MVA – Probabilistic Graphical Models

Homework 2

Wilson Jallet*

December 12, 2019

1 K-means and the EM algorithm

Question 1. We consider a mixture model of K components for a dataset (X_i) where $X_i \in \mathbb{R}^d$. We denote $Z_i \in \{1, \dots, K\}$ the latent hidden label. Each component occurs with probability $p_k = \mathbb{P}(Z_i = k)$ and is distributed as

$$X_i \sim \mathcal{N}(\mu_k, D_k)$$

i.e. $p(x|k) = \frac{1}{((2\pi)^d |D_k|)^{1/2}} \exp(-\frac{1}{2}(x-\mu_k)^T D_k^{-1}(x-\mu_k))$.

The data log-likelihood under parameters $\Theta = ((p_1, \mu_1, D_1), \dots, (p_K, \mu_K, D_K))$ is

$$\mathcal{L}(X_1, \dots, X_n; \Theta) = \sum_{i=1}^n \log\left(\sum_{k=1}^K p_k p(X_i|k; \mu_k, D_k)\right)$$

We seek to compute the MLE

$$\widehat{\Theta} \in \operatorname*{argmax}_{\Theta} \mathcal{L}(X_1, \dots, X_n; \Theta)$$

This optimization problem is intractable when using straightforward methods.

The EM algorithm goes as follows:

- *Expectation* Compute the posterior probability of the latent variables Z_i:

$$q_{k,i}^{(t)} = p(Z_i = k | X_i; \Theta^{(t)}) = \frac{p_k^{(t)} p(X_i | Z_i = k; \Theta^{(t)})}{\sum_{\ell=1}^K p_\ell^{(t)} p(X_i | Z_i = \ell; \Theta^{(t)})} \tag{1}$$

  and denote $w_k^{(t)} = \sum_{i=1}^n q_{k,i}^{(t)}$ – we then have $\sum_{k=1}^K w_k^{(t)} = n$.

- *Maximization* Update the parameters $\Theta^{(t)}$ by maximizing the lower bound objective:

$$\max_{\Theta} \mathcal{J}(q^{(t)}, \Theta) = \sum_{i=1}^n \left(\sum_{k=1}^K q_{k,i}^{(t)} \left(\log p_k - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|D_k| - \frac{1}{2}(X_i - \mu_k)^T D_k^{-1}(X_i - \mu_k)\right)\right) \tag{2}$$

*wilson.jallet@polytechnique.org

subject to $\sum_{k=1}^{K} p_k = 1$ (associated to a multiplier ν). The KKT conditions give the null gradient condition:

$$\frac{1}{p_k} w_k^{(t)} - \nu = 0 \tag{3a}$$

$$-\sum_{i=1}^{n} q_{k,i}^{(t)} D_k^{-1}(\mu_k - X_i) = 0 \tag{3b}$$

$$-\frac{1}{2}\sum_{i=1}^{n} q_{k,i}^{(t)} (D_k^{-1} - D_k^{-2}\operatorname{diag}(X_i - \mu_k)^2) = 0 \tag{3c}$$

Which leads to the updates:

$$p_k = \frac{1}{n} w_k^{(t)} \tag{4a}$$

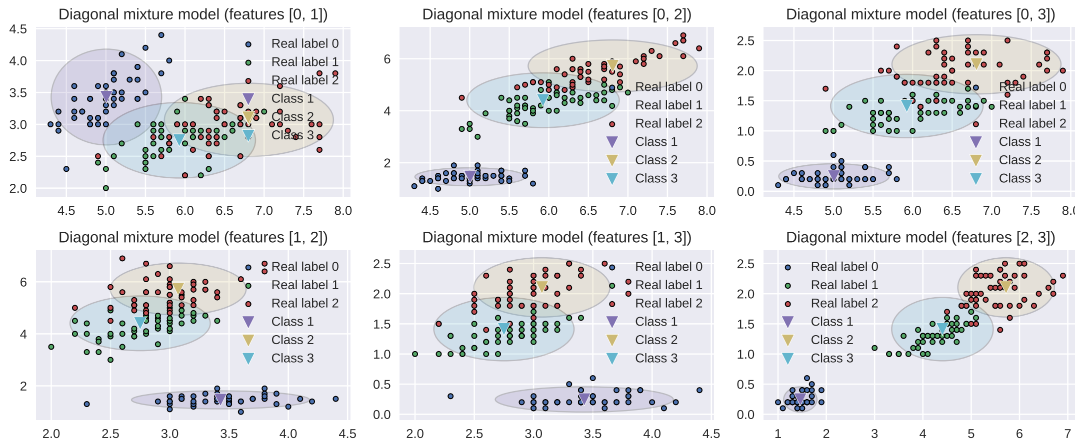
$$\mu_k = \frac{1}{w_k^{(t)}} \sum_{i=1}^{n} q_{k,i}^{(t)} X_i \tag{4b}$$

$$D_k = \frac{1}{w_k^{(t)}} \sum_{i=1}^{n} q_{k,i}^{(t)} \operatorname{diag}(X_i - \mu_k)^2 \tag{4c}$$

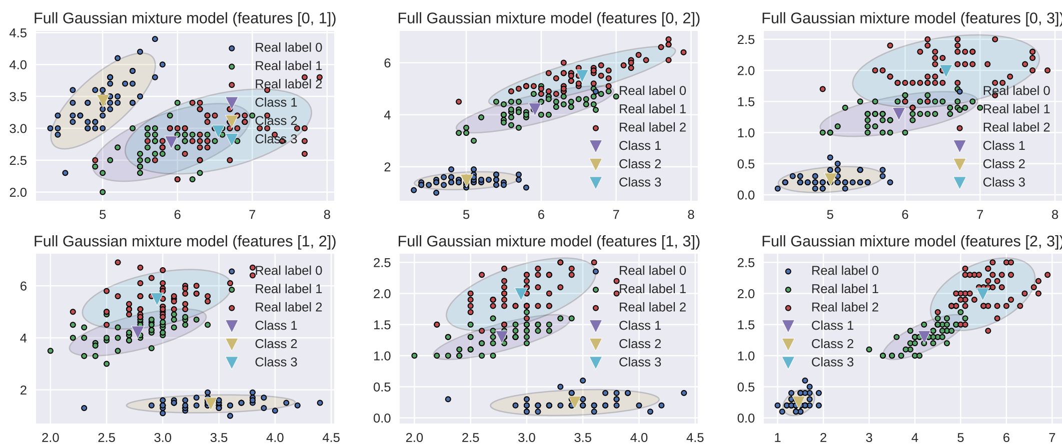
Question 2. The main advantage of this "reduced" covariance mixture model is that it is more sparse: it uses far fewer parameters ($K(2d + 1)$) than the its full counterpart, which has $K(1 + d + d(d + 1)/2)$ parameters. For datasets with relatively independent features (conditionally on the latent class), this can give performance very close to the full covariance while having a smaller, simpler model (meaning better AIC or BIC scores).

Question 3. Figure 1 compares the obtained latent class centroids and confidence ellipsoids (where applicable) for the diagonal and full covariance mixture models and K-means, on the Iris dataset, for a small number of classes $K = 3$ (the actual number of classes in the data). Figures 2 and 3 represent the same for $K = 2, 4$ classes.

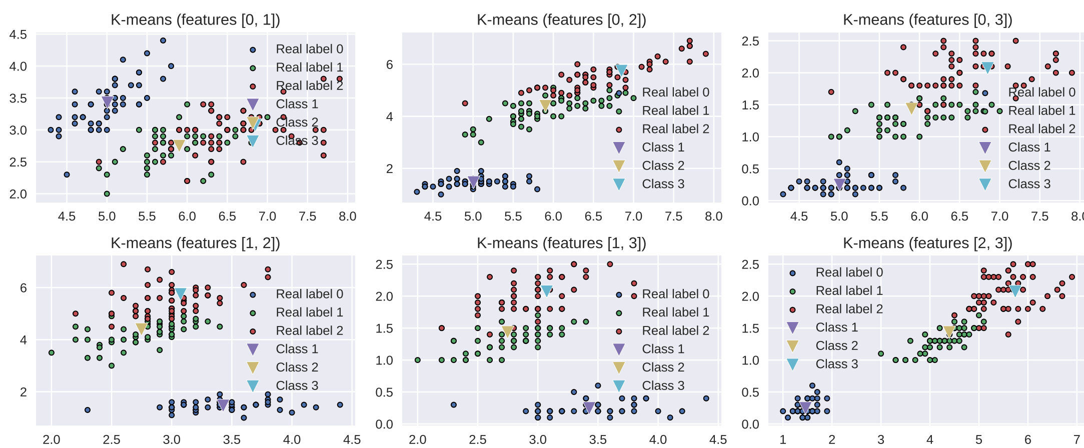
Question 4. I created synthetic data made up of an ellipsis and annulus: Figure 4 shows how the mixture models and K-means compare.



(a) Diagonal Gaussian mixture model on the Iris dataset (our implementation). $K = 3$ classes.

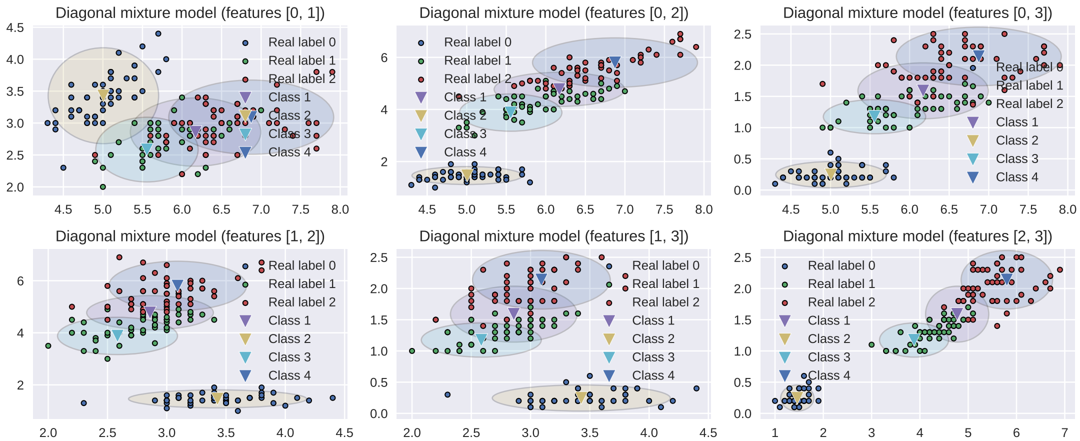


(b) Full Gaussian mixture on the Iris dataset using `scikit-learn`. $K = 3$ classes.

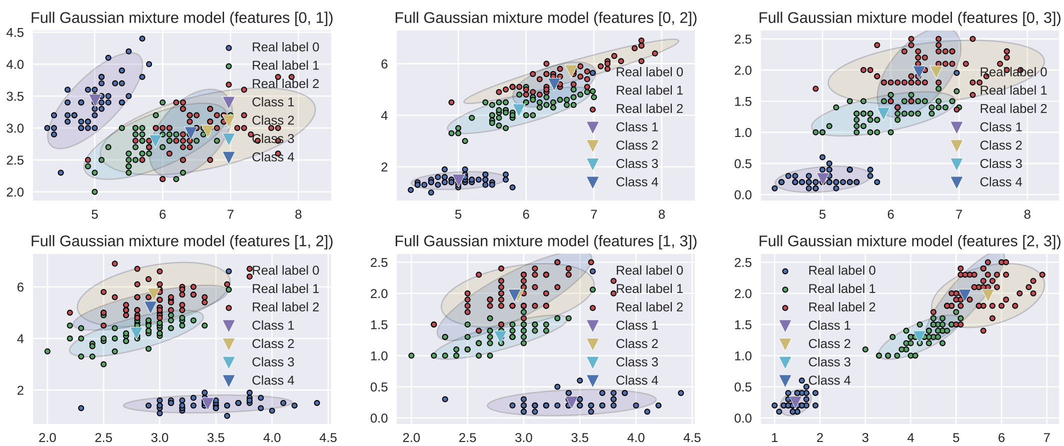


(c) Centroids of the K-means model.

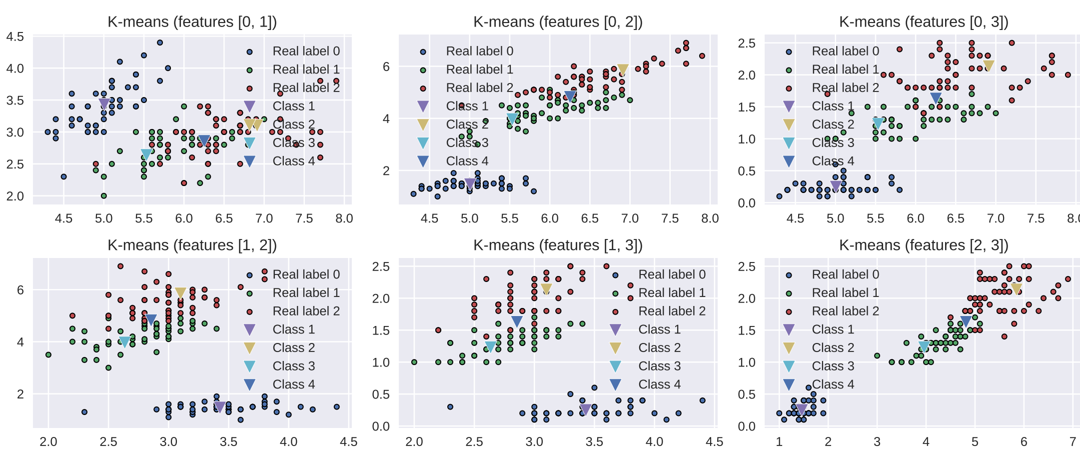
Figure 1: Comparison of the diagonal and full covariance mixture models and K-means for $K = 3$ classes.



(a) Diagonal Gaussian mixture model on the Iris dataset (our implementation).

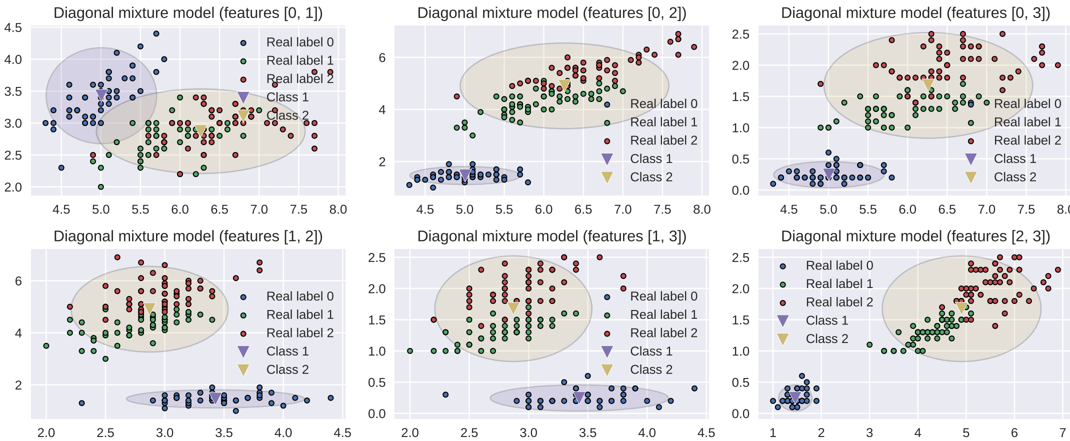


(b) Full Gaussian mixture on the Iris dataset using `scikit-learn`.

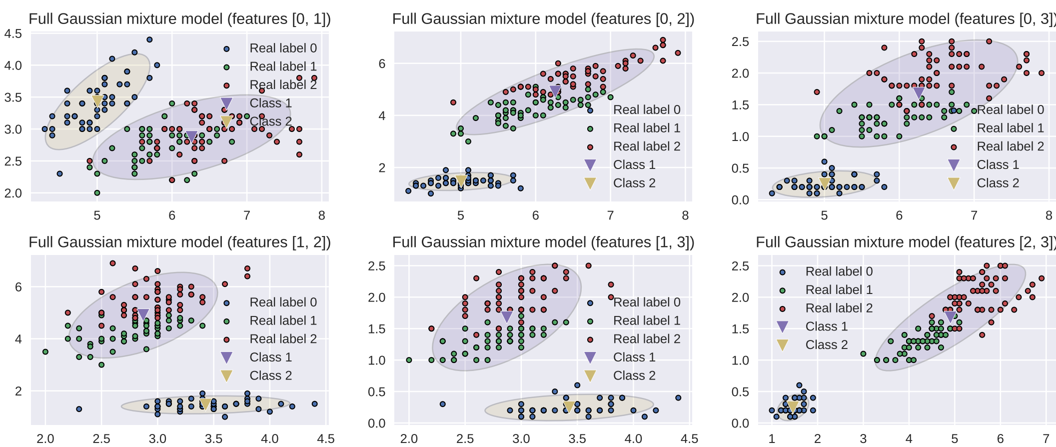


(c) K-means.

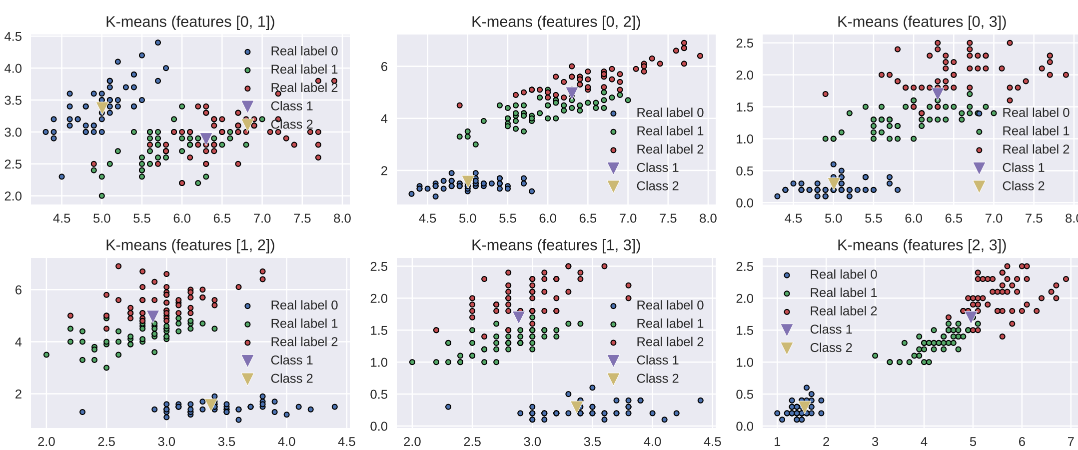
Figure 2: Comparison of the models for $K = 4$ classes.



(a) Diagonal Gaussian mixture model on the Iris dataset (our implementation).

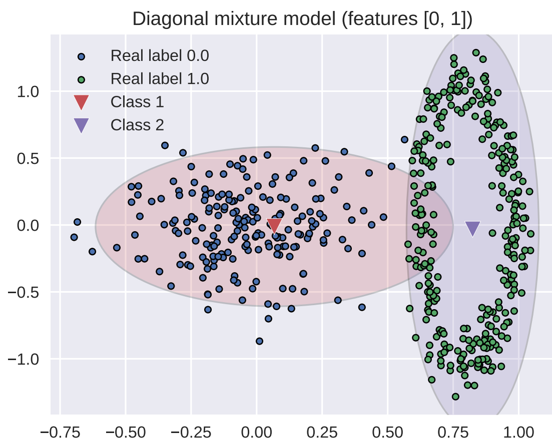


(b) Full Gaussian mixture on the Iris dataset using `scikit-learn`.

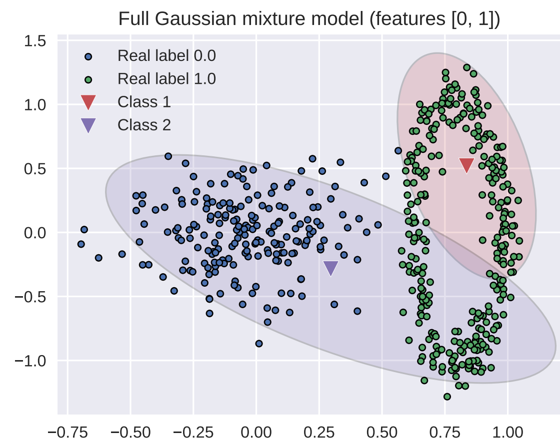


(c) Full Gaussian mixture on the Iris dataset using `scikit-learn`.

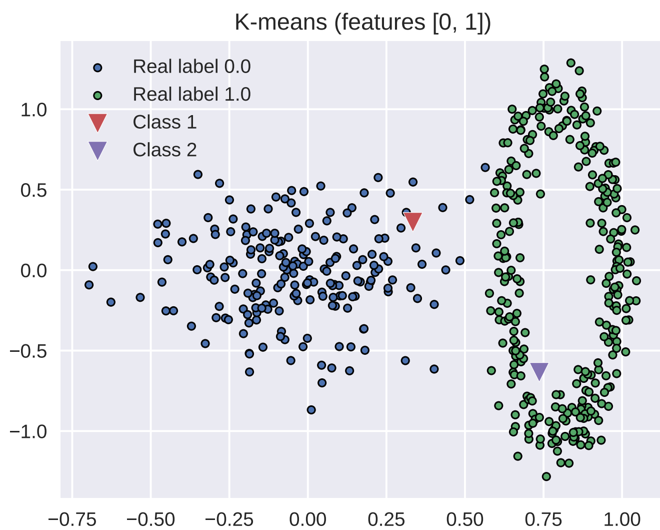
Figure 3: Comparison of the models for $K = 2$ classes.



(a) Diagonal Gaussian mixture model.



(b) Full Gaussian mixture model.



(c) Result of the K-means model: notice how the class centroids are offset from the actuel center of the class.

Figure 4: Comparison of the models on synthetic data where K-means performs worse than mixtures.

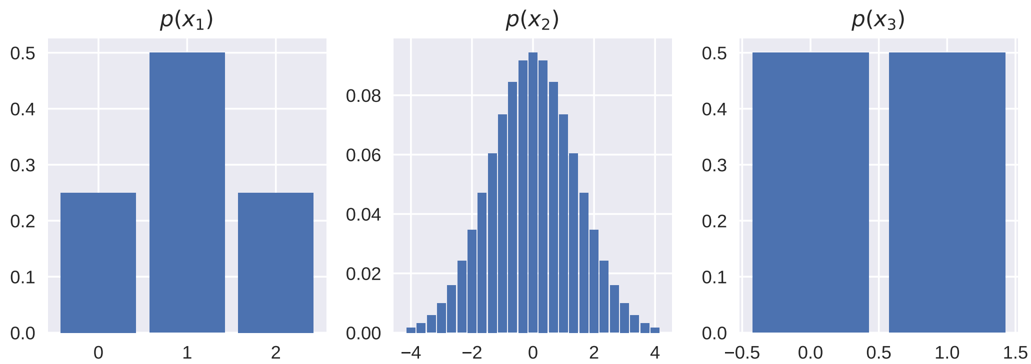


Figure 5: Sanity test: undirected graph with $n = 3$ nodes, and a probability distribution p with factors $\psi_1 = (1, 2, 1)$, $\psi_2(x_2) = \exp(-x_2^2/4)$, $\psi_3 = (1, 1)$ and $\psi_{i,i+1} = 1$ (i.e. independence).

2 Graphs, algorithms and Ising

Question 1. We recall that for an undirected chain graph G probability distributions factor as

$$p(x) = \frac{1}{Z} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=1}^{n-1} \psi_{i,i+1}(x_i, x_{i+1}) \tag{5}$$

The algorithm. The marginal distribution of X_i can be rewritten as

$$p(x_i) = \frac{1}{Z} \mu_{i-1,i}(x_i) \psi_i(x_i) \mu_{i+1,i}(x_i)$$

where $\mu_{i-1,i}, \mu_{i+1,i}$ are (forward, backward) messages from $i-1$ to i and $i+1$ to i. They are propagated as:

$$\mu_{i,i+1}(x_{i+1}) = \sum_{x_i} \psi_i(x_i) \psi_{i,i+1}(x_i, x_{i+1}) \mu_{i-1,i}(x_i) \tag{6a}$$

$$\mu_{i,i-1}(x_{i-1}) = \sum_{x_i} \psi_i(x_i) \psi_{i-1,i}(x_{i-1}, x_i) \mu_{i+1,i}(x_i) \tag{6b}$$

Practical implementation. If the state space $\mathcal{X}$ of the variables $X_1, \ldots, X_n$ (for instance for binary variables) is discrete we can represent the input functions ψ_i and $\psi_{i,i+1}$ as arrays. If not (continuous variables for instance), we can discretize a grid over $\mathcal{X}$ and precompute an array of values for the factors. Denoting the arrays in bold letters, we forward-propagate by

$$\boldsymbol{\mu}_{i,i+1} = (\boldsymbol{\mu}_{i-1,i} \odot \boldsymbol{\psi}_i) \boldsymbol{\psi}_{i,i+1}$$

and back-propagate by

$$\boldsymbol{\mu}_{i,i-1} = (\boldsymbol{\mu}_{i,i+1} \odot \boldsymbol{\psi}_i) \boldsymbol{\psi}_{i-1,i}^T$$

which allows to compute the marginal distributions using vectorized operations.

Figure 5 shows an implementation for independent edges. The expected marginals are $X_1 \sim \mathcal{M}(1; 1/4, 1/2, 1/4)$, $X_2 \sim \mathcal{N}(0, 2)$ and $X_3 \sim \mathcal{B}(1/2)$.

Question 2. The vertex set V of the graph $G = (V, E)$ induced by the grid has vertices of the form $v = (j, k)$ where $1 \le j \le w$ and $1 \le k \le h$. An easy junction tree to extract from G is given by collapsing the rows into supernodes: the resulting tree T has vertex set $V_T = \{c_k\}$ where $c_k = \{(j, k)\}_j$ and looking at the resulting edges shows T is actually an undirected chain $c_1 - c_2 - \cdots - c_h$. Computationally, since there are $w = 10$ columns and $h = 100$ rows, this leads to a reasonably-sized state space (2^{10} different states) for each supernode.

$$p(x) = \frac{1}{Z} \prod_k \tilde{\psi}_k(x_{c_k}) \prod_k \tilde{\psi}_{k,k+1}(x_{c_k}, x_{c_{k+1}})$$

where

$$\tilde{\psi}_k(x_{c_k}) = \prod_{j=1}^{w} \psi_{(j,k)}(x_{(j,k)}) \prod_{j=1}^{w-1} \psi_{(j,k),(j+1,k)}(x_{(j,k)}, x_{(j+1,k)})$$

$$\tilde{\psi}_{k,k+1}(x_{c_k}, x_{c_{k+1}}) = \prod_{j=1}^{w} \psi_{(j,k),(j,k+1)}(x_{(j,k)}, x_{(j,k+1)})$$