Exercise 1.1

Algorithm was implemented in Ipython notebook (attached)

I vectorized all of the operations over n samples to increase efficiency.

Thus, vectorized version of line 3 (finding $\log(r_{ik})$):

$$\boxed{\log_{:,k} = \log(\pi_k) - \frac{1}{2}\sum_{i=1}^{d}\log(\underset{\sim}{S_k}) - \frac{1}{2}(X - \mu_k)^2 \cdot \frac{1}{\underset{\sim}{S_k}}}$$

pointwise

broadcasted

This simplification is possible because S_k is a diagonal matrix (which was represented as d-dim vector).

Also, since S_k is a diagonal matrix, updating formula for it becomes

$$S_j = \frac{\sum_{i=1}^{n} r_{ik}(x_{ij} - \mu_j)^2}{\sum_{i=1}^{n} r_{ik}} = \frac{\sum_{i=1}^{n} r_{ik}\, x_{ij}^2}{\sum_{i=1}^{n} r_{ik}} - \mu_j$$

Derivation:

$$\min \sum_{i=1}^{n}\sum_{k=1}^{k} r_{ik}\left[-\log\pi_k + \frac{1}{2}\log|S_k| + \frac{1}{2}(x_i - \mu_k)^T S^{-1}(x_i - \mu_k)\right] =$$

$$= \min \sum_{i=1}^{n}\sum_{k=1}^{K} r_{ik}\left[-\log\pi_k + \frac{1}{2}\log(S_{k_1}\cdot\ldots\cdot S_{k_d}) + \frac{S_{k_i}^{-1}}{2}(x_i - \mu_k)(x_i - \mu_k)\right]$$

$$\frac{d}{dS_{k_j}} = \sum_{i=1}^{n} r_{ik}\left[\frac{1}{2}\,\frac{\cancel{S_{k_1}}\ldots \cancel{S_{k_{j-1}}}\ \cancel{S_{k_{j+1}}}\ \cancel{S_{k_d}}}{\cancel{S_{k_1}}\ldots S_{k_j}\ldots \cancel{S_{k_d}}} - \frac{S_k^{-2}}{2}(x_{ij} - \mu_{k_j})\right] \Rightarrow$$

$$\sum_{i=1}^{n} r_{ik}\left[\frac{1}{2}\,\frac{1}{S_{k_j}} - \frac{S^{-2}}{2}(x_{ij} - \mu_k)^2\right] = 0 \quad \Big|\cdot S_{k_j}^2$$

$$\sum_{i=1}^{n} r_{ik}\cdot S_{k_j} = \sum_{i=1}^{n} r_{ik}(x_{ij} - \mu_{k_j})^2$$

doesn't depend on i

$$\boxed{S_{k_j} = \frac{\sum_{i=1}^{n} r_{ik}(x_{ij} - \mu_{k_j})^2}{\sum_{i=1}^{n} r_{ik}}} = \frac{\sum_{i=1}^{n} r_{ik}\, x_{ij}^2 - \sum_{i=1}^{n} r_{ik}\,\mu_{k_j}^2}{\sum_{i=1}^{n} r_{ik}} = \boxed{\frac{\sum_{i=1}^{n} r_{ik}\, x_{ij}^2}{\sum_{i=1}^{n} r_{ik}} - \mu_{k_j}^2}$$

__Space complexity__

The algorithm needs to store $\overset{R^{k\times d}}{\mu}, \overset{R^{k\times d}}{S}, \overset{R^{n\times k}}{r}$ - matrices and $\overset{R^{1\times k}}{\pi}$ - vector. Therefore, space complexity is $\boxed{O(kd + nk)}$

__Time complexity__

$\boxed{O(nkd^2)}$ for updating responsibility matrix r, which is the most expensive operation

$\boxed{\text{Exercise 1.2}}$

I was using 45,000 samples for training because for some reason jupyter notebook was crushing when I tried to do PCA on a larger data set.

I reduced dimensions to 100 with PCA.

$\boxed{\text{For } k = 5, \text{ accuracy} = 14.371\,\%}$ on the test set.