Assignment4 Report

Author: 余永琦 Student Number: ID 120090761

Contents

Part1: Written Questions

1. Basic step: assume the random variable has only 2 values.

$f(E[X]) = f(p_1 x_1 + p_2 x_2)$, $E[f(X)] = p_1 f(x_1) + p_2 f(x_2)$

We know $p_1 + p_2 = 1$, $p_1, p_2 \geq 0$, by the convexity of f,

we can conclude that $f(E[X]) \leq E[f(X)]$.

Assumption: $f(E[X]) \leq E[f(x)]$ is true for X has $2, \cdots n$ points.

Inductive step: X has $n+1$ points.

$f(E[X]) = f(p_1 x_1 + p_2 x_2 + \cdots + p_{n+1} x_{n+1})$

Consider $p_n x_n + p_{n+1} x_{n+1} = (p_n + p_{n+1}) \cdot \left(\frac{p_n}{p_n + p_{n+1}} x_n + \frac{p_{n+1}}{p_n + p_{n+1}} x_{n+1}\right)$

as one data point of X.

Then by our assumption, X has n points, we know that

$f(E[X]) = f\left(p_1 x_1 + p_2 x_2 + \cdots + (p_n + p_{n+1}) \cdot \left(\frac{p_n}{p_n + p_{n+1}} x_n + \frac{p_{n+1}}{p_n + p_{n+1}} x_{n+1}\right)\right)$

$\leq p_1 f(x_1) + p_2 f(x_2) + \cdots + (p_n + p_{n+1}) f\left(\frac{p_n}{p_n + p_{n+1}} x_n + \frac{p_{n+1}}{p_n + p_{n+1}} x_{n+1}\right)$

From basic step, we know that $f\left(\frac{p_n}{p_n + p_{n+1}} x_n + \frac{p_{n+1}}{p_n + p_{n+1}} x_{n+1}\right) \leq \frac{p_n}{p_n + p_{n+1}} f(x_n) + \frac{p_{n+1}}{p_n + p_{n+1}} f(x_{n+1})$

$\therefore f[E[X]] \leq p_1 f(x_1) + p_2 f(x_2) + \cdots + (p_n + p_{n+1}) \left(\frac{p_n}{p_n + p_{n+1}} f(x_n) + \frac{p_{n+1}}{p_n + p_{n+1}} f(x_{n+1})\right) = E[f(X)]$.

$\therefore$ We've proved $f[E[X]] \leq E[f(X)]$ when X has $n+1$ values from n values.

$\therefore$ We've proved $f(E[X]) \leq E[f(x)]$ by induction.

2. Bernoullis distribution: $p(x_i|\theta_k) = \mu_{kj}^{x_{ij}}(1-\mu_{kj})^{1-x_{ij}}$

① For M-step for ML estimation, ~~need~~ we need to optimize $Q(\theta, \theta^{t-1})$

with $Q(\theta, \theta^{t-1}) = \sum_i \sum_k r_{ik} \log \pi_k + \sum_i \sum_k r_{ik} \log p(x_i|\theta)$

$$\frac{\partial Q}{\partial \mu_{kj}} = \frac{\partial \left(\sum_i \sum_k r_{ik}(x_{ij}\log\mu_{kj} + (1-x_{ij})\log(1-\mu_{kj}))\right)}{\partial \mu_{kj}} = \sum_i r_{ik}\left(\frac{x_{ij}}{\mu_{kj}} - \frac{1-x_{ij}}{1-\mu_{kj}}\right)$$

$$= \frac{\sum_i r_{ik}(x_{ij} - \mu_{kj})}{\mu_{kj}(1-\mu_{kj})} = 0 \Rightarrow \mu_{kj} = \frac{\sum_i r_i x_{ij}}{\sum_i r_{ik}}$$

② For M-Step for MAP estimation of a mixture of Bernoullis with a $\beta(\alpha, \beta)$ prior: $p(\mu_{kj}) = \mu_{kj}^{\alpha-1}(1-\mu_{kj})^{\beta-1}$.

~~Need~~ We need to maximize $L = Q(\theta, \theta^{t-1}) + \log p(\theta)$

$$\frac{\partial L}{\partial \mu_{kj}} = \frac{\sum_i r_{ik}(x_{ij} - \mu_{kj})}{\mu_{kj}(1-\mu_{kj})} + \frac{\alpha-1}{\mu_{kj}} - \frac{\beta-1}{1-\mu_{kj}} = \frac{\sum_i r_{ik}(x_{ij} - \mu_{kj}) + \alpha - 1 - 2\mu_{kj} + \mu_{kj} - \beta\mu_{kj} + \mu_{kj}}{\mu_{kj}(1-\mu_{kj})}$$

$$= \frac{\left(\sum_i r_{ik} x_{ij}\right) + \alpha - 1 - \left(\left(\sum_i r_{ik}\right) + \alpha + \beta - 2\right)\mu_{kj}}{\mu_{kj}(1-\mu_{kj})} = 0$$

$$\Rightarrow \mu_{kj} = \frac{\left(\sum_i r_{ik} x_{ij}\right) + \alpha - 1}{\left(\sum_i r_{ik}\right) + \alpha + \beta - 2}$$

3. $\sum_{i':z_{i'}=k}(x_i-x_{i'})^2 = \sum_{i':z_{i'}=k}(x_i-\bar{x}_k+\bar{x}_k-x_{i'})^2$

$$= \sum_{i':z_{i'}=k}(x_i-\bar{x}_k)^2 + \sum_{i':z_{i'}=k}(\bar{x}_k-x_{i'})^2 + 2\sum_{i':z_{i'}=k}(x_i-\bar{x}_k)(\bar{x}_k-x_{i'})$$

$\because 2\sum_{i':z_{i'}=k}(x_i-\bar{x}_k)(\bar{x}_k-x_{i'}) = 2(x_i-\bar{x}_k)\sum_{i':z_{i'}=k}(\bar{x}_k-x_{i'}) = 0$

$\therefore \sum_{i':z_{i'}=k}(x_i-x_{i'})^2 = \sum_{i':z_{i'}=k}(x_i-\bar{x}_k)^2 + \sum_{i':z_{i'}=k}(\bar{x}_k-x_{i'})^2$

$$= n_k(x_i-\bar{x}_k)^2 + \sum_{i':z_{i'}=k}(\bar{x}_k-x_{i'})^2$$

$$\sum_{i:z_i=k}\sum_{i':z_{i'}=k}(x_i-x_{i'})^2 = \sum_{i:z_i=k} n_k(x_i-\bar{x}_k)^2 + \sum_{i:z_i=k}\sum_{i':z_{i'}=k}(\bar{x}_k-x_{i'})^2$$

$$= n_k\sum_{i:z_i=k}(x_i-\bar{x}_k)^2 + n_k\sum_{i:z_{i'}=k}(\bar{x}_k-x_{i'})^2$$

$\because \sum_{i:z_i=k}(x_i-\bar{x}_k)^2 = \sum_{i':z_{i'}=k}(\bar{x}_k-x_{i'})^2$

$\therefore \sum_{i:z_i=k}\sum_{i':z_{i'}=k}(x_i-x_{i'})^2 = 2n_k\sum_{i:z_i=k}(x_i-\bar{x}_k)^2$

$\therefore J_W(z) = \frac{1}{2}\sum_{k=1}^{k}\sum_{i:z_i=k}\sum_{i':z_{i'}=k}(x_i-x_{i'})^2 = \frac{1}{2}\sum_{k=1}^{k}2n_k\sum_{i:z_i=k}(x_i-\bar{x}_k)^2$

$$= \sum_{k=1}^{k} n_k\sum_{i:z_i=k}(x_i-\bar{x}_k)^2$$

4. $\frac{m^+(m^++1)}{2} = 1+2+\cdots+m^+ = \sum_{i=1}^{m^+} i$.

For each data point x_i^+, rank $i - i$ counts the number of negative class data points x_j^- with rank j lower than rank i.

If $g(x_i^+) > g(x_j^-)$, then ~~the~~ it contributes 1 to rank $i - i$.

If $g(x_i^+) = g(x_j^-)$, if we sort the data randomly, there is 50% possibility that x_j^- has lower rank than x_i^+, so this data point contributes 0.5 to rank $i - i$.

If $g(x_i^+) < g(x_j^-)$, it contributes 0 to rank $i - i$.

From above we can know that ~~can~~ for a point x_i^+,

rank $i - i = \sum_{j=1}^{m^-} \mathbb{1}(e_{ij})$.

$$\therefore AUC = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} \mathbb{1}(e_{ij}) = \frac{\sum_{i=1}^{m^+} \text{rank } i - i}{m^+ m^-} = \frac{\left(\sum_{i=1}^{m^+} \text{rank } i\right) - \frac{m^+(m^++1)}{2}}{m^+ m^-}$$

5.

First we calculate the mean μ:

$$\mu = \frac{1}{10}\sum_{i=1}^{10} x_i = [-0.4, 0.8, 0.2, 0.2, -1.3]^T$$

Then we calculate the covariance matrix:

$$\Sigma = \frac{1}{10}\sum_{i=1}^{10}(x_i - \mu)(x_i - \mu)^T$$

The covariance matrix is:

```
[[ 3.04,  0.82, -0.02, -0.82, -0.12],
 [ 0.82,  3.76, -2.16,  1.04,  1.04],
 [-0.02, -2.16,  3.56,  0.76, -0.84],
 [-0.82,  1.04,  0.76,  5.56,  1.16],
 [-0.12,  1.04, -0.84,  1.16,  2.21]]
```

Now SVD decompose the covariance matrix, we can get:

Q =

```
[[-0.02625442,  0.29809153, -0.87848693,  0.16894209, -0.33192081],
 [ 0.57873744,  0.39493632, -0.17625638, -0.33313495,  0.60584079],
 [-0.32862419, -0.58025264, -0.40532527,  0.08283824,  0.61980825],
 [ 0.65356276, -0.64618454, -0.14382432, -0.15734291, -0.33144989],
 [ 0.35949345,  0.03031732,  0.11054589,  0.91041788,  0.16960016]]
```

Each column of **Q** is a unit-length principle components of X.
We let **U** be the first two column because they correspond to the largest eigenvalues, and we can calculate the projection of each data:

$$x_i = U^T(x_i - \mu)$$

The projections of each data are:

```
[-3.13194616  1.98183693]
[-0.60746566  4.49653708]
[ 1.97315969 -1.40348923]
[ 0.4956134  -2.87861204]
[-0.72008587  0.48232827]
[1.87576557 1.74241301]
[5.38313097 0.53945236]
[-0.591651   -4.45776178]
[-4.46907149 -1.07184006]
[-0.20744945  0.56913546]
```

Part2: Programming Questions

Part 1

Implementation of 3 clustering algorithms

I wrote 3 classes to implement the clustering, all of them have parameters **n_clusters** to determines the number of clusters we generate, and **random** to determines whether we fix a seed to the random state. (We fix a seed when comparing the performance on evaluation metrics or run times or iterations, and do not fix a seed when comparing sensitivities)

All of the 3 classes have two methods, one is **fit**(data) to fit the dataset and get the parameters of that algorithm. After running **fit**(data), the classes will have new attributes **time** to record the time of running the algorithm cost, and **iterations** to represent the number of iterations to finish the algorithm. The other method is **predict**(data) to output the clustering labels of the algorithm. Note that **predict**(data) must run after **fit**(data).

Choosing initial points strategy

Randomly pick n points in the dataset to be the initial cluster centers, n is the number of clusters for the kmeans and accelrated-kmeans, and use kmeans to generate initial centers for GMM.

Kmeans

The implementation of kmeans is simple. First we randomly get n initial centers. Then, we keep repeating the procedure:

1. Assign the data point to its closest center.
2. Update the center using the mean of assigned data points.

Stop until the centers converge.

Basic K-means Clustering

1. First, you choose K — the number of clusters. Then you randomly put K feature vectors, called **centroids**, to the feature space.
2. Next, compute the distance from each example $\mathbf{x}$ to each centroid $\mathbf{c}$ using some metric, like the Euclidean distance. Then we assign the closest centroid to each example (like if we labeled each example with a centroid id as the label).
3. For each centroid, we calculate the average feature vector of the examples labeled with it. These average feature vectors become the new locations of the centroids.
4. We recompute the distance from each example to each centroid, modify the assignment and repeat the procedure until the assignments don't change after the centroid locations are recomputed.
5. Finally we conclude the clustering with a list of assignments of centroids IDs to the examples.

Accelerated Kmeans

The procedure of accelerated kmeans is exactly the same with Kmeans, however, it utilize the triangle inequality to avoid unnecessary calculations to accelerate the algorithm.

$$\text{Lemma 1: if } d(b,c) \geq 2d(x,b), \text{ then } d(x,c) > d(x,b)$$

$$\text{Lemma 2: } d(x,c) > \max\{0, d(x,b) - d(d,c)\}$$

The implementation strictly follow the reference in *https://www.aaai.org/ Papers/ICML/ 2003/ICML03-022.pdf .*

Putting the observations above together, the accelerated k-means algorithm is as follows.

First, pick initial centers. Set the lower bound $l(x,c) = 0$ for each point x and center c. Assign each x to its closest initial center $c(x) = \text{argmin}_c\, d(x,c)$, using Lemma 1 to avoid redundant distance calculations. Each time $d(x,c)$ is computed, set $l(x,c) = d(x,c)$. Assign upper bounds $u(x) = \min_c d(x,c)$.

Next, repeat until convergence:

1. For all centers c and c', compute $d(c,c')$. For all centers c, compute $s(c) = \frac{1}{2}\min_{c' \neq c} d(c,c')$.
2. Identify all points x such that $u(x) \leq s(c(x))$.
3. For all remaining points x and centers c such that
   (i) $c \neq c(x)$ and
   (ii) $u(x) > l(x,c)$ and
   (iii) $u(x) > \frac{1}{2}d(c(x),c)$:
   3a. If $r(x)$ then compute $d(x,c(x))$ and assign $r(x) =$ false. Otherwise, $d(x,c(x)) = u(x)$.
   3b. If $d(x,c(x)) > l(x,c)$ or $d(x,c(x)) > \frac{1}{2}d(c(x),c)$ then
   Compute $d(x,c)$
   If $d(x,c) < d(x,c(x))$ then assign $c(x) = c$.
4. For each center c, let $m(c)$ be the mean of the points assigned to c.
5. For each point x and center c, assign
   $l(x,c) = \max\{l(x,c) - d(c,m(c)), 0\}$.
6. For each point x, assign
   $u(x) = u(x) + d(m(c(x)), c(x))$
   $r(x) =$ true.
7. Replace each center c by $m(c)$.

In my code I mark each steps 1-7 explicitly.

GMM—EM

There are two ways to get initial parameters, according to the sklearn documents.Here we choose initial centers using by Kmeans.

init_params : *{'kmeans', 'random'}, default='kmeans'*

The method used to initialize the weights, the means and the precisions. Must be one of:

```
'kmeans' : responsibilities are initialized using kmeans.
'random' : responsibilities are initialized randomly.
```

The GMM class has an additional attribute **tolerance** to determine the tolerance when consider convergence of the log likelihood. Set default value = 1e-6.

E-step:

$$
\begin{aligned}
\gamma_k = p(z = k \mid \mathbf{x}) &= \frac{p(z = k)p(\mathbf{x} \mid z = k)}{p(\mathbf{x})} \\
&= \frac{p(z = k)p(\mathbf{x} \mid z = k)}{\sum_{j=1}^{K} p(z = j)p(\mathbf{x} \mid z = j)} \\
&= \frac{\pi_k \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}
\end{aligned}
$$

M-step:

$$
\max_{\Theta} \sum_k \sum_n \gamma_k^{(n)} \ln\left(\pi_k\right) + \sum_k \sum_n \gamma_k^{(n)} \ln\left(\mathcal{N}\left(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right)
$$

$$
\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \mathbf{x}^{(n)}
$$

$$
\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \left(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k\right)\left(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k\right)^{\top}
$$

$$
\pi_k = \frac{\sum_{n=1}^{N} \gamma_k^{(n)}}{N}
$$

Repeat the procedure until convergence.

Outputs(fixed seed 12):

```
Kmeans output labels:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1
 2 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1 1 3 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 1 1 1 2 1 1 1 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3]
Accelerated Kmeans output labels:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1
 2 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1 1 3 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 1 1 1 2 1 1 1 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3]
GMM-EM output labels:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 3 1 1 3 3 1 3 3 1 1 1 1 1 2
 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 3 3 3 3 3 3 1 1 1 3 2 2 2 2
 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 2 2 1 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

The output of kmeans and accelerated kmeans should always the same with fixed seed, since they have the same procedure.

Part 2

Silhouette coefficient

Given a clustering, we define

- a: The mean distance between a point and all other points in the **same** cluster.
- b: The mean distance between a point and all other points in the next **nearest** cluster.

Silhouette coefficient s for a single sample is formulated as:

$$s = \frac{b-a}{\max(a,b)} \Rightarrow s = \begin{cases} 1-\frac{a}{b} & \text{if } a < b \\ 0 & \text{if } a = b \\ \frac{b}{a}-1 & \text{if } a > b \end{cases}$$

First calculate the Silhouette coefficient of each point, then output then mean Silhouette coefficient.

Rand index

We can calculate the following values:

- a: The number of pairs of elements in S that are in the **same** subset in X and in the **same** subset in Y
- b: The number of pairs of elements in S that are in the **different** subset in X and in the **different** subset in Y
- c: The number of pairs of elements in S that are in the **same** subset in X and in the **different** subset in Y
- d: The number of pairs of elements in S that are in the **different** subset in X and in the **same** subset in Y

The rand index (RI) can be computed as follows:

$$\mathrm{RI} = \frac{a+b}{a+b+c+d} = \frac{a+b}{\frac{n(n-1)}{2}}$$

```
The Silhouette coefficient of Kmeans is:0.47193373191268945
The Rand Index of Kmeans is:0.8743677375256322
The Silhouette coefficient of accelerated Kmeans is:0.47193373191268945
The Rand Index of accelerated  Kmeans is:0.8743677375256322
The Silhouette coefficient of GMM-EM is:0.3993538185123337
The Rand Index of GMM-EM is:0.8619731146046935
```

Part 3

Running 10 times random initialization with and we calculate the variance of evaluation scores of these clustering results:

```
Variance of Silhouette coefficient of Kmeans is:3.455851796646978e-06
Variance of Rand Index of Kmeans is:2.1708406354231118e-06
Variance of Silhouette coefficient of accelerated Kmeans is:3.5912555774418542e-06
Variance of Rand Index of accelerated Kmeans is:2.9084081607735e-06
Variance of Silhouette coefficient of GMM-EM is:0.0005151252722427254
Variance of Rand Index of GMM-EM is:0.0006342240987418524
```

We can see the GMM-EM algorithm is much more sensitive to the initial point compare to the kmeans and accelerated kmeans algorithms.

Since running the GMM-EM cost a lot of time, I set a variable calculateSensitivity as False, if you want to run it, set it as True.

```
calculateSensitivity = False
if calculateSensitivity:
    se1,ri1 = sensitivity(kmeans(3,random=True),X,10)
    print("Variance of Silhouette coefficient of Kmeans is:"+str(se1))
    print("Variance of Rand Index of Kmeans is:"+str(ri1))
    se2, ri2 = sensitivity(accelerated_kmeans(3, random=True), X, 10)
    print("Variance of Silhouette coefficient of accelerated Kmeans is:"+str(se2))
    print("Variance of Rand Index of accelerated Kmeans is:"+str(ri2))
    se3, ri3 = sensitivity(GMM(3, random=True), X, 10)
    print("Variance of Silhouette coefficient of GMM-EM is:"+str(se3))
    print("Variance of Rand Index of GMM-EM is:"+str(ri3))
```

Part 4

Under the fixed seed 12:

```
The iterations of kmeans is:5
The required time of kmeans is:0.02863001823425293
The iterations of accelerated kmeans is:5
The required time of accelerated kmeans is:0.0368800163269043
The iterations of GMM-EM is:29
The required time of GMM-EM is:3.288856029510498
```

We notice that the required time of accelerated kmeans is larger than kmeans, the reason is the dataset we use is small so the advantage of accelerated means is inexplicit.