# SURF 2017 INTERIM REPORT 2

V. CHEN
ADVISED BY A. M. STUART AND M. M. DUNLOP
THIS SURF IS FUNDED BY THE AEROSPACE CORPORATION

**1. Introduction.** Over the past month, I have been implementing different hierarchical algorithms for the semi-supervised clustering problem. In the Bayesian formulation, recall the prior distribution,

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

as well as the general form of the prior,

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

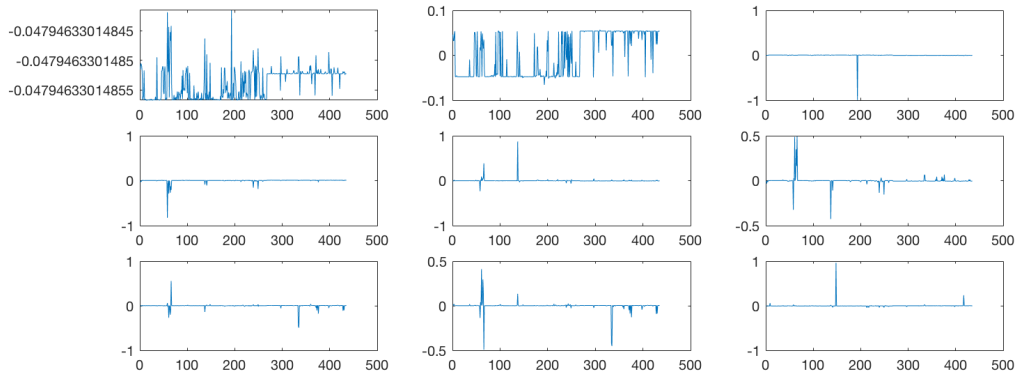
The models to be considered for this project evolve as follows:
For models (A), (B), and (C), the hyperparameters $\tau, \alpha, M$ refer to the prior in (1)
(A) Use fixed $\tau, \alpha, M = N - 1$ to cluster data.
(B) Learn $\tau, \alpha$; fix $M = N - 1$.
(C) Learn $\tau, \alpha, M$.
For models (D) 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 $f(\lambda), M$.
So far, I have implemented models (A) through (D). The past month has been devoted to experiments testing individual models and comparing their performances, using metrics such as classification accuracy and variance. Classification accuracy is well defined for the model problems that I tested the data on. These problems are outlined in section 2.

**2. Model problems.** Applications to be considered include the following:

**2.1. Voting records.** A data set of the voting records on 16 votes of 435 individuals from the U.S. House of Representatives. 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 $w_{ij} = \exp\left(-\frac{d(x_i,x_j)}{2l^2}\right)$, $d_{p,q}(x,y) = ||x-y||_p^q$ with $p = 2, q = 2, l = 1$. Figure 1 is part of the spectrum of this choice of Laplacian.

**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

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




be using the self-tuning, symmetric Laplacian for this data set, introduced in [5]. 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,* $\sigma = 0.04$.

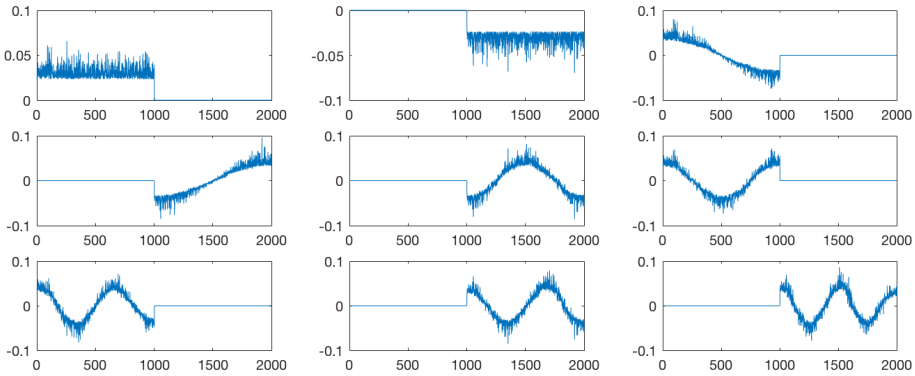
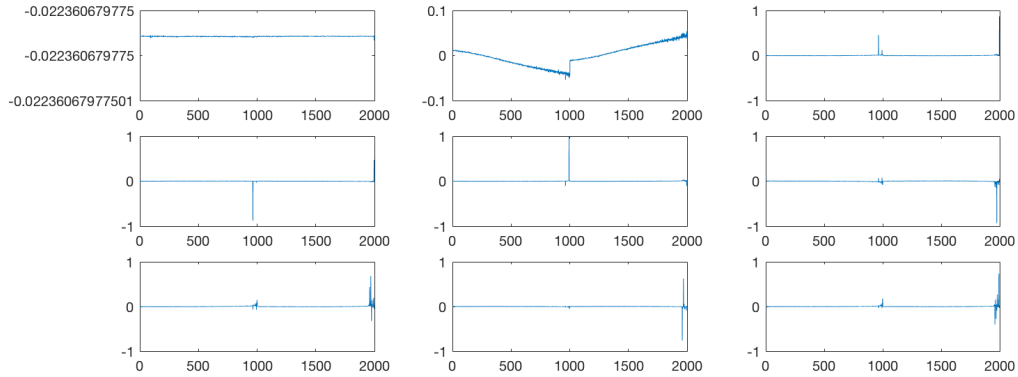
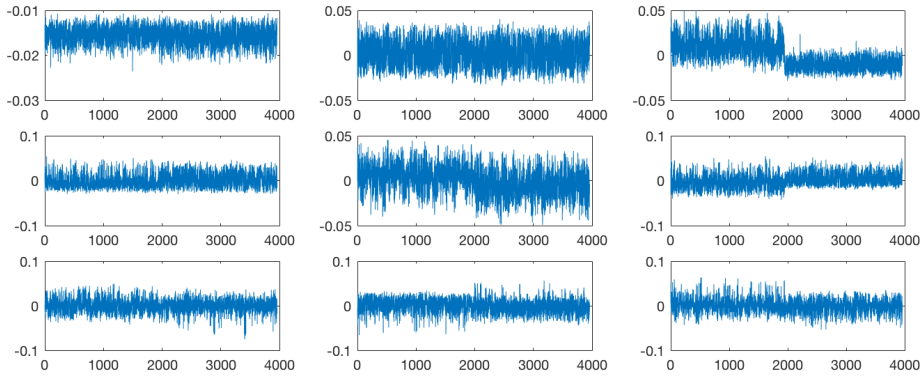


FIG. 3. *Lowest* 9 *eigenvectors of unnormalized fixed length* $L$ *of two moons,* $\sigma = 0.04$.



**2.3. MNIST data sets.** This data set contains 70,000 images of $28 \times 28$ 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 will focus on binary classification between the digits 4 and 9. We will use the self-tuning, symmetric normalized Laplacian again. See Figure 4.

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



**3. Algorithms.** This section discusses the algorithms that I have implemented for the hierarchical models.

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

**Algorithm 1** General pCN adapted from [3]

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

**3.2. Model (B).** To implement model (B), which is hierarchical, we need to sample conditional distributions that govern $u, \tau$, and $\alpha$. 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 use $\theta$ to represent the hyperparameters, which could include $\tau, \alpha$, and $M$. However, we cannot sample the 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 $u^{(n+1)}, \tau^{(n+1)}$ and $\alpha^{(n+1)}$ each with MCMC to target these conditional distributions. One method is to update $u, \tau, \alpha$ 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, as we will independently propose $u \to \hat{u}, \tau \to \hat{\tau}, \alpha \to \hat{\alpha}, M \to \hat{M}$, 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 algorithm that we will be using for the following hierarchical algorithms.

Our first of two hierarchical algorithms for model (B) is deemed "centered," as contrasted with the second algorithm described later in this section. Using Metropolis-within-Gibbs, we will require an expression for the posterior, so define $f(u, \tau, \alpha)$ as the joint posterior distribution. By Bayes' theorem,

$$f(u, \tau, \alpha) \propto \exp(-\Phi(u)) \times \mathbb{P}(u, \tau, \alpha) = \exp(-\Phi(u))\mathbb{P}(u|\theta)\pi_0(\theta).$$

Recall from (1) that the prior is distributed as $\mathsf{N}(0, C)$ with the covariance matrix $C(\theta) = C(\tau, \alpha) = (L + \tau^2 I)^{-\alpha}$. We can write

$$f(u, \tau, \alpha) \propto \frac{1}{\sqrt{(2\pi)^d \det C(\tau, \alpha)}} \exp\left(-\Phi(u) - \frac{1}{2}\langle u, C(\tau, \alpha)^{-1}u\rangle + \log(\pi_0(\tau, \alpha))\right). \tag{3}$$

The normalization constant $\det(C(\theta))$ depends on $\tau, \alpha$ now and does not cancel out. Using this expression for the posterior, I implemented the Metropolis-within-Gibbs method for model (B) in Algorithm 2.

**Algorithm 2** Hierarchical on $\tau, \alpha$

Initialize $u^{(0)} = q_1$, the Fiedler vector expressed in the standard basis.
Initialize $\tau^{(0)}, \alpha^{(0)}$. Select $\beta \in [0, 1]$
Pick $\epsilon_1, \epsilon_2$, the jump sizes for $\tau, \alpha$ respectively.
**for** $k = 0$ to $S$ **do**
    Sample $v$ from the prior distribution and expressed in the eigenbasis ▷ $u|y, \tau, \alpha$.
    Expressing $u$ in the eigenbasis, set a proposal $\hat{u}^{(k)} = (1 - \beta^2)^{1/2}u^{(k)} + \beta v$
    Set $u^{(k+1)} = \hat{u}^{(k)}$ with probability

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

    Set a proposal $\hat{\tau}^{(k)} = \tau^{(k)} + \epsilon_1 t$ for $t \sim \mathsf{N}(0, 1)$ ▷ $\tau|y, u, \alpha$
    Set $\tau^{(k+1)} = \hat{\tau}^{(k)}$ with probability

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

    (Using the eigenbasis representation of $u$ for computation) ▷ $f$ defined in (3)
    Set a proposal $\hat{\alpha}^{(k)} = \alpha^{(k)} + \epsilon_2 a$ for $a \sim \mathsf{N}(0, 1)$ ▷ $\alpha|y, u, \tau$
    Set $\alpha^{(k+1)} = \hat{\alpha}^{(k)}$ with probability

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

**end for**
**return** $\{u^{(k)}, \tau^{(k)}, \alpha^{(k)}\}$

We also looked at a different parameterization for model (B). Algorithm 3 uses the variable $\xi$ that is related to the classifying function $u$ by

$$T(\xi, \tau, \alpha) = \sum_{i=0}^{M} \frac{1}{(\lambda_i + \tau^2)^{\alpha/2}}\xi_i q_i = u \tag{4}$$

following (1). This algorithm is "non-centered" compared to the centered parameterization given in Algorithm 2, meaning that the unknown classifying variable $\xi$ and the parameters $\theta = (\tau, \alpha)$ are a priori independent [4].

This model assumes the prior $\xi \sim \mathsf{N}(0, I)$. Similar to the centered case, define $g(\xi, \tau, \alpha)$ as the joint posterior distribution. By Bayes' theorem,

$$g(\xi, \tau, \alpha) \propto \exp(-\Phi(T(\xi, \tau, \alpha))) \times \mathbb{P}(\xi, \tau, \alpha) = \exp(-\Phi(T(\xi, \tau, \alpha)))\mathbb{P}(\xi)\pi_0(\tau, \alpha).$$

Note that $\mathbb{P}(\xi) = \frac{1}{\sqrt{(2\pi)^d \det I}} \exp(-\frac{1}{2}\langle \xi, I\xi \rangle) \propto \exp(-\frac{1}{2}\langle \xi, \xi \rangle)$ and the normalization constants drop out. We obtain:

$$g(\xi, \tau, \alpha) \propto \exp\left(-\Phi(T(\xi, \tau, \alpha)) - \frac{1}{2}\langle \xi, \xi \rangle + \log(\pi_0(\tau, \alpha))\right). \tag{5}$$

**Algorithm 3** Non-centered parameterization: sampling $\xi, \tau, \alpha$

Choose $\xi^{(0)} \in \mathbb{R}^N, \alpha^{(0)}, \tau^{(0)} > 0, \beta \in (0, 1]$ 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

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

▷ $T$ defined in (4)

  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

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

▷ $g$ defined in (5)

  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

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

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

**3.3. Model (C).** In this algorithm, we aim to learn $M$, the number of eigenvectors we want to use, as well. This algorithm seems promising given the problem we encountered with the irregularity of higher eigenvectors with the algorithms in 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 for $M$, typically $\mathsf{U}(1, 70)$ or similar. It is convenient to implement this algorithm with the non-centered approach since we do not need to worry about $M$ affecting terms such as the determinant of the covariance. The new equation for $T$, which relates $\xi, \tau, \alpha, M$ to $u$, is:

$$T(\xi, \tau, \alpha, M) = \sum_{i=0}^{M} \frac{1}{(\lambda_i + \tau^2)^{\alpha/2}}\xi_i q_i = u. \tag{6}$$

We can compute the joint posterior on $\xi, \tau, \alpha, M$ similarly to the derivation of (5):

$$g(\xi, \tau, \alpha, M) \propto \exp\left(-\Phi(T(\xi, \tau, \alpha, M)) - \frac{1}{2}\langle \xi, \xi \rangle + \log(\pi_0(\tau, \alpha, M))\right). \tag{7}$$

The only modification from Algorithm 3 is to add a random walk proposal for $M$ and compute the acceptance probability as the ratio of the joint posteriors. This algorithm is given in Algorithm 4.

**Algorithm 4** Non-centered parameterization, hierarchical with $\tau, \alpha, M$

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 (6)

  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 (7)

  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** $\{T(\xi^{(k)}, \tau^{(k)}, \alpha^{(k)}, M^{(k)}), \tau^{(k)}, \alpha^k\}$

**3.4. Model (D).** This algorithm reparameterizes the problem in terms of the random vectors $v$ and $\xi$. $v_j$ modifies the scale of influence of $q_j$, the $j$th eigenvector of the graph Laplacian, on the classifying function $u$.

$$T(v, \xi) = \sum_{i=0}^{M} v_i\xi_i q_i = u. \tag{8}$$

Here, $M$ is fixed. We take $\xi \sim \mathsf{N}(0, I)$. Using good estimates for $\tau, \alpha$, perhaps obtained by algorithms that learn $\tau, \alpha$, set the prior on $v$ to be:

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

where $a$ is a fixed scalar.
Finally, we derive an expression for the posterior with Bayes' theorem.

$$\begin{aligned}\mathbb{P}(v, \xi|y) &\propto \mathbb{P}(y|v, \xi)\mathbb{P}(v, \xi) \\ &\propto \exp\left(-\Phi(T(v, \xi))\right)\mathbb{P}(v)\mathbb{P}(\xi) \\ &\propto \exp\left(-\Phi(T(v, \xi)) + \log(\pi_0(v)) - \frac{1}{2}\langle\xi, \xi\rangle\right).\end{aligned}$$

Let $h(v, \xi)$ denote the joint posterior on $v$ and $\xi$. Then,

$$h(v, \xi) \propto \exp\left(-\Phi(T(v, \xi)) + \log(\pi_0(v)) - \frac{1}{2}\langle\xi, \xi\rangle\right). \tag{9}$$

The updates for this algorithm will be done as follows:

- Update $\xi^{(k+1)}|v^{(k)}, y$ with a pCN proposal.
- Update $v^{(k+1)}|\xi^{(k+1)}, y$ with a random walk on the uniform prior for $v$.

This algorithm is given in Algorithm 5.

**Algorithm 5** Non-centered parameterization, hierarchical with $v$

Choose $v^{(0)}, \xi^{(0)} \in \mathbb{R}^N, \beta \in (0,1], \epsilon > 0$.
**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

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

  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

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

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

**3.5. Model (E).** This model extends model (D), trying to learn $M$ as well. Define:

$$T(v, \xi, M) = \sum_{i=0}^{M} v_i \xi_i q_i = u. \tag{10}$$

Taking the uniform prior on $v$, we obtain:

$$h(v, \xi, M) \propto \exp\left(-\Phi(T(v, \xi, M)) + \log(\pi_0(v, M)) - \frac{1}{2}\langle \xi, \xi \rangle\right). \tag{11}$$

This approach is described in Algorithm 6.

**4. Experiments comparing hierarchical and nonhierarchical algorithms.** Inspired by the experiment comparing the different clustering algorithms given in [1], we ran a similar set of experiments to determine if a hierarchical approach was beneficial to clustering. The clustering algorithms we want to compare are: Fiedler thresholding, nonhierarchical pCN, and two hierarchical algorithms for learning $\tau, \alpha$ as well as the classifying function $u$. For each of the four algorithms tested, 50 random realizations of the two moons data set were generated with $r = 1, N = 2000, d = 100$, and varying $\sigma$. We used the same RNG seed for the trials across different algorithms so that these 50 realizations are consistent over the different algorithms tested. With each two moons realization, we tested the classification accuracy with varying fidelity percents for the labeled data. Again, the chosen labeled set $Z'$ is consistent across the trials for different algorithms. Our results are summarized in Figure 5.

**4.1. Learning $\tau$ and $\alpha$.** In these experiments, the hierarchical algorithms performed much better in classifying the two moons dataset than the nonhierarchical algorithm. We fixed $\tau =$

**Algorithm 6** 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

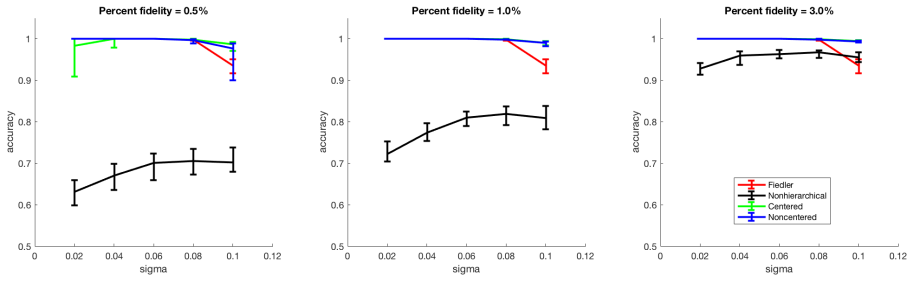
$$\begin{aligned} A(v^{(k)} \to \hat{v}^{(k)}) &= \min\left\{1, \frac{h(\hat{v}^{(k)}, \xi^{(k+1)}, M^{(k)})}{h(v^{(k)}, \xi^{(k+1)}, M^{(k)})}\right\} \\ &= \min\left\{1, \exp\left(\Phi(T(v^{(k)}, \xi^{(k+1)}, M^{(k)})) - \Phi(T(\hat{v}^{(k)}, \xi^{(k+1)}, M^{(k)}))\right)\right\} \end{aligned}$$

  **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

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

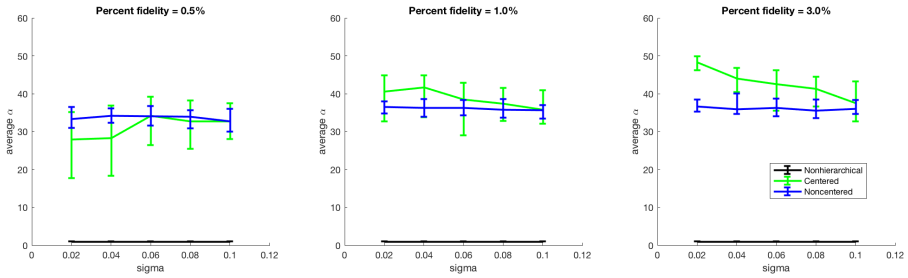
**end for**
**return** $\{T(v^{(k)}, \xi^{(k)}), v^{(k)}, \xi^{(k)}\}$

FIG. 5. *Classification accuracy of different algorithms for two moons dataset compared with $\sigma$ and percent fidelity. Plotted is the median classification accuracy with error bars that indicate the 25 and 75-th quantiles over the 50 trials for each parameter combination. For the pCN and hierarchical algorithms, we used spectral projection up to the first 50 eigenvectors. We ran 100000 iterations with a burn-in period of 1000, and we fixed $\gamma = 0.1$.*




$\alpha = 1$ in the nonhierarchical pCN, while we set $\tau^{(0)} = \alpha^{(0)} = 1$ for the hierarchical algorithms. In the hierarchical algorithms, we allowed $\tau \in [0.01, 60]$ and $\alpha \in [0.1, 60]$ as the uniform priors. The medians of the average values of $\alpha$ given by the hierarchical algorithms are shown in Figure 6. It appears that fixing $\alpha = 1$ in the nonhierarchical algorithm is partially responsible for the discrepancy in performance. With small $\alpha$, the prior on $u$ drops off more slowly with the higher eigenvectors, while larger $\alpha$ enforces that these higher eigenvectors have less influence on $u$. This

FIG. 6. *Average values of $\alpha$ from the MCMC. Plotted is the median of the averages of $\alpha$ with error bars that indicate the 25 and 75-th quantiles over the 50 trials for each parameter combination. Notice that $\alpha$ is on average much larger in the hierarchical algorithms.*

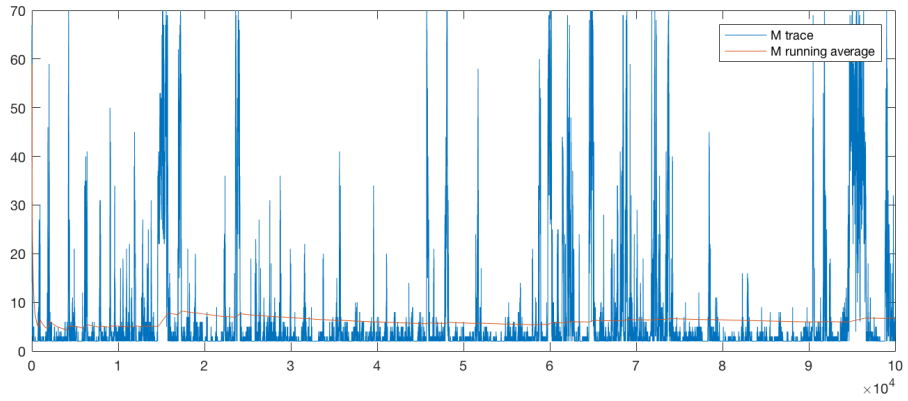



suggests that for the two moons data set, information for binary classification is concentrated in the lower eigenvectors, as expected.

## 5. Experiments with model (C).

**5.1. $M$ and $\sigma$ relation.** We tested Algorithm 4 on the two moons dataset with varying $\sigma$, fixing 1% labeled nodes. We also fixed $\tau = 2, \alpha = 35$ by setting $\epsilon_1 = \epsilon_2 = 0$, so the algorithm is only learning $\xi$ and $M$. We initialized $M^{(0)} = 50$, and the uniform prior of $M \sim \mathsf{U}(1, 70)$. We can see that for small $\sigma = 0.06$, $M$ is small as only the first few eigenvectors are necessary (see Figure 7). However, when we choose a larger $\sigma = 0.2$, $M$ needs to be larger (see Figure 8). For $\sigma = 0.06$, the classification accuracy was 100%. For $\sigma = 0.2$, the classification accuracy was 91.97%.

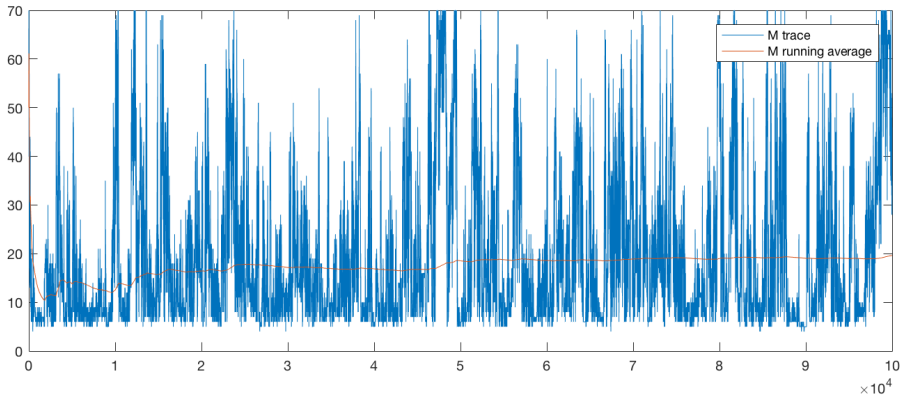
FIG. 7. $\sigma = 0.06$, *trace of* $M$




**5.2. Nonhierarchical vs. Hierarchical.** We will compare Algorithm 1, pCN, with fixed $\tau$ and $\alpha$ against Algorithm 4 with the same fixed $\tau$ and $\alpha$, but learning $M$.

**5.2.1. Voting records.** The nonhierarchical algorithm for learning $\tau$ and $\alpha$ gave expected values of $\tau$ and $\alpha$ as $\tau \approx 2$ and $\alpha \approx 35$, so I chose those parameters for the pCN and for Algorithm 4. For the pCN, all 435 eigenvectors were used. I narrowed the range of $M$ allowed in Algorithm 4 to 1 to 70. 5 labeled nodes were selected, consistent across the two methods. In this particular realization (seeded with rng(5)), the hierarchical algorithm achieves 87.74% while the nonhierarchical algorithm achieves 87.67% classification accuracy. Figure 9, Figure 10 , and Figure 11 show the results of the pCN. Figure 15, Figure 12, Figure 13 show the results the hierarchical algorithm. Notice in Figure 15 that there seems to be an important eigenvector for classification indexed around 30, as $M$ seldom drops below 30. Looking at the eigenvectors around index 30, I plotted the 34th eigenvector in Figure 14. From a purely visual perspective,

FIG. 8. $\sigma = 0.2$, *trace of* $M$




it does appear that this eigenvector corresponds decently to the final classification in Figure 12 by looking at where some of the "spikes" are. After trying some other realizations, it seems that being hierarchical may have some benefits, but more experiments are needed.

FIG. 9. *Algorithm* 1, *average of* $S(u)$

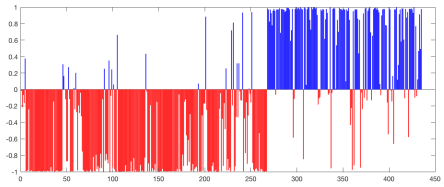


FIG. 10. *Algorithm* 1, $u$ *acceptance probability*

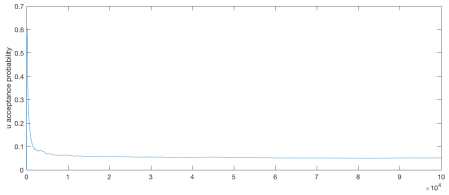



FIG. 11. *Algorithm* 1, *running average of* $S(u(i))$ *for select* $i$

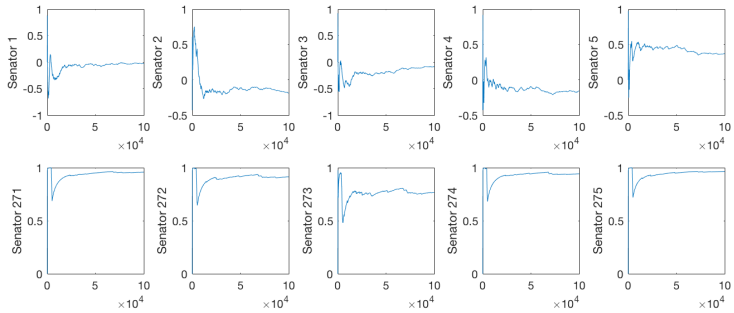



FIG. 12. *Algorithm* 4, *average of* $S(u)$

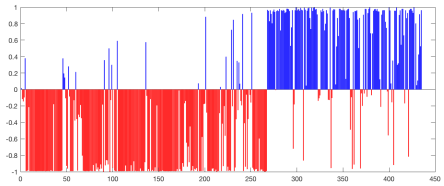


FIG. 13. *Algorithm* 4, $M$ *and* $\xi$ *acceptance probabilities*

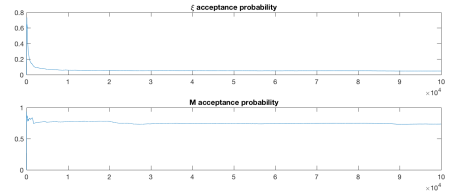


FIG. 14. *34th eigenvector of unnormalized fixed length-scale L*

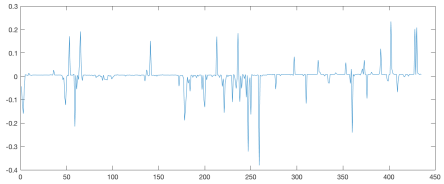
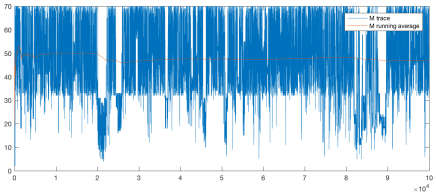


FIG. 15. *Algorithm 4, trace of M*




**5.2.2. Two moons.** I tried a similar set of experiments on the two moons data. I fixed $\tau = 2, \alpha = 35$, again following the expected values of these hyperparameters from the noncentered algorithm that learns $\tau, \alpha$. I set the pCN to use the first 100 eigenvectors, and I set the range for $M$ in Algorithm 4 to be $[1, 70]$. The two moons data was generated with $N = 2000, d = 100, \sigma = 0.2$ and 1% fidelity. The nodes selected for fidelity are random but consistent between the different methods, as is the data itself. The colored diamonds in the scatter plots are the labeled nodes. One realization of this test is shown in Figure 16, Figure 17 for pCN, and Figure 18, Figure 19, Figure 20 for the hierarchical algorithm. In this particular realization, the two algorithms performed similarly, with 90.56% accuracy for pCN and 91.97% accuracy for the hierarchical algorithm as shown in the final clusterings in Figure 16 and Figure 18. In Figure 20, $M$ seems to have moved from its uniform prior distribution, preferring a mean of around 20. The results again suggest that there is some value to being hierarchical with $M$ even with good choices of $\tau$ and $\alpha$, but more experiments are needed.

FIG. 16. *pCN, final classification projected into first two dimensions.*

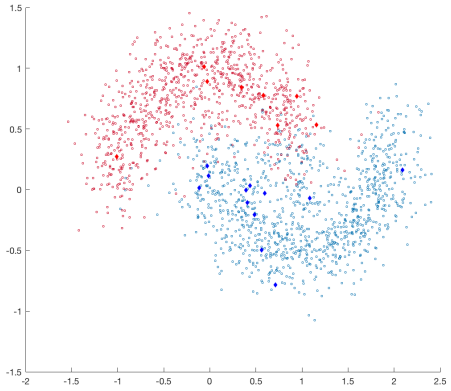
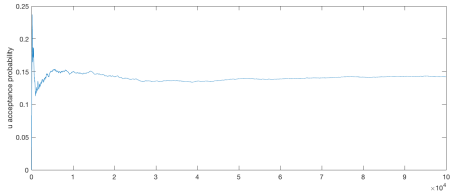


FIG. 17. *pCN, u acceptance probability.*




**6. Experiments with model (E).** We run some experiments with model (E) on the two moons dataset. We again fix $\tau = 2, \alpha = 35$, and pick $a = 0.5$ to create the prior on $v$. We pick the walk size for $v$ to be $\epsilon = 1.0 \times 10^{-13}$. We allow the prior on $M$ to be uniform, $\mathsf{U}(1, 70)$. If we compare this algorithm against model (C) with fixed $\tau, \alpha$, this is equivalent to setting $\epsilon = 0$ and fixing $v$ to be at the mean of its prior. Learning $v$ and $M$ does not seem to greatly affect the classification accuracy compared to just learning $M$. Consider the following run on two moons, with fidelity = 1%, $\sigma = 0.2$, in Figure 21 and Figure 22. The model that learns $v, M$ obtained 85.45% accuracy while the one that only learns $M$ obtained 86.52% accuracy. The histograms of $M$ appear very similar, with the same cutoff observed at the 14th index. The average values of the $u_j$ coefficients also seem to agree in magnitude and sign between the two algorithms.

**7. Goals.** One research goal for the remainder of the summer is to gather more data comparing the algorithm that learns $M$ against the nonhierarchical algorithm to determine if the

FIG. 18. *Hierarchical, final classification projected into first two dimensions.*

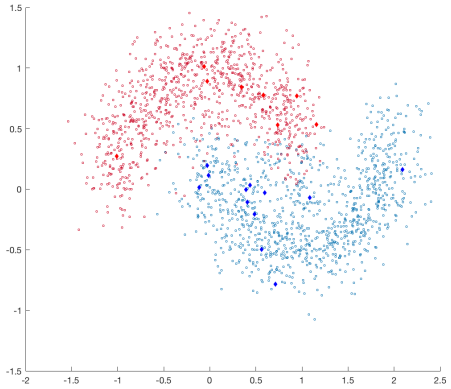


FIG. 19. *Hierarchical, $\xi$ acceptance probability.*

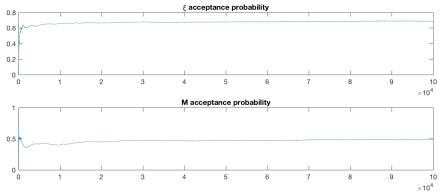
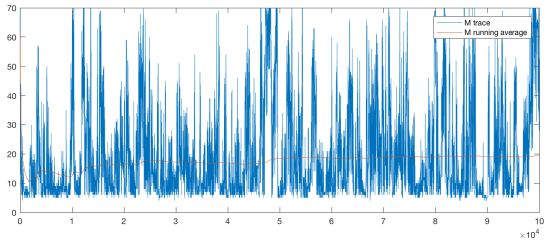



FIG. 20. *Hierarchical, seed rng(3). $M$ trace.*




method is beneficial. It will also be interesting to better analyze the relationship between $\sigma$ and $M$ in the two moons dataset. Preliminary experiments suggest that in low $\sigma$ data, small $M$ is enough for an accurate clustering, but in large $\sigma$, higher $M$ is necessary as more eigenvectors are needed to fit the problem. It could be interesting to create a plot of average $M$ versus increasing $\sigma$ to test this hypothesis.

Another goal is to study the effectiveness of the hierarchical methods on multiclass datasets. In particular, I am working on implementing these algorithms to handle the MNIST data set, which features up to 10 different classes.

## 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] O. Papaspiliopoulos, G. O. Roberts, and M. Sköld, *A general framework for the parametrization of hierarchical models*, Statistical Science, (2007), pp. 59–73.

[5] L. Zelnik-Manor and P. Perona, *Self-tuning spectral clustering*, in Advances in neural information processing systems, 2005, pp. 1601–1608.

FIG. 21. *Learning $v, M$. The bottom left figure was one observation of the $v_j$.*

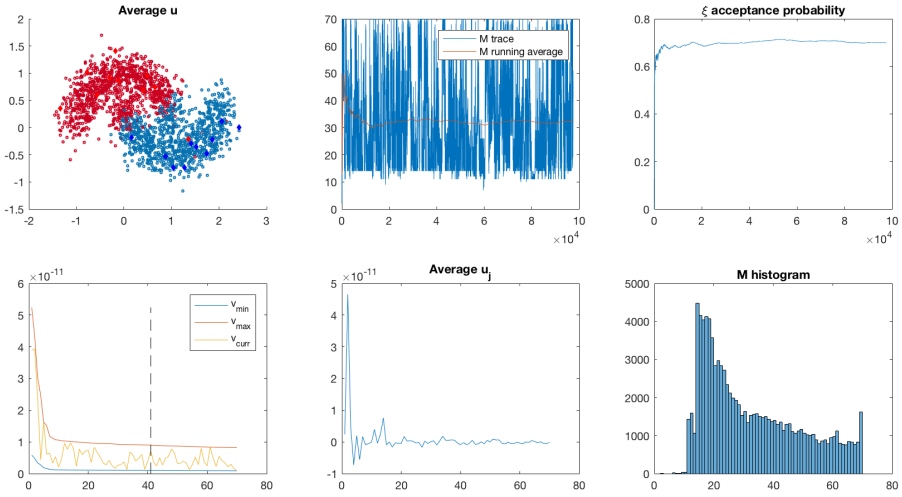



FIG. 22. *Learning only $M$*