**Probabilistic GSFA:**

**Vishal Rishi MK, CH18B013**

**Intuition:**

Let $\boldsymbol{y}[k] = \left[y_1[k], y_2[k], \dots, y_M[k]\right]^T,\ k \geq 0$ be the observation signal. In standard linear SFA, we obtain the filtered observation signal $\boldsymbol{y}_f[k]$ by using the backward-difference operator,

$$\boldsymbol{y}_f[k] = \boldsymbol{y}[k] - \boldsymbol{y}[k-1]$$

With this approach in mind, the probabilistic SFA has an AR(1) process for the latent slow features $x_j[k],\ k \geq 0,\ j \in \{1,2, \dots, P\}$

$$e_j[k] = x_j[k] - \lambda_j x_j[k-1]$$

Here, $e_j[k] \sim GWN\left(0,\ \sigma_j^2\right)$ and $\lambda_j$ is the AR(1) coefficient. We use the same idea to come up with a probabilistic model for GSFA.

**Prior – Latent slow features:**

Let us assume that all signals are zero at negative time instants. Let $G(q^{-1})$ be a filter and $g[k]$, $(g[k] = 0, k < 0)$ be its impulse response coefficients. Hence, the filtered observation signal $\boldsymbol{y}_f[k]$ is given by,

$$\boldsymbol{y}_f[k] = G(q^{-1})\boldsymbol{y}[k]$$

In the convolution form,

$$\boldsymbol{y}_f[k] = \sum_{l=0}^{k} g[k-l]\boldsymbol{y}[l]$$

$$g[0]\boldsymbol{y}[k] = -\sum_{l=0}^{k-1} g[k-l]\boldsymbol{y}[l] + \boldsymbol{y}_f[k]$$

From this equation, we write a probabilistic model for the latent slow features

$x_j[k],\ j \in \{1,2, \dots, P\}$

$$x_j[k] = -\sum_{l=0}^{k-1} \lambda_j[l] x_j[k-l] + e_j[k]$$

Here, $e_j[k] \sim GWN\left(0,\ \sigma_j^2\right)$ and $\lambda_j[k]$ be the impulse response coefficients of $\Lambda_j(q^{-1})$.

If $\lambda_j[0] = 0$,

$$x_j[k] = -\sum_{l=0}^{k} \lambda_j[l] x_j[k-l] + e_j[k]$$

$$x_j[k] = -\Lambda_j(q^{-1}) x_j[k] + e_j[k]$$

Note that $\Lambda_j(q^{-1})$ should have a delay of at least one, since $\lambda_j[0] = 0$. We can also write,

$$x_j[k] = H_j(q^{-1}) e_j[k]$$

$$H_j(q^{-1}) = \frac{1}{1 + \Lambda_j(q^{-1})}$$

The marginal and conditional distributions are given by,

$$f\left(x_j[k]\right) = \frac{1}{\sqrt{2\pi\sigma_j^2 \sum_{l=0}^{k} h_j^2[l]}} \exp\left\{-\frac{1}{2\sigma_j^2 \sum_{l=0}^{k} h_j^2[l]} x_j^2[k]\right\}$$

$$f\left(x_j[k] | k-1\right) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{-\frac{1}{2\sigma_j^2}\left(x_j[k] + \sum_{l=0}^{k} \lambda_j[l] x_j[k-l]\right)^2\right\}$$

Here, $h_j[k]$ is the impulse response coefficients of $H_j(q^{-1})$. Note that $h_j[0] = 1$. We would require absolute convergence of $h_j[k]$ for $x_j[k]$ to be second-order stationary at large $k$.

**Synthesis or Generative model:**

The generative equation for the observation signal $\boldsymbol{y}[k]$ is,

$$\boldsymbol{y}[k] = \phi(\{\boldsymbol{x}[0], \boldsymbol{x}[1], \dots, \boldsymbol{x}[k]\}) + \boldsymbol{\varepsilon}[k]$$

Here, $\boldsymbol{\varepsilon}[k] \sim GWN(\boldsymbol{0},\ \sigma_\varepsilon^2 \boldsymbol{I}_{M\times M})$ and $\boldsymbol{x}[k] = \left[x_1[k], x_2[k], \dots, x_P[k]\right]^T$

In general, $\phi(.)$ can be a non-linear and dynamic function of $\boldsymbol{x}[k]$.

When $\phi(\{\boldsymbol{x}[0], \boldsymbol{x}[1], \dots, \boldsymbol{x}[k]\}) = \boldsymbol{W}\boldsymbol{x}[k]$, $\boldsymbol{W} \in \mathbb{R}^{M\times P}$, we get a linear generative model that does not use past latent variables to generate the observation signal (linear GSFA).

The conditional distribution is given by,

$$f(\boldsymbol{y}[k]|k-1) = \frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}} \exp\left\{-\frac{1}{2\sigma_\varepsilon^2} \|\boldsymbol{y}[k] - \phi(\{\boldsymbol{x}[0], \boldsymbol{x}[1], \dots, \boldsymbol{x}[k]\})\|^2\right\}$$

**Estimation of parameters:**

The parameters to be estimated are $\left\{\sigma_{j=1:P}^2,\ \left(\lambda_{j=1:P}[k]|k>0\right),\ \phi(.),\ \sigma_\varepsilon^2\right\}$. We can use variational inference to estimate the parameters as described in this paper. By parameterizing $\Lambda_j(q^{-1})$, we can reduce the number of parameters.

For example,

$$\Lambda_j(q^{-1}) = \frac{b_j q^{-1}}{1 - a_j q^{-1}}$$

Note that when $a_j = 0$ for all $j$, we obtain the probabilistic model for standard SFA. We should keep in mind that the estimated values for $\{a_{j=1:P},\ b_{j=1:P}\}$ should result in a stable and invertible $H_j(q^{-1})$. Hence, we should be solving a constrained (and non-linear) optimization problem.

**Numerical Simulation - Subspace Identification:**

**Simulation 1:**

The data generating process (DGP) is,

$$y[k] = -1.3986x_1[k] + \varepsilon[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.7q^{-1}}{1 - 0.5q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.6621)$ and $\varepsilon[k] \sim GWN(0, 0.1956)$

The number of datapoints generated, $N = 500$ and SNR is 10.

The estimated model is,

$$y[k] = 1.3555x_1[k] + \varepsilon[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.662q^{-1}}{1 - 0.2337q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.6508)$ and $\varepsilon[k] \sim GWN(0, 0.0818)$

The negative average loglikelihood as a function of iterations is shown below,

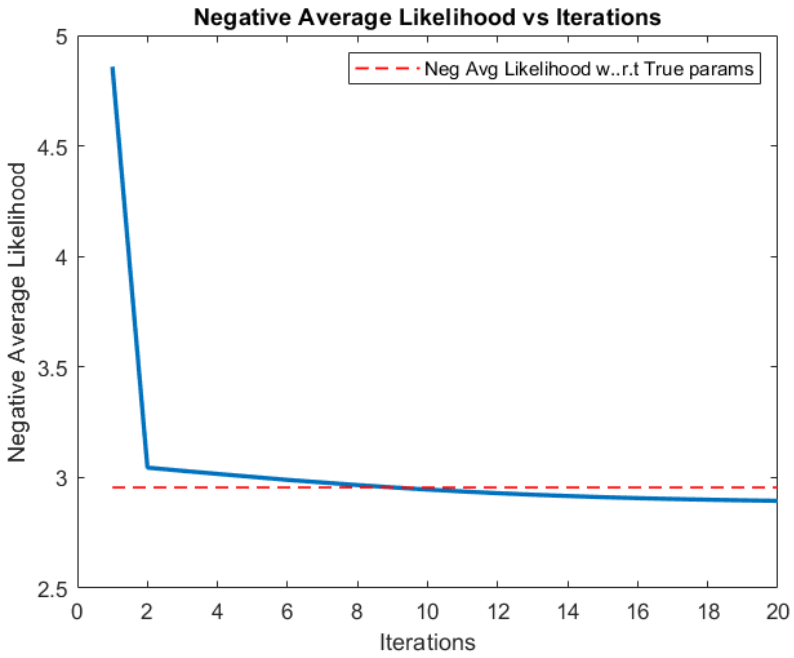


*Figure 1: Negative Average Likelihood vs Iterations*

The estimated states against the true values are shown below,

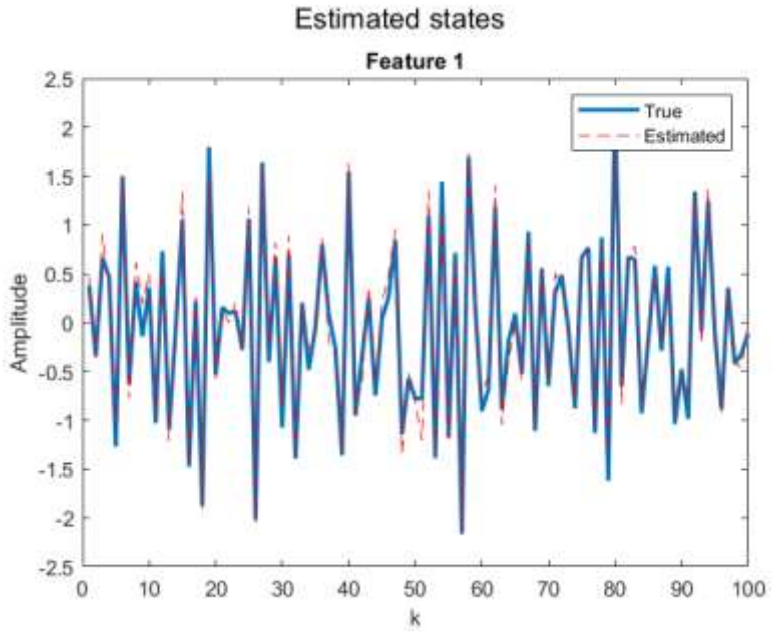


*Figure 2: Estimated vs True states*

**Simulation 2:**

The data generating process (DGP) is,

$$y[k] = 0.8205x_1[k] + \varepsilon[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.7q^{-1}}{1 - 0.5q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.6621)$ and $\varepsilon[k] \sim GWN(0, 0.0337)$

The number of datapoints generated, $N = 1500$ and SNR is 20.

The estimated model is,

$$y[k] = -0.7791x_1[k] + \varepsilon[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.7989q^{-1}}{1 - 0.4187q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.5727)$ and $\varepsilon[k] \sim GWN(0, 0.0092)$

The negative average loglikelihood as a function of iterations is shown below,

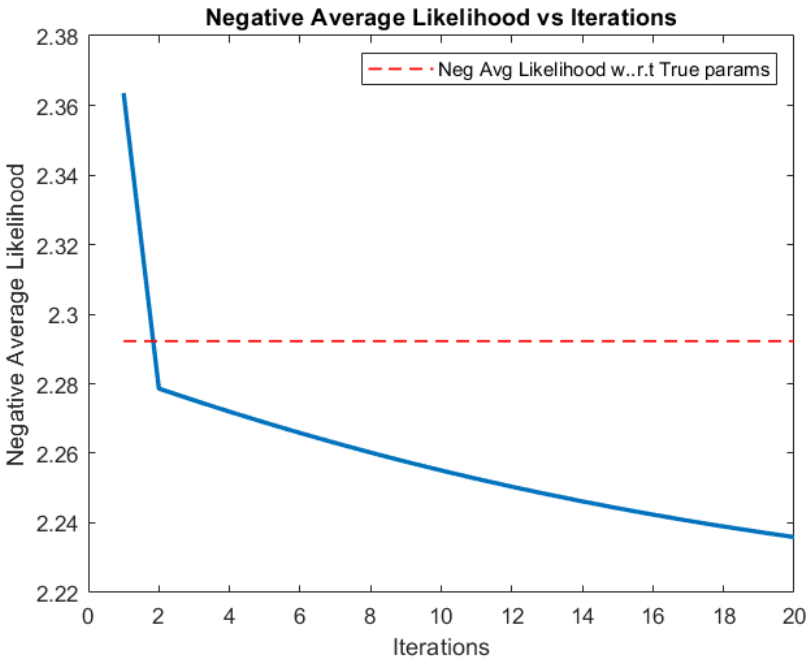


*Figure 3: Negative Average Likelihood vs Iterations*

The estimated states against the true values are shown below,

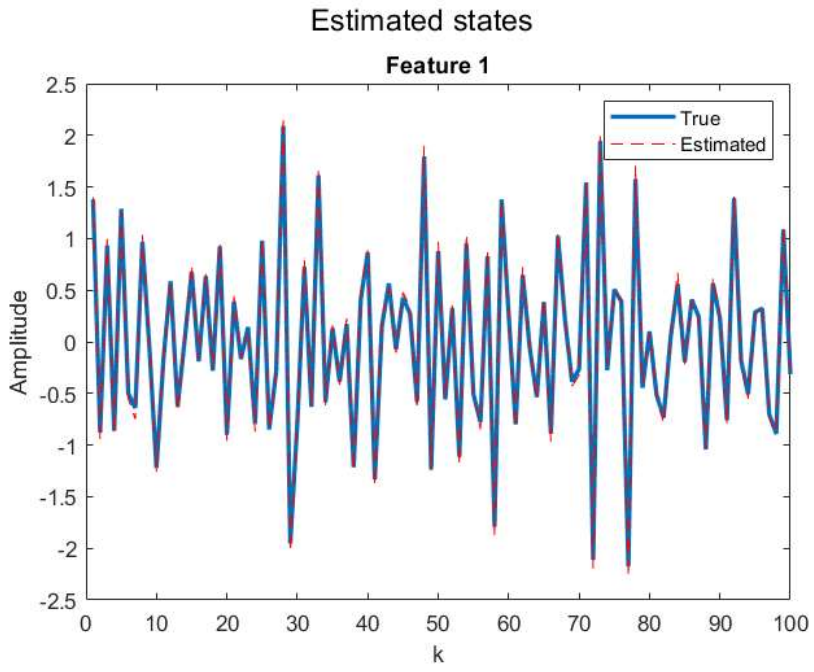


*Figure 4: Estimated vs True states*

**Simulation 3:**

The data generating process (DGP) is,

$$\boldsymbol{y}[k] = \begin{bmatrix} -0.734 & -0.023 \\ -0.368 & -1.30 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.7q^{-1}}{1 - 0.5q^{-1}}$$
$$\Lambda_2(q^{-1}) = \frac{0.2q^{-1}}{1 - 0.8q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.6621)$, $e_2[k] \sim GWN(0, 0.9412)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.0539\boldsymbol{I}_{2\times2})$

The number of datapoints generated, $N = 200$ and SNR is 10.

The estimated model is,

$$\boldsymbol{y}[k] = \begin{bmatrix} -0.173 & -0.407 \\ 0.7983 & -0.816 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.1728q^{-1}}{1 - 0.6318q^{-1}}$$
$$\Lambda_2(q^{-1}) = \frac{0.318q^{-1}}{1 - 0.3648q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.963)$, $e_2[k] \sim GWN(0, 0.908)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.1641\boldsymbol{I}_{2\times2})$

The negative average loglikelihood as a function of iterations is shown below,

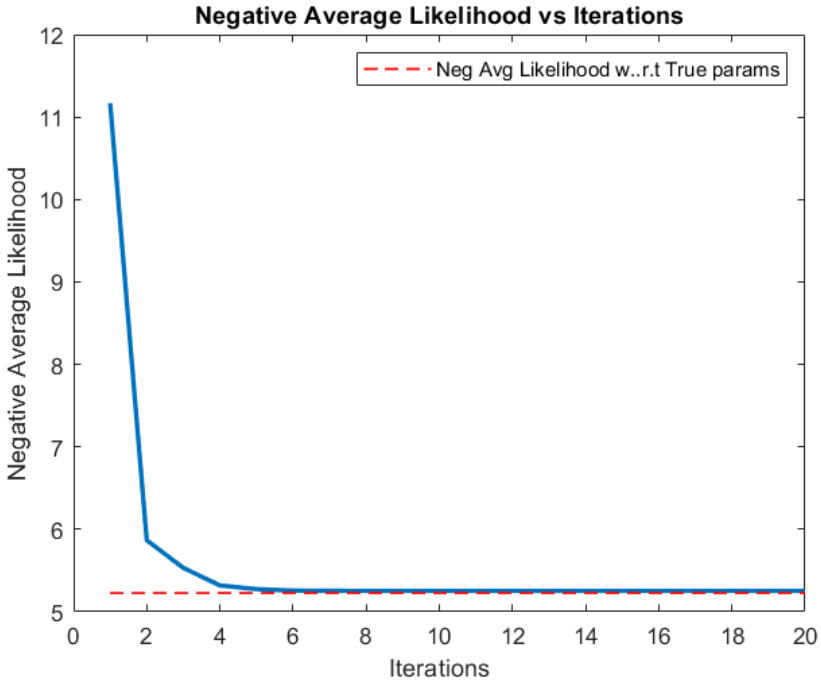


*Figure 5: Negative Average Likelihood vs Iterations*

The estimated states against the true values are shown below,

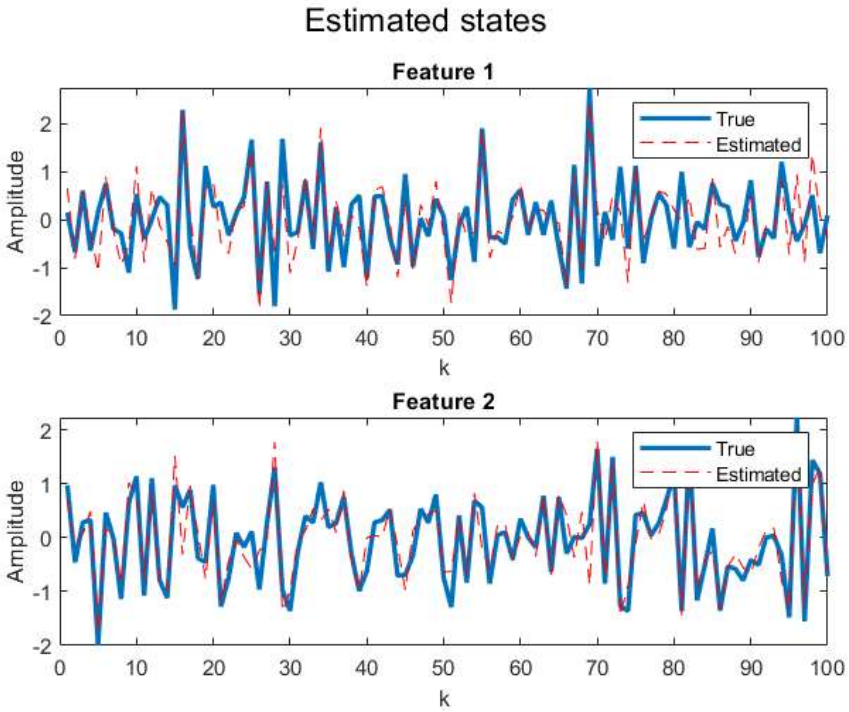


*Figure 6: Estimated vs True states*

**Simulation 4:**

The data generating process (DGP) is,

$$\boldsymbol{y}[k] = \begin{bmatrix} -0.734 & -0.023 \\ -0.368 & -1.30 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.7q^{-1}}{1 - 0.5q^{-1}}$$
$$\Lambda_2(q^{-1}) = \frac{0.2q^{-1}}{1 - 0.8q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.6621)$, $e_2[k] \sim GWN(0, 0.9412)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.0539\boldsymbol{I}_{2\times2})$

The number of datapoints generated, $N = 1000$ and SNR is 10.

The estimated model is,

$$\boldsymbol{y}[k] = \begin{bmatrix} -0.4547 & -0.117 \\ -0.8383 & 0.946 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.324q^{-1}}{1 - 0.102q^{-1}}$$
$$\Lambda_2(q^{-1}) = \frac{0.1266q^{-1}}{1 - 0.695q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.901)$, $e_2[k] \sim GWN(0, 0.977)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.1758\boldsymbol{I}_{2\times2})$

The negative average loglikelihood as a function of iterations is shown below,

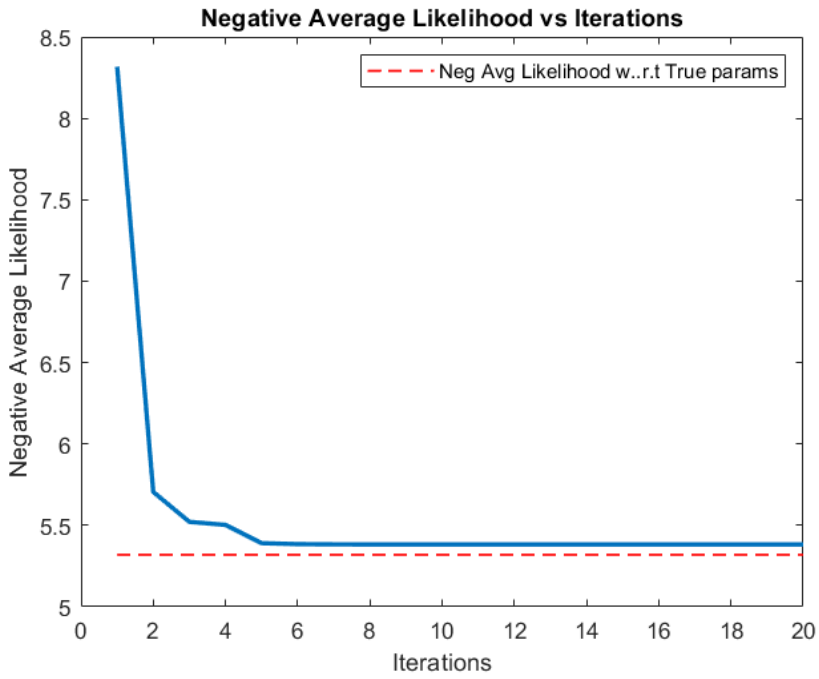


*Figure 7: Negative Average Likelihood vs Iterations*

The estimated states against the true values are shown below,

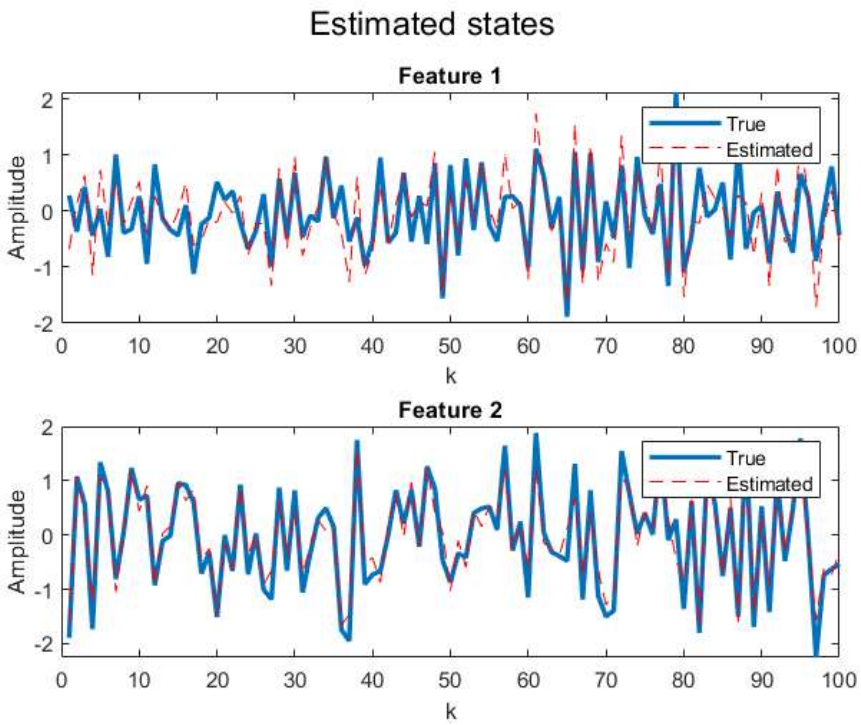


*Figure 8: Estimated vs True states*

**Simulation 5:**

The data generating process (DGP) is,

$$\boldsymbol{y}[k] = \begin{bmatrix} 0.867 & 0.1154 & 0.517 \\ 1.059 & -0.925 & 0.829 \\ -0.735 & 1.411 & -0.710 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$
$$x_3[k] = -\Lambda_3(q^{-1})x_3[k] + e_3[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.7q^{-1}}{1 - 0.5q^{-1}}$$
$$\Lambda_2(q^{-1}) = \frac{0.2q^{-1}}{1 - 0.8q^{-1}}$$
$$\Lambda_3(q^{-1}) = \frac{0.5q^{-1}}{1 - 0.1q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.6621)$, $e_2[k] \sim GWN(0, 0.9412)$, $e_3[k] \sim GWN(0, 0.7706)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.1033\boldsymbol{I}_{3\times3})$

The number of datapoints generated, $N = 200$ and SNR is 10.

The estimated model is,

$$\boldsymbol{y}[k] = \begin{bmatrix} -0.304 & 0.241 & -0.799 \\ -0.688 & -0.639 & -1.301 \\ 0.709 & 1.135 & 1.136 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$
$$x_3[k] = -\Lambda_3(q^{-1})x_3[k] + e_3[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.068q^{-1}}{1 - 0.068q^{-1}}$$

$$\Lambda_2(q^{-1}) = \frac{0.293q^{-1}}{1 - 0.561q^{-1}}$$

$$\Lambda_3(q^{-1}) = \frac{0.465q^{-1}}{1 - 0.251q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.9953)$, $e_2[k] \sim GWN(0, 0.9149)$, $e_3[k] \sim GWN(0, 0.8150)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.018\boldsymbol{I}_{3\times3})$

The negative average loglikelihood as a function of iterations is shown below,

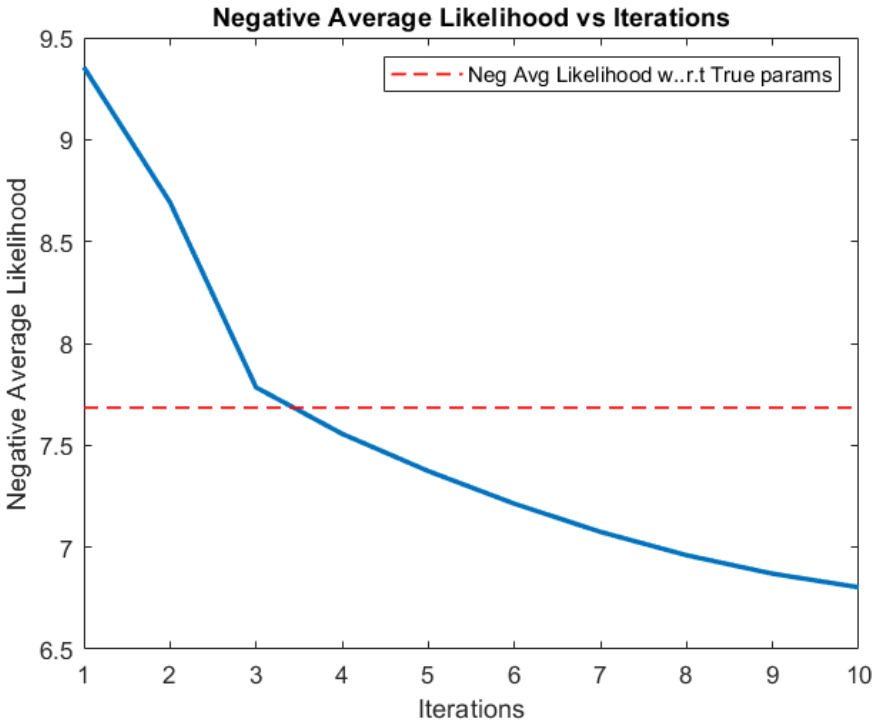


*Figure 9: Negative Average Likelihood vs Iterations*

The estimated states against the true values are shown below,

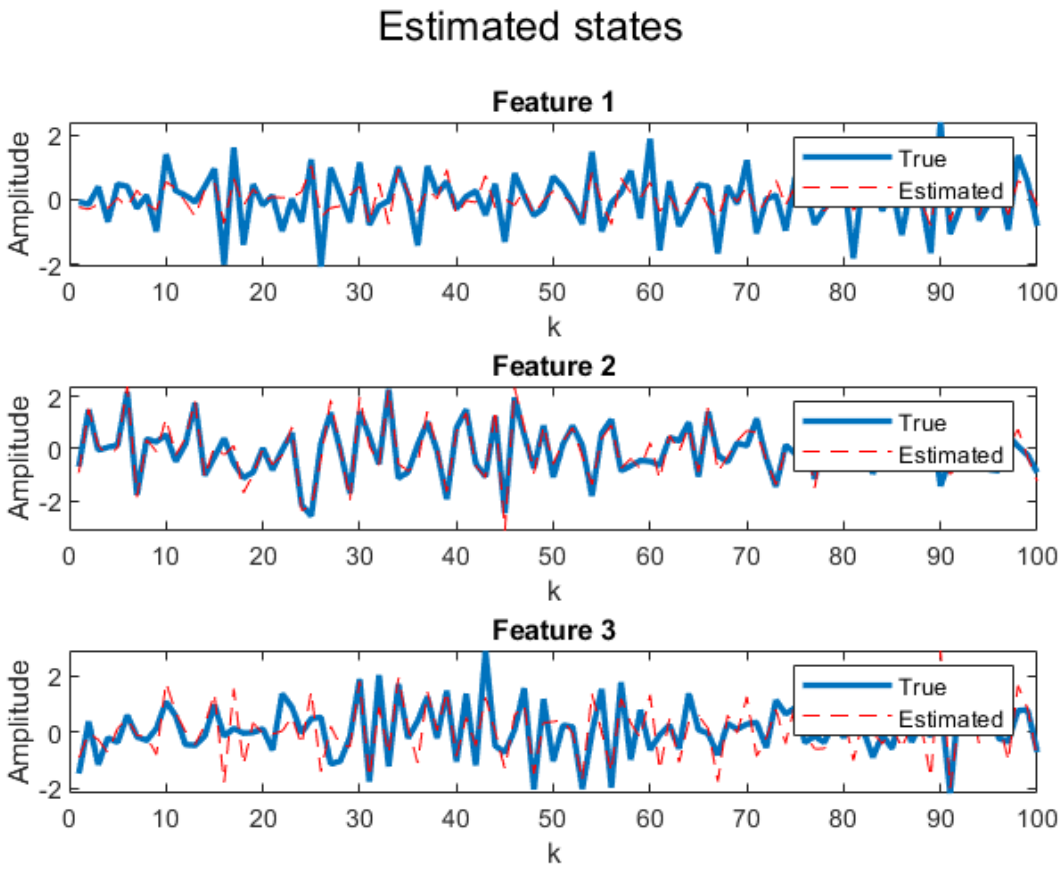


*Figure 10: Estimated vs True states*

**Simulation 6:**

The data generating process (DGP) is,

$$\boldsymbol{y}[k] = \begin{bmatrix} 1.539 & -0.742 & 0.783 \\ 0.467 & -0.664 & -0.861 \\ 0.786 & -0.894 & -0.428 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$
$$x_3[k] = -\Lambda_3(q^{-1})x_3[k] + e_3[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.7q^{-1}}{1 - 0.5q^{-1}}$$
$$\Lambda_2(q^{-1}) = \frac{0.2q^{-1}}{1 - 0.8q^{-1}}$$
$$\Lambda_3(q^{-1}) = \frac{0.5q^{-1}}{1 - 0.1q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.6621)$, $e_2[k] \sim GWN(0, 0.9412)$, $e_3[k] \sim GWN(0, 0.7706)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.2802\boldsymbol{I}_{3\times3})$

The number of datapoints generated, $N = 200$ and SNR is 5.

The estimated model is,

$$\boldsymbol{y}[k] = \begin{bmatrix} 1.393 & 0.426 & 1.714 \\ -0.422 & 0.857 & 0.392 \\ 0.076 & 0.891 & 0.897 \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\varepsilon}[k]$$

$$x_1[k] = -\Lambda_1(q^{-1})x_1[k] + e_1[k]$$
$$x_2[k] = -\Lambda_2(q^{-1})x_2[k] + e_2[k]$$
$$x_3[k] = -\Lambda_3(q^{-1})x_3[k] + e_3[k]$$

$$\Lambda_1(q^{-1}) = \frac{0.361q^{-1}}{1 + 0.276q^{-1}}$$

$$\Lambda_2(q^{-1}) = \frac{0.252q^{-1}}{1 - 0.427q^{-1}}$$

$$\Lambda_3(q^{-1}) = \frac{0.302q^{-1}}{1 - 0.257q^{-1}}$$

Here, $e_1[k] \sim GWN(0, 0.8191)$, $e_2[k] \sim GWN(0, 0.9385)$, $e_3[k] \sim GWN(0, 0.9164)$, and $\boldsymbol{\varepsilon}[k] \sim GWN(\mathbf{0}, 0.0871\boldsymbol{I}_{3\times3})$

The negative average loglikelihood as a function of iterations is shown below,

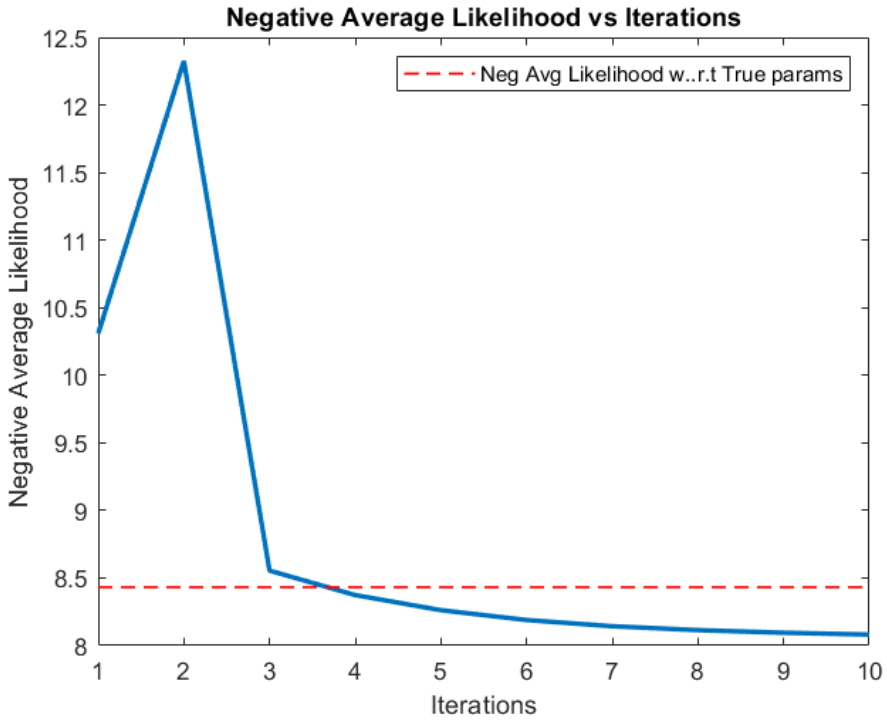


*Figure 11: Negative Average Likelihood vs Iterations*

The estimated states against the true values are shown below,

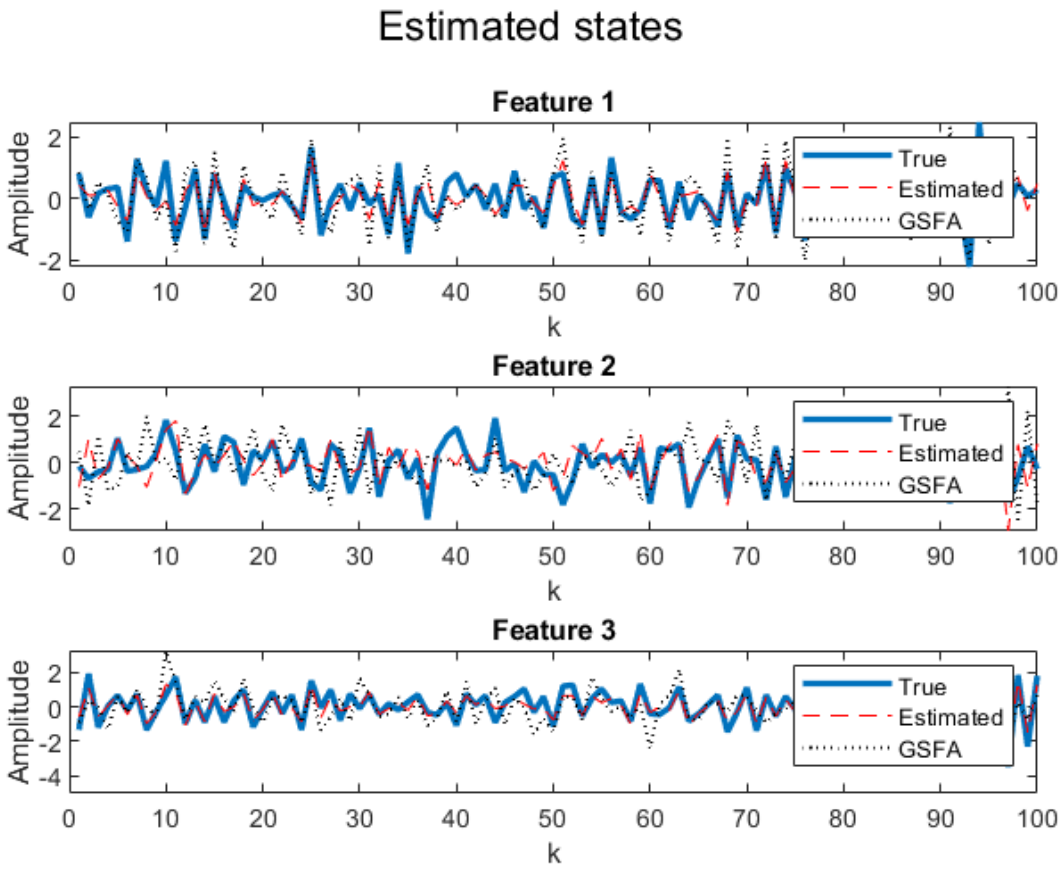


*Figure 12: Estimated vs True states*

**German Traffic Sign Recognition Benchmark (GTSRB):**

The GTSRB consists of photographs of 43 different traffic signs taken on German roads under uncontrolled conditions with variations in lighting, sign size, and distance. It has 39,209 labeled images for training and 12,630 images without label for evaluation.

**Preprocessing:**

The images were resized to 64 x 128 dimension and the HOG features were computed (which resulted in a vector of size 3780). We used PCA to further reduce the dimensionality to 200 (as it explained >75% of the total variance of the training data). To employ methods like SFA, GSFA, probabilistic GSFA, we sort the training data (and not the test data) according to their class labels.

**Methods:**

**SFA:**

We use a two-layer, cascaded, SFA with non-linear expansion (a compact expansion that only doubles the data dimension was employed, $\boldsymbol{x}^T \rightarrow \boldsymbol{x}^T, (|\boldsymbol{x}|^{0.8})^T$, where the absolute value and exponent 0.8 are computed component-wise). At each layer, we retain only the top 400 slow features. Once the slow features are computed, classification head is trained on it.

**GSFA:**

We use a two-layer, cascaded, GSFA with non-linear expansion (a compact expansion that only doubles the data dimension was employed, $\boldsymbol{x}^T \rightarrow \boldsymbol{x}^T, (|\boldsymbol{x}|^{0.8})^T$, where the absolute value and exponent 0.8 are computed component-wise). The filter matrix $\boldsymbol{G}$ is constructed from $G(q^{-1}) = 1 + \sum_{i=1}^{79} q^{-i}$. At each layer, we retain only the top 400 slow features. Once the slow features are computed, classification head is trained on it.

**Probabilistic GSFA:**

The parameterization of the filters is as follows:

$$H_j(q^{-1}) = \frac{1}{1 + b_j q^{-1} + d_j q^{-2}}$$

Here, $j \in \{1, \dots, P\}$ and $P = 1200$.

**Classification head:**

We use the Gaussian Naïve Bayes (GNB) classifier for the classification head.

**Evaluation metrics:**

Taking into account the fact that the dataset is imbalanced, we report the following classification metrics:

a) Accuracy

b) Weighted F1 score

We also report the above scores for a baseline model, which is simply the Gaussian Naïve Bayes classifier trained on the raw data (without any extraction of slow features).

**Results on the test data:**

| | **Accuracy** | | | **Weighted F1 score** | | |
|---|---|---|---|---|---|---|
| | **Value of M** | | | **Value of M** | | |
| | **200** | **300** | **500** | **200** | **300** | **500** |
| **Baseline** | 0.8204 | 0.8154 | 0.7950 | 0.8225 | 0.8187 | 0.8030 |
| **SFA** | 0.8359 | 0.8379 | 0.8248 | 0.8399 | 0.8419 | 0.8299 |
| **GSFA** | 0.8581 | 0.8555 | 0.8482 | 0.8636 | 0.8617 | 0.8566 |
| **Prob. GSFA** | 0.8283 | - | - | 0.8312 | - | - |