SEMI-SUPERVISED LEARNING USING BAYESIAN HIERARCHICAL METHODS

V. CHEN
ADVISED BY A. M. STUART AND M. M. DUNLOP

Abstract.
In semi-supervised machine learning, the task of clustering is to divide the data into groups using a labeled subset. Our Bayesian approach to graph-based clustering views the classifying function as a random variable with a distribution that combines the label model with prior beliefs about the classification. In Bayesian hierarchical methods, hyperparameters governing the prior distribution are introduced and can be sampled as well, with the goals of both deriving a classification and learning the distribution of the hyperparameters. We apply Markov Chain Monte Carlo methods for indirectly sampling the posterior distribution of these random variables, as direct sampling is generally challenging. We focus on priors derived from the graph Laplacian, a matrix whose eigenvectors are known to contain cluster information. We implemented Bayesian hierarchical models that learn different sets of hyperparameters, including ones that govern the scale of the eigenvectors or the number of eigenvectors used. We tested these models on real and synthetic data sets. Our results indicate that there is information to be learned about the distribution of these hyperparameters, which could be used to improve classification accuracy.

1. Introduction. In the context of semi-supervised learning, the *clustering* problem refers to attempting to group a set of data or objects into clusters of similar objects. Applications of clustering include image segmentation, social network analysis, and voter classification, but the reader will readily identify numerous other potential applications. We approach the clustering problem from a graph-theoretical point of view, which assigns a node to each data value and weights on edges between nodes to indicate how related the data points are.

Let Z be a set of nodes $\{1, \dots N\}$. The feature vectors are given by $x : Z \to \mathbb{R}^d$. $Z' \subset Z$ is a subset of labeled nodes, for which we have label data given by $y : Z' \to \{-1, 1\}$ for **binary** data, and $y : Z' \to \{e_1, \dots e_k\}$ for k**-ary** (multiclass) data.

The classifying function is given by the vector u, and the label model is given by $y(\ell) = S(u(\ell))$ for $l \in Z'$ (with possible additional corruption by noise). In the case of binary data, $y(\ell) \in \{\pm 1\}$, and for $v \in \mathbb{R}$,

$$S(v) = \begin{cases} 1 & v > 0 \\ 0 & v = 0 \\ -1 & v < 0. \end{cases}$$

In the case of multiclass data, $y(\ell) \in \{e_1, \dots e_k\}$, $v \in \mathbb{R}^k$ and

$$S(v) = e_p, \quad p = \underset{r=1,\dots,k}{\operatorname{argmax}} v_r$$

where $\{e_l\}_{l=1}^k$ denotes the standard basis of $\mathbb{R}^k$.

Given this problem, one goal is to come up with a classification that best fits the data. The research group in which I am working is currently looking at the effectiveness of incorporating Bayesian inference into semi-supervised learning. In the Bayesian approach, the unknown classifying function u is viewed as a random variable, whose distribution is determined by combining the labeling model y with a probability distribution representing our prior beliefs about the classification and derived from the feature vectors. Hierarchical Bayesian methods provide additional flexibility by introducing hyperparameters governing the prior distribution, which are in turn treated as random variables to be inferred.

Our prior beliefs come from *spectral clustering*, which uses the clustering properties of the eigenvalues and eigenvectors of the graph Laplacian, a matrix defined on the similarity graph constructed from the feature vectors as follows:

Let the similarity graph be G with weights $w_{ij} \geq 0$. A larger weight indicate a higher degree of similarity between the two nodes. Define $\delta_i = \sum_{j=1}^N w_{ij}$. Then, define matrices W, D as:

$$W = w_{ij}, \quad D = \begin{cases} \delta_i & i = j \\ 0 & i \neq j. \end{cases}$$

Then, let $L = D - W$ denote the unnormalized graph Laplacian on $\mathbb{R}^N$. Also, denote $L_{\text{sym}} = D^{-1/2}LD^{-1/2}$ as the symmetric normalized graph Laplacian, and $L_{\text{rw}} = D^{-1}L$ as the random walk normalized graph Laplacian. Denote the eigenbasis of L by

$$Lq_j = \lambda_j q_j, \quad j = 0, \dots, N-1. \tag{1}$$

We order the eigenvectors $q_0, \dots q_{N-1}$ in increasing order of their corresponding eigenvalues. These eigenvalues are always non-negative [6]. The graph Laplacian is important because the eigenvectors corresponding to the smaller eigenvalues are known to hold information about cluster data. In particular, if the graph has "ideal" clusters so that the edges between clusters all have weight 0, then the graph Laplacian would have as many eigenvalues 0 as there are clusters, and corresponding eigenvectors would be indicators for the clusters [6]. Even if the data is not ideal, the eigenvectors corresponding to small eigenvalues should hold clustering information. This includes the important Fiedler vector, the second-smallest eigenvector which is useful for bi-partitioning the graph. The eigenvalues and eigenvectors of the graph Laplacian are used to construct the prior probability distribution about u.

For the Bayesian approach, the prior, likelihood and posterior are defined as follows:

$\mathbb{P}(u)$: **Prior** distributed as $\mathsf{N}(0, C)$ where the covariance matrix $C = f(L)^2$ in general, where f is a function that we choose. A sample u from this prior can be made through a realization of the following random sum:

$$u = \sum_{j=0}^{M} f(\lambda_j)\xi_j q_j, \quad \xi_j \sim \mathsf{N}(0,1) \quad \text{i.i.d.} \tag{2}$$

Looking at the form of the sample, it would make sense to impose the restriction that f should be monotonically decreasing so that lower eigenvectors dominate the distribution, a belief consistent with our discussion about graph Laplacian eigenvectors above.

We will be mostly using a fixed function f, chosen as $f(\lambda) = (\lambda + \tau^2)^{-\alpha/2}$. In this case, $C = (L + \tau^2 I)^{-\alpha}$, where τ and α are hyperparameters that govern the prior, and a sample u from the prior can be made through the following:

$$u = \sum_{j=0}^{M} (\lambda_j + \tau^2)^{-\alpha/2}\xi_j q_j, \quad \xi_j \sim \mathsf{N}(0,1) \quad \text{i.i.d.} \tag{3}$$

Note that this f is indeed monotonically decreasing.

Another possibility not explored in this paper is to learn f as well.

$\mathbb{P}(y|u)$: **Likelihood**, given by $\exp(-\Phi(u))$.

If we believe there is uncertainty in the labels, we could represent this uncertainty as additive Gaussian noise. For $\gamma > 0$, $\Phi(u)$ is chosen as

$$\Phi(u) = \frac{1}{2\gamma^2} \sum_{\ell \in Z'} |y(\ell) - S(u(\ell))|^2 \tag{4}$$

as introduced in [4]. This definition can be derived from the **Bayesian level-set** model for label uncertainty, which assumes $y(j) = S(u(j)) + \eta_j, \eta_j \sim \mathsf{N}(0, \gamma^2)$ i.i.d. Then, the conditional distribution $y(j)|u \sim \mathsf{N}(S(u(j)), \gamma^2)$. We can compute the likelihood as follows:

$$\mathbb{P}(y|u) = \prod_{j \in Z'} \mathbb{P}(y(j)|u) \propto \prod_{j \in Z'} \exp\left(-\frac{(y(j) - S(u(j)))^2}{2\gamma^2}\right).$$

So $\Phi(u) = -\log \mathbb{P}(y|u) = \frac{1}{2\gamma^2} \sum_{\ell \in Z'} |y(\ell) - S(u(\ell))|^2$ as desired.

If we believe that the label data is certain, $\Phi(u)$ is defined as

$$\Phi(u) = \begin{cases} 0 & S(u(l)) = y(l) \quad \forall l \in Z' \\ \infty & \text{otherwise.} \end{cases} \tag{5}$$

This case can be viewed as the $\gamma \to 0$ limit of (4) to enforce that u must perfectly respect the labels. Notice that $\exp(-\Phi(u)) = 1$ if and only if the classifying function u respects the label data. Otherwise, $\exp(-\Phi(u)) = 0$.
One other model for the likelihood is the **probit** model, also used in [1], in which we add noise to the classifying function before taking the sign function. Specifically, we assume $y(j) = S(u(j) + \eta_j), \eta_j \sim \mathsf{N}(0, \gamma^2)$ i.i.d.. Denote the normal cumulative distribution function

$$\psi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-t^2/2\right) dt$$

Then,

$$\begin{aligned}
\mathbb{P}(y(j) = 1|u) = \mathbb{P}(\mathsf{N}(u(j), \gamma^2) > 0) &= \frac{1}{\sqrt{2\pi\gamma^2}} \int_0^{\infty} \exp\left(\frac{-(z - u(j))^2}{2\gamma^2}\right) dz \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\frac{u(j)}{\gamma}}^{\infty} \exp\left(-t^2/2\right) dt \\
&= 1 - \psi(-u(j)/\gamma) \\
&= \psi(u(j)/\gamma)
\end{aligned}$$

where we used the fact that $1 - \psi(-x) = \psi(x)$.
Similarly, we can compute $\mathbb{P}(y(j) = -1|u) = \psi(-u(j)/\gamma)$. Since the labels $y(j)$ are either 1 or -1, we can write $\mathbb{P}(y(j)|u) = \psi(y(j)u(j)/\gamma)$. Finally,

$$\mathbb{P}(y|u) = \prod_{j \in Z'} \mathbb{P}(y(j)|u) = \prod_{j \in Z'} \psi(y(j)u(j)/\gamma)$$

and we can see that we would define

$$\Phi(u) = -\sum_{j \in Z'} \log\left(\psi(y(j)u(j)/\gamma)\right) \tag{6}$$

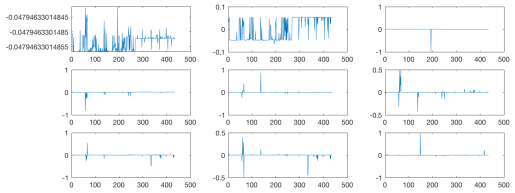
$\mathbb{P}(u|y)$: **Posterior**, which by Bayes' theorem is proportional to $\mathbb{P}(u)\mathbb{P}(y|u)$

1.1. Models. Now that we have established the groundwork for the Bayesian methods, we describe the models of interest for this project. For all of the following models, we aim to learn u or ξ (the classifying function), as well as any of the listed hyperparameters. The models evolve as follows:
For models (A), (B), and (C), the hyperparameters τ, α, M refer to the prior in (3).
(A) Use fixed $\tau, \alpha, M = N - 1$ to cluster data.
(B) Learn τ, α; fix $M = N - 1$.
(C) Learn τ, α, M.
For models (D) and (E), samples from a new prior are given by $u = \sum_{j=0}^{M} v_j \xi_j q_j$.
(D) Learn $\{v_j\}_{j=0}^{M}$ with M fixed.
(E) Learn $\{v_j\}_{j=0}^{M}$ and M.
Model (F) can apply to all of the previous models.
(F) Multiclass, hierarchical on number of classes.
Model (G) focuses on learning the arbitrary function f defined in (2).
(G) Learn $f(\lambda), M$.
Models (A) through (E) were fully implemented. Model (F) applied to model (C) has been implemented, but not with a hierarchical learning of the number of classes. Model (G) was not implemented.

1.2. Model problems. Applying the discussed models on datasets, real or synthetic, allow us to compare models in terms of convergence rate and classification accuracy. The datasets employed include:

1.2.1. Voting records. This dataset contains the voting records on 16 votes of 435 individuals from the U.S. House of Representatives, in 1984 from the 98$^{\text{th}}$ United States Congress, 2$^{\text{nd}}$ session. The data is ordered so that all the representatives from the party are contiguous in the indexing. The graph has $N = 435$ nodes, with feature vectors that are $d = 16$ dimensional. Each component of the feature vectors is either $+1, -1$, or 0 to indicate voting for, voting against, or abstaining on a particular bill, respectively. On this data set, we want to perform binary classification with a subset of labeled data. For this problem, we will use the unnormalized Laplacian with fixed length-scale weights. That is, we will define $w_{ij} = \exp\left(-\frac{d(x_i,x_j)}{2l^2}\right)$, $\quad d_{p,q}(x,y) = ||x-y||_p^q$ with $p = 2, q = 2, l = 1$. Figure 1 is part of the spectrum of this choice of Laplacian.

FIG. 1. *Lowest* 9 *eigenvectors of unnormalized L of voting records.*



1.2.2. Two moons. This is a synthetic data set constructed with two half circles in $\mathbb{R}^2$ with radius one. These are embedded in $\mathbb{R}^{100}$, and data points are sampled from the circles with Gaussian noise distributed as $\mathsf{N}(0, \sigma^2)$ added to each of the 100 dimensions. In this data set, we will be generating realizations of two moons with $N = 1000$ or $N = 2000$ nodes, each associated with $d = 100$ dimensional feature vectors. This data is again ordered, with the first $N/2$ nodes generated from the first half circle, and the latter $N/2$ from the second. Again, here we want to perform binary classification into the two half circles from which the data was generated. We will be using the self-tuning, symmetric Laplacian for this data set, introduced in [7]. This Laplacian matrix has weights that infer the local spatial scale from the data, removing the need to choose a fixed length-scale parameter. The use of this self-tuning Laplacian in the two-moons data set seems to encode more information in the eigenvectors. Compare Figure 2 with Figure 3.

FIG. 2. *Lowest* 9 *eigenvectors of self-tuning symmetric L of two moons,* $\sigma = 0.04$.

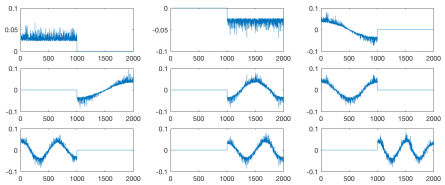
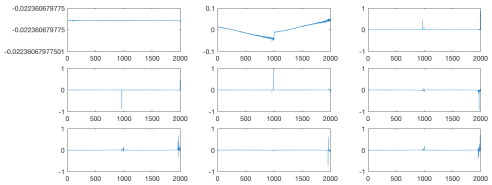


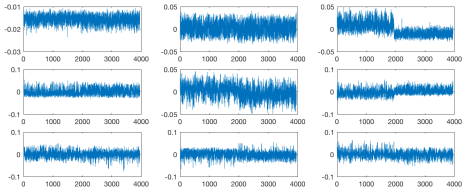
FIG. 3. *Lowest* 9 *eigenvectors of unnormalized fixed length L of two moons,* $\sigma = 0.04$.



1.2.3. MNIST data sets. This data set contains 70,000 images of 28×28 pixels with handwritten digits 0 through 9. The feature vectors are $d = 400$ dimensional and are formed by projection onto the first 50 PCA components. We can use this dataset to test binary classification if we choose two digits to identify. In particular, 4 and 9 are the hardest digits to distinguish, and we refer to this binary problem as MNIST49. This dataset can also be used for multiclass algorithms, up to $k = 10$ with the classes $0, \ldots 9$. N depends on the digits that we wish to use and is typically around 4000. We will use the self-tuning, symmetric normalized Laplacian again. See Figure 4.

1.3. Findings. Our first finding is that some hierarchical models are preferable to others with respect to metrics such as convergence rate and classification accuracy. In particular, our experiments indicate that the hyperparameters (τ, α, M) in model (C) appear to be better than the formulation with (v, ξ, M) in model (E).

FIG. 4. *Lowest 9 eigenvectors of self-tuning symmetric L of MNIST, digits 4 and 9.*



We can observe through experiments that τ, α, M are important for selecting the weight and number of eigenvectors used to formulate a classifying function. These hyperparameters show promising behavior when applied to the model problems.
We finally find that it is preferable to learn certain combinations of the hyperparameters of (τ, α, M), and not necessarily learn all three at once. Some combinations perform better than others with respect to convergence rate, classification accuracy, and behavior of the hyperparameters.
The remaining sections describe these findings and the experiments that support our claims.

2. Algorithms. This section discusses the algorithms that we have implemented for the hierarchical models. Note that model (B) can be thought of as a special case of model (C), and model (D) as a special case of model (E). Therefore, we provide the algorithms for model (A), (C), (E), and (F) only. These algorithms are implemented in MATLAB and tested on the listed model problems.

2.1. Model (A). For the nonhierarchical model (A), we implemented a pCN algorithm first introduced in [2] and described in Algorithm 1.

Algorithm 1 General pCN adapted from [3]

Select $u^{(0)}$. Select τ, α. Select $\beta \in [0, 1]$
for $k = 0$ to S **do**
  Sample v from the prior distribution given in (3)
  Set $\hat{u}^{(k)} = (1 - \beta^2)^{1/2} u^{(k)} + \beta v$
  Set $\alpha(u^{(k)} \to \hat{u}^{(k)}) = \min(1, \exp(\Phi(u^{(k)}) - \Phi(\hat{u}^{(k)})))$
  Set $u^{(k+1)} = \hat{u}^{(k)}$ with probability $\alpha(u^{(k)} \to \hat{u}^{(k)})$, and set $u^{(k+1)} = u^{(k)}$ otherwise
end for
return $\{u^{(k)}\}$

2.2. Hierarchical algorithms. To implement hierarchical algorithms, we need to sample conditional distributions that govern u and θ, where θ is the set of hyperparameters. θ could be (τ, α, M) or (v, M) depending on if we choose model (C) or model (E). When we attempt to sample the posterior distribution in the hierarchical methods, we could use the Gibbs sampler. The basic form of Gibbs sampling has two repeated steps:

- Update $u^{(n+1)} \sim u|\theta^{(n)}, y$
- Update $\theta^{(n+1)} \sim \theta|u^{(n+1)}, y$

We cannot sample these conditional distributions directly, so we replace direct sampling with Markov Chain Monte Carlo (MCMC) indirect sampling methods, which are invariant with respect to each of the exact conditional distributions. With Metropolis-within-Gibbs, at every step we update $u^{(n+1)}, \theta^{(n+1)}$ each with MCMC to target these conditional distributions. One method is to update u, θ in a block, by proposing $(u, \theta) \to (\hat{u}, \hat{\theta})$ and computing the transition probability for this step.
The algorithms that we will implement will be slightly different from the block update, as we will independently propose $u \to \hat{u}$ and $\theta_i \to \hat{\theta}_i$ for each hyperparamter θ_i, and perform these transitions in separate steps. At each step, we fix all but the proposed parameter, and we compute

the transition probability. This is the approach that we will use for the following hierarchical algorithms.

2.3. Model (C). In this algorithm, we aim to learn τ, α, M. This algorithm seems promising given the problem we encountered with the irregularity of higher eigenvectors with the experiments with model (B), which we resolved by truncating the number of eigenvectors. Essentially, we want to be hierarchical about our level of truncation. We take a uniform prior over the integers for M such as $\mathsf{U}[1, 70]$, and uniform priors for τ, α such as $\mathsf{U}(0.01, 60)$ and $\mathsf{U}(0.1, 90)$, respectively. It is convenient to implement this algorithm with a "non-centered" approach (see [5] for this terminology), meaning that the unknown classifying variable ξ and the parameters $\theta = (\tau, \alpha, M)$ are a priori independent. Theoretical results backed with our own experiments suggest that this non-centered approach is preferable to the centered approach of sampling u and θ, which are not independent. Benefits of the non-centered approach include the cancellation of the normalization constant in the multivariate Gaussian density function and overall better mixing of the parameters, leading to faster convergence. The new equation for T, which relates ξ, τ, α, M to u, is:

$$T(\xi, \tau, \alpha, M) = \sqrt{\frac{N}{\mathrm{Tr}((L+\tau^2)^{-\alpha})}} \sum_{i=0}^{M} \frac{1}{(\lambda_i + \tau^2)^{\alpha/2}} \xi_i q_i = u. \tag{7}$$

The $\sqrt{\frac{N}{\mathrm{Tr}((L+\tau^2)^{-\alpha})}}$ scale ensures that $\mathbb{E}(u^2) = N$, or $\mathbb{E}(u_j^2) = 1$. We can compute the joint posterior on ξ, τ, α, M, using the fact that they are independent to simplify:

$$g(\xi, \tau, \alpha, M) \propto \exp\left(-\Phi(T(\xi, \tau, \alpha, M)) - \frac{1}{2}\langle \xi, \xi \rangle + \log(\pi_0(\tau, \alpha, M))\right). \tag{8}$$

This algorithm is given in Algorithm 2.

2.4. Model (E). This algorithm reparameterizes the problem in terms of the random vectors v and ξ. v_j modifies the scale of influence of q_j, the jth eigenvector of the graph Laplacian, on the classifying function u. This model aims to learn M as well. Define:

$$T(v, \xi, M) = \sqrt{\frac{N}{\mathrm{Tr}((L+\tau^2)^{-\alpha})}} \sum_{i=0}^{M} v_i \xi_i q_i = u. \tag{9}$$

Using good estimates for τ, α, perhaps obtained by algorithms that learn τ, α, set the prior on v to be:

$$v_j \sim \mathsf{U}\left((1-a)(\lambda_j + \tau^2)^{-\alpha/2}, (1+a)(\lambda_j + \tau^2)^{-\alpha/2}\right)$$

Taking this uniform prior on v and $\xi \sim \mathsf{N}(0, I)$, we obtain:

$$h(v, \xi, M) \propto \exp\left(-\Phi(T(v, \xi, M)) + \log(\pi_0(v, M)) - \frac{1}{2}\langle \xi, \xi \rangle\right). \tag{10}$$

This approach is described in Algorithm 3.

2.5. Model (F). We describe the algorithm to solve the multiclass clustering problem. Recall that for $v \in \mathbb{R}^k$, we define $S(v) = e_p, \quad p = \mathrm{argmax}(v_r)$ where $\{e_1, \dots e_k\}$ is the standard basis of $\mathbb{R}^k$. Now, the classifying function is of the form $u : Z \to \mathbb{R}^k$, or $u = (u^{[1]}, u^{[2]}, \dots u^{[k]})$ with functions $u^{[j]} : Z \to \mathbb{R}$. Each of the $u^{[j]}$ has prior distribution given by $\mathsf{N}(0, C)$ with $C = (L + \tau^2 I)^{-\alpha}$. We retain

$$\Phi(u) = \frac{1}{2\gamma^2} \sum_{l \in Z'} \mathrm{norm}(y(l) - S(u(l)))^2.$$

Algorithm 2 Non-centered parameterization, hierarchical with τ, α, M

Choose $\xi^{(0)} \in \mathbb{R}^N, \tau^{(0)}, \alpha^{(0)}, M^{(0)} > 0, \beta \in (0, 1]$ and $\epsilon_1, \epsilon_2 > 0$.
for $k = 0$ to S **do**
  Propose $\hat{\xi}^{(k)} = (1-\beta^2)^{\frac{1}{2}}\xi^{(k)} + \beta\zeta^{(k)}$, $\zeta^{(k)} \sim \mathsf{N}(0, I)$
  Make transition $\xi^{(k)} \to \hat{\xi}^{(k)}$ with probability

$$A(\xi^{(k)} \to \hat{\xi}^{(k)}) = \min\left\{1, \exp\left(\Phi(T(\xi^{(k)}, \tau^{(k)}, \alpha^{(k)}, M^{(k)})) - \Phi(T(\hat{\xi}^{(k)}, \tau^{(k)}, \alpha^{(k)}, M^{(k)}))\right)\right\}$$

▷ T defined in (7)

  Propose $\hat{\tau}^{(k)} = \tau^{(k)} + \epsilon_1\rho^{(k)}, \rho^{(k)} \sim \mathsf{N}(0, 1)$
  Make transition $\tau^{(k)} \to \hat{\tau}^{(k)}$ with probability

$$A(\tau^{(k)} \to \hat{\tau}^{(k)}) = \min\left\{1, \frac{g(\xi^{(k+1)}, \hat{\tau}^{(k)}, \alpha^{(k)}, M^{(k)})}{g(\xi^{(k+1)}, \tau^{(k)}, \alpha^{(k)}, M^{(k)})}\right\}$$

▷ g defined in (8)

  Propose $\hat{\alpha}^{(k)} = \alpha^{(k)} + \epsilon_2\sigma^{(k)}, \sigma^{(k)} \sim \mathsf{N}(0, 1)$
  Make transition $\alpha^{(k)} \to \hat{\alpha}^{(k)}$ with probability

$$A(\alpha^{(k)} \to \hat{\alpha}^{(k)}) = \min\left\{1, \frac{g(\xi^{(k+1)}, \tau^{(k+1)}, \hat{\alpha}^{(k)}, M^{(k)})}{g(\xi^{(k+1)}, \tau^{(k+1)}, \alpha^{(k)}, M^{(k)})}\right\}$$

  Propose $\hat{M}^{(k)} = M^{(k)} + Q$, with jump Q distributed as $\mathbb{P}(Q = k) \propto \frac{1}{1+|k|}$, $|Q|$ bounded.
  Make transition $M^{(k)} \to \hat{M}^{(k)}$ with probability

$$A(M^{(k)} \to \hat{M}^{(k)}) = \min\left\{1, \frac{g(\xi^{(k+1)}, \tau^{(k+1)}, \alpha^{(k+1)}, \hat{M}^{(k)})}{g(\xi^{(k+1)}, \tau^{(k+1)}, \alpha^{(k+1)}, M^{(k)})}\right\}$$

end for
return $\{T(\xi^{(k)}, \tau^{(k)}, \alpha^{(k)}, M^{(k)}), \tau^{(k)}, \alpha^k\}$

We will implement the non-centered approach. Here, the variable is $\xi = (\xi^{[1]}, \dots \xi^{[k]})$ where each $\xi^{[j]}$ is associated with $u^{[j]}$ by the following:

$$T(\xi^{[j]}) = \sum_{i=1}^{M} \frac{1}{(\lambda_j + \tau^2)^{\alpha/2}}\xi_i^{[j]}q_i = u^{[j]}.$$

Then, define:

$$T(\xi) = (T(\xi^{[1]}), \dots T(\xi^{[k]})) = u. \tag{11}$$

Refer to Algorithm 4.
We can extend this algorithm to be hierarchical with τ, α, M as in the binary case. Define T as

$$T(\xi^{[j]}, \tau, \alpha, M) = \sum_{i=1}^{M} \frac{1}{(\lambda_j + \tau^2)^{\alpha/2}}\xi_i^{[j]}q_i = u^{[j]}.$$

One choice we have to make is whether there should be one universal copy of τ, α, M for all of the k copies of the prior, or if each copy should have its own independently-evolving hyperparameters.

3. Comparing Models (C) and (E). Here, we compare models (C) and (E) on the binary classification problem MNIST49. We set $\gamma = 0.0001$ for the label noise, which seems to improve accuracy for both models. This is reasonable because the labeling of digits should have little error, except when some digits are poor representatives of their class.

Algorithm 3 Non-centered parameterization, hierarchical with v, M

Choose $v^{(0)}, \xi^{(0)} \in \mathbb{R}^N, M^{(0)}, \beta \in (0,1], \epsilon > 0$.
for $k = 0$ to S **do**
  Propose $\hat{\xi}^{(k)} = (1-\beta^2)^{\frac{1}{2}}\xi^{(k)} + \beta\zeta^{(k)}$, $\zeta^{(k)} \sim \mathsf{N}(0, I)$
  Make transition $\xi^{(k)} \to \hat{\xi}^{(k)}$ with probability

$$A(\xi^{(k)} \to \hat{\xi}^{(k)}) = \min\left\{1, \exp\left(\Phi(T(v^{(k)}, \xi^{(k)}, M^{(k)})) - \Phi(T(v^{(k)}, \hat{\xi}^{(k)}, M^{(k)}))\right)\right\}$$

  Propose $\hat{v}^{(k)} = v^{(k)} + \epsilon\rho^{(k)}, \rho^{(k)} \sim \mathsf{N}(0, I)$
  if $\hat{v}_j^{(k)}$ is outside of its prior interval for any j **then**
    Reject and set $v^{(k+1)} = v^{(k)}$.
  else
    Make transition $v^{(k)} \to \hat{v}^{(k)}$ with probability

$$\begin{aligned} A(v^{(k)} \to \hat{v}^{(k)}) &= \min\left\{1, \frac{h(\hat{v}^{(k)}, \xi^{(k+1)}, M^{(k)})}{h(v^{(k)}, \xi^{(k+1)}, M^{(k)})}\right\} \\ &= \min\left\{1, \exp\left(\Phi(T(v^{(k)}, \xi^{(k+1)}, M^{(k)})) - \Phi(T(\hat{v}^{(k)}, \xi^{(k+1)}, M^{(k)}))\right)\right\} \end{aligned}$$

  end if
  Propose $\hat{M}^{(k)} = M^{(k)} + Q$, with jump Q distributed as $\mathbb{P}(Q = k) \propto \frac{1}{1+|k|}$, $|Q|$ bounded.
  Make transition $M^{(k)} \to \hat{M}^{(k)}$ with probability

$$\begin{aligned} A(M^{(k)} \to \hat{M}^{(k)}) &= \min\left\{1, \frac{h(v^{(k+1)}, \xi^{(k+1)}, \hat{M}^{(k)})}{h(v^{(k+1)}, \xi^{(k+1)}, M^{(k)})}\right\} \\ &= \min\left\{1, \exp\left(\Phi(v^{(k+1)}, \xi^{(k+1)}, M^{(k)}) - \Phi(v^{(k+1)}, \xi^{(k+1)}, \hat{M}^{(k)})\right)\right\} \end{aligned}$$

end for
return $\{T(v^{(k)}, \xi^{(k)}), v^{(k)}, \xi^{(k)}\}$

3.1. Model (C). We fix $M = 50$ and allow τ, α to be learnt from uniform priors $[0.01, 20]$ and $[0.1, 60]$, respectively. The results are shown in Figure 5. τ is initialized at 20 but finds a small value at around step 20000 and stays near there. Note that this corresponds with the sharp increase in classification accuracy after this value of τ was found. The mean and median of τ after it seems to converge is around 0.7. The accuracy is around 96%.

3.2. Model (E). Fix $M = 50$ again. With $\tau = 0.7$, which is "cheating" by using the τ learnt from model (C), we obtain Figure 6. Note the similar final accuracy of around 96%, with faster convergence to that accuracy since we cheated with the initialization.
With $\tau = 0.3$, we obtain Figure 7. Note the slow convergence compared to $\tau = 0.7$. In fact, it appears that the accuracy is still climbing even at 100000 iterations.

3.3. Remarks. These experiments suggest that model (E) is sensitive to the initial conditions of τ, α, while model (C) is able to effectively learn these hyperparameters. In addition, using the values of τ, α suggested by model (C) to calibrate model (E) does not seem to improve either the convergence rate or the classification accuracy. These factors suggest that model (C) is a better formulation for the clustering problem.

4. Learning A Single Hyperparameter. As the previous section suggests, we will focus on model (C) and the hyperparameters (τ, α, M). This section summarizes the observations on learning τ, α, M independently and explains why these parameters seem to be important to learn. In these experiments, we remove the zero eigenvalue and its corresponding constant eigenvector.

Algorithm 4 Multiclass, Metropolis-within-Gibbs updates

Choose $\xi^{(0)} = (\xi^{[1],(0)}, \dots \xi^{[k],(0)}), \xi^{[j],(0)} \in \mathbb{R}^M$. Choose $\tau, \alpha, \beta \in (0, 1]$. ▷ We refer to $\xi^{[j],(i)}$ as the ith observation of $\xi^{[j]}$.
for $i = 0$ to S **do**
  for $j = 1$ to k **do**
    Propose $\hat{\xi}^{[j],(i)} = (1 - \beta^2)^{\frac{1}{2}} \xi^{[j],(i)} + \beta \zeta^{(i)}$, $\zeta^{(i)} \sim \mathsf{N}(0, I)$
    Define $\hat{\xi} = (\xi^{[1],(i+1)}, \dots \xi^{[j-1],(i+1)}, \hat{\xi}^{[j],(i)}, \xi^{[j+1],(i)}, \dots)$
    Make transition $\xi^{(i)} \to \hat{\xi}$ with probability

$$A(\xi^{(i)} \to \hat{\xi}) = \min \left\{ 1, \exp \left(\Phi(T(\xi^{(i)})) - \Phi(T(\hat{\xi})) \right) \right\}$$

  end for
end for
return $\{T(\xi^{(i)})\}$

FIG. 5. *Model (C) on MNIST49. Figures from top to bottom, left to right: Final classification obtained, ξ running acceptance probability, final average of u_j, τ trace, α trace, running classification accuracy (updated every 2500 trials).*

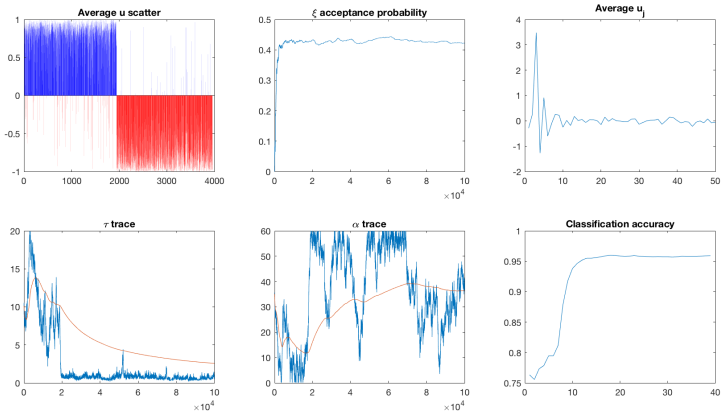


FIG. 6. *Model (E) on MNIST49 with $\tau = 0.7, \alpha = 35, a = 0.8$. Figures from left to right, top to bottom: Final classification obtained, ξ running acceptance probability, v acceptance probability, final v_j observation, final average of u_j, running classification accuracy (updated every 2500 trials).*

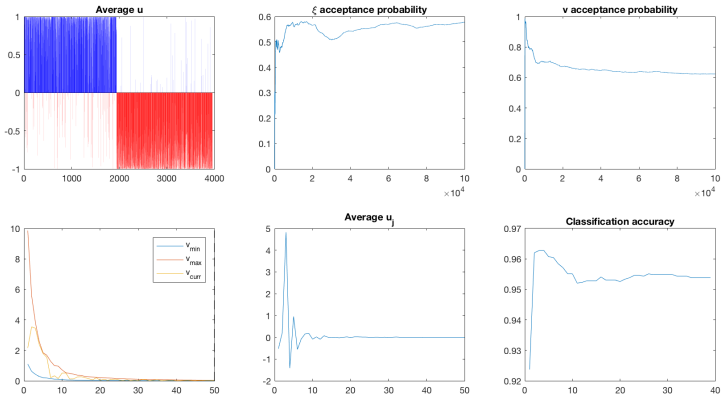
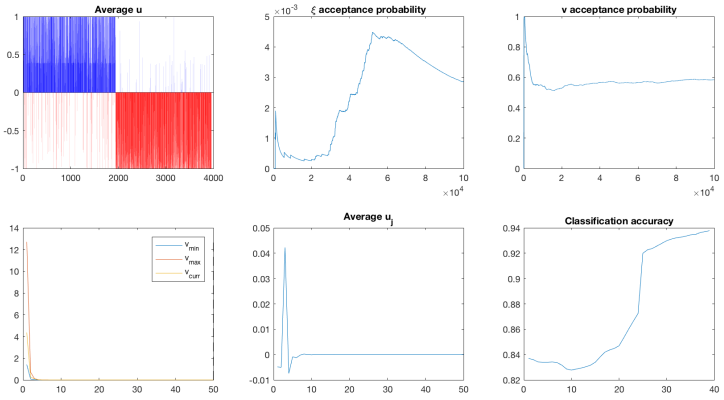
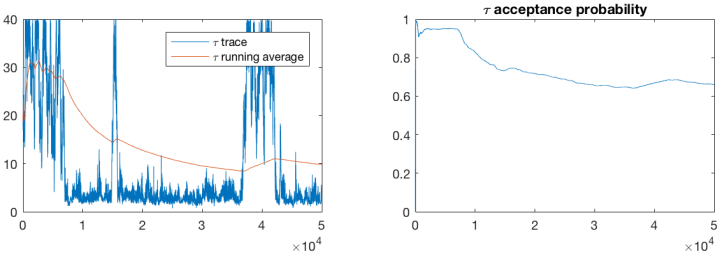


FIG. 7. *Model (E) on MNIST49 with* $\tau = 0.3, \alpha = 35, a = 0.8$.



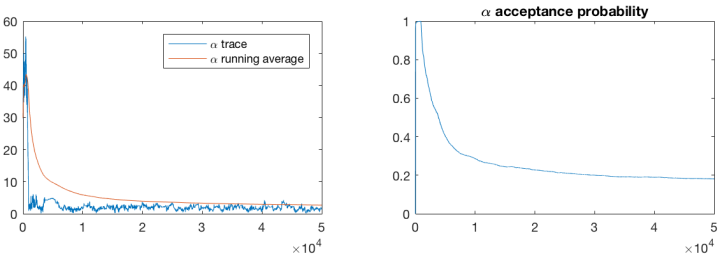
4.1. Learning τ. τ has proven to be relatively easy to sample, either alone or in addition to learning α. Testing this model on the voting records data set, with $\alpha = 30, M = 435$, and a uniform prior $\mathsf{U}[0, 40]$ for τ, results in the trace of τ shown in Figure 8. $M = 435$ is using all of the eigenvectors, and the eigenvectors with larger eigenvalues tend to be noisier. Hence, τ concentrates on a relatively small value to reduce the effects of these higher eigenvectors.

FIG. 8. *Trace of* τ *on voting records, 3% fidelity and label noise* $\gamma = 0.0001$.



4.2. Learning α. α controls the rate of decrease of the weights of the eigenvectors. α behaves similar to τ when $\tau = 0$ is fixed. Consider the experiment shown in Figure 9 on the two moons dataset with $\tau = 0, M = 50$, and a uniform prior $\mathsf{U}[0.1, 60]$ on α. Here, α walks until it is small enough so that the first few eigenvectors that are needed for the classification is of similar scale, but remains large enough so that the unnecessary eigenvectors are relatively small.

FIG. 9. *Trace of* α *on two moons* $N = 2000, \sigma = 0.15$*, 1% fidelity and label noise* $\gamma = 0.15$.



4.3. Learning M. M controls the number of eigenvectors used. In a way, small M plays a similar role as large α in truncating the eigenvectors, but M and α could be used together to control which eigenvectors to include and the relative scale of these selected eigenvectors. We

observed an interesting relation between M and the noise in the data. When the data is noisier, we find that M tends to converge to a larger value as well, since more eigenvectors are needed to explain the clustering. We demonstrate this relation in an experiment using the two moons dataset, since we can directly control the noise of the synthetic data. We fix $\tau = 0, \alpha = 1$ and learn M on the interval $\mathsf{U}[1, 70]$. Compare Figure 10 with Figure 11 and note the higher average value of M when $\sigma = 0.20$. A pronounced minimum value of M is shown in Figure 10, suggesting that an eigenvector around that index is important for classification.

FIG. 10. *Trace of M on two moons $N = 2000, \sigma = 0.15$, 1% fidelity and $\gamma = 0.15$.*

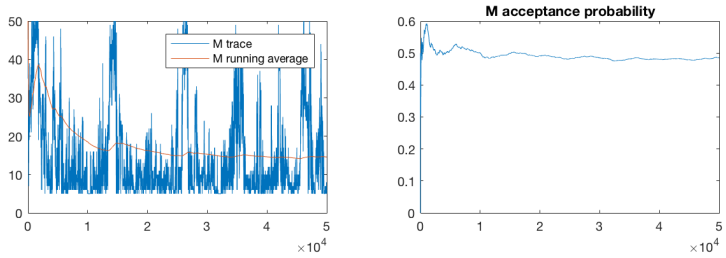
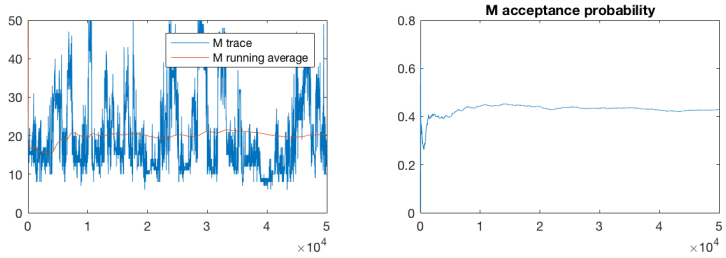
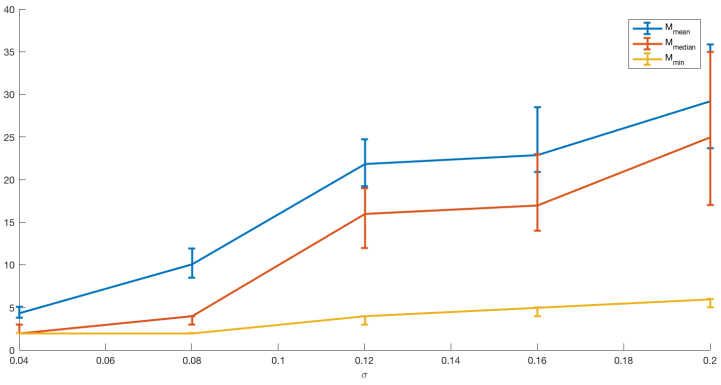


FIG. 11. *Trace of M on two moons $N = 2000, \sigma = 0.20$, 1% fidelity and $\gamma = 0.20$.*



Performing the same experiment over a larger number of trials and noise levels, we obtain the graph shown in Figure 12. Again, the same direct relation between M and σ is observed.

FIG. 12. *Statistics of M compiled over 50 trials on the two moons dataset. The plotted statistics are the mean, median, and minimum of M. Error bars indicate 25 and 75-th quantiles of the statistic.*



5. Learning Pairs of Hyperparameters. Having established our beliefs in the importance of the hyperparameters (τ, α, M), we turn our attention to selecting the optimal subset of these hyperparameters to learn. For these experiments, we focus on multiclass, using the MNIST dataset.

5.1. Learning (τ, α). We had promising results learning (τ, α) for the binary problems of voting records and two moons. In those experiments, we observed a consistent learning of a small τ, and we wished to see if this would hold in multiclass. We find that in MNIST, we can indeed learn a small value for τ, shown in Figure 13.
As the labels in MNIST should have no noise (since an image with label 4 will always be a picture of a 4), it is natural to consider the $\gamma \to 0$ limit. This limit is considered in Figure 14, with 10% fidelity and $\epsilon_\tau = 0.1, \epsilon_\alpha = 3$, and $\gamma = 10^{-7}$. However, the mixing of τ is poor with an acceptance probability of around 9%.

FIG. 13. *Learning small τ on MNIST* [1, 4, 9].

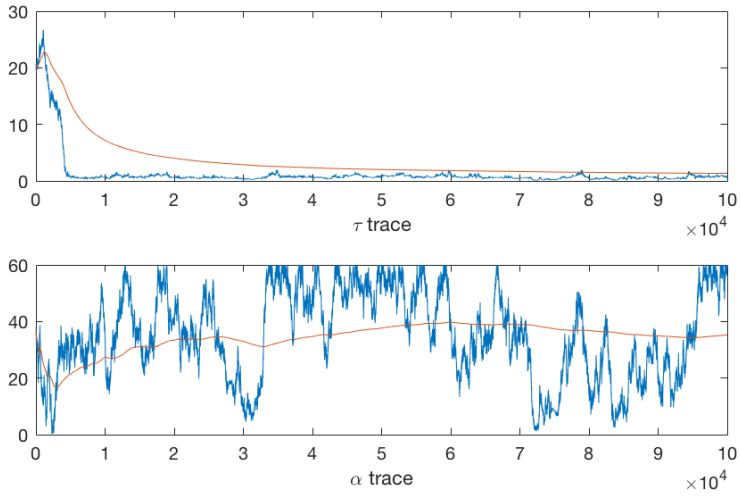
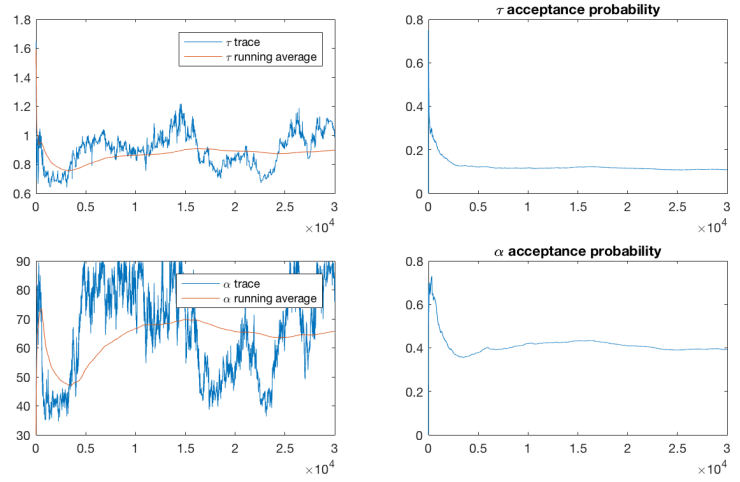


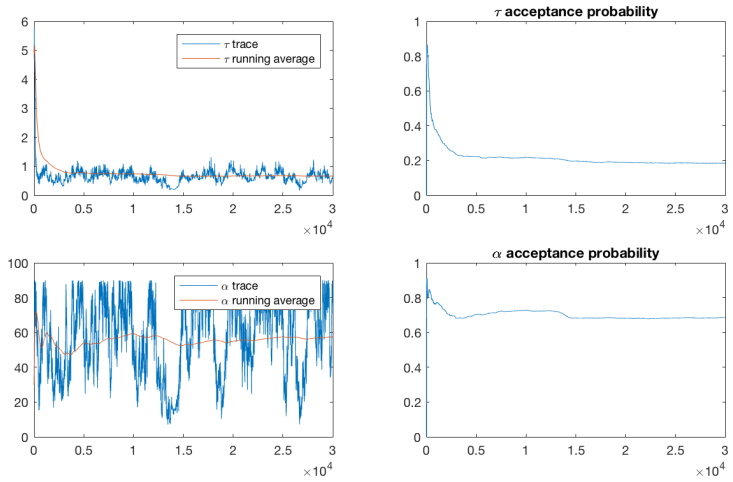
FIG. 14. *Traces of τ, α on MNIST* [1, 4, 9], 10% *fidelity and label noise* $\gamma = 10^{-7}$.



These experiments suggest that we may need larger γ to accommodate a higher fidelity. Taking $\gamma = 0.1$ still resulted in acceptance probabilities under 10%, but $\gamma = 0.9$ resulted in better mixing. This choice is shown in Figure 15. τ is accepted 20% of the time even with a larger jump of $\epsilon_\tau = 0.2$, while the same classification accuracy of around 97% is achieved. Even with initialization at $\tau = 5$, a small value of τ is found with good mixing.
Finally, notice the correlation between τ and α, for example in Figure 14 and Figure 15. There appears to be a direct relation between τ and α. This behavior is in line with our beliefs about

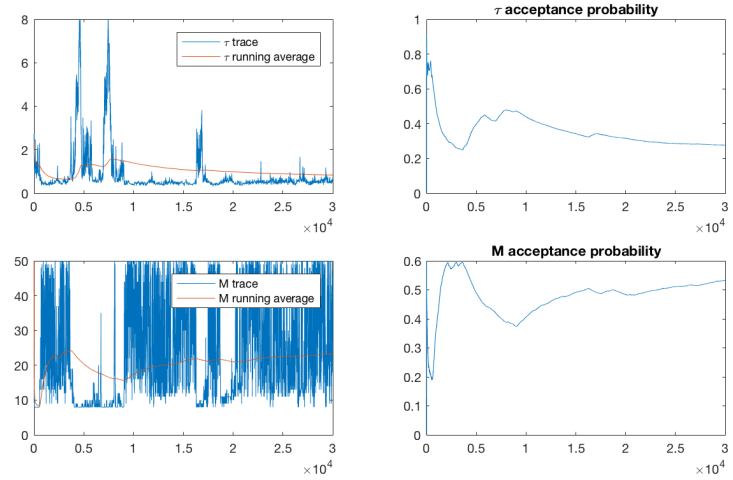
the effects of τ and α: when τ is larger, the values of $(\lambda+\tau^2)$ are relatively closer, and so a larger α is needed to distinguish between these values.

FIG. 15. *Traces of τ, α on MNIST* $[1, 4, 9]$*, 10% fidelity and label noise* $\gamma = 0.9$.



5.2. Learning (τ, M). Again with MNIST $[1, 4, 9]$, we learn τ and M with fixed α. As above, we set $\gamma = 0.9$ to allow for 10% fidelity. We first fix $\alpha = 30$ in the experiment shown in Figure 16. Note that when τ is large, M drops to around 10. An explanation for this relation is that large τ causes $(\lambda+\tau^2)^{-\alpha/2}$ to appear the same, so all eigenvectors are assigned equal weight in the prior. Then, M must be small to truncate the unnecessary eigenvectors. Similarly, when τ is small, M could be larger since the unnecessary eigenvectors already have small weight.

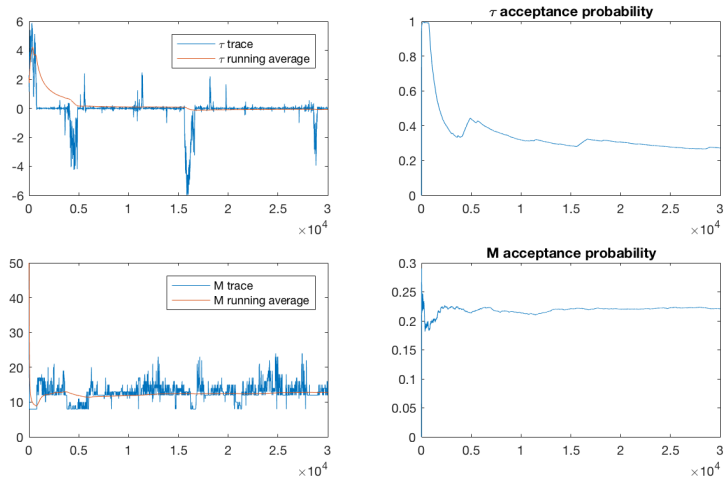
FIG. 16. *Traces of τ, M on MNIST* $[1, 4, 9]$*, $\alpha = 30$, 10% fidelity and label noise* $\gamma = 0.9$.



In Figure 17, with $\alpha = 1$, we see that M and τ are both learned with low variance. τ prefers to be close to 0, which makes sense since α is so small that a very small value of τ is needed to give weight to the lowest eigenvectors. M also does not fluctuate as high as it did in $\alpha = 30$, since α being small allows the higher eigenvectors to remain in the prior, so M is used to remove them.

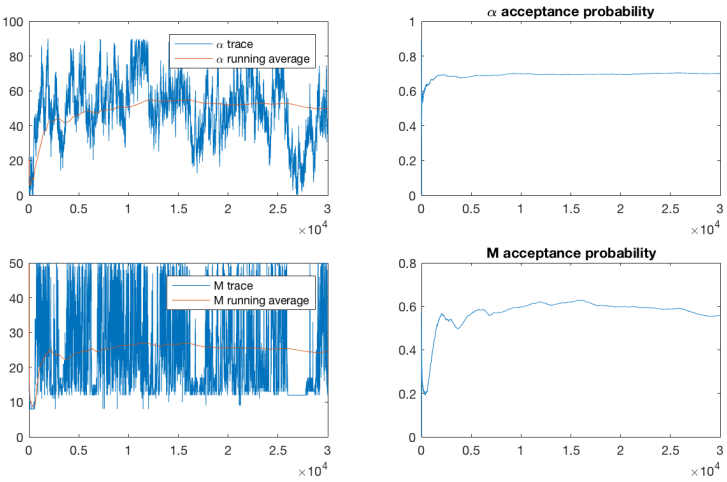
5.3. Learning (α, M). The last pair is to fix τ and learn (α, M). With $\tau = 0.7$ and $\gamma = 0.9$ to allow for 10% fidelity, we attempt to learn α from the prior $\mathsf{U}[0.1, 90]$ and M from the prior

FIG. 17. *Traces of τ, M on MNIST $[1, 4, 9]$, $\alpha = 1$, 10% fidelity and label noise $\gamma = 0.9$.*



$\mathsf{U}[1, 70]$. The results are shown in Figure 18, and the behavior of α with M appears to be inversely related as well. When α is small, M needs to be smaller to truncate the unnecessary eigenvectors.

FIG. 18. *Traces of α, M on MNIST $[1, 4, 9]$, $\tau = 0.7$, 10% fidelity and label noise $\gamma = 0.9$.*



5.4. Remarks. In these experiments, we observe that γ, the label noise parameter, is very important for both classification accuracy and convergence rate. With a better choice of γ, the algorithm learning (τ, α) exhibited better mixing and a better classification accuracy is attained. This choice of γ depends on the fidelity.
In addition, reasonable solutions for hyperparameter values are learned for each of the three pairs of hyperparameters, suggesting that these algorithms may all have their own merit.

6. Learning All Three Hyperparameters. In experiments on the two moons dataset, we find that learning all the hyperparameters together, (τ, α, M), gave results that were worse in both classification accuracy and convergence rate.

7. Adaptive β for Multiclass. In the multiclass algorithm, when learning the k independent copies of the classifying function, we observed that different choices of β for each copy is

sometimes necessary to achieve healthy acceptance probabilities in the pCN proposal. For example, take $\beta = 0.1, \tau = 0, \alpha = 1, \gamma = 1$ on the dataset MNIST [1, 3, 4, 9]. We obtain the running acceptance probability for the 4 different ξ copies shown in Figure 19. The acceptance probabilities of 2 of the copies approach 0.7, while the other two linger around 0.4. One solution for this problem is to update the values of B according to the observed running acceptance probabilities for each of the k copies. First, choose p, the target acceptance probability for each of the k copies. Empirically, we can observe that $p \approx 0.5$ allows for fast convergence. With p fixed, we make updates to each β every D samples. During the update, for each of the k copies we compute p', the acceptance probability of ξ over the last D samples, and we set $\beta \to \min(\beta(1 + p' - p), 1)$. This means that if $p' > p$, we increase β because we wish to lower the acceptance probability. Similarly, if $p' < p$, we decrease β to increase the acceptance probability. We perform these updates j times, after which the different β values are fixed for the rest of the computation. The results of adjusting β is shown in Figure 20. Notice that the acceptance probabilities for the 4 different copies are now very similar and around 0.5.

FIG. 19. *ξ acceptance probability without adaptive β updates, MNIST* [1, 3, 4, 9].

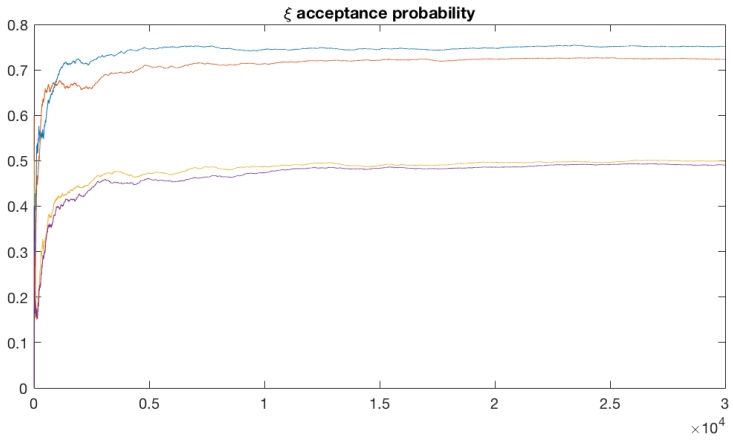
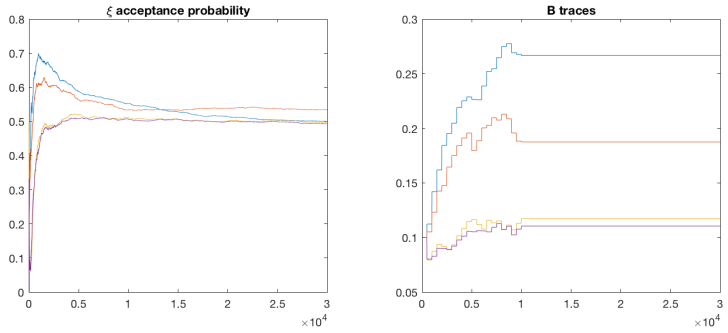


FIG. 20. *ξ acceptance probability and B traces with adaptive β updates, MNIST* [1, 3, 4, 9]. *The β updates are turned off after* 10000 *samples.*



8. Comparing Hyperparameter Choices. Having established that learning pairs or even all three hyperparameters could all be reasonable approaches, we compared the classification accuracy and convergence rate of these methods in experiments. We needed to benchmark these algorithms against a nonhierarchical algorithm to see if there are merits to a hierarchical approach. We used the MNIST dataset with digits [3, 4, 5, 9] and $N = 4854$ digits total. We fixed the fidelity to be 1% and $\gamma = 0.0001$, since the fidelity is low. We ran a number of trials, randomly selecting a subset of digits to label for each trial and testing each algorithm with that labeled subset. We incorporated the tuning of β into each algorithm since we believe that it

should help with convergence rate. For each algorithm, we will additionally throw away the zero eigenvalue and its corresponding eigenvector. We took 25000 samples with a burn-in period of 2000, and calculated the classification accuracy from the running average of $S(u)$ every 5000 samples. The priors for τ, α, M were $\mathsf{U}[-6,6], \mathsf{U}[1,110]$, and $\mathsf{U}[1,80]$, respectively. The details of the different parameter choices for the different algorithms follow:
The nonhierarchical algorithm will fix $\tau = 0, \alpha = 1, M = 50$.
Learning (τ, α), we fix $M = 50$. We set $\epsilon = 0.2$ for τ and $\epsilon = 5$ for α.
Learning (τ, M), we fix $\alpha = 60$. We set $\epsilon = 0.2$ for τ and $|Q| \leq 15$ for M.
Learning (α, M), we fix $\tau = 1$. We set $\epsilon = 5$ for α and $|Q| \leq 15$ for M.
Learning (τ, α, M), we set $\epsilon = 0.2$ for τ, $\epsilon = 5$ for α and $|Q| \leq 15$ for M.
The results are shown ???. Plotted is the median of the classification accuracies of the algorithm against the sample number.

9. Acknowledgments. This SURF is funded by The Aerospace Corporation.

REFERENCES

[1] A. L. Bertozzi, X. Luo, A. M. Stuart, and K. C. Zygalakis, *Uncertainty quantification in the classification of high dimensional data*, ArXiv e-prints, (2017), https://arxiv.org/abs/1703.08816.
[2] A. Beskos, G. O. Roberts, A. M. Stuart, and J. Voss, *MCMC methods for diffusion bridges*, Stochastics and Dynamics, 8 (2008), pp. 319–350.
[3] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White, *MCMC methods for functions: modifying old algorithms to make them faster*, Statistical Science, 28 (2013), pp. 424–446.
[4] M. Iglesias, Y. Lu, and A. M. Stuart, *A Bayesian level set method for geometric inverse problems*, Interfaces and Free Boundaries, 18 (2016), pp. 181–217.
[5] O. Papaspiliopoulos, G. O. Roberts, and M. Sköld, *A general framework for the parametrization of hierarchical models*, Statistical Science, (2007), pp. 59–73.
[6] U. von Luxburg, *A tutorial on spectral clustering*, Statistics and Computing, 17 (2007), pp. 395–416.
[7] L. Zelnik-Manor and P. Perona, *Self-tuning spectral clustering*, in Advances in neural information processing systems, 2005, pp. 1601–1608.