correspond to different stages (possibly unrolled in time) of a causal process, the ordered blockmodel may be more appropriate. If we expect nodes of a given class to include some connections with others of the same class, as in the study of gene regulatory networks where some regulators influence others, then the blockmodel should perform better. Additionally,

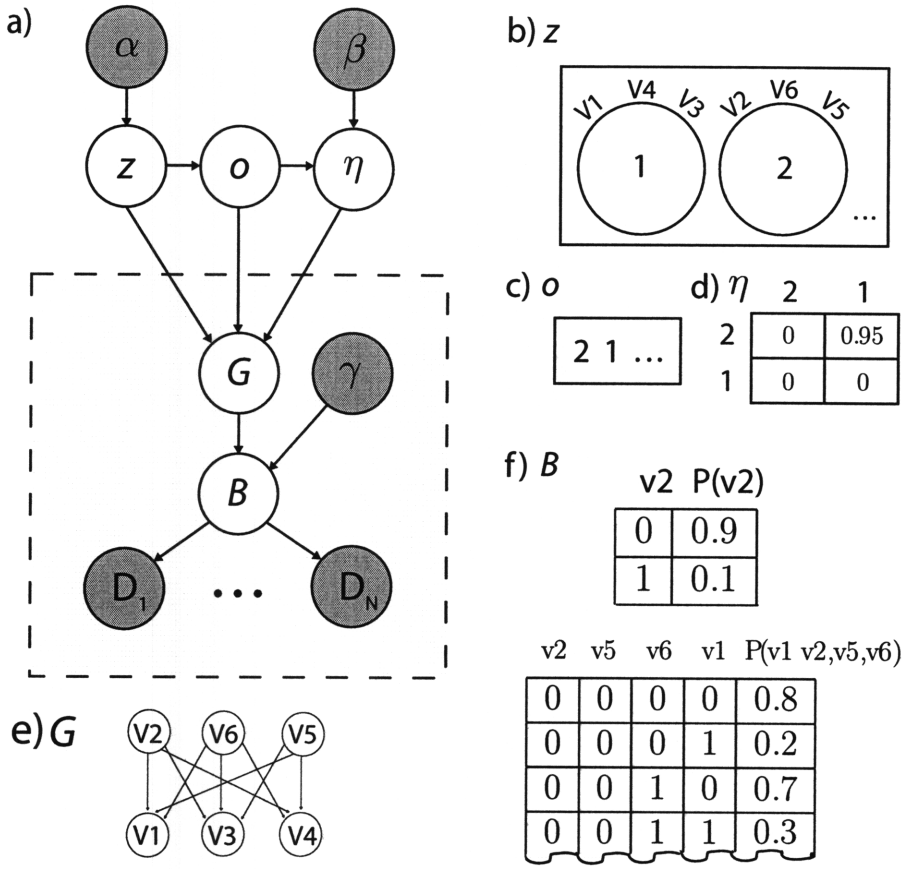


Figure 2-1: (a) Graphical meta-model for the *ordered blockmodel.* The dashed line indicates the components of the meta-model corresponding to traditional Bayes net structure learning with a uniform prior. (b)-(d) Latent variables representing abstract structural knowledge.

both models can generate any directed acyclic graph. The blockmodel can do this either by assigning all nodes to the same class, learning the sparsity of the graph, or by assigning all nodes to different classes (which may be useful for dense, nearly cyclic graphs). The ordered blockmodel has the option of assigning nodes to different classes corresponding to layers in the topological ordering of an arbitrary DAG.

Our data likelihood is the standard marginal likelihood for complete discrete observations, assuming the CPTs (represented by the parameters B) have been integrated out; γ plays the role of the pseudo-counts setting the degree of expected determinism of the CPTs [13]. Overall, then, our model has three free parameters (in the symmetric β case) — α, β and γ — which each give us the opportunity to encode weak prior knowledge at different levels of abstraction. Of course, in principle, one could assign hyperpriors to these quantitites and

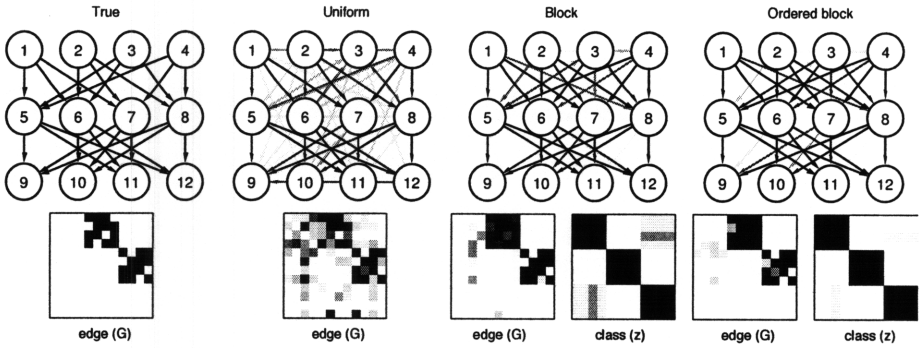


Figure 2-2: Results for three models on a 12-node layered topology with 75 data samples provided for training. The upper row displays the marginal probabilities of each edge in the graph under each model, obtained via selective model averaging over 100 samples. The bottom row displays the same edge marginals as an adjacency matrix, along with the marginal probabilities that $z_i = z_j$, indicating inferences about the class structure.

integrate them out.

This nonparametric, hierarchical approach is attractive for several reasons. The CRP gracefully handles a spectrum of class granularities, generally preferring to produce clumps but permitting each object to be in its own class if necessary. $\vec{z}$, $\vec{\alpha}$ and η provide convenient locations for the insertion of strong or weak prior knowledge, if we have it, ranging from complete template knowledge to only the expectation that some reasonable template can be found. They also represent additional outputs of learning, which are of interest in both cognitive applications (e.g. the concept *disease* is at least as much about the patterns in the causal relationships present across many particular diseases as it is about those relationships in particular) and in scientific data analysis (e.g. the notion that regulators exist is useful even if one isn't entirely sure about which genes are regulated by which other ones).

Throughout this paper, we will compare our approach with Bayes net structure learning given a conventional, uniform prior on DAGs. Formally, this corresponds to the subcomponent of the meta-model in Figure 2-1 indicated by the dashed box, and a prior on graph structures G given by Equation 2.4 with a single class and η (now a scalar) fixed to 0.5.

2.2 Inference

Having framed our approach to structure learning as inference in the graphical model of Figure 2-1, we must provide an effective procedure for performing it. Note that when deter-

mining the contribution of a given structure to the predictive distribution, we will explicitly represent the posterior on parameters for that structure, again using conjugacy. To further reduce the size of our hypothesis space for the experiments in this paper, we also integrate out the η matrix. Let $n^+_{a,b}$ be the number of edges between nodes of class a and nodes of class b, $n^-_{a,b}$ the corresponding number of missing edges, with K^+ the number of classes as before. Then we have:

$$p(G|\vec{z}) = \prod_{a=1}^{K^+} \prod_{b=1}^{K^+} \frac{B(\beta_1 + n^+_{a,b}, \beta_2 + n^-_{a,b})}{B(\beta_1, \beta_2)} \tag{2.5}$$

Note that in the case of the blockmodel, where η is not itself constrained to generate acyclic graphs, this equality is instead only a proportionality, due to renormalization of the prior over acyclic graphs.

Inference, in general, will entail finding posterior beliefs about $\vec{z}$, $\vec{o}$ and G. We organize our inference around a Markov chain Monte Carlo procedure, consisting of both Gibbs and Metropolis-Hastings (MH) kernels, to retain theoretical correctness guarantees in the limit of long simulations. Of course, it is straightforward to anneal the Markov chain and to periodically restart if we are only interested in the MAP value or a set of high-scoring states for selective model averaging [12].

Our overall MCMC process decouples into moves on graphs and, if relevant, moves on the latent states of the prior (a partition of the nodes, and in the ordered case, an ordering of the groups in the partition). Our graph moves are simple: we Gibbs sample each potential edge, conditioned on the rest of the hidden state. This involves scoring only two states — the current graph and the graph with a given edge toggled — under the joint, renormalizing, and sampling, and is therefore quite computationally cheap. As all our priors place 0 probability mass on cyclic graphs, cyclic graphs are never accepted. Furthermore, much of the likelihood can be cached, since only nodes whose parent set changed will contribute a different term to the likelihood. We found these Gibbs moves to be more effective than the classic neighborhood based graph moves discussed in [8], though in general we expect combinations of our moves, MH-moves that randomly propose reversing edges, and neighborhood-based moves to be most effective.

Conditionally sampling the latent states that describe the block structure is also straightforward. In the unordered case, we simply use the standard Gibbs sampler for the Chinese restaurant process, fixing the class assignments of all nodes but one — call it i — and run-

ning the free node through the CRP [19]. Specifically, as the process is exchangeable, we can Gibbs sample a given entry by taking i to be the last person to enter the restaurant, and sampling its class assignment z_i from the normalized product of the CRP conditionals (below) and Equation 5:

$$P(z_i = k|\alpha, \vec{z}^{-i}) = \begin{cases} \frac{m_k}{i-1+\alpha} & \text{if } m_k > 0 \\ \frac{\alpha}{i-1+\alpha} & k \text{ is a new class} \end{cases} \tag{2.6}$$

In the ordered case, we must be more careful. There, we fix the relative ordering of the classes of all objects except i, whose z_i we will be resampling. When considering the creation of a new class, we consider insertions of it into $\vec{o}$ at all possible free spots in the relative order. This leaves us with an exhaustive, mutually exclusive set of possibilities for $\vec{z}$ and $\vec{o}$ which we can score under the joint distribution and renormalize to get a Gibbs sampler.

For very large problems, obtaining even approximate posteriors is beyond our current computational capabilities, but search techniques like those in [26] would no doubt be very useful for approximate MAP estimation. Furthermore, additional temperature and parallelization schemes, as well as more sophisticated moves, including splitting and merging classes, could all be used to improve mixing.

2.3 Evaluation

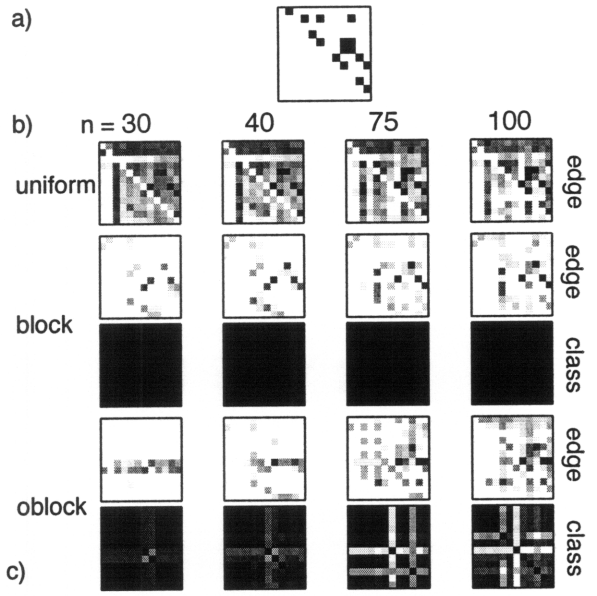
We evaluate our approach in three ways, comparing throughout to the *uniform* model, a baseline that uses a uniform prior over graph structures. First we consider simple synthetic examples with and without strong block structure. Second, we explore networks with topologies and parameterizations inspired by networks of real-world interest, including the QMR-DT network and a gene regulatory network. Finally, we report model performance on data sampled from HEPAR II [16], an engineered network that captures knowledge about liver disorders. Because our interest is in using structured priors to learn predictive structure from very small sample sizes, we usually cannot hope to identify any structural features definitively. We thus evaluate learning performance in a Bayesian setting, looking at the marginal posterior probabilities of edges and class assignments. We have focused our initial studies on fairly small networks (between 10 and 40 variables) where approximate Bayesian inferences about network structure can be done quickly, but scaling up to larger networks is

an important goal for future work.

In all cases, we used our MCMC scheme to explore the space of graphs, classes, and orders, and report results based on an approximate posterior constructed from the relative scores of the 100 best models found. The pool of models we chose from was typically constructed by searching 10 times for 2000 iterations each; we found this was generally sufficient to find a state that at least matched the score of the ground truth structure.

We sampled several training and test sets from each structure; here, we report representative results. Hyperparameters were set to the same values throughout: $\alpha = 0.5, \beta = 1.0$ and $\gamma = 0.5$. For real-world applications we might adjust these parameters to capture prior knowledge that the true graph is likely to be sparse, and that the number of underlying classes is either large or small, but fixing the hyperparameters allows a fairer comparison with the uniform model.

Compared to the uniform model, we expect that the block models will perform well when the true graph is block-structured. Figure 2-2 shows results given 75 samples from the three-layered structure on the left. CPTs for the network were sampled from a symmetric Dirichlet distribution with hyperparameter 0.5. The true graph is strongly block structured, and even though the number of samples is small, both block models discover the three classes, and make accurate predictions about the edges that appear in the true graph. The inferences made by the uniform model are less accurate: in particular, it is relatively confident about the existence of some edges that violate the feed-forward structure of the true network. No prior will be appropriate for all datasets, and we expect that the uniform model will beat the block models in some cases when the true graph is not block-structured. Ideally, however, a learning algorithm should be flexible enough to cope with many kinds of data, and we might hope that the performance of the block models does not degrade too badly when the true graph does not match their priors. To explore this setting we generated data from graphs with sparse but non-block-structured connectivity, like the one shown in Figure 2-3. This graph was sampled by including each edge with probability 0.3, where we rejected all cyclic graphs and all graphs with in- or out-degrees greater than 4; CPTs were sampled from a Dirichlet distribution with hyperparameter 0.5. As expected, the block model discovers no block structure in the data, remaining confident throughout that all nodes belong to a single class. To our surprise, the block model performed better than the uniform model, making fewer mistaken predictions about the edges that appear in the true structure, and matching the true distribution more closely than the uniform model. Even though the block



	30	40	50	75	100	1000
U	0.78	0.60	0.49	0.44	0.38	0.03
B	**0.53**	**0.37**	**0.31**	**0.33**	0.33	**0.03**
OB	0.71	0.64	0.60	0.34	**0.32**	0.03

Figure 2-3: Learning results for data sampled from a graph without block structure. (a) adjacency matrix representing the true graph (the (i, j) entry is black if there is an edge from i to j) (b) edge and class assignment marginal probabilities for a subset of the sample sizes, as in Figure 2-2. (c) estimated Kullback-Leibler divergences between the posterior predictive distributions of each model and ground truth.

model found only one class, it learned the density of connections is low, winning over the uniform model (which reserves much of its probability mass for highly connected graphs) on very sparse datasets. Since the ordered block model allows no connections within classes, it cannot offer the same advantage, and Figure 2-3 shows that its performance is comparable to that of the uniform model.

Our approach to modeling abstract structure was motivated in part by common-sense medical knowledge, and networks like QMR-DT. The QMR network is proprietary, but we created a QMR-like network with the connectivity shown in Figure 2-4. The network has two classes, corresponding to diseases and the edges all appear between diseases and symptoms. CPTs for the network were generated using a noisy-or parameterization. The results in Figure 2-4 show that both block models recover the classes given a small set of examples, and it is particularly striking that the unordered model begins to make accurate predictions about class membership when only 20 examples have been provided. By the time 80 examples have been provided, the blockmodel makes accurate predictions about the edges in the true graph, and achieves a predictive distribution superior to that of the uniform model.

As [23] have shown, genetic expression data may be explained in part by the existence of underlying classes, or modules. Unlike the QMR example, links within classes should be expected: in particular, we expect that genes belonging to the class *regulator* will sometimes regulate each other. To test our models on a simple setting with this structure, we generated data from the network shown in Figure 2-5, with CPTs sampled from a Dirichlet distribution with hyperparameter 0.5. After 1000 samples, the block model is confident of the correct class structure (with 4 regulators), and is considering one of the two correct within-regulator edges (between variables 1 and 2), although it is uncertain about the orientation of this edge. The uniform model has similar beliefs about this edge after 1000 samples, but it has equally confident incorrect beliefs about other influences on the regulators. The ordered blockmodel begins to infer class differences earlier, and is considering grouping various subsets of the regulators.

As already suggested, knowledge about medical conditions can sometimes be organized as knowledge about interactions between three classes of variables: *risk factors*, *diseases* and *symptoms*. The HEPAR II network captures knowledge about liver diseases in a structure close to this three-part template. The structure of HEPAR II was elicited from medical experts, and the CPTs were learned from a database of medical cases. We generated training data from a subset of the full network that includes the 21 variables that appear in Figure

1 of [16] and all edges between these nodes from the true model. The network structure is shown in Figure 2-6. When provided with 1000 training examples, the unordered block model finds two classes, one which includes 4 of the 5 diseases, and a second which includes all the remaining variables. Note that the missing disease has a pattern of connections that is rather unusual: unlike the other 4 diseases, it has a single outgoing edge, and that edge is sent to another disease rather than a symptom. The model fails to distinguish between the risk factors and the symptoms, perhaps because the connectivity between the risk factors and the diseases is very sparse. Although neither block model recovers the three part structure described by the creators of the network, both discover sufficient structure to allow them to match the generating distribution better than the uniform model.

2.4 Discussion and Conclusions

In this paper we have explored two formalisms for representing abstract structural constraints in Bayes nets and shown how this knowledge can support structure learning from sparse data. Both formalisms are based on nonparametric, hierarchical Bayesian models that discover and characterize graph regularities in terms of node classes. We have seen how this approach succeeds when block structure is present without incurring a significant cost when it is not. We have also seen how this approach can, at the same time as it is learning Bayes net structure, recover abstract class-based patterns characterizing an aspect of the causal roles that variables play.

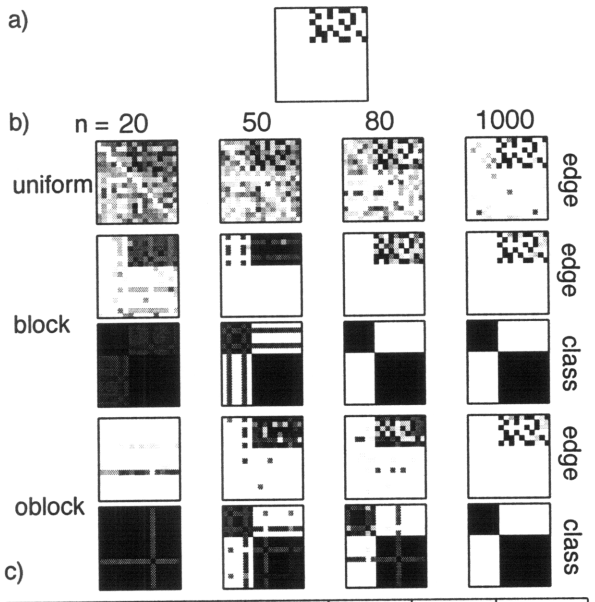
Our approach admits several straightforward extensions. First, the blockmodel prior, without modifications for acyclicity, should be directly applicable to discovering latent types in undirected graphical models and Markov models for time series, such as dynamic Bayesian networks or continuous-time Bayesian networks. Second, we expect our approach to provide additional benefits when observational data is incomplete, so that prior knowledge about likely roles becomes more significant. Such data could be incorporated via approximations to the marginal likelihood such as those from [1].

Weaknesses that should be addressed in future work include a more detailed empirical evaluation, with particular attention to sample complexity and computational complexity. For example, the Gibbs kernels for MCMC inference over structures should be compared with annealed and blocked variants, to assess susceptibility to local minima. Furthermore, the robustness of the "blessing of abstraction" phenomenon - where it sometimes seems easier

to learn abstract knowledge like causal theories than concrete knowledge like individual networks - should also be evaluated. Finally, comparisons to a one-class blockmodel (where we learn the degree of sparseness of a given network) may serve as a better baseline.

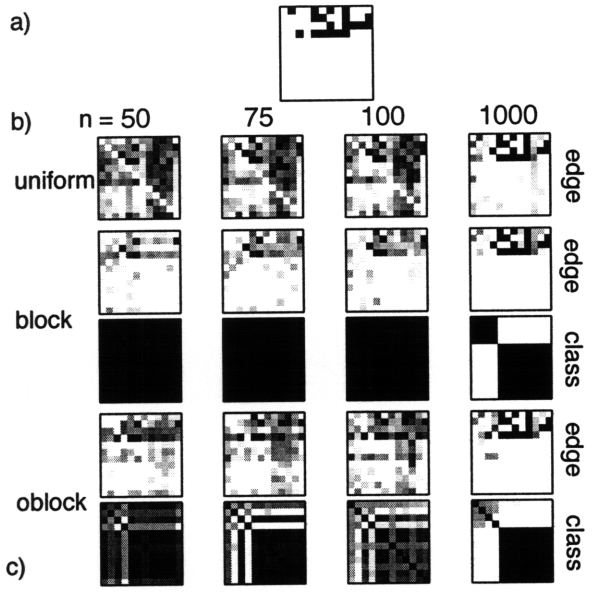
An exploration of richer pattern languages should also be useful. For example, the notion that nodes of a certain class *can* connect to nodes of another type (but may do so only rarely, or with parameters unrelated to the types), might be more appropriate for some domains. The literature on social networks, e.g. [27], provides many examples of similar, interesting relational patterns ripe for probabilistic formalization. Closer dependence of parameterization on class would also be interesting to explore: for example, some classes might project only excitatory or inhibitory edges. Finally, we note that transfer to new systems of variables, an important function of abstract knowledge, could be implemented by adding another layer to our hierarchical model. In particular, we could flexibly share causal roles across entirely different networks by replacing our CRP with the Chinese restaurant franchise of [25].

A central feature of this work is that it attempts to negotiate a principled tradeoff between the expressiveness of the space of possible abstract patterns (with the attendant advantages of particular patterns as inductive constraints) and the possibility of learning the abstract patterns themselves. Our nonparametric approach was inspired by a striking capacity of human learning, which should also be a desideratum for any intelligent agent: the ability to learn certain kinds of "simple" or "natural" structures very quickly, while still being able to learn arbitrary — and arbitrarily complex — structures given enough data. We expect that exploring these tradeoffs in more detail will be a promising area for future research.



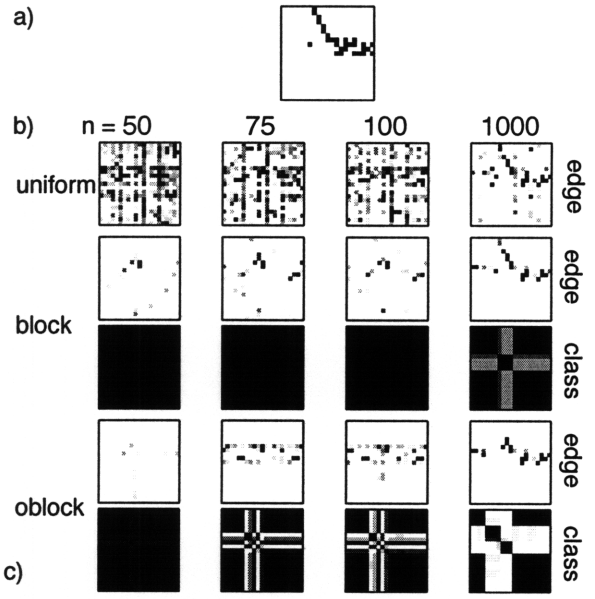
	20	30	50	80	100	1000
U	1.56	1.34	**0.82**	0.50	0.48	0.05
B	**1.19**	**1.12**	1.08	**0.28**	**0.26**	0.04
OB	2.32	2.11	1.03	0.48	0.82	**0.04**

Figure 2-4: Learning results for data sampled from a noisy-OR QMR-like network.



	30	40	50	75	100	1000
U	1.16	0.79	0.72	0.68	0.49	0.04
B	**0.73**	**0.64**	**0.49**	**0.42**	**0.35**	**0.03**
OB	1.60	0.79	0.62	0.68	0.42	0.03

Figure 2-5: Learning results for data sampled from a model inspired by a genetic regulatory network.



	20	30	50	75	100	1000
U	2.88	2.21	1.51	1.01	0.70	0.07
B	**2.01**	**1.62**	**1.06**	**0.58**	0.53	**0.06**
OB	2.06	1.66	1.36	0.63	**0.44**	0.06

Figure 2-6: Learning results for data sampled from a subset of the HEPAR II network.

Chapter 3

AClass: An online algorithm for generative classification

3.1 Introduction

Classification is a foundational problem in machine learning, and classic approaches are full of apparent tradeoffs. For example, the ability to handle missing or unlabeled data via simple inductive biases is considered the strong point of generative approaches while discriminative methods often exploit flexible, nonlinear decision boundaries that generalize well while remaining computationally tractable. In this paper, we present the AClass algorithm, a simple but novel approach that combines several strengths of previous classifiers that have not typically been available together in a single algorithm. We also perform preliminary experiments in synthetic and benchmark settings, showing competitive performance.

AClass is an online, parallelizable algorithm for multiclass classification based on approximate, sequential Monte Carlo based inference in a probabilistic generative model. AClass operates by learning a model of the class-conditional densities of the data by assuming they are Chinese restaurant process (CRP) mixtures. Chinese restaurant process mixture models are a standard tool from the nonparametric Bayesian statistics literature, and enable the number of mixture components to be discovered dynamically during posterior inference. Intuitively, each component can be thought of as a mode or prototype associated with its class. The full generative model embeds these class-conditional mixtures in an outer mixture over class labels. As this space permits many modes per class, we can obtain nonlinear decision boundaries, without sacrificing the appealing tractability of popular discriminative

approaches. The model can handle both missing labels and missing features, although the approximate inference methods we present requires all labels during training to be observed. The training and testing procedures derived from our Monte Carlo approximation constitute the AClass algorithm.

Our work builds on the approach of [15], which used conventional finite mixture models (trained with EM) to represent class-conditional densities. AClass addresses some of the basic inductive and algorithmic limitations of that approach. First, by grounding our algorithm in posterior inference in a bank of CRP mixtures, model complexity is automatically adjusted in a probabilistically coherent, self-consistent way as new datapoints arrive. Second, by approximating posterior inference using sequential Monte Carlo methods [4], we can efficiently and accurately train and test online, without bounding the model complexities we consider. In particular, as new data are encountered, our online inference scheme avoids the computationally expensive re-training (and re-crossvalidation) associated with finite mixture approaches.

3.1.1 Related Work

The model underlying AClass is a nonparametric extension of the mixture-of-finite-mixtures approach to supervised and semisupervised classification (see [15] for applications to text data and [9] for applications to speech recognition). Instead of finite mixture models for each class-conditional density, we use CRP mixture models with conjugate priors [19], which are typically fit to data using Markov chain Monte Carlo (MCMC) methods [14]. However, instead of computationally expensive, batch MCMC, we perform inference via the conditional-posterior particle filter for conjugate CRP mixtures (see e.g. [5] and [22]).

The Forgetron [3] is one recently developed discriminative approach that provides similar functionality to AClass. The Forgetron is a bounded-memory, online, kernelized, binary classification scheme with attractive theoretical properties, but it cannot naturally handle missing data. Widely used discriminative algorithms like regularized least squares (RLS) and logistic regression also provide reasonable performance benchmarks [20], although they lack some of the desirable features of our approach, especially online training and testing.

In principle, an efficient implementation of AClass should be applicable to large scale classification problems with mixed discrete and continuous features, such as classifying web pages, large image databases or astronomical data. In this paper, we focus on exploring the

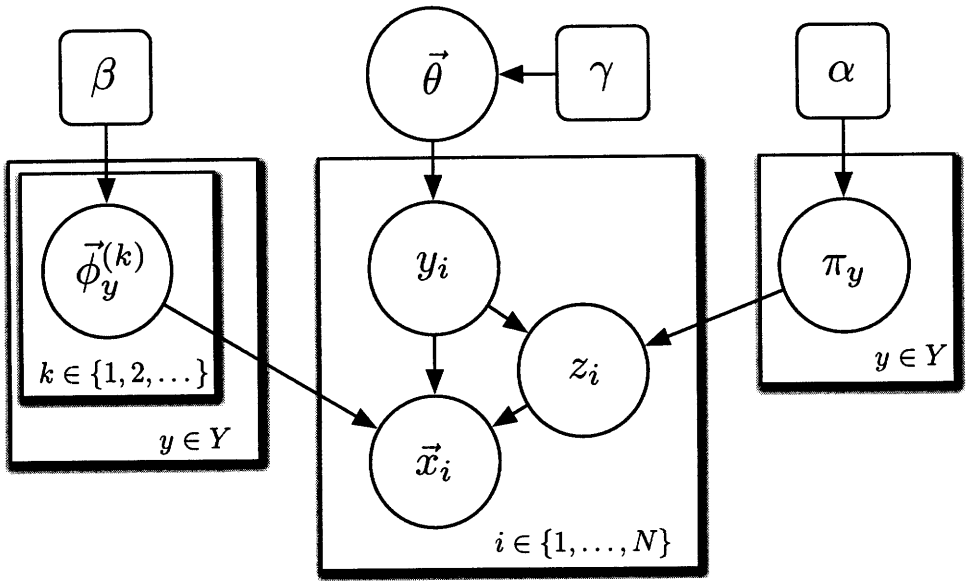


Figure 3-1: Graphical model for the joint probability distribution over latent parameters, labels and feature vectors in which AClass operates. The training and testing procedures comprising the AClass algorithm are derived from sequential Monte Carlo inference in this model.

properties of the basic model algorithm in simple cases.

3.2 Model and Inference

We describe our model in terms of a generative process for data sets, comprised of class labels, $y \in Y$, and feature vectors, $\vec{x}$.

1. Let $\vec{\phi}_y^{(k)}$ be the vector of parameters for the kth mixture component in the class-conditional density for class y. For $y \in Y$, $k \in \{1, 2, \ldots\}$, sample $\vec{\phi}_y^{(k)}$ from an appropriate conjugate prior (with hyperparameter β).

2. Sample mixing weights over class labels, $\vec{\theta} = \{\theta_y\}$, from a Dirichlet prior with hyperparameter γ.

3. For each datapoint $i \in \{1, \ldots, N\}$,

   (a) Sample the class label, y_i, from a multinomial distribution with parameters $\vec{\theta}$.

   (b) Sample the group (mixture component) membership for this datapoint, z_i, according to the CRP for class y_i. Formally, if n_y^g is the number of datapoints with

label y assigned to group g, and $n_y = \sum_g n_y^g$ is the total number of datapoints observed with label y so far, we choose each existing group with probability $\frac{n_y^g}{n_y+\alpha}$ and a new group (i.e. create a new mixture component in this class-conditional density) with probability $\frac{\alpha}{n_y+\alpha}$.

(c) Sample the feature vector $\vec{x}_i$ from the appropriate conjugate likelihood with parameters $\vec{\phi}_{y_i}^{(z_i)}$.

In this paper, we focus on binary features, modeled via the conjugate Beta-Bernoulli model. Continuous features (either independent or correlated, via the Normal-Inverse-Chisquared and Normal-Inverse-Wishart models, respectively) are a straightforward modification.

In essence, the per-class Chinese restaurant process induces a prior on partitions of our training data, ranging from the case where all datapoints are in one group (recovering a naive Bayes classifier or single-mode model for the class conditional density) to the case where each datapoint has its own mode (recovering a kernel density estimate, with the kernel derived from the posterior predictive distribution of the conjugate model for mixture components).

Our model has few free parameters. We set α to the standard value of 1.0, and leave the Beta hyperparameters at reference values of 0.5 throughout.

3.2.1 Inference

The structure of our sequential Monte Carlo algorithm for inference closely follows our generative process; this conceptual simplicity is an attractive feature of our approach. In the case of fully supervised data, the asserted class labels separate the parameters of the CRP mixture models for each class, so inference reduces to separate problems for each class-conditional CRP. To solve these decoupled problems, we employ a particle filtering approach based on [5].

The intuition behind a CRP mixture particle filter is that each particle represents a hypothesis about the assignment of the past data points to groups. When a new datapoint is processed by a particle, it is stochastically assigned to a group based on how well it fits with the data previously assigned to that group (via its probability under the group's posterior predictive distribution). Old datapoints are *not* reassigned; instead, the resampling step for the particle filter discourages the propagation of particles which seem to be relatively poor explanations of the data.

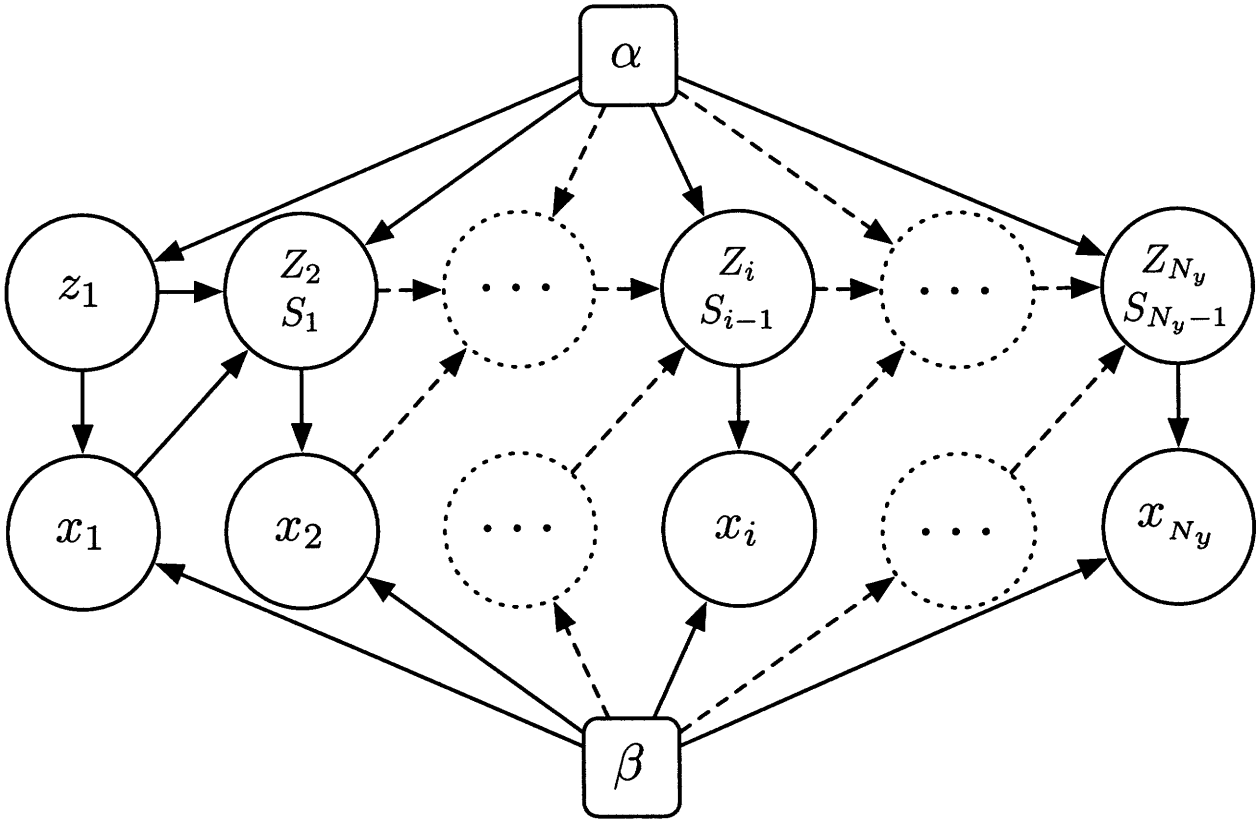


Figure 3-2: Dynamic model obtained by arbitrarily ordering the datapoints within each class and integrating out class-conditional component parameters. Z_i is the group (mixture component) membership vector for datapoints $1...i$. S_{i-1} is the collection of sufficient statistics for all groups present before datapoint i is processed. Note that the dimensionality of the latent variables Z and S grows as the chain unrolls over datapoints. The AClass inference algorithm can be interpreted as a bank of particle filters for this dynamic Bayesian network, one for each class.

To better understand the structure of our inference approach, it is illustrative to view the dynamical graphical model shown in Figure 3-2. This model can be obtained from our original model by focusing on one class, arbitrarily ordering the datapoints for that class, and integrating out the class-conditional mixture component parameters $\vec{\phi}$ (retaining only their associated per-mixture-component sufficient statistics). Our sequential Monte Carlo algorithm can be directly interpreted as a bank of independent filters for this model, one for each class.

The essence of the transformation from the original model to a dynamic one (where approximate filtering yields approximate posterior inference) is the collection of the z variables, or group memberships, for all datapoints up to i into a single group membership vector Z_i. This transformation directly exploits the self-consistency of the CRP prior on

partitions for varying numbers of datapoints to yield well-defined unrollings for an arbitrary number of datapoints. However, since we are using the CRP as a prior on partitions (or component assignment vectors) in a mixture model, we must also similarly transform the sufficient statistics; accordingly, we also track the sufficient statistics for all groups (mixture components) when datapoint i is processed in S_{i-1}. Thus the latent state of this dynamic model tracks a density estimate for the datapoints in terms of a partition of datapoints into mixture components and the parameters associated with the components. Accordingly, S_0 is empty (there are no pre-existing sufficient statistics when processing the first datapoint), and $Z_1 = z_1 = [0]$, since the first datapoint must be in its own group.

To sample from the "motion model" on latent variables, $P(Z_i, S_{i-1}|x_{i-1}, Z_{i-1}, S_{i-2})$, first copy Z_{i-1} into Z_i, then incorporate x_{i-1} into the appropriate part of S (based on its group assignment z_{i-1}), then sample a component assignment for datapoint i from the CRP prior. The "observation model" measures compatibility of x_i with the running density estimate; so, $P(x_i|Z_i, S_{i-1})$ is just the posterior predictive distribution of the group to which i was assigned (obtained by looking up the entry of S_{i-1} for the group z_i).

Particle filtering yields running Monte Carlo estimates of the latent variable in this chain; after processing all datapoints up to i, then, we have samples from $P(z_{1\ldots i}, s_{1\ldots(i-1)}|x_1, \cdots, x_i)$, namely the posterior distribution of interest. AClass is simply a conditional-posterior particle filter for this chain structured model. Since the particle filter can be asked for posterior estimates at any point in time, our algorithm is online. Since the only interactions between particles take place during aggregation steps such as resampling and testing, our algorithm is massively parallelizable with relatively inexpensive communication costs, and is, in principle, suitable for large scale classification tasks.

In the pseudocode shown in Figure 3.2.1, we use $\vec{x}$ to denote the F features of an observation (possibly with missing values), $y \in Y$ for class labels, α for the CRP hyperparameter, γ for the hyperparameter on the multinomial distribution over class labels, $A.m[l]$ to denote the counts for class l, and $p.n[g]$ to denote the counts for group g (part of a CRP partition) in particle p. We have N total datapoints. All values are assumed to be initialized to zero unless otherwise specified. Furthermore, we assume some basic utility procedures, such as SAMPLE-DISCRETE, which returns a pseudorandom sample from a vector of probabilities); AVERAGE, which returns a weighted average of its inputs; and RESAMPLE-PARTICLES($\mathcal{F}, \vec{w}$), which resamples the particles in filter $\mathcal{F}$ according to the weights in $\vec{w}$ via the standard re-

sampling algorithm from Sampling-Importance-Resampling[1] [4]. This step focuses effort of the sampler on particles that score well locally (i.e. explain the last datapoint well). In practice, all functions manipulating probabilities/weights should be implemented in the log domain to avoid underflow. Note that all vectors are assumed to be initialized to zeros and growable dynamically.

Because we assume conjugate priors on the parameters of the mixture components in each CRP-based class-conditional density estimator, we can integrate out these parameters analytically and only represent the sufficient statistics associated with previous datapoints assigned to a given group. UPDATE-SUFFSTATS and POSTERIOR-PREDICTIVE are standard probability computations in terms of these statistics; missing features are handled simply by ignoring them in these functions. For example, with binary data, the sufficient statistics for each feature are simply counts of observed heads, h_g^p, and observed tails, t_g^p, one for each group in each particle. Then UPDATE-SUFFSTATS increments those counts, and POSTERIOR-PREDICTIVE computes $P(\vec{x}) = \prod_{f=1}^{F} \frac{\beta+h^{x^f}t^{1-x^f}}{2\beta+h+t}$. See e.g. [7] for details for other conjugate models.

The memory requirements are worst-case linear, when every datapoint is assigned to its own group in its appropriate class-conditional density. However, this case is exceedingly unlikely in practice - in expectation, the Chinese restaurant process generates $\log \alpha$ groups, and generating as many groups as datapoints would be a signal that no generalization from one datapoint to another is licensed. In practice, we conjecture bounds of a few hundred groups should be adequate for very large scale real-world discrete feature classification tasks, based on [15].

We can characterize the complexity per iteration of the algorithm in terms of p_i (the number of particles per CRP density estimator), G (the maximum number of groups found in a class-conditional CRP) and the other variables defined above. In particular, TRAIN-CRP and PREDICTIVE-DENSITY both take $O(p_i GD)$ time. This subroutine must be called $O(N)$ times. Overall, then, we get an algorithm whose runtime can be expected to be linear in N (although the extremely unlikely worst-case complexity is quadratic).

[1]Throughout, we resample when the number of effective particles, measured as $1/(\sum \vec{w}_i^2)$, reaches half of the original number of particles in the filter

```
ACLASS-TRAIN(A:aclass, y:class-label, x:observation)
    TRAIN-CRP(A.F[y], x)
```

```
ACLASS-TEST(A:aclass, x:observation)
    foreach label y in Y
        prob[y] += (A.m[y] + γ) / (Sum(A.m) + |Y|·γ)
                   · PREDICTIVE-DENSITY(A.F[y], x)
    prob ← NORMALIZE(prob)
```

```
TRAIN-CRP(F:filter, x:observation)
    foreach particle p in F
        foreach group g in 1, 2, ..., p.numGroups
            prob[g] ← p.n[g] / SUM(p.n + α) · POSTERIOR-PREDICTIVE(p.stats[g], x)
        prob[g + 1] ← α / (SUM(p.n) + α) · POSTERIOR-PREDICTIVE(p.stats[g + 1], x)
        F.w[p] ← SUM(prob)
        g' ← SAMPLE-DISCRETE(prob)
        p.n[g'] ← p.n[g'] + 1
        UPDATE-SUFFICIENT-STATISTICS(p, g', x)
    RESAMPLE-PARTICLES(F, w)
```

```
PREDICTIVE-DENSITY(F:filter, x:observation)
    foreach particle p in F
        foreach group g in 1, 2, ..., p.numGroups
            prob[p][g] ← p.n[g] / (SUM(p.n) + α) · POSTERIOR-PREDICTIVE(p.stats[g], x)
        prob[p][g + 1] ← α / (SUM(p.n) + α) · POSTERIOR-PREDICTIVE(p.stats[g + 1], x)
        prob[p] ← SUM_g(prob[p][g])
    return AVERAGE(F.w[p], prob[p])
```

Figure 3-3: Pseudocode for the AClass algorithm. Suitable for use with generic conjugate models for mixture components. Depends on standard statistical procedures for sampling, averaging and predictive density evaluation.

3.3 Experiments

We performed several preliminary experiments to address the basic performance of our algorithm in synthetic and real-world settings. Throughout, the discriminative benchmarks we compared against were linear logistic regression and nonlinear regularized least squares with a polynomial kernel of degree 2. For the regularized least squares algorithm, we used a one-vs-all scheme and optimized the regularization parameter λ for each classifier individually via efficient leave-one-out cross-validation methods [20]. For logistic regression, we set $\lambda = 10^{-2}$ in all trials. To handle missing data with discriminative methods, we used online mean imputation - that is, we filled in each missing feature value with the running average of the observed values seen before it, starting with 0.5. It is worth noting that this mean imputation strategy can be interpreted as fitting a single mixture component to each class, with independent feature values, and imputing with the mean of the posterior predictive distribution on features.

Our core approach to classification is grounded in class-conditional density estimation. Accordingly, we first report the performance of our density estimator (the TRAIN-CRP and PREDICTIVE-DENSITY subroutines) on synthetic data. We generated synthetic data from mixture models over 50-dimensional binary feature vectors with varying numbers of components (from 1 to 30), to test our particle filter inference approximation (and model robustness to the setting of α, which was fixed to 1 throughout). The mixing weights (over the components) were fixed to be uniform, yielding groupings a priori disfavored under the Chinese restaurant process and therefore a conservative test of particle filter accuracy. For each component in each mixture, we generated parameters for each binary feature (corresponding to a single coin weight) from a Beta(0.5, 0.5) distribution, yielding moderately noisy clusters. We then measured accuracy at component discovery and density estimation under our model.

Figure 3-4 summarizes our results with respect to discovery of the right complexity mixture. We found that the particle filter approximation discovered complexities accurately, though for very small training sizes, we underestimate the number of groups if the true number is large enough that some may not have been observed in the training sample.

We also computed a Monte Carlo estimate of the KL divergence between the density learned by our particle filters and the true mixture density. For comparison, we show the KL divergence between truth and a one-mode model (which, if used for class-conditional density estimation and classification, would yield a naive Bayesian classifier). When the true

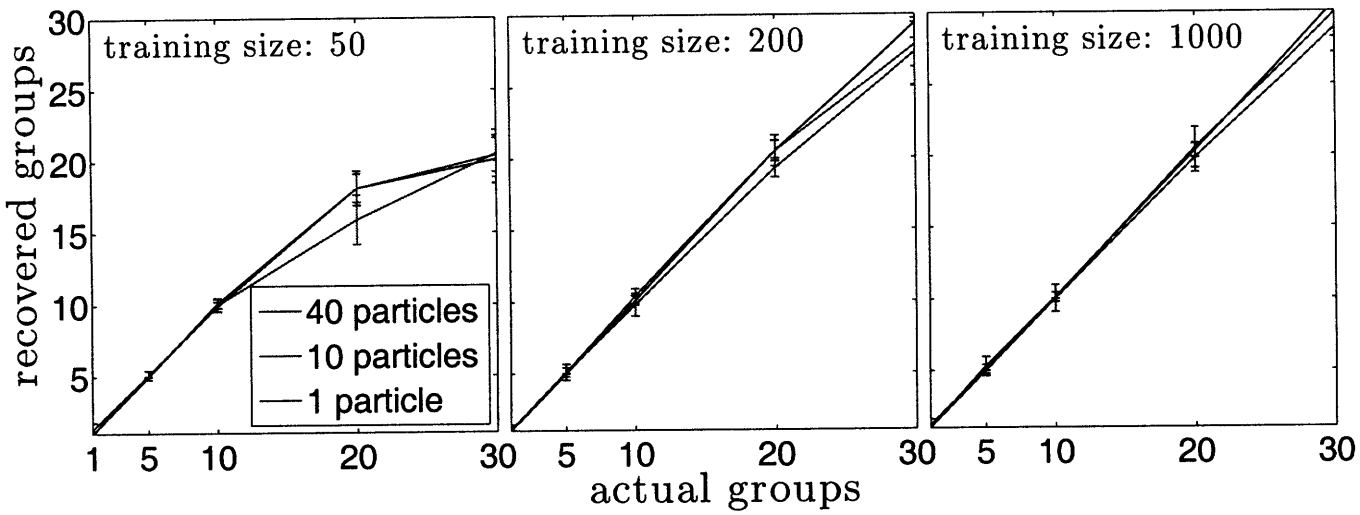


Figure 3-4: Accuracy of sequential Monte Carlo for Chinese restaurant process mixture models on synthetic data. Each plot shows the number of recovered groups (mixture components) versus the number of groups in the true model from which synthetic data was sampled. Each plot shows accuracy for a different number of training points. Each line shows results for inference with a different number of particles. All results are averages over 10 training sets.

density has multiple modes, the naive Bayesian model performs substantially worse than a CRP mixture, though they agree when the true distribution is unimodal. Otherwise, we find all particle filters yield accurate density estimators with small numbers of samples, though the 40 particle version is roughly twice as accurate as the 1-particle version when the true density has many groups. This is unsurprising, as the 1-particle filter will be sensitive to random errors in the assignment of early datapoints, while similarly erroneous particles die in resampling in the 40-particle filter.

Our second experiment explored the issue of whether these density estimators can be used as the basis of competitive generative classifiers. We investigated several synthetic classification problems, and report here the results on one of them: a four-class problem, where the class-conditional density for each class was a mixture model (as in experiment 1, with uniform mixing weights and Beta(0.5, 0.5) priors on the Bernoulli parameter for each dimension). The mixtures for the four classes had 4, 10, 5 and 20 modes respectively, representing a wide range of complexities requiring cross-validation over a broad range under traditional approaches. We report average 0-1 loss on a held-out test set of 500 datapoints for a range of training set sizes, as well as performance in a missing data setting, where varying fractions of feature values were missing at random during training *and* testing. At

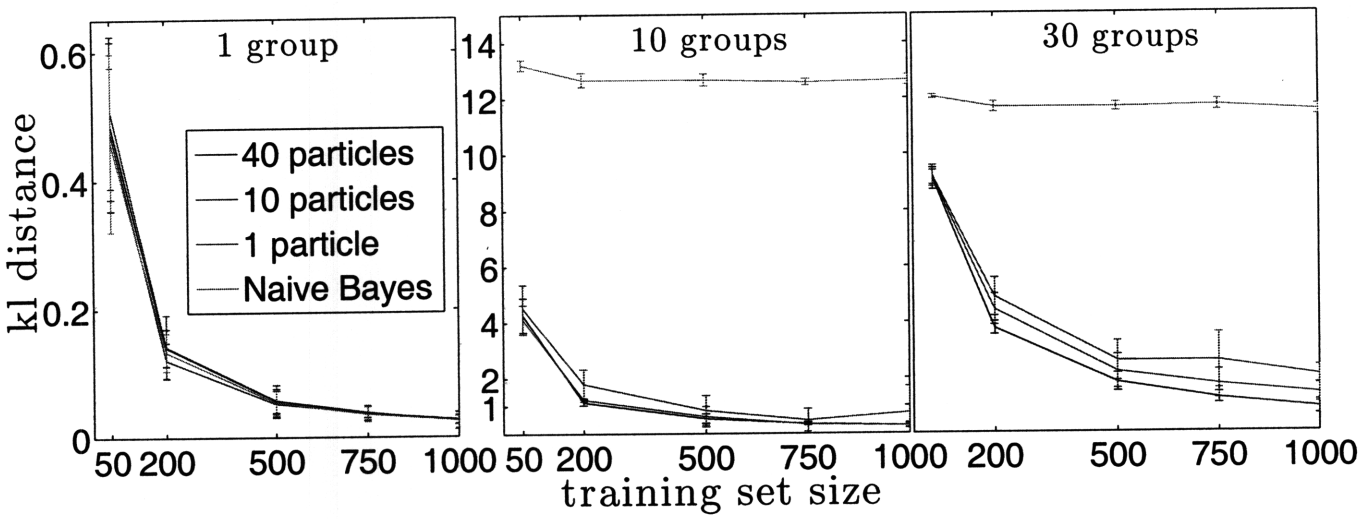


Figure 3-5: Effectiveness of sequential Monte Carlo inference in Chinese restaurant process mixture models at producing accurate density estimators. Each plot shows a Monte Carlo estimate of the KL divergence between the learned density model and ground truth versus the number of training data points. Each plot shows accuracy for a different number of groups (mixture components) in the true density. Each line shows results for inference with a different number of particles (and a one-component model, equivalent to the density estimation done as part of a naive Bayes classifier, as a baseline). All results are averages over 10 training/testing sets. Note the scale difference in the 1 group case.

all three levels of missing data, AClass outperformed the other algorithms. Naive Bayes was the least effective. With no missing data, the 1-particle AClass performs essentially as well as AClass, since classification is easier than density estimation (at least under 0-1 loss, so predictive uncertainties are unimportant). However, with 50% data missing, the 1-particle AClass suffers slightly, as small errors in class-conditional density estimation have a greater impact on test accuracy.

Our third experiment assessed the performance of our approach on a standard classification task based on a subset of the 20 Newsgroups data set collected by [21]. This dataset consists of text documents belonging to one of four broad categories (the `comp.*`, `rec.*`, `sci.*`, and `talk.*` usenet hierarchies). For each document, there are 100 binary features indicating the presence or absence of each of 100 common non-stop words. Figure 3-7 summarizes these results. As above, we report average 0-1 loss on a held-out test set of 500 for all algorithms. Qualitative properties of the performance comparison are consistent with the synthetic tests above, with performance degrading gracefully (and remaining above the discriminative approaches) as features are made missing at random. It is worth noting that

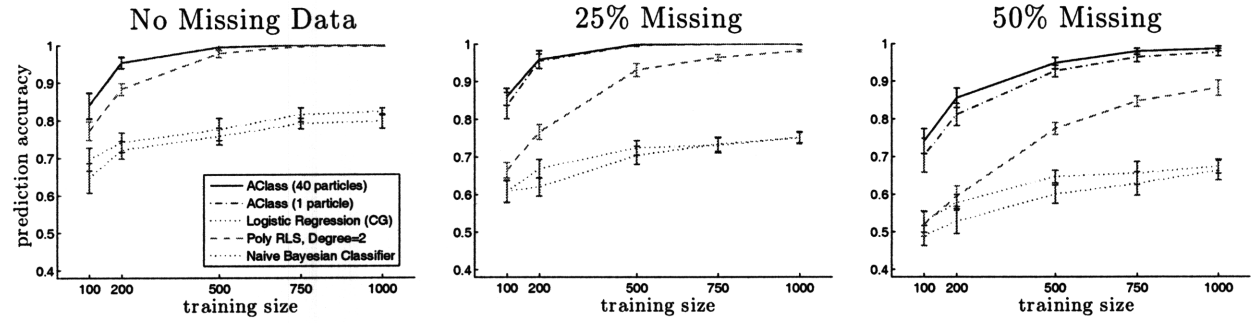


Figure 3-6: Classification performance (via average 0-1 loss) versus training set size on a 100 feature synthetic benchmark with 4 classes. The class-conditional densities had 4 modes, 5 modes, 10 modes and 20 modes respectively. Each plot shows performance when a different fraction of feature values are missing during training and testing. Each line shows classification performance for a different classification algorithm. All results are averages over 10 training/test sets.

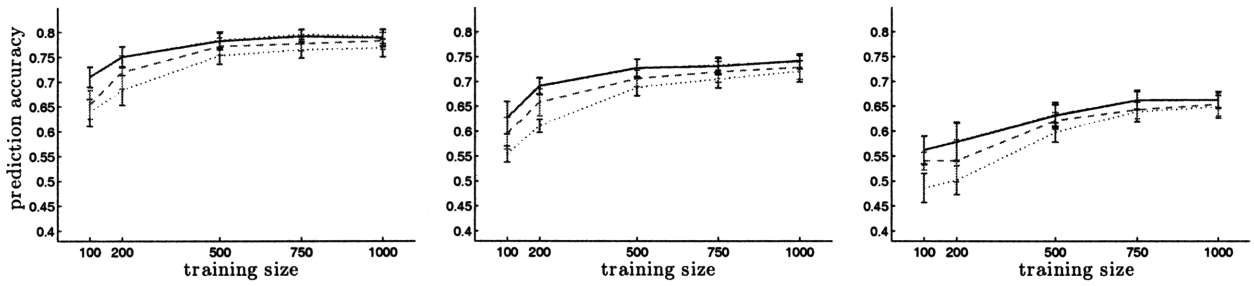


Figure 3-7: Classification performance, as above, on a 4-class problem on a 20-Newsgroup dataset with 100 binary features. Note the effectiveness of the naive Bayes classifier on this task, and the recovery of that performance by AClass, due to sensible regularization.

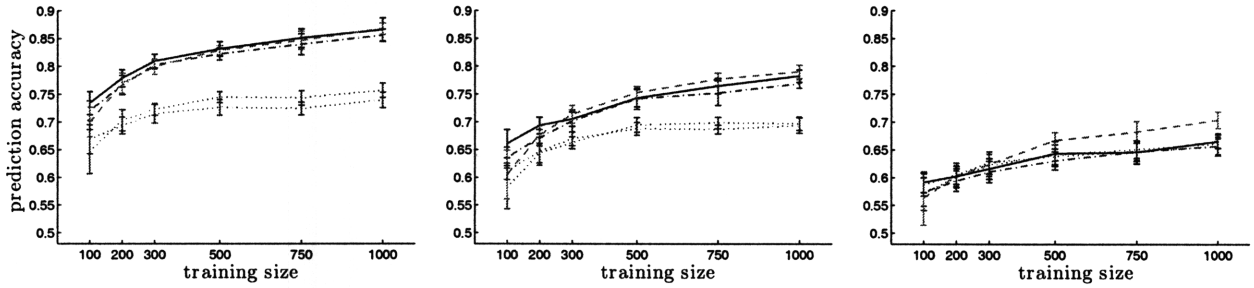


Figure 3-8: Classification performance, as in Figure 3-6, on a 2-class, odd versus even MNIST digit classification task. Feature vectors were obtained by binarizing coefficients with respect to the first 50 principal components of MNIST. Note AClass' competitive performance with polynomial regularized last squares when 0% and 25% of feature values are missing, and the crossover to naive Bayes' lower performance when 50% of feature values are missing.

in this data set, AClass appears to be finding a single mode per class, likely due to the inherent feature sparsity and lack of hyperparameter inference, recovering the performance of naive Bayes.

For our fourth experiment, we assess performance on a binarized variant of MNIST, collapsed into an odd-vs-even digits classification task. Following standard preprocessing procedures, we first extracted the top 50 principal components from 5000 randomly chosen examples, each originally 784-dimensional vectors with discrete values in the range [0,255]. We then binarized the components (by taking their sign); this binary data was the input to all algorithms. Our results are in Figure 3-8. When all data is observed, AClass performs competitively with polynomial RLS and significantly outperforms naive Bayes and logistic regression, suggesting the helpfulness of a properly regularized nonlinear decision boundary. However, once half the feature values are missing, AClass and naive Bayes perform almost identically, likely because AClass no longer has sufficient evidence to sustain multimodal class conditional densities.

Without preprocessing via PCA, on fully observed data, AClass performs as naive Bayes, and is beaten by polynomial RLS. We suspect this is because we are not performing hyperparameter learning - or modeling the continuous data directly - and are thus getting mislead by the large number of background pixels that are always off for each training example. In our experience, accurate hyperparameter estimation can often be an important enabler of sampler accuracy. Incorporating hyperparameter estimation into our approach might well require the inclusion of Gibbs moves, and could therefore require further research to obtain online approximations.

3.4 Discussion

Our model and inference algorithm are both conceptually attractive, as they each address a significant limitation of previous work – namely, principled choice of the complexity of the class-conditional density estimators and support for tractable, online inference. It is the combination that enables us to construct ACLASS, an algorithm that yields good computational and generative performance while still being grounded in a simple procedure and intuitive inductive bias.

This model/algorithm pair is also interesting from the standpoint of the broader generative/discriminative debate in prediction. By 'generative model', we mean a model for

datapoints and labels which includes a model for the class-conditional densities, where prediction (at test time) is done by Bayes' rule. By 'discriminative model', we mean a model for only the probability of a label given a datapoint[2].

Typically, discriminative models admit a larger space of possible decision boundaries than generative ones, and are often nonparametric (e.g. SVMs, RLS). Therefore discriminative models often perform better asymptotically, but can overfit on very small data sets. Generative models, on the other hand, can perform poorly in the large data limit if they don't allow arbitrarily complex class-conditional densities via nonparametric density estimators; this is the case with e.g. finite mixture models and certainly with naive Bayes. Furthermore, if the inference approximations used to train a generative model don't accurately incorporate Bayesian model averaging, as is the case with standard approaches like EM-trained finite mixture models, prediction performance may be quite poor for very small data sets due to overfitting. This leaves little room for generative methods to perform well, supporting the conventional wisdom that discriminative approaches are more appropriate for classification problems.

This conventional wisdom can be refined based on our algorithm and experiments. Generative methods such as AClass can in fact exploit the power of nonparametrics via appropriate class-conditional modeling, as well as perform competitively on a range of data set sizes via tractable approximate inference. However, they can still suffer relative to discriminative methods in the presence of large numbers of noisy or irrelevant features (as in raw MNIST, without PCA preprocessing), by modeling aspects of the data that do not vary across classes. One possible remedy would be to construct generative models where only part of the features of each datapoint (e.g. a subset of features, or an additive component for all features) is modeled conditioned on the class label, while the rest is modeled using a single set of parameters shared across all classes. This would in some sense be a probabilistic elaboration on PCA-based preprocessing. Hyperparameter learning, ideally involving coupling across classes to support ignoring universally noisy or irrelevant features, is one very simple means of addressing this. Our polynomial RLS implementation, for example, used leave-one-out cross-validation to adjust kernel bandwidth, a step in this direction.

Another approach, supported by the surprisingly good performance of polynomial RLS

[2]Training procedures for discriminative models might well be explicitly Bayesian or have Bayesian interpretations (as with regularized least squares and logistic regression), but this simply means they are *probabilistic*, not generative.

with online mean imputation, would be to build hybrid classifiers out of one generative model and one discriminative model. The trained generative model would provide priors on missing labels and features, which could be used in fitting and testing a discriminative model. One simple, heuristic way to realise this combination, using existing generative and discriminative algorithms, would be to train a generative model, fill in missing features (or even labels, during training) according to the mean of its posterior predictive distribution, and use this filled-in data to train and test a discriminative algorithm.

The most salient avenue for future work on AClass is in empirical evaluation. Developing a parallel implementation and doing an in-depth classification benchmark study, including other discriminative methods such as the Forgetron and other kernel machines, will be key; these were omitted due to the preliminary nature of our present experiments. We also note that constant-memory applications (such as classification on embedded devices) could enforce a hard bound by constraining the Chinese restaurant process to produce no more than some constant number (say G^+) of groups for each class. This is *not* equivalent to fixing the number of groups in advance; each partition with a number of groups less than or equal to G^+ remains a distinct hypothesis in the space, is considered during the course of inference, and has finite probability.

Furthermore, our inference algorithm can be elaborated in many ways to improve its performance and increase its scope. For example, the transformation from the AClass model to the chain in Figure 3-2 motivates a general strategy for sequentializing Bayesian inference (over datapoints and over numbers of variables). This strategy permits more sophisticated particle filters, including ones that use proposal distributions obtained via exactly or approximately summing out over future values of the latent states, as well as filters that alternate between particle filtering updates and Gibbs sampling of past assignments. We are currently developing this general case in a separate paper, as well as working on their application to the AClass model to reduce the sensitivity of the one-particle approximation to mistakes in the beginning of a data set.

Finally, it should be possible to develop approximations to inference in the AClass model suitable for situations with missing labels as well as missing features. In particular, one could attempt to estimate missing class labels by a nested particle filtering scheme, where an outer particle filter assigns labels to unlabeled datapoints, and each particle in this filter contains an instance of the AClass algorithm. Even in data sets where AClass falls back on a single mode per class, such a strategy remains online, and therefore a potentially appealing

alternative to EM.

3.5 Conclusion

We have presented AClass, a simple, online learning algorithm for supervised multiclass classification in the presence of missing data. Our algorithm operates by sequential Monte Carlo inference in a mixture of Chinese restaurant process mixtures. This generative model encodes a simple inductive bias, namely prototype discovery for each class, and our algorithm performs well in preliminary experiments. Our nonparametric formulation allows AClass to learn arbitrary complex decision boundaries given sufficient evidence, while generalizing rapidly when a small number of prototypical examples suffice to describe the data.

Classification algorithms grounded in generative probabilistic models can support principled treatment of missing feature values and inference over unlabeled training examples, both of which are important in real-world applications. However, these advantages must be achieved while retaining algorithmic tractability and good generalization performance. We hope that AClass (and the generative model in which it operates) represents a useful step in this direction.

Bibliography

[1] M. Beal and Z. Ghahramani. The Variational EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures. *Bayesian Statistics*, 7, 2003.

[2] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[3] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a fixed budget. In *NIPS 18*. 2006.

[4] A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. 2001.

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

[6] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In *IJCAI 16*, pages 1300–1309, 1999.

[7] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman & Hall, 2003.

[8] P. Giudici and R. Castelo. Improving Markov Chain Monte Carlo Model Search for Data Mining. *Machine Learning*, 50:127–158, 2003.

[9] A. Halberstadt and J. Glass. Heterogeneous measurements and multiple classifiers for speech recognition. In *Proceedings of ICSLP*, 1998.

[10] C. Kemp, T. Griffiths, and J. Tenenbaum. Discovering latent classes in relational data. Technical report, MIT CSAIL, 2004.

[11] P. Kraaijeveld, M. Druzdzel, A. Onisko, and H. Wasyluk. GeNIeRate: An Interactive Generator of Diagnostic Bayesian Networks, 2005.

[12] D. Madigan, A. Raftery, C. Volinsky, and J. Hoeting. Bayesian model averaging. In *Proc. AAAI Workshop on Integrating Multiple Learned Models*, Portland, OR, 1996.

[13] K. Murphy. Learning bayes net structure from sparse data sets. Technical report, Comp. Sci. Div., UC Berkeley, 2001.

[14] Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

[15] Kamal Nigam. Using Unlabeled Data to Improve Text Classification. Technical Report CMU-CS-01-126, Computer Science Department, Carnegie-Mellon University, 2001.

[16] A. Onisko, M. J. Druzdzel, and H. Wasyluk. Learning Bayesian network parameters from small data sets: Application of noisy-OR gates. *International Journal of Approximate Reasoning*, 27:165–182, 2001.

[17] D. Pe'er, A. Tanay, and A. Regev. MinReg: A Scalable Algorithm for Learning Parsimonious Regulatory Networks in Yeast and Mammals. *Journal of Machine Learning Research*, 7:167–189, 2006.

[18] J. Pitman. Combinatorial stochastic processes. (Notes for the Saint Flour Summer School), 2002.

[19] C. Rasmussen. The Infinite Gaussian Mixture Model. In *NIPS 12*, pages 554–560, 2000.

[20] R. Rifkin and R. Lippert. What you should know about matrix decompositions and regularized least squares. Unpublished manuscript, 2006.

[21] Sam Roweis. 20 newsgroups fragment, 2006.

[22] A. N. Sanborn, T. L. Griffiths, and D. J. Navarro. A more rational model of categorization. In *COGSCI*, 2006.

[23] E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning Module Networks. *Journal of Machine Learning Research*, 6:557–588, 2005.

[24] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base I. - The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255, 1991.

[25] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet Processes. Technical report, UC Berkeley Statistics Department, 2004.

[26] M. Teyssier and D. Koller. Ordering-based Search: A Simple and Effective Algorithm for Learning Bayesian Networks. In *UAI 21*, pages 584–590, Edinburgh, Scottland, UK, July 2005.

[27] S. Wasserman and K. Faust. *Social network analysis: Methods and applications.* Cambridge University Press, 1994.