Probabilistic Temporal Subspace Clustering: Supplementary Material

Behnam Gholami
Department of Computer Science
Rutgers University
bb510@cs.rutgers.edu

Vladimir Pavlovic
Department of Computer Science
Rutgers University
vladimir@cs.rutgers.edu

1. Non-parametric EM Algorithm for Probabilistic Temporal Subspace Clustering

The proposed model can be formally defined as

$$f^s \sim GP(0, \mathcal{K}), \quad s = 1, 2, ..., S \tag{1}$$

$$\beta_s^n = \sigma(f^s(t_n)) \prod_{l=1}^{s-1} (1 - \sigma(f^l(t_n))), \quad n = 1, ..., N, \; s = 1, 2, ..., S \tag{2}$$

$$c_n | \beta_1^n, ..., \beta_S^n \sim Multi(\beta_1^n, \beta_2^n, ..., \beta_S^n), \quad n = 1, ..., N, \tag{3}$$

$$\pi_{ks} \sim Beta(a/K, b(K-1)/K), \quad s = 1, ..., S, \; k = 1, ..., K, \tag{4}$$

$$z_{ks} | \pi_{ks} \sim Ber(\pi_{ks}), \quad s = 1, ..., S, \; k = 1, ..., K, \tag{5}$$

$$w_{s,n} \sim \mathcal{N}(0, \gamma_{s,n}^{-1} \boldsymbol{I}), \quad s = 1, ..., S, \; n = 1, ..., N, \tag{6}$$

$$x_n | z_s, w_{s,n}, \boldsymbol{\Phi}, \boldsymbol{\mu}, \alpha_s \sim \sum_{s=1}^{S} \beta_s \mathcal{N}\big(x_n; \boldsymbol{\Phi}_s (z_s \odot w_{s,n}) + \mu_s, \alpha_s^{-1} \boldsymbol{I}\big), s = 1, ..., S, \; n = 1, ..., N, \tag{7}$$

where $Multi$ denotes the multinomial distribution.

E step:

In this step the parameters are fixed, and the variational distributions are updated by maximizing the lower bound using a coordinate ascent algorithm.

Update for $w_{s,n}$:

One can show that $q(w_{s,n})$ is a $\mathcal{N}(m_{s,n}, \Sigma_{s,n})$ with parameters

$$(\Sigma_{s,n})^{-1} = \gamma_{s,n} \boldsymbol{I} + (\alpha_s)^{-1} Z_s (\boldsymbol{\Phi}_s)^\top \boldsymbol{\Phi}_s Z_s \tag{8}$$

$$m_{s,n} = \gamma_{s,n} \Sigma_{s,n} (\boldsymbol{\Phi}_s Z_s)^\top (x_n - \mu_s) \tag{9}$$

where Z_s is a diagonal matrix with entries taken from the vector z_s.

Update for $\pi_s = [\pi_{1s}, ..., \pi_{Ks}]$:

One can show that $q(\pi_{ks})$ is a $Beta(\pi_{ks}; a_{ks}, b_{ks})$ where

$$a_{ks} = a/K + z_{ks}, \quad b_{ks} = b(K-1)/K + 1 - z_{ks} \tag{10}$$

M step:

In this step, the parameters are computed by maximizing the expected complete-data log likelihood $\log p(\boldsymbol{X}, \mathcal{T}, \Theta, \Omega)$ while keeping the parameters of the posterior distribution $Q(\Omega)$ fixed.

Update for $\boldsymbol{\Phi}_s$:
One can show that $\boldsymbol{\Phi}_s$ can be analytically updated as

$$\boldsymbol{\Phi}_s = \Big[\sum_{n=1}^{N} \mathbf{1}[c_n = s](x_n - \mu_s) m_{s,n}^{\top} Z_s \Big] \times \Big[\alpha_s^{-1} \boldsymbol{I} + \sum_{n=1}^{N} \mathbf{1}[c_n = s] Z_s \big(m_{s,n} m_{s,n}^{\top} + \Sigma_{s,n} \big) Z_s \Big]^{-1}. \tag{11}$$

where $\mathbf{1}[x]$ denotes the indicator function ($\mathbf{1}[x] = 0$ if x is true, 0 otherwise).

Update for μ_s:
One can show that μ_s can be analytically updated as

$$\mu_s = \sum_{n=1}^{N} \mathbf{1}[c_n = s] \big(x_s - \boldsymbol{\Phi}_s (z_s \odot m_{s,n} \big) / \sum_{n=1}^{N} \mathbf{1}[c_n = s]. \tag{12}$$

Update for α_s:
One can show that α_s can be analytically updated as

$$\alpha_s^{-1} = \frac{\sum_{n=1}^{N} \mathbf{1}[c_n = s] \Big(\big\| x_n - \mu_s - \boldsymbol{\Phi}_s (z_s \odot m_{s,n}) \big\|^2 + tr \big(\boldsymbol{\Phi}_s Z_s \Sigma_{s,n} Z_s \big) \Big)}{d \sum_{n=1}^{N} \mathbf{1}[c_n = s]}. \tag{13}$$

Update for $\gamma_{s,n}$:
One can show that $\gamma_{s,n}$ can be analytically updated as

$$\gamma_{s,n}^{-1} = \frac{m_{s,n}^{\top} m_{s,n} + tr \big(\Sigma_{s,n} \big)}{K}. \tag{14}$$

Update for c_n:

$$c_n = \underset{c_n \in \{1,2,\ldots,S\}}{\arg\max} \ \mathcal{L}(c_n), \tag{15}$$

where

$$\begin{aligned} \mathcal{L}(c_n) &= \sum_{s=1}^{S} \mathbb{E}_{q(w_{s,n})} [\log p(x_n | c_n, z_s, w_{s,n}, \boldsymbol{\Phi}_s, \mu_s, \alpha_s)] + \log p(c_n | t_n, f^1, ..., f^S) \\ &= \sum_{s=1}^{S} -\frac{\alpha_s}{2} \mathbf{1}[c_n = s] \| x_n - \mu_s - \boldsymbol{\Phi}_s (z_s \odot m_{s,n}) \|^2 \\ &+ \sum_{s=1}^{S} \mathbf{1}[c_n = s] \Big[\frac{D(\log \alpha_s - \log 2\pi)}{2} - \frac{\alpha_s}{2} tr \big(\boldsymbol{\Phi}_s Z_s \Sigma_{s,n} Z_s \big) \Big] \\ &+ \sum_{s=1}^{S} \mathbf{1}[c_n = s] \Big[\log \sigma \big(f^s(t_n) \big) + \sum_{l=1}^{s-1} \log \big(1 - \sigma(f^l(t_n)) \big) \Big] \end{aligned} \tag{16}$$

Intuitively, the first two terms in Eq. 16 are data-dependent terms and the last term corresponds to the (approximate) **GP-GEM** prior penalty. The value of the last term will be very negative for low-probability cluster indices, as learned through inference.

Algorithm 1 Obtaining z_s

Require: $\{x_n\}, \mathbf{\Phi}_s, \alpha_s, \mu_s, q(w_{s,n}), q(\pi_{1s}, ..., \pi_{Ks}),$

set $z_s = 0$ and index set $\mathcal{I} = \emptyset$.
for $k = 1, 2, ..., K$ **do**
  set $\rho_k^+ = \sum_{n=1}^N -\frac{\alpha_s}{2}\mathbf{1}[c_n = s]\Big[\|x_n - \mu_s - [m_{s,n}]_k \mathbf{\Phi}_{s,k}\|^2 + [\mathbf{\Phi}_s]_{kk}[\Sigma_{s,n}]_{kk}\Big] + \Psi(a_{s,k}) - \Psi(b_{s,k})$
  set $\rho_k^- = \sum_{n=1}^N -\frac{\alpha_s}{2}\mathbf{1}[c_n = s]\|x_n\|^2$
end for
while $\max_k \ \rho_k^+ - \rho_k^- > 0$ **do**
  set $k' = \arg\max_k \ \rho_k^+ - \rho_k^-, \ \mathcal{I} \leftarrow \mathcal{I} \cup \{k'\}, \ z_{k's} = 1, \ \rho_k^+ = -\infty$
  for all $k \notin \mathcal{I}$ **do**
    set $\rho_k^+ = \sum_{n=1}^N -\frac{\alpha_s}{2}\mathbf{1}[c_n = s]\Big[\|x_n - \mu_s - [\mathbf{\Phi}_s]_{\mathcal{I}}[m_{s,n}]_{\mathcal{I}} - [m_{s,n}]_k \mathbf{\Phi}_{s,k}\|^2 + tr\big([\mathbf{\Phi}_s]_{\mathcal{I}}[\Sigma_{s,n}]_{\mathcal{I}}\big) + [\mathbf{\Phi}_s]_{kk}[\Sigma_{s,n}]_{kk}\Big] + \Psi(a_{s,k}) - \Psi(b_{s,k})$
    set $\rho_k^- = \sum_{n=1}^N -\frac{\alpha_s}{2}\mathbf{1}[c_n = s]\Big[\|x_n - \mu_s - [\mathbf{\Phi}_s]_{\mathcal{I}}[m_{s,n}]_{\mathcal{I}}\|^2 + tr\big([\mathbf{\Phi}_s]_{\mathcal{I}}[\Sigma_{s,n}]_{\mathcal{I}}\big)\Big]$
  end for
end while
return z_s

It therefore eliminates these clusters from the model by shrinking $\mathcal{L}(c_n)$ to lower values.

Update for z_s:

$$z_s = \arg\max_{z_s} \mathcal{L}(z_s), \quad s.t. \quad z_s \in \{0, 1\}^K, \tag{17}$$

where

$$\begin{aligned}\mathcal{L}(z_s) &= \sum_{n=1}^N \mathbb{E}_{q(w_{s,n})}[\log p(x_n | c_n, z_s, w_{s,n}, \mathbf{\Phi}_s, \mu_s)] + \mathbb{E}_{q(\pi_{1s}, ..., \pi_{Ks})}[\log p(z_s | \pi_{1s}, ..., \pi_{Ks})] \\ &= \sum_{n=1}^N -\frac{\alpha_s}{2}\mathbf{1}[c_n = s]\Big[\|x_n - \mu_s - \mathbf{\Phi}_s(z_s \odot m_{s,n})\|^2 + tr\big(\mathbf{\Phi}_s Z_s \Sigma_{s,n} Z_s\big)\Big] \\ &+ \sum_{k=1}^K z_{ks}\big(\Psi(a_{s,k}) - \Psi(b_{s,k})\big), \quad s.t. \ \ z_s \in \{0, 1\}^K.\end{aligned} \tag{18}$$

where $\Psi(.)$ denotes the **digamma** function. Intuitively, the first term (second line) in Eq. 18 is data dependent term, and the last term corresponds to the Beta distribution penalty. More precisely, the value of the last term will be very small (negative) for low-probability subspace bases, as learned through the EM inference. Hence, it removes these subspace bases from the model by shrinking $\mathcal{L}(z)$ to smaller values. Since (18) is a combinatorial optimization problem, we use a greedy algorithm (Algorithm 1) similar to Orthogonal Maching Persuit (OMP) [2] to solve (18).

In Algorithm 1, $[\boldsymbol{X}]_{\mathcal{I}}/[x]_{\mathcal{I}}$ denotes the submatrix/subvector of $\boldsymbol{X}/x$ formed by the columns/dimensions indexed by $\mathcal{I}$. Intuitively, we initialize z with zero and set sequentially each entry of z to one, scoring each entry to determine which to set to one.

Update for $f^s = [f^s(t_1), ..., f^s(t_N)]$

In the following, we denote $f^s(t_n)$ with f_n^s for notational simplicity.

$$f^s = \arg\max_{f^s} \mathcal{L}(f^s) \tag{19}$$

where

$$\begin{aligned}\mathcal{L}(f^s) &= \log(f^s|\mathbb{K}) + \sum_{n=1}^{N} \log p(c_n = s|f^s) \\ &= -\frac{1}{2} f^{s\top} \mathbb{K}^{-1} f^s + \sum_{n=1}^{N} \mathbf{1}[c_n = s] \log \sigma(f_n^s) + \sum_{n=1}^{N} \sum_{l=s+1}^{S} \mathbf{1}[c_n = l] \log \big(1 - \sigma(f_n^s)\big)\big) \\ &= -\frac{1}{2} f^{s\top} \mathbb{K}^{-1} f^s + \sum_{n=1}^{N} \mathbf{1}[c_n = s] \log \sigma(f_n^s) + \sum_{n=1}^{N} \mathbf{1}[c_n > s]\big(\log \sigma(f_n^s)) - f_n^s\big) \\ &\geq -\frac{1}{2} f^{s\top} \mathbb{K}^{-1} f^s + \sum_{n=1}^{N} \mathbf{1}[c_n = s]\big(\frac{1}{2} f_n^s - \lambda(\xi_{sn}) f_n^{s2}\big) + \sum_{n=1}^{N} \mathbf{1}[c_n > s]\big(-\frac{1}{2} f_n^s - \lambda(\xi_{sn}) f_n^{s2}\big) \\ &\geq -\frac{1}{2} f^{s\top} \mathbb{K}^{-1} f^s + C_s^\top f^s - \frac{1}{2} f^{s\top} \mathbb{A} f^s \end{aligned} \tag{20}$$

where

$$C_s = \big[\frac{1}{2}(\mathbf{1}[c_1 = s] - \mathbf{1}[c_1 > s]), ..., \frac{1}{2}(\mathbf{1}[c_N = s] - \mathbf{1}[c_N > s])\big]^\top \tag{21}$$

and $\mathbb{A}$ is a $N \times N$ diagonal matrix defined as

$$\mathbb{A} = \begin{pmatrix} 2\lambda(\xi_{s1})(\mathbf{1}[c_1 = s] - \mathbf{1}[c_1 > s]) & & & \\ & 2\lambda(\xi_{s2})(\mathbf{1}[c_2 = s] - \mathbf{1}[c_2 > s]) & & \\ & & \ddots & \\ & & & 2\lambda(\xi_{sN})(\mathbf{1}[c_N = s] - \mathbf{1}[c_N > s]) \end{pmatrix} \tag{22}$$

and $\{\xi_{sn}\}, s = 1, ..., S,\ n = 1, ..., N$ are the lower bound variational parameters.

One can show that f^s is updated as

$$f^s = \big(\mathbb{K}^{-1} + \mathbb{A}\big)^{-1} C_s \tag{23}$$

It should be noted that since $\mathbb{K}^{-1}$ is a tri-diagonal matrix and $\mathbb{A}$ is a diagonal matrix, (23) can be efficiently computed in $O(N^2)$ time.

Update for ξ_{sn}

$$\begin{aligned}\xi_{sn} &= \arg\max_{\xi_{sn}} \mathbf{1}[c_n = s] \log \sigma(f_n^s) + \mathbf{1}[c_n > s]\big(\log \sigma(f_n^s))\big) \\ &= \arg\max_{\xi_{sn}} \mathbf{1}[c_n > (s-1)]\bigg(\log \sigma(\xi_{sn}) - \frac{\xi_{sn}}{2} + \lambda(\xi_{sn})\xi_{sn}^2 - \lambda(\xi_{sn}) f^{s2}\bigg)\end{aligned} \tag{24}$$

By setting the derivative of the objective function of the Eq. 24 with respect to ξ_{sn} equal to zero, we have

$$1 - \sigma(\xi_{sn}) - 1/2 + 2\xi_{sn}\lambda(\xi_{sn}) + \lambda'(\xi_{sn})\xi_{sn}^2 - \lambda'(\xi_{sn}) f_n^{s2} = 0 \tag{25}$$

Using definitions of $\sigma(\xi_{sn})$ and $\lambda(\xi_{sn})$, the above Equation can be simplified as

$$\lambda'(\xi_{sn})\big(\xi_{sn}^2 - f_n^{s2}\big) = 0 \tag{26}$$

Since $\lambda'(\xi_{sn})$ is a monotonic function of ξ_{sn} for $\xi_{sn} > 0$ and the negative values for ξ_{sn} can be ignored [1], $\lambda'(\xi_{sn}) \neq 0$ and hence the update equations for the variational parameters can be obtained as $\xi_{sn} = f_n^s$.

Update for η

$$\eta = \arg\max_{\eta}\ -\frac{1}{2}\sum_{s=1}^{S} f^{s\top}\mathbb{K}^{-1} f^s - \frac{1}{2}\log|\mathbb{K}| \tag{27}$$

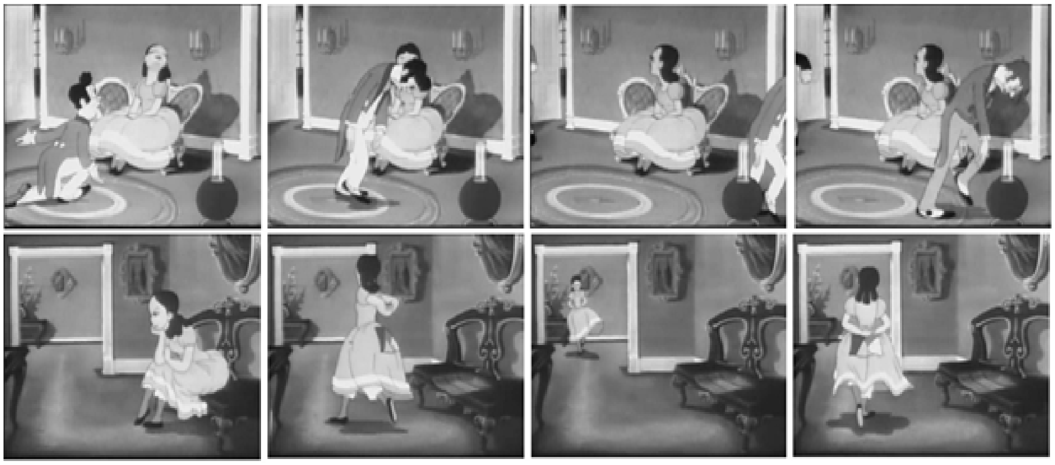

Figure 1. Sample frames from the video scene segmentation dataset. Each row contains some frames belonging to the same scene.

Table 1. ACC with standard deviation on scene segmentation dataset. The best (bold red), the second best (red).

method	Sequence-1	Sequence-2	Sequence-3	Sequence-4	Sequence-5	Sequence-6
SSC	63.27 ±3.67	65.38± 2.82	60.03 ± 3.21	71.06 ± 2.28	64.08 ± 2.81	69.99 ± 3.03
LRR	67.31 ± 2.99	70.83±2.81	62.01±2.82	70.08±2.69	65.31±2.78	70.43 ±2.67
LSR	63.28 ± 2.88	70.45±2.76	60.32 ±2.68	70.71 ±2.42	66.32 ±2.74	71.34 ± 2.12
OSC	75.37 ± 2.49	80.03 ± 3.27	63.21 ±2.60	75.36 ±2.31	70.11 ±3.96	78.44 ± 3.11
TSC	73.12 ±3.20	80.11 ±2.77	69.32 ±2.66	**83.21** ±3.00	**80.10** ±2.55	82.00 ±2.90
PM (our)	**82.11** ±4.10	**87.21** ±3.31	**75.31** ±3.84	81.32 ± 2.99	79.87 ±2.81	**84.64** ± 2.65

where $|\mathbb{K}|$ denotes the determinant of $\mathbb{K}$. By setting the derivative of the objective function of the Eq. 27 respect to η equal to zero, we have

$$\frac{1}{2}tr\left((\beta\beta^\top - \mathbb{K}^{-1})(\hat{\mathbb{K}} \odot \mathbb{K})\right) = 0 \tag{28}$$

where $\hat{\mathbb{K}}$ is a $N \times N$ matrix such that

$$\hat{\mathbb{K}}(i,j) = \exp(-|t_i - t_j|), \quad i = 1, ..., N, j = 1, ..., N \tag{29}$$

and $\beta = \mathbb{K}^{-1} f^s$. Since (28) is a highly nonlinear function of η, its update cannot be computed in closed form. However, since η is a scalar, we simply use the one dimensional gradient descent algorithm to solve (27). It should be noted that The complexity of computing the derivative of the objective function in eq. 28 is dominated by the need to invert the $\mathbb{K}$ matrix. Since $\mathbb{K}^{-1}$ can be computed in $O(N)$ time, the computation of the derivative of the objective function in eq. 28 requires only $O(N^2)$ time per iteration.

2. Video Scene Segmentation Experiments

The goal of this experiment is to segment individual scenes from a video sequence. The video sequence is drawn from a short animation freely available from the Internet Archive[1]. See Fig. 1 for some examples of two sequences to be segmented. Six video sequences, containing three scenes each, are drawn from the dataset. The sequences are around 15-30 seconds in length (approximately 450-900 frames). For this data set, we build a dictionary of the frames with 300 bases using the Orthogonal matching Pursuit (OMP) algorithm [2] and encode each frame as a 300 dimensional sparse vector.

We also set the truncation level for the number of subspaces and their dimension to $(K = 10, S = 10)$.

The mean performance along with the standard deviation of each method over 5 runs on the different sequences of the datasets is shown in Tables 1, and 2, from which we can see that the proposed method has better performance than the other

[1] http://archive.org

Table 2. NMI with standard deviation on scene segmentation dataset. The best (bold red), the second best (red).

method	Sequence-1	Sequence-2	Sequence-3	Sequence-4	Sequence-5	Sequence-6
SSC	0.5327 ± 0.013	0.5038 ± 0.008	0.4903 ± 0.016	0.5406 ± 0.009	0.5708±0.004	0.5299±0.012
LRR	0.5231±0.009	0.5083 ±0.003	0.5001±0.008	0.5608 ±0.011	0.5731±0.008	0.5143±0.009
LSR	0.5028±0.008	0.5145 ± 0.009	0.4932 ± 0.008	0.5571 ±0.006	0.5832±0.006	0.5434 ±0.012
OSC	0.5437±0.021	0.5503 ±0.017	0.5321 ±0.015	0.6036±0.004	0.6111 ± 0.006	0.5544±0.011
TSC	0.5512 ±0.011	0.5811 ±0.013	0.5232 ±0.012	0.6421± 0.008	0.6510 ±0.005	0.6100 ±0.009
PM (our)	**0.6111** ±0.021	**0.6321** ±0.018	**0.6357** ±0.017	0.6933 ±0.011	**0.6833** ±0.015	**0.6877** ±0.020

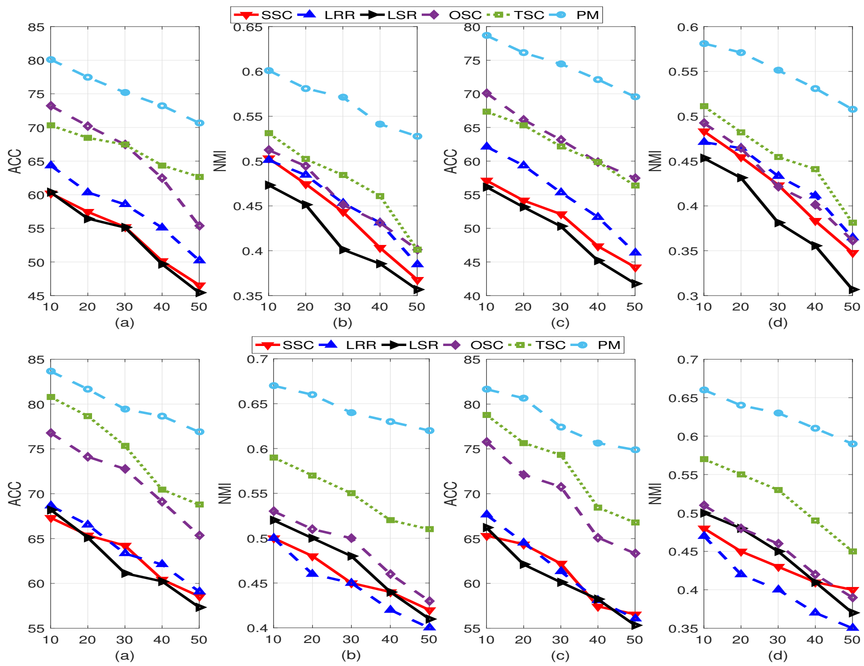

Figure 2. The results of different methods on the scene dataset when the data suffer from the loss of features. First Row: **Sequence 1**. Second Row: **Sequence 6**. Horizontal axes denote the missing rates (%). (a),(b): ACC and NMI results for MAR features, respectively. (c),(d): ACC and NMI results for NMAR features, respectively.

competing methods. This is due the fact that the Optimization-based methods learn the features and cluster them separately (sequentially), while our generative model simultaneously learns the representations and clusters the data points. The key observation is that good representations are beneficial to data clustering, with clustering results providing supervisory signals to representation learning.

3. Additional Experiments for missing features

MAR and MNAR results (averaged over 5 runs) for subjects 13, 54, 80 and 113 of Mocap dataset and sequences 1 and 6 of the scene segmentation dataset are provided in the Figs. 3 and 2 respectively. Not surprisingly, for both MAR and MNAR cases, the **PM** is more robust to missing features than other competing methods.

References

[1] C. M. Bishop. Pattern recognition. *Machine Learning*, 2006. 4

[2] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007. 3, 5

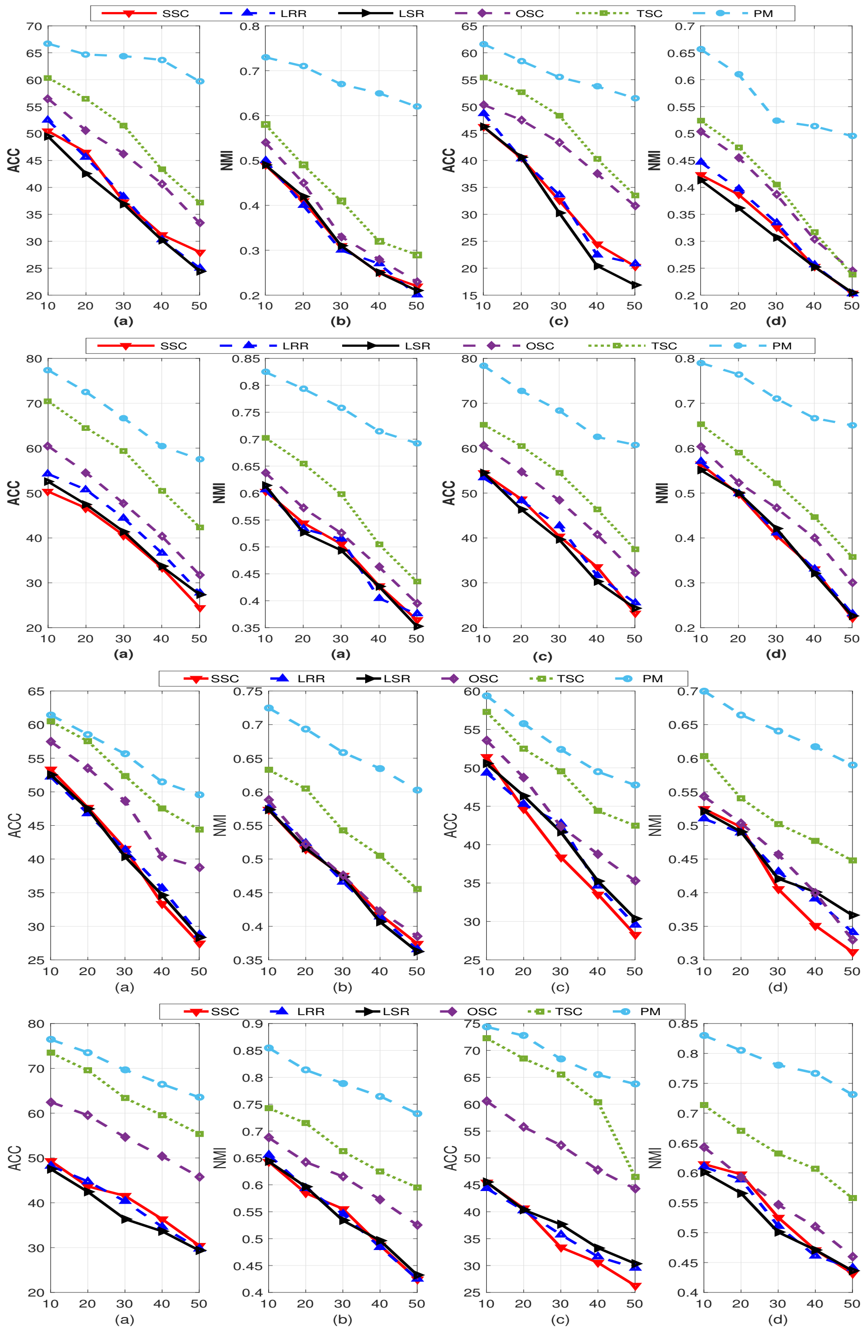

Figure 3. The results of different methods on Mocap dataset when the data suffer from the loss of features. First Row: **Subject 13**. Second Row: **Subject 54**. Third Row: **Subject 80** . Forth Row: **Subject 113**. Horizontal axes denote the missing rates (%). (a),(b): ACC and NMI results for MAR features, respectively. (c),(d): ACC and NMI results for NMAR features, respectively.