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 τ, α, M refer to the prior in (1)
(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 $f(\lambda), M$.
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:

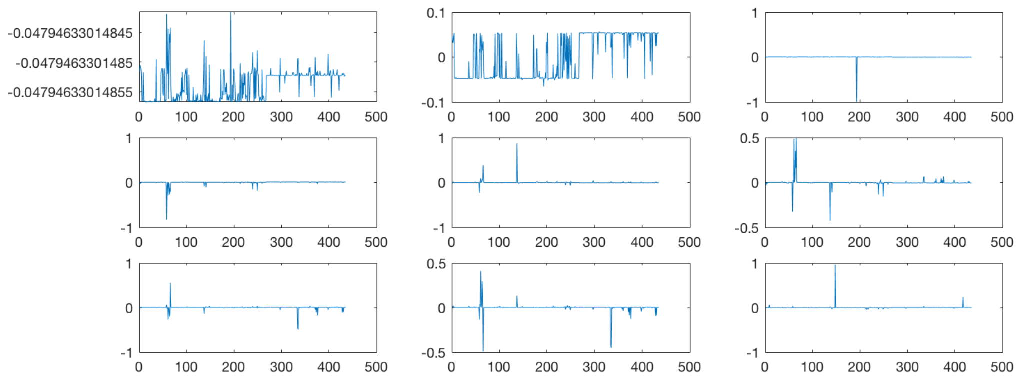
2.1. Voting records.

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 $w_{ij} = \exp\left(-\frac{d(x_i,x_j)}{2l^2}\right), \quad d_{p,q}(x,y) = ||x-y||_p^q$ with $p = 2, q = 2, l = 1$. Figure 1 is part of the spectrum of this choice of Laplacian.

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 [4]. 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,* $\sigma = 0.04$.

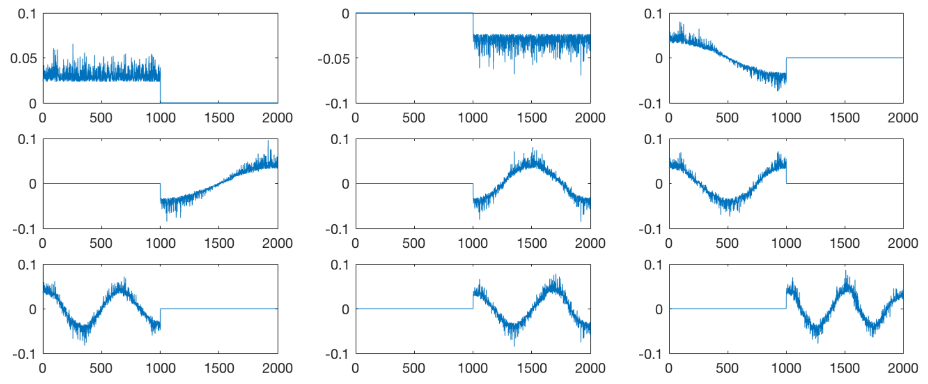
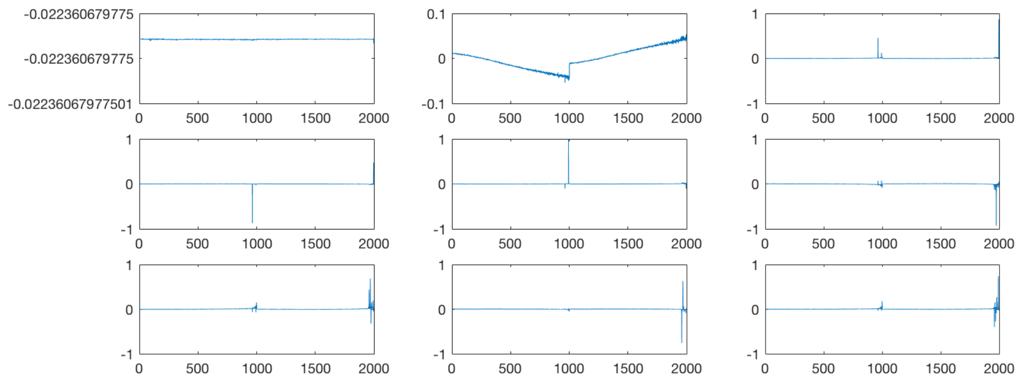
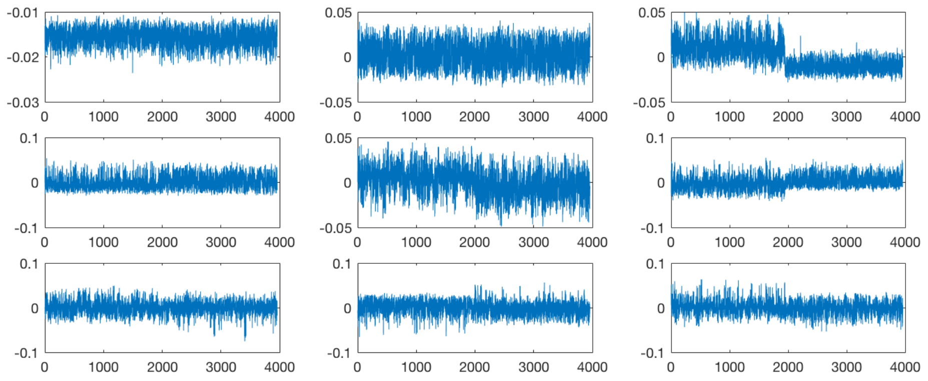


FIG. 3. *Lowest* 9 *eigenvectors of unnormalized fixed length* L *of two moons,* $\sigma = 0.04$.



2.3. MNIST data sets. This data set contains 70,000 images of 28×28 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 [1] and described in Algorithm 1.

Algorithm 1 General pCN adapted from [2]

```
Select $u^{(0)}$. Select $\tau, \alpha$. Select $\beta \in [0,1]$
for $k = 0$ to $S$ do
    Sample $v$ from the prior distribution given in (1)
    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 $\{u^{(k)}\}$
```

3.2. Model (B). To implement model (B), which is hierarchical, we need to sample conditional distributions that govern u, τ, and α. 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 θ to represent the hyperparameters, which could include τ, α, 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, τ, α 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 τ, α 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 τ, α

```
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 ξ 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 ξ and the parameters $\theta = (\tau, \alpha)$ are a priori independent [3].

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 ξ, τ, α

```
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).

4. Voting records experiments with models (A) and (B).

In each of the following experiments, the labeled data Z' was chosen consistently: members 20-30 was labeled one party and members 280-290 as the other.
For Algorithm 1, fixing $\gamma = 0.0001, \beta = 0.4, \tau = 1, \alpha = 1$, and running for 100000 iterations with a burn-in period of 1000, I could get clustering at about 85% accuracy. One final clustering obtained is shown in Figure 5, with relevant figures Figure 6 and Figure 7 that suggest that the chain has converged.
Experiments with Algorithm 2 on the voting records data indicated poor mixing in the hyper-parameters by looking at traces of τ, α. We decided that the problem was in part due to the irregularity of the larger eigenvectors, perhaps because of the accuracy of MATLAB's eig solver. Truncating the list of eigenvectors to only consider the first 50 seemed to help the problem, and we were able to observe better clustering. Fixing $\gamma = 0.0001, \beta = 0.4, \tau^{(0)} = 20, \alpha^{(0)} = 20, \tau \in [0.1, 60], \alpha \in [0.1, 60], \epsilon_\tau = 0.1, \epsilon_\alpha = 0.1$ and running 100000 iterations with a burn-in period of 1000, the accuracy is about 83%. See Figure 8, Figure 9, Figure 10, Figure 11.
Finally, the non-centered Algorithm 3 seems to converge faster than Algorithm 2, and truncation of the eigenvectors is not necessary. Fixing $\gamma = 0.0001, \beta = 0.1, \tau^{(0)} = 20, \alpha^{(0)} = 20, \tau \in$

FIG. 5. *Algorithm 1 final average*

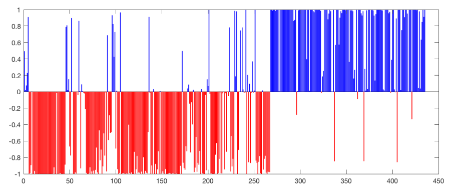


FIG. 6. *Algorithm 1 average u acceptance probability*

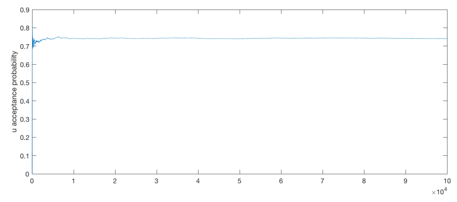


FIG. 7. *Running averages of classifications of delegates*

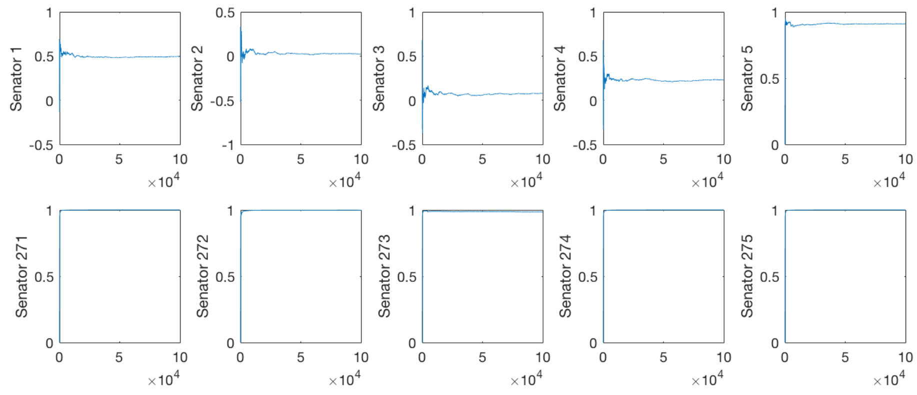


FIG. 8. *Algorithm 2 after truncating eigenvectors, final average*

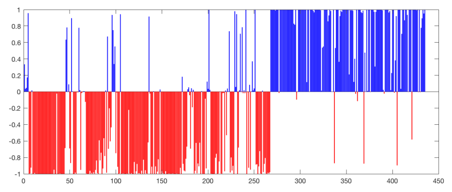


FIG. 9. *Algorithm 2 average u acceptance probability*

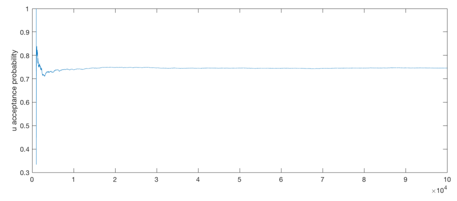


FIG. 10. *Algorithm 2 trace of τ*

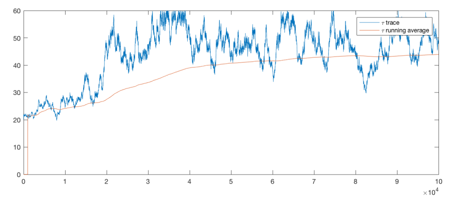
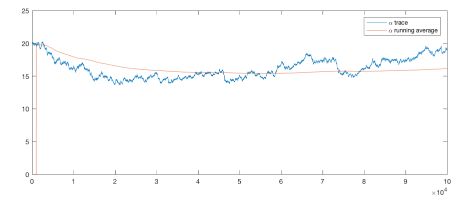


FIG. 11. *Algorithm 2 trace of α*



$[0.1, 60], \alpha \in [0.1, 60], \epsilon_\tau = 1, \epsilon_\alpha = 1$ and running 100000 iterations with a burn-in period of 1000, the accuracy is about 87%. See Figure 12, Figure 13, Figure 14, Figure 15.

5. Two moons experiments with models (A) and (B). Using these algorithms, I also ran experiments to cluster the two-moons data set, with $r = 1, N = 1000, d = 100, \sigma = 0.1$. We used the self-tuning Laplacian introduced in [4].

For each of the following experiments, we labeled the same 42 nodes, 21 in each half-moon, approximately uniformly spaced. Running Algorithm 1 on this data set using only the first 50 eigenvectors with $\gamma = 0.0001, \beta = 0.4, \tau = 1, \alpha = 1$ with 100000 iterations and a burn-in period of 1000, we could get around 90% accuracy.

We now compare the results from the centered (Algorithm 2) and non-centered (Algorithm 3)

FIG. 12. *Algorithm 3 final average*

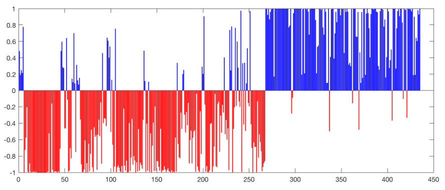


FIG. 13. *Algorithm 3 average ξ acceptance probablity*

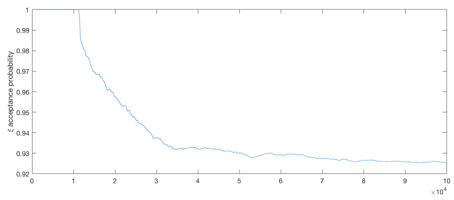


FIG. 14. *Algorithm 3 trace of τ*

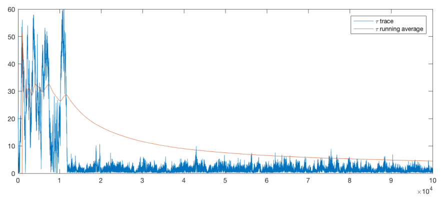


FIG. 15. *Algorithm 3 trace of α*

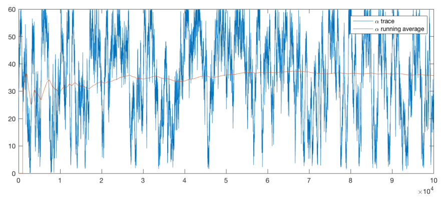


FIG. 16. *Algorithm 1 final average*

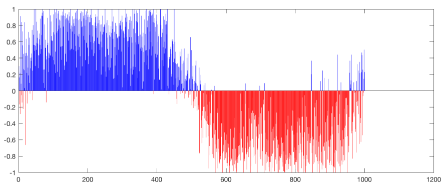
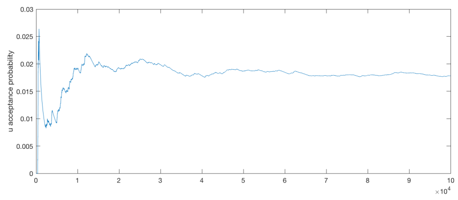


FIG. 17. *Algorithm 1 average u acceptance probability*



algorithms, truncating the number of eigenvectors to 50 in both cases. For both algorithms, we fixed $\gamma = 0.1, \beta = 0.4, \tau^{(0)} = 20, \alpha^{(0)} = 20, \tau \in [0.01, 60], \alpha \in [1, 60], \epsilon_\tau = 0.5, \epsilon_\alpha = 1$ and ran 100000 iterations with a burn-in period of 1000. With these settings, Algorithm 2 achieves around 90% accuracy, while Algorithm 3 achieves around 98%. It still appears that the non-centered algorithm converges faster than the centered algorithm. If the centered algorithm is initialized at $\tau^{(0)} = 1$, the final classification accuracy is very similar to that of the non-centered algorithm. See Figure 18, Figure 19, Figure 20, Figure 21, Figure 22, Figure 23, Figure 24, Figure 25.

FIG. 18. *Algorithm 2 final average*

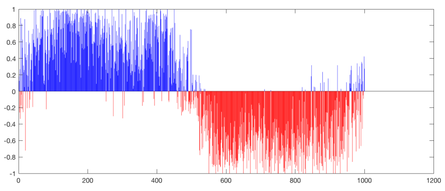
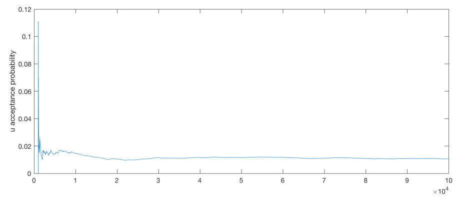


FIG. 19. *Algorithm 2 average u acceptance probability*



Notice the convergence of τ and its low variance in the samples for the non-centered self-tuning algorithm.

6. Experiments comparing hierarchical and nonhierarchical algorithms.

7. Experiments with model (C).

8. Experiments with model (D).

REFERENCES

Fig. 20. *Algorithm 2, trace τ*

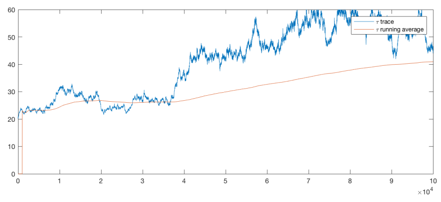


Fig. 21. *Algorithm 2, trace α*

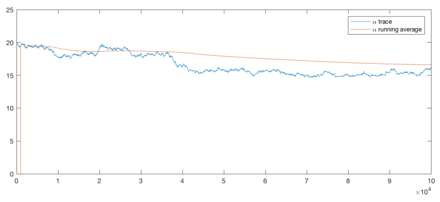


Fig. 22. *Algorithm 3 final average*

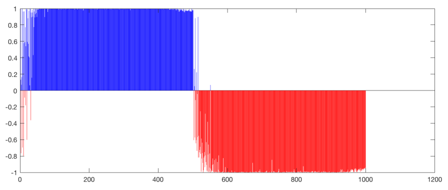


Fig. 23. *Algorithm 3 average ξ acceptance probability*

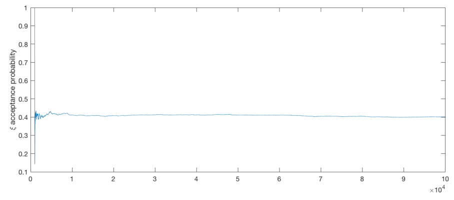


Fig. 24. *Algorithm 3, trace τ*

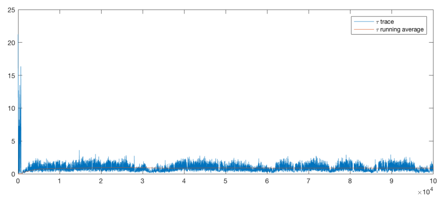
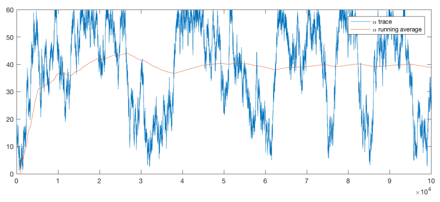


Fig. 25. *Algorithm 3, trace α*



[1] A. Beskos, G. O. Roberts, A. M. Stuart, and J. Voss, *MCMC methods for diffusion bridges*, Stochastics and Dynamics, 8 (2008), pp. 319–350.

[2] 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.

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

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