IE 581 - Simulation Methods for Statistical Inference Term Project Report

Gibbs Sampling for Clustering

An implementation on
Iris Data Set

Submitted by

Soner Aydın
Özgün Elçi
Süleyman Topdemir

Under the guidance of
Assist. Prof. Sinan Yıldırım

Faculty of Engineering and Natural Sciences
SABANCI UNIVERSITY
Istanbul

Fall 2016

Gibbs Sampling for Clustering

Submitted on January 12, 2017

1 Introduction

The project covers implementations of two different algorithms which are Gibbs sampling algorithm and Expectation-Maximization algorithm with the purpose of clustering a multivariate data set regarding class of each sample in the data set. Gaussian Mixture Model is assumed for determining what kind of structure/distribution the data set has. Based on the model, the required conditional probability distributions for each unknown parameters is derived from the joint distribution to conduct the Gibbs sampling algorithm. Also the Bayesian approaches are put into practiced to implement Expectation-Maximization algorithm. Relatively, high accuracy values are achieved for two algorithms in the cluster analysis.

Cluster analysis is one of the most common tasks in the machine learning-related fields. The main objective in this analysis is, basically, to group a set of objects which can be coordinates or indicates a point that has more than two features in such a way that the similar objects should be assigned to the same cluster. In the clustering tasks, the labeled data are not given (Everitt, 1977). Clustering is also known as unsupervised classification because of this case. The main difference between clustering and classification is that whether or not you know prior knowledge of classes in the data set. In this project, the Iris flower data set was used as an input for both algorithms. Normally this data set is one of the labeled ones, which means that each sample has a labeled output attribute. Notwithstanding the labeled data, the labels were not used in the algorithms, however, the class information were utilized to calculate the accuracy values at the end.

In data mining field, several algorithms such as k-means algorithm, DBSCAN, fuzzy c-means are commonly used to obtain appropriate clusters according to data set suitability (Xu and Wunsch, 2005). Additionally, Mixture of Gaussian is one of the models that are used for classification and clustering tasks. In this project, we assumed that the Iris data sets attributes are distributed like multivariate Gaussian distribution. The reason why one can make such an assumption can be seen at Figure 1 visually.

The first reason that we made this project is how one can take advantage of Monte Carlo approaches to make a data mining analysis, specifically cluster analysis in this project. From this point of view, we looked forward a data set whose attributes are distributed from multivariate Gaussian close enough. Gibbs sampling, which is one of Markov Chain Monte Carlo approaches, is utilized. After achieving the accuracy results, Expectation-Maximization (EM) algorithm was implemented in order to evaluate the output of the Gibbs Sampling approach. The reason why we chose this algorithm to compare is that EM has more deterministic way to cluster compared to Gibbs Sampling. We would like to show the advantages and disadvantages of Gibbs Sampling by using a benchmark.

In the next section, we review the literature that we make use of in this project. In the third section, we describe how we applied Gibbs sampling to a clustering problem. Fourth section is dedicated to our computational studies and we give concluding remarks in the fifth section.

2 Literature Review

2.1 Expectation Maximization

The expectation maximization method is widely used in maximum likelihood estimation (Yildirim, 2012). We used the pseudo-code given in Zaki and Meira Jr (2014).

```
$t \leftarrow 0$;
//Initialization;
Randomly initialize $\mu_1^t, ..., \mu_J^t$;
$\Sigma_j^t \leftarrow I, \ \forall j = 1, ..., J$;
$P^t(z_j) \leftarrow \frac{1}{J}, \ \forall j = 1, ..., J$;
repeat
    $t \leftarrow t + 1$;
    //Expectation Step;
    for $j$ = 1,...,$J$ and $i$ = 1,...,$N$ do
        $w_{ij} \leftarrow \frac{f(x_i|\mu_j,\Sigma_j)P(z_j)}{\sum_{a=1}^J f(x_i|\mu_a,\Sigma_a)P(z_a)}$ //Posterior Probability $P^t(z_j|x_i)$;
    end
    for $j$ = 1,...,$J$ do
        $\mu_j^t \leftarrow \frac{\sum_{i=1}^N w_{ij}x_i}{\sum_{i=1}^N w_{ij}}$ //re-estimate mean;
        $\Sigma_j^t \leftarrow \frac{\sum_{i=1}^N w_{ij}(x_i-\mu_j)(x_i-\mu_j)^\top}{\sum_{i=1}^N w_{ij}}$ //re-estimate covariance matrix;
        $P^t(z_j) \leftarrow \frac{\sum_{i=1}^N w_{ij}}{N}$ //re-estimate priors;
    end
until $\sum_{j=1}^J \|\mu_j^t - \mu_j^{t-1}\|^2 \leq \epsilon$;
```

Algorithm 1: Expectation Maximization

Here we give the EM algorithm in its very basic form, but our R code for this algorithm is given in the appendix.

2.2 Gibbs Sampling

We describe the Gibbs algorithm in more detail in the next section, where we elaborate on the details of the conditional distributions. Gibbs sampling is used when the dimension of the random variable is larger than 1. If we can sample from the conditional distributions, then the resulting process produces a Markov Chain (Yildirim, 2016). Thus, the main application of Gibbs sampling has been in the areas of image reconstruction, neural networks, and expert systems (Gelfand and Smith, 1990). Two of the resources we used for Gibbs sampling are MacKay (1998) and Robert and Casella (2013).

3 An Application on Iris Data Set

Iris flower data set is a multivariate data set and is well-known in data mining field (Lichman, 2013). The data set consists of 50 samples from each threes species of Iris (setosa, virginica and versicolor). Each sample has 4 attributes which are the length and the width of the sepals and petals. Scatter plot of the data set is shown at Figure 1.

In this project, firstly we assume that each cluster z_j is distributed by a multivariate Gaussian distribution, that is,

$$P(X|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left\{ \frac{-1}{2}(x-\mu_j)^\top \Sigma_j^{-1}(x-\mu_j) \right\}$$

where the cluster mean $\mu_j \in R^d$ and the covariance matrix $\Sigma_j \in R^{d\times d}$ are both unknown parameters, where d indicates the dimension of the multivariate Gaussian distribution. Our problem has 4 attributes and 1 output attribute, which refer to iris flower type. It means that d is equal to 4.

We used Iris data set in our computational study. We first describe the notation we used in our methods.

Notationally, $Z = \{z_i\}_{i=1}^N$ is the collection of all class indicator variables, $\Theta = \{\theta_j\}_{j=1}^J$ is the collection of all class parameters, $\theta = \{\mu_j, \Sigma_j\}$ are the mean and covariance for class j. $w = \{w_j\}_{j=1}^J, w_j = P(z_i = j)$ are the class prior probabilities where $w \sim \text{Dirichlet}(\alpha)$. We also have $\Theta \sim N(\mu, \Sigma)$, multivariate Gaussian distribution. In addition, y represents the data points and covers all the attribute values for each data sample. Moreover, C parameter refers to the count number for each class j.

In Bayesian statistics, the conjugate prior of the mean vector of multivariate Gaussian distribution is another multivariate Gaussian distribution, and the conjugate prior of the covariance matrix is an Inverse-Wishart distribution. $\Sigma_j \sim \text{Inverse-Wishart}(\Lambda_0, v_0)$ and $\mu \sim \text{Gaussian}(\mu_0, \Sigma_0)$.

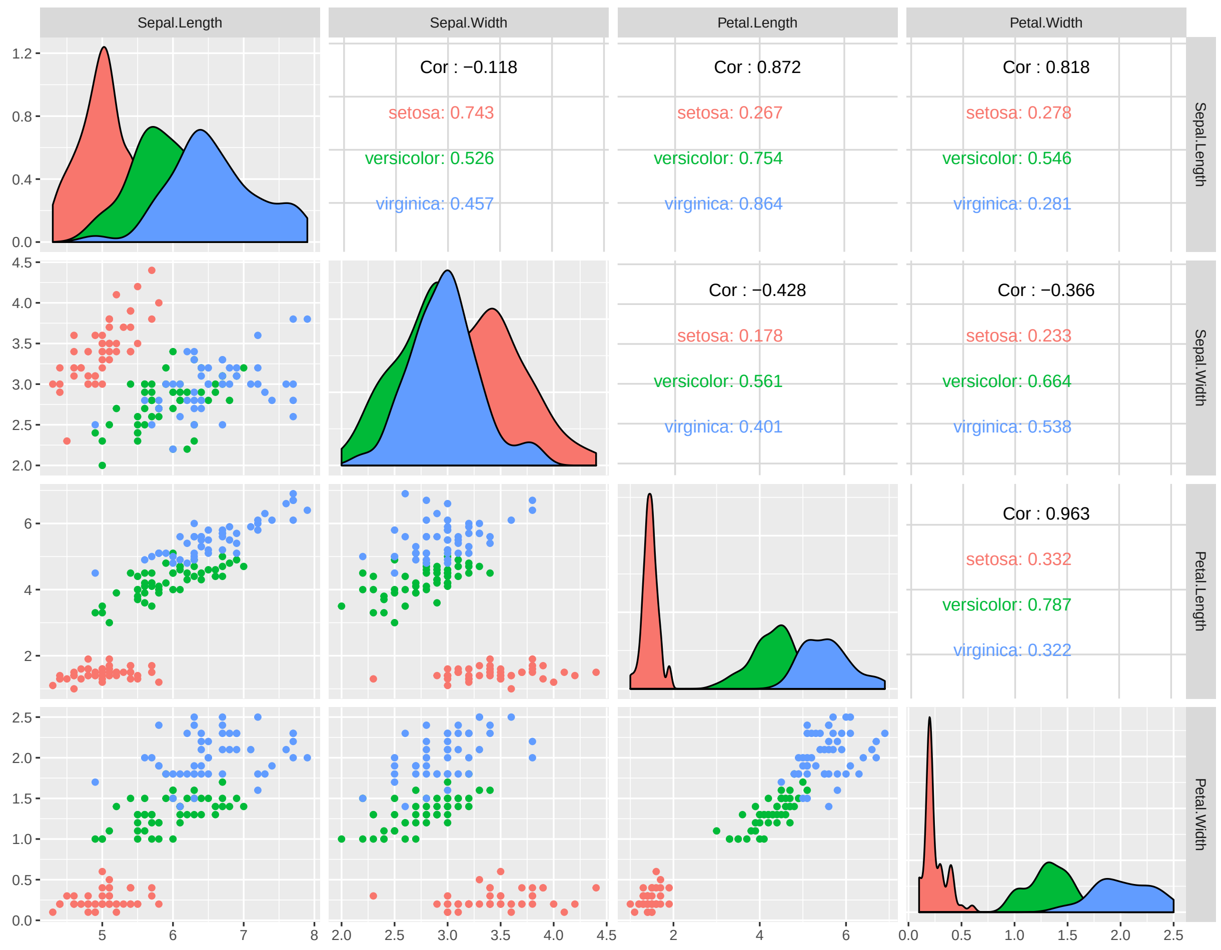


Figure 1: Iris Data Set

The joint density function in compact form is as follows:

$$\Psi(\mathbf{w},\mu,\mathbf{\Sigma},\mathbf{y},\mathbf{z}) = p(\mathbf{w})\ p(\mathbf{z}|\mathbf{w})\ p(\mu,\mathbf{\Sigma}|\mathbf{z})\ p(y|\mu,\mathbf{\Sigma}) \tag{1}$$

In this compact form, we assume that μ and $\mathbf{\Sigma}$ are conditionally independent from $\mathbf{w}$ given $\mathbf{z}$, and $\mathbf{y}$ is conditionally independent from $\mathbf{z}$ given μ and $\mathbf{\Sigma}$.

$$\begin{aligned}\Psi(\mathbf{w},\mu,\mathbf{\Sigma},\mathbf{y},\mathbf{z}) &= p(\mathbf{w})\ p(\mathbf{z}|\mathbf{w})\ p(\mu,\mathbf{\Sigma}|\mathbf{z})\ p(y|\mu,\mathbf{\Sigma})\\ \Psi(\mathbf{w},\mu,\mathbf{\Sigma}|\mathbf{y},\mathbf{z})\ p(\mathbf{y},\mathbf{z}) &= p(\mathbf{w})\ p(\mathbf{z}|\mathbf{w})\ p(\mu,\mathbf{\Sigma}|\mathbf{z})\ p(y|\mu,\mathbf{\Sigma})\\ \Psi(\mathbf{w},\mu,\mathbf{\Sigma}|\mathbf{y},\mathbf{z}) &\propto p(\mathbf{w})\ p(\mathbf{z}|\mathbf{w})\ p(\mu,\mathbf{\Sigma}|\mathbf{z})\ p(y|\mu,\mathbf{\Sigma}).\end{aligned}$$

In our problem $J = 3$ and $N = 150$. And we will be focusing on the following unnormalized posterior distribution (ϕ):

$$\phi = p(w)\prod_{t=1}^{150} p(z_t|w)\prod_{i=1}^{3} p(\mu_i)p(\Sigma_i)\prod_{i=1}^{150} p(y_i|\mu_{z_i},\Sigma_{z_i}) \tag{2}$$

We will be dealing with the following joint density function:

$$\prod_{j=1}^{3} w_j^{C_j+\alpha_j-1} \prod_{j=1}^{3} \frac{1}{(2\pi)^{d/2}} |\Sigma|^{-d/2} \exp\left\{ \frac{-1}{2} (\mu_j - \mu_0)^\top \Sigma_0^{-1} (\mu_j - \mu_0) \right\} \frac{|\Lambda_0|^{V_0/2}}{2^{\frac{V_0 d}{2}} \Gamma_d (V_0/2)} |\Sigma_j|^{\frac{-V_0+d+1}{2}}$$
$$\exp\left\{ \frac{-1}{2} \operatorname{tr}\left(\Lambda_0 \Sigma_j^{-1} \right) \right\} \prod_{i=1}^{150} \frac{1}{(2\pi)^{d/2}} |\Sigma_{z_i}|^{-d/2} \exp\left\{ \frac{-1}{2} (y_i - \mu_{z_i})^\top \Sigma_{z_i}^{-1} (y_i - \mu_{z_i}) \right\}$$

For any given $j \in J$, we can write the following conditional mean of Gaussian distribution where we used another Gaussian random variable as a prior. We began our derivations by first assuming the case where $d = 1$ for simplicity, and then extended our result for the general case. Note that when $d = 1$ we used $\sigma_j,\ j \in J$ to denote the standard deviation of the cluster distribution, whereas in the general case we used Σ to denote the same parameter.

$$\begin{aligned}
P(\mu_j | y, w, z, \sigma_j) &\propto \exp\left\{ \frac{-1}{2\sigma_0^2} (\mu_j - \mu_0)^2 \right\} \exp\left\{ \frac{-1}{2\sigma_j^2} \sum_{i \in I|(z_i=j)} (y_i - \mu_{z_i})^2 \right\} \\
&= \exp\left\{ \frac{-1}{2\sigma_0^2} \left(\mu_j^2 - 2\mu_j\mu_0 + \mu_0^2 \right) + \frac{-1}{2\sigma_j^2} \sum_{i \in I|(z_i=j)} \left(y_i^2 - 2y_i\mu_{z_i} - \mu_{z_i}^2 \right) \right\} \\
&\propto \exp\left\{ \frac{-1}{2} \left(\frac{\mu_j^2}{\sigma_0^2} - \frac{2\mu_j\mu_0}{\sigma_0^2} + \frac{\mu_j^2 C_j}{\sigma_j^2} - \frac{2\mu_j}{\sigma_j^2} \sum_{i \in I|(z_i=j)} y_i \right) \right\} \\
&= \exp\left\{ \frac{-1}{2} \left[\mu_j^2 \left(\frac{1}{\sigma_0^2} + \frac{C_j}{\sigma_j^2} \right) - 2\mu_j \left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma_j^2} \sum_{i \in I|(z_i=j)} y_i \right) \right] \right\}
\end{aligned}$$

This is similar to Gaussian cdf, with parameters $\hat{\sigma}$ and $\hat{\mu}$, where

$$\hat{\sigma} = \left(\frac{1}{\sigma_0^2} + \frac{C_j}{\sigma_j^2} \right)^{-1}, \quad \text{and} \quad \frac{\hat{\mu}}{\hat{\sigma}^2} = \left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma_j^2} \sum_{i \in I|(z_i=j)} y_i \right)$$

In multivariate case, the variance is equal to:

$$\hat{\Sigma} = \left[\Sigma_0^{-1} + C_j \Sigma^{-1} \right]^{-1} \tag{3}$$

The mean is equal to

$$\hat{\mu} = \hat{\Sigma} \left(\mu_0 \Sigma_0^{-1} + \Sigma^{-1} \sum_{i \in I} y_i \right) \tag{4}$$

For any given $j \in J$, we can write the following conditional variance of Gaussian distribution where we used inverse-wishart distribution as a prior.

$$
\begin{aligned}
P(\Sigma_j|y,w,z,\mu) \propto& |\Sigma_j|^{\frac{-V_0+d+1}{2}} \exp\left\{\frac{-1}{2}\operatorname{tr}\left(\Lambda_0\Sigma_j^{-1}\right)\right\} \prod_{i\in I|z_i=j} |\Sigma_{z_i}|^{-d/2} \exp\left\{\frac{-1}{2}\left(y_i-\mu_{z_i}\right)^\top \Sigma_{z_i}^{-1}\left(y_i-\mu_{z_i}\right)\right\} \\
\propto& |\Sigma_j|^{\frac{-V_0+dC_j+d+1}{2}} \exp\left\{\frac{-1}{2}\operatorname{tr}\left(\Lambda_0\Sigma_j^{-1}\right)+\frac{-1}{2}\sum_{i\in I|z_i=j}\operatorname{tr}\left(\Sigma_{z_i}^{-1}\left(y_i-\mu_{z_i}\right)\left(y_i-\mu_{z_i}\right)^\top\right)\right\} \\
=& |\Sigma_j|^{\frac{-V_0+dC_j+d+1}{2}} \exp\left\{\frac{-1}{2}\operatorname{tr}\left(\Lambda_0\Sigma_j^{-1}\right)+\frac{-1}{2}\sum_{i\in I|z_i=j}\operatorname{tr}\left(\left(y_i-\mu_{z_i}\right)\left(y_i-\mu_{z_i}\right)^\top\Sigma_{z_i}^{-1}\right)\right\} \\
=& |\Sigma_j|^{\frac{-V_0+dC_j+d+1}{2}} \exp\left\{\frac{-1}{2}\left(\operatorname{tr}\left(\Lambda_0\Sigma_j^{-1}\right)+\operatorname{tr}\left(\left(\sum_{i\in I|z_i=j}\left(y_i-\mu_{z_i}\right)\left(y_i-\mu_{z_i}\right)^\top\right)\Sigma_{z_i}^{-1}\right)\right)\right\}
\end{aligned}
$$

This is similar to inverse-wishart distribution with parameters $\hat{\Lambda}$ and $\hat{v}$.

$$
\hat{\Lambda} = \Lambda_0 + \sum_{i\in I|z_i=j}\left(y_i-\mu_{z_j}\right)^\top\left(y_i-\mu_{z_j}\right) \tag{5}
$$

Moreover we compute the following:

$$
\hat{v} = v_0 + C_j d \tag{6}
$$

As a result, in this sampling method we used the following distributions for μ and Σ:

$$
\begin{aligned}
&\mu \sim \mathcal{N}\left(\hat{\mu},\hat{\Sigma}\right) = \mathcal{N}\left(\hat{\Sigma}\left(\mu_0\Sigma_0^{-1}+\Sigma^{-1}\sum_{i\in I} y_i\right), \left[\Sigma_0^{-1}+C_j\Sigma^{-1}\right]^{-1}\right) \\
&\Sigma \sim \mathcal{IW}\left(\hat{\Lambda},\hat{v}\right) = \mathcal{IW}\left(\Lambda_0+\sum_{i\in I|z_i=j}\left(y_i-\mu_{z_j}\right)^\top\left(y_i-\mu_{z_j}\right), v_0+C_j d\right) \\
&w \sim \operatorname{Dir}(\alpha+C)
\end{aligned}
$$

In particular, we use these parameters while calculating the Mahalanobis distance between each data point and clusters. To calculate C value at each iteration, we used the following method.

$$
\begin{aligned}
q &\sim \operatorname{Categorical}\left(\hat{w}_{\text{Mahalanobis}}|\mu,\Sigma,w\right) \\
C_j &= E_q\left(I=\{z_i=j\}\right)N.
\end{aligned}
$$

Here, our aim is to assign cluster membership probability to data points inversely proportional to their Mahalanobis distance to cluster centroids.

4 Computational Results

Iris dataset contains 150 data points. First 50 points belong to class-1, points between 51 - 100 belong to class-2, and points between 101 150 belong to class-3. We used this information to assess the performance of clustering. For each of these classes, we checked if the clustered data points are lumped together. Here is an example result for demonstration:

Data Points	Cluster 1	Cluster 2	Cluster 3
1-50	1	49	0
51-100	27	0	23
101-150	22	0	28

Table 1: Example Run

In this example run, $1 + 23 + 22 = 46$ data points are wrongly clustered. So, our clustering accuracy is $(150 - 46)$ / $150 = 0.6933$ (69.33 percent). We will compare the EM algorithm and Gibbs sampling approach in this manner.

For both methods, we ran the code with 2000 iterations.

Results for the EM algorithm:

Data Points	Cluster 1	Cluster 2	Cluster 3
1-50	0	1	49
51-100	19	31	0
101-150	29	21	0

Table 2: Results of the EM Algorithm

Accuracy: 72.67 percent.

Results for the Gibbs sampling method:

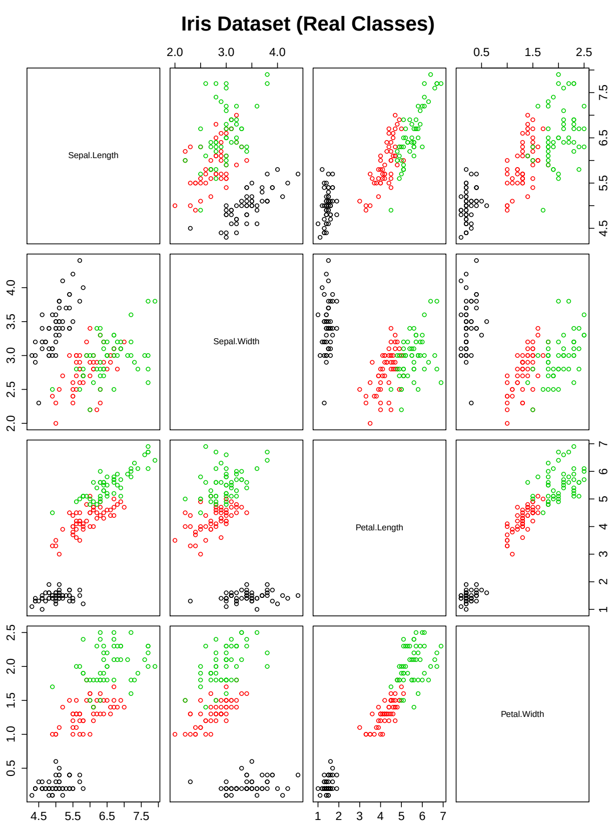
Data Points	Cluster 1	Cluster 2	Cluster 3
1-50	0	2	48
51-100	12	38	0
101-150	34	16	0

Table 3: Results of the Gibbs Algorithm

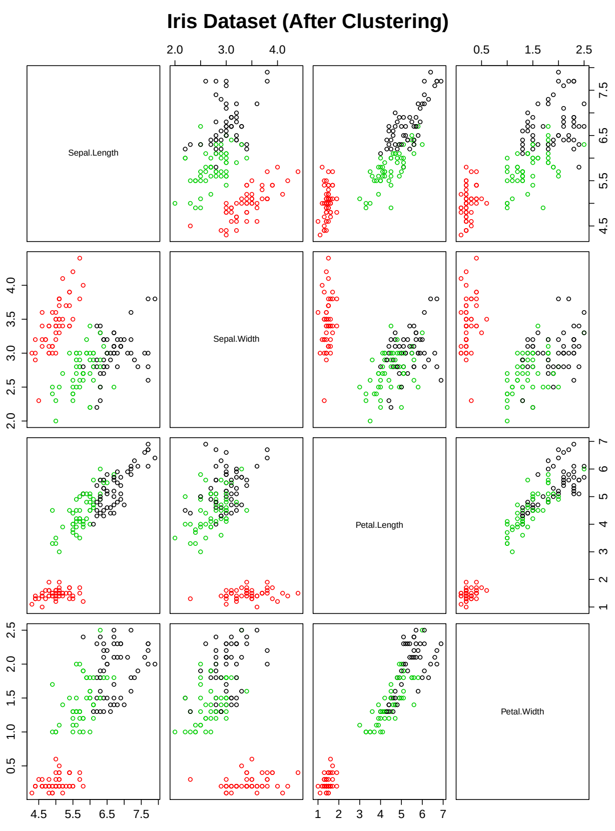
Accuracy: 80.00 percent.

We can observe the effect of inherent structure of the data points on the accuracy by looking at the following pairs plots:

Below, one of the clusters is inherently well-seperated from the other ones, whereas two of them are too much intertwined, thus having more contribution to total error than the isolated one.



(a) Real Classes



(b) After Clustering

Figure 2: Comparison of the real class labels and the clustering results

5 Conclusion

In this project, we successfully implemented an expectation maximization algorithm and a Gibbs sampling algorithm for clustering. We used our algorithms on the famous Iris data set, and obtained results which have reasonable accuracy in terms of correctly clustering the data points. When we used Gibbs sampling we observed perturbations in the values of the estimated parameters, whereas when we used expectation maximization, the estimated parameters did not fluctuate, but converged to a value which stayed stable after the convergence. This result was expected since Gibbs sampling is a Markov Chain Monte Carlo method.

During our computational study, we observed that both of these methods suffered from the assumption that our the flower attributes have Gaussian distribution. If we were to find a more suitable distribution, we could have obtained results with a better accuracy. Our Gaussian assumption has shortcomings considering the fact that the cluster of flower species are not linearly separable.

References

Everitt, B. (1977). Cluster analysis.

Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409.

Lichman, M. (2013). UCI machine learning repository.

MacKay, D. J. (1998). Introduction to monte carlo methods. In *Learning in graphical models*, pages 175–204. Springer.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.

Yildirim, S. (2012). Maximum likelihood parameter estimation in time series models using sequential monte carlo. PhD Thesis, University of Cambridge.

Yildirim, S. (2016). Simulation methods for statistical inference. Available at http://people.sabanciuniv.edu/sinanyildirim/Lecture_notes.pdf.

Zaki, M. J. and Meira Jr, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press.

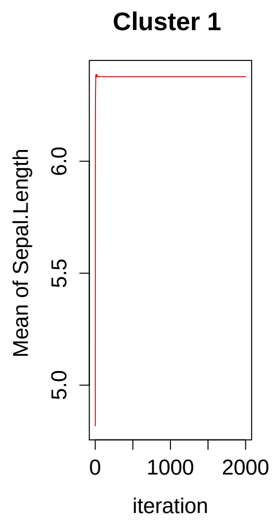
R Code for Expectation Maximization

```
# install.packages("MCMCpack")
library(MCMCpack)

# Preprocessing
Y <- iris[, 1:4]; Y <- as.matrix(Y)
Z <- matrix(rep(0, 450), 150, 3)
Cj <- factor(iris[, 5], labels = 1:3)
colnames(Z) <- c("setosa", "versicolor", "virginica")

# Preparing label matrix Z
# This part is optional to view real class labels
for(i in 1:150) {
  j <- Cj[i]
  Z[i, j] <- 1
}

# Initialization of parameters
w <- matrix(rep(1/3, 450), 150, 3)
mu <- list(m_1 = rnorm(4, colMeans(Y), 1),
           m_2 = rnorm(4, colMeans(Y), 1),
           m_3 = rnorm(4, colMeans(Y), 1))
sigma <- list(s_1 = cov(Y)/3, s_2 = cov(Y)/3, s_3 = cov(Y)/3)

maxiter = 2000

# Expectation-Maximization main algorithm
for(t in 1:maxiter) {
  # Assigning cluster membership probabilities
  # By using Mahalanobis distance between a data point and cluster means
  d <- matrix(rep(450), 150, 3)
  for(i in 1:3) {
    d[, i] <-  sqrt(mahalanobis(x = Y, center = mu[[i]], cov = sigma[[i]]))
  }
  for(i in 1:150) {
    d[i, ] <- prod(d[i, ]) / d[i, ]
  }
  d <- d / rowSums(d)
  w <- w * d
  # Updating cluster probabilities by using Dirichlet-Categorical model
  for(i in 1:150) {
    w[i, ] <- rdirichlet(1, w[i, ]*d[i, ]*150)
  }
  w <- w / rowSums(w)
  # Updating the parameters of Gaussians
  for(i in 1:3) {
    mu[[i]] <- w[, i] %*%  Y / sum(w[, i])
    X <- Y - matrix(mu[[i]], 150, 4, byrow = T)
    z <- 0
    for(j in 1:150) {
      z <- z + (w[j, i] * X[j, ] %*% t(X[j, ]))
    }
    sigma[[i]] <- z / sum(w[, i])
  }
}

# Test of results
colSums(w[1:50,])
colSums(w[51:100,])
colSums(w[101:150,])
```

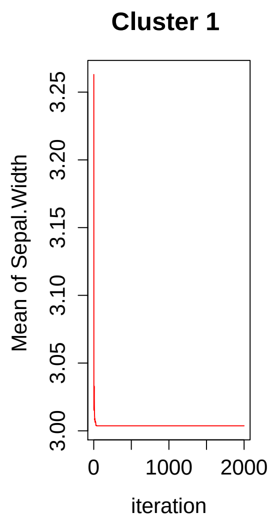
R Code for Gibbs Sampling

```r
# install.packages("MCMCpack")
# install.packages("mvtnorm")
library(MCMCpack)
library(mvtnorm)

# Preprocessing
Y <- iris[, 1:4]; Y <- as.matrix(Y)
Z <- matrix(rep(0, 450), 150, 3)
cj <- factor(iris[, 5], labels = 1:3)
colnames(Z) <- c("setosa", "versicolor", "virginica")

# Preparing label matrix Z
# This part is optional to view real class labels
for(i in 1:150) {
  j <- cj[i]
  Z[i, j] <- 1
}

# Initialization of parameters
w <- matrix(rep(1/3, 450), 150, 3)
mu <- list(m_1 = as.matrix(colMeans(Y) + rnorm(4, 0, 1), 4, 1),
           m_2 = as.matrix(colMeans(Y) + rnorm(4, 0, 1), 4, 1),
           m_3 = as.matrix(colMeans(Y) + rnorm(4, 0, 1), 4, 1))
mu_0 <- list(m_1 = as.matrix(colMeans(Y), 4, 1),
             m_2 = as.matrix(colMeans(Y), 4, 1),
             m_3 = as.matrix(colMeans(Y), 4, 1))
sigma <- list(s_1 = cov(Y)/3, s_2 = cov(Y)/3, s_3 = cov(Y)/3)
sigma_0 <- list(s_1 = cov(Y)/3, s_2 = cov(Y)/3, s_3 = cov(Y)/3)

maxiter <- 2000 # Maximum number of iterations

w.result <- matrix(rep(1/3, 450), 150, 3)
mu.result <- list(m_1 = as.matrix(colMeans(Y) + rnorm(4, 0, 1), 4, 1),
                  m_2 = as.matrix(colMeans(Y) + rnorm(4, 0, 1), 4, 1),
                  m_3 = as.matrix(colMeans(Y) + rnorm(4, 0, 1), 4, 1))
sigma.result <- list(s_1 = cov(Y)/3, s_2 = cov(Y)/3, s_3 = cov(Y)/3)

# Gibbs sampling main algorithm
for(t in 1:maxiter) {
  # Assigning cluster membership probabilities
  # By using Mahalanobis distance between a data point and cluster means
  d <- matrix(rep(0, 450), 150, 3)
  for(i in 1:3) {
    d[, i] <-  sqrt(mahalanobis(x = Y, center = mu[[i]], cov = sigma[[i]]))
  }
  for(i in 1:150) {
    d[i, ] <- prod(d[i, ]) / d[i, ]
  }
  d <- d / rowSums(d)
  w <- w * d; w <- w / rowSums(w)
  I <- matrix(rep(0, 450), 150, 3)
  # Updating cluster probabilities by using Dirichlet-Categorical model
  for(i in 1:150) {
    theta <- rdirichlet(1, w[i, ]*150)
    I[i, ] <- as.numeric(theta == max(theta))
  }
  Cj <- colSums(I)
  # Updating the parameters of Gaussians
  sigma_hat <- list(s_1 = 0, s_2 = 0, s_3 = 0)
  mu_hat <- list(s_1 = 0, s_2 = 0, s_3 = 0)
  for(i in 1:3) {
    y <- Y - matrix(rep(1, 150), 150, 1) %*% t(mu[[i]])
    lambda <- 0
    for(j in 1:150) {
      lambda <- lambda + y[j, ] %*% t(y[j, ])
    }
    sigma[[i]] <- riwish(4 + Cj[i]*4, lambda)
    sigma_hat[[i]] <- solve((solve(sigma_0[[i]]) + Cj[i]*solve(sigma[[i]])))
    mu_hat[[i]] <- sigma_hat[[i]] %*% (t(t(mu_0[[i]]) %*% solve(sigma_0[[i]]
    mu[[i]] <- t(rmvnorm(1, mu_hat[[i]], sigma_hat[[i]]))
  }
  # Accumulating the parameters
  # To be divided by number of iterations later on
  # In order to take their average
  if(t > 1) {
    w.result <- w.result + w
    for(i in 1:3) {
      mu.result[[i]] <- mu.result[[i]] + mu[[i]]
      sigma.result[[i]] <- sigma.result[[i]] + sigma[[i]]
    }
  }
}
```

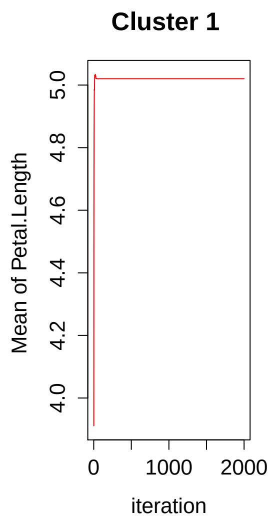
```
# Dividing the sum of parameters along the path by number of iterations
# In order to obtain their estimate (average)
w.result <- w.result / maxiter
for(i in 1:3) {
  mu.result[[i]] <- mu.result[[i]] / maxiter
  sigma.result[[i]] <- sigma.result[[i]] / maxiter
}

# Test of results
# With respect to cluster probabilities in the last iteration
colSums(w[1:50,])
colSums(w[51:100,])
colSums(w[101:150,])
```



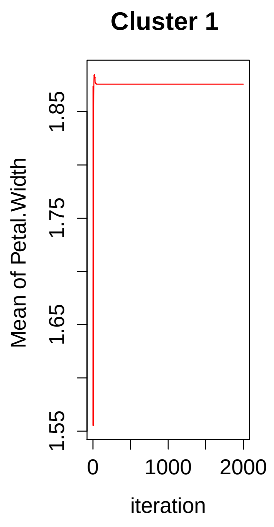
(a) μ_{11} with EM



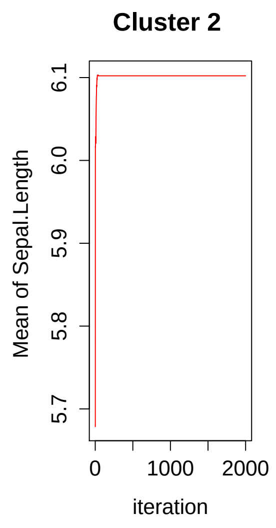
(b) μ_{12} with EM



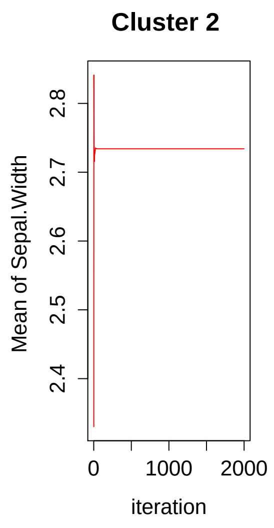
(c) μ_{13} with EM



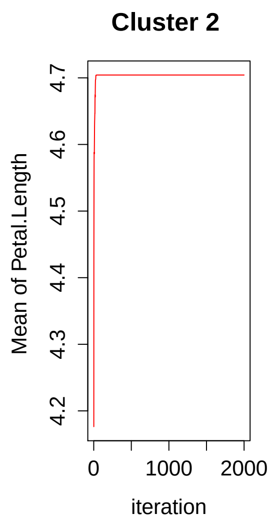
(d) μ_{14} with EM



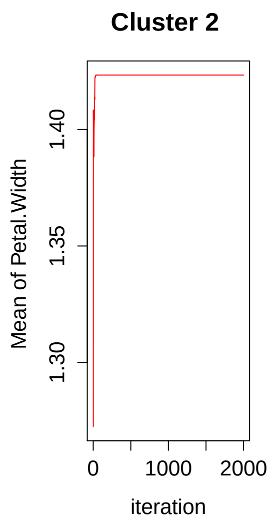
(e) μ_{21} with EM



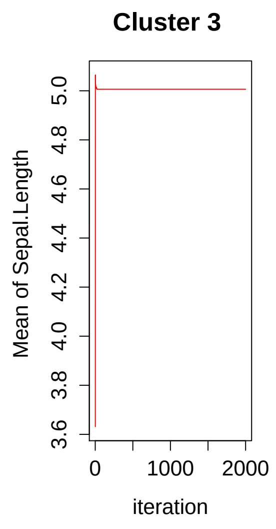
(f) μ_{22} with EM



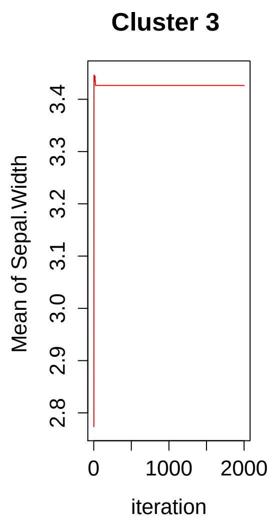
(g) μ_{23} with EM



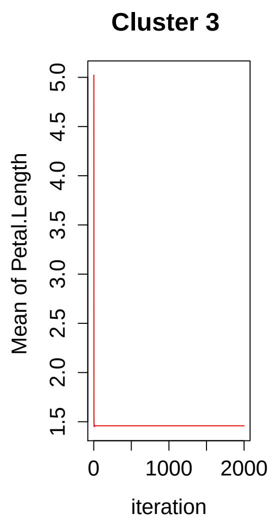
(h) μ_{24} with EM



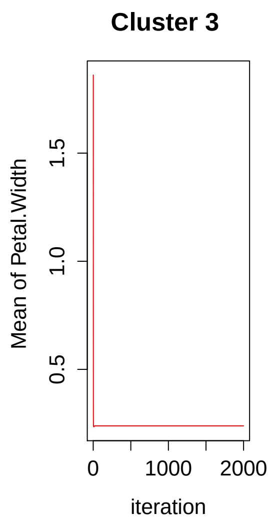
(i) μ_{31} with EM



(j) μ_{32} with EM

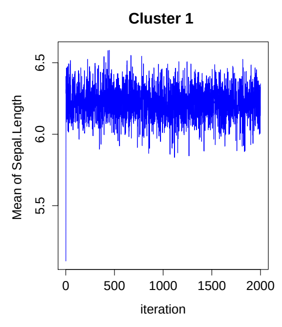


(k) μ_{33} with EM

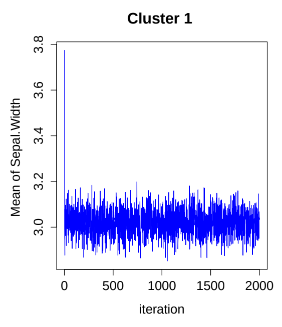


(l) μ_{34} with EM

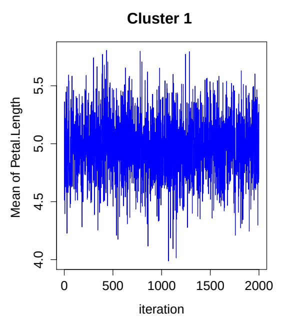
Figure 3: Results of mean parameters obtain from EM algorithm



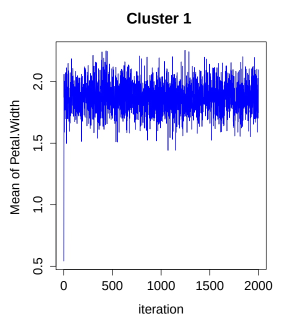
(a) μ_{11} with Gibbs



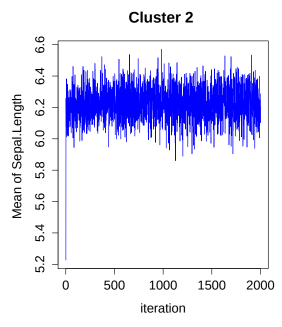
(b) μ_{12} with Gibbs



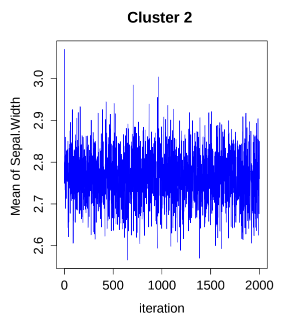
(c) μ_{13} with Gibbs



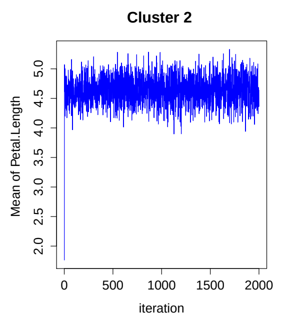
(d) μ_{14} with Gibbs



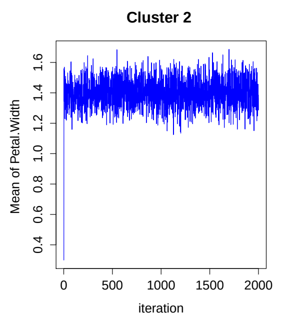
(e) μ_{21} with Gibbs



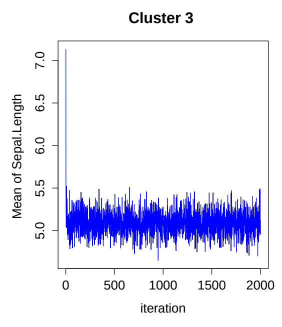
(f) μ_{22} with Gibbs



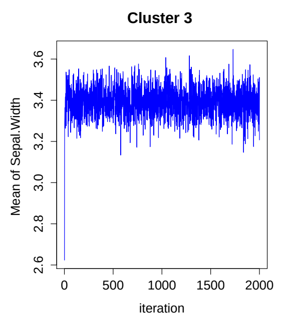
(g) μ_{23} with Gibbs



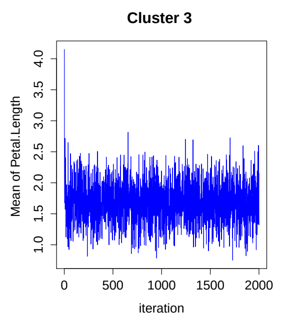
(h) μ_{24} with Gibbs



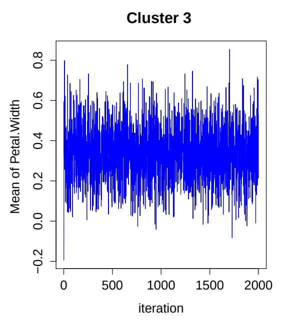
(i) μ_{31} with Gibbs



(j) μ_{32} with Gibbs



(k) μ_{33} with Gibbs



(l) μ_{34} with Gibbs

Figure 4: Results of mean parameters obtain from Gibbs algorithm