Homework 3

Hong Gan hg383@cornell.edu
Zhen Liu zl557@cornell.edu

November 16, 2016

1 Sentiment analysis of online reviews

1.1 Download the data set

Those labels are balanced. In each file there are 500 reviews with positive label as 1 and 500 reviews with negative label as 0. The ratio between positive review counts and negative review counts is 1. For file processing, each line was read as an entry and split with $\overset{\frown}{.}$ The first part is reviews and the second part as label, saved in two different matrices. Then we proceed to the second step to start to pre-processing reviews before split them into matrices.

1.2 Pick your preprocessing strategy

Since these sentences are reviews, we would like to remove noise in it before save them into matrices as data. In this step, firstly we applied a regular expression filter to remove any Non-alphabetic tokens, allowing only [a-zA-Z] be passed into next strategy. Numbers and punctuation are not carry a message in most reviews and usually confusing, ex. ! could be use in both angry and happy situation, then there is no reason to keep them. And in this step, we transferred all letters into lower cases because orders is not considered in our algorithms. Then we removed all stop words in sentence using NLTK stop word dict. Because words as the do and usually do not contain a message, or the message is too broad (Like want).
After all pre-processing strategies, we split sentence into words and use a matrix to represent them. We maintain a matrix which contains all words existed in the training set, and then use the index of the word in the list to represent whether the word shows in the list or not.

1.3 Split training and testing set

We process each .txt file one at a time. For the final step of pre-processing, we initialized two counters to count 400 instances for both negative and prositive. After each instance pool reach 400 counts, the reset of reviews would be placed into testing data.

1.4 Bag of Words model

In this part, firstly we pass all reviews in the training set once to record all words exist in the training set, making a word list. Then we pass all reviews in the training set again. In this pass, we make a matrix N*d with all 0s for each review word, and marking the index of each word shows in specific review as 1. For the testing set, we do the same procedure with the same word list but we only mark words show up in the training set, for words only shows in the testing set, we simply drop it. Here we could not use the testing set because it would cause over fit problems there. Two examples of the feature vectors are reported.

$$[1.1.1...., 0.0.0.]\,(1 \text{ x } 4091)$$

$$[0.0.0...., 0.0.0.]\,(1 \text{ x } 4091)$$

1.5 Pick your post-processing strategy

We tried l1-norm and l2-norm. Actually the result between l1-norm and l2-norm is not largely different, but l2 normalization is better and easier numerical analysis than l1. L2 is more sensitive to out-liners and usually generate an better overall result than l1. But after finished the program part, we realized in this situation log-normalization might be an even better choice than l2-norm. Because log-norm could better deal with smaller numbers and generate a log-liked result, which would fit sporadic data better.

1.6 Training set clustering

In this step, we implemented a K-means algorithm to divide the training set into 2 clusters. For the k-Means algorithm, the center is a matrix. For the first cluster, the center is:

$$0.001515725576468671, 0.001515725576468671,$$
$$0.001515725576468671, 0.001515725576468671,$$
$$0.001515725576468671, 0.001515725576468671..$$

And another cluster's center vector is like:

$$0.0, 0.0, 0.0, 0.0, 0.0...0.0$$

We believe this situation is caused by the k-Means cluster's result. In our result, almost all result was clustered into one, so that in this situation, the center is just the center of all data. And because another cluster get an average of 0, thus div by 0 give 0 to NaN to all elements in the result.

For the N-gram, the center report is very like the normal k-means. In a typical run we could have a result as:

$$[0.00045765, 0.00045765, 0.00045765..., 0.00045765]$$
$$[nan, nan, nan, nan, nan, nan..., nan]$$

It is because we got similar result in the N-gram as the k-Means, which clustered all result into one cluster and left a div by 0 problem to another cluster.

1.7 Sentiment prediction

In the Bag of Word Model, the Logistic Regression report accuracy: 0.80333. We run this model for a few times. Due to the random centroids initialization result, the result is slightly different. One confusion matrix is reported as below:

	Cluster 1	Cluster 2
Cluster 1	257	43
Cluster 2	75	225

Table 1: Confusion Matrix 1

In the N-gram Model, the Logic Regression reported an accuracy: 0.63833. A confusion matrix is reported as below:

	Cluster 1	Cluster 2
Cluster 1	267	33
Cluster 2	184	116

Table 2: Confusion Matrix 2

In the weight vector, we actually take a look in the coefficient of the logistic regression result. Then we picked 10 results of each model, then use their index to retrieve the actual word. The result is reported as below:

1.8 N-gram model

After we implemented the N-gram model, we achieved an accuracy of 0.63833, whereas the bag of word model achieved an accuracy of 0.80333 under logistic regression under the same settings. In this situation, we believe the problem is caused by the N-gram split. The N-gram split makes the distribution of the model become much more sporadic. Thus, many phrase only shows once in the training data and there are many words cannot be matched in the testing data. So that we could expect a lower accurate rate and the result proved our hypothesis.

Bag of Word Model	N-Gram
great	work great
delicious	highly recommend
love	one best
excellent	great phone
nice	great product
loved	food good
fantastic	really good
awesome	easy use
amazing	good price
liked	great food

Table 3: Top weighted words in each method

1.9 PCA for bag of words model

For the PCA part, three runs with 10, 50 and 100 are executed and classification result are reported below:

Accuracy with r = 100: 0.6966667 Accuracy with r = 50 : 0.6816667 Accuracy with r = 10 : 0.6133333

We could see that, with the increment of the different r, the accuracy rate is increasing. We also ran some larger r number, noticed that when r = 1000, the accuracy is very close to 0.80.

1.10 Algorithms comparison and analysis

Comparing all algorithms, we realized that logistic regression model under Bag of Words model given the best results overall. In most cases, k-Means model just simply clustered all data into one cluster and given a 50% accurate rate. And N-gram model just not work in this situation at all. After we applied N-gram model, the logistic regression model returns a lower accurate rate, and k-Means model just return another result with all data into one cluster with different center.

The reason caused this problem, in our opinion, is due to the distribution of the data. In fact, relative to the data size, the way we processing it (bag of word) makes the matrix become very sporadic, thus it is very hard for k-means to cluster it.

For the N-gram model, the reason it fails here is it just separate the data even further than the bag of words model, it makes the algorithm even harder to calculate the distance. Thus it returned a worse result as we expected.

2 EM algorithm and implementation

2.1 Show that the alternating algorithm for K-means is a special case of the EM algorithm

The K-means algorithm turns out to be a special case of clustering with a mixture of Gaussians where all variances are equal (and covariances are 0 and mixture weights are equal, as well see below): the underlying assumption is that clusters are essentially spherical.[1]

E-step:

$$\hat{\gamma_i} = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)} \quad \text{for} \quad i = 1, ..., N$$

M-step:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^{N}(1-\hat{\gamma_i})y_i}{\sum_{i=1}^{N}(1-\hat{\gamma_i})}$$

$$\hat{\sigma}_1^2 = \frac{\sum_{i=1}^{N}(1-\hat{\gamma_i})(y_i-\hat{\mu}_i)^2}{\sum_{i=1}^{N}(1-\hat{\gamma_i})}$$

and similarly for $\hat{\mu}_2$, $\hat{\sigma}_2^2$ using $\hat{\gamma_i}$ in place of $(1-\hat{\gamma_i})$.[2]

2.2 About the data

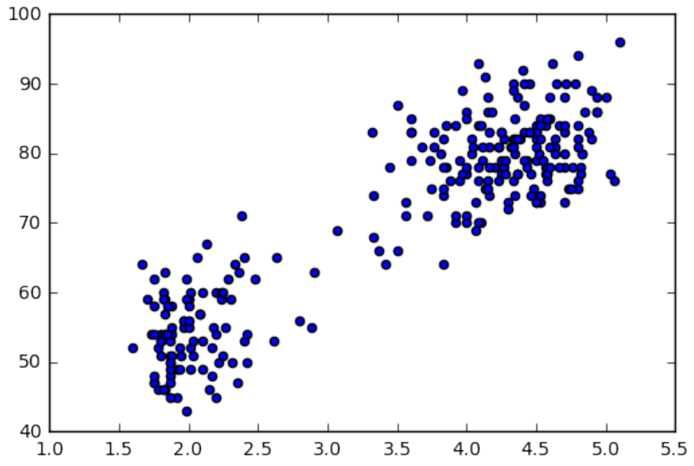
The input data can be parsed into a 272*2 matrix. Each column is the corresponding eruption time and waiting time. We strip the index column and plot the data points on 2d plane.

2.3 Implement a bimodal GMM model

In this part a generic GMM model is implemented, in iteration() function it use EM algorithms to fit in the data. The generic GMM model can work as a bimodal GMM model by indicating k = 2. We noticed that in each step, the corresponding update of mu and sigma depend the posterior probability matrix gamma. If in one iteration, the posterior probability matrix before and after the E-step are the same. Then we can draw conclusion that the algorithm has reached converged. So the termination criteria is to check the each entry of the gamma matrix, to see it's each entry doesnt change. mu is initialed by choose a value between the maximum and minimum. The sigma is set to the co-variance matrix of the whole data set.

[1]http://bit.ly/2gesqaO

[2]From scribe notes 11, CS 5785 – Applied Machine Learning



```
take 12 iterations to converge
Plot the trajectories of two mean vectors in 2 dimensions
```

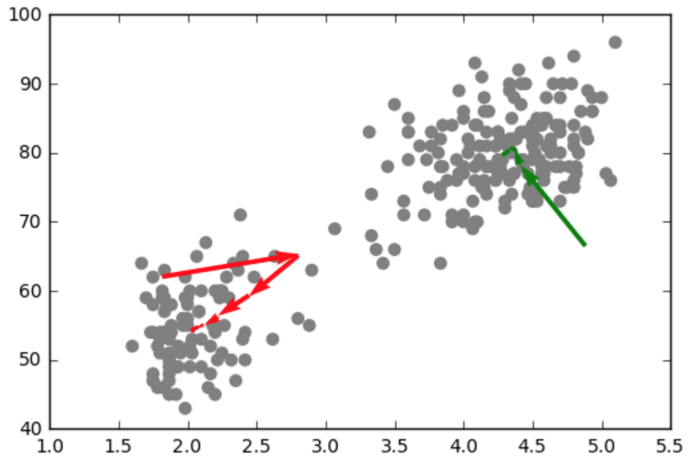


Figure 1: Trajectories of two mean vectors in 2 dimensions

Run the program for 50 times with different initial parameter guesses. And plot the distribution of the total number of iterations needed for algorithm to converge.

```
(array([  1.,   3.,   4.,  10.,   6.,   9.,   8.,   6.,   2.,
 array([  9. ,   9.9,  10.8,  11.7,  12.6,  13.5,  14.4,  15.3,
        17.1,  18. ]),
 <a list of 10 Patch objects>)
```

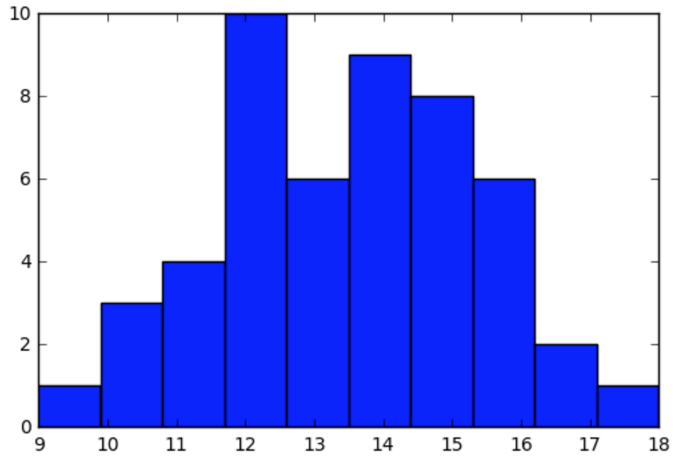
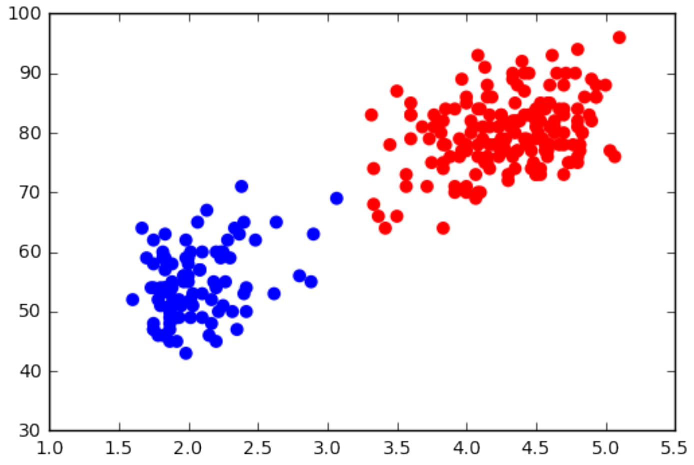


Figure 2: Total number of iterations needed for algorithm to converge

2.4 Repeat

This time run K-means algorithm over all the data points with K = 2 and label each point with one of two clusters. Input data need to be normalized along feature axis first. The data points are separated into two clusters.



For the labeled points, we can use maximum likelihood for Gaussian to initialize the first guess of the mean and co-variance matrices. Based on this first guess, the way EM algorithm converge is different with the random guess version. We found that this time it took less step to reach converge.Because the initialization of the parameters are optimized during the process of the k-means.

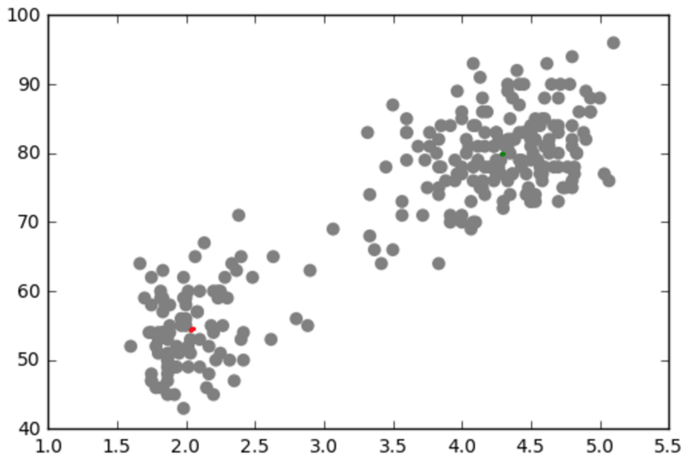
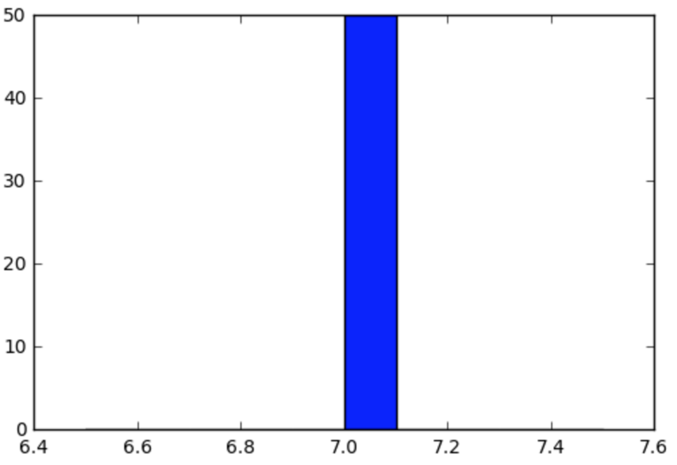


Figure 3: Distribution of the total number of iterations

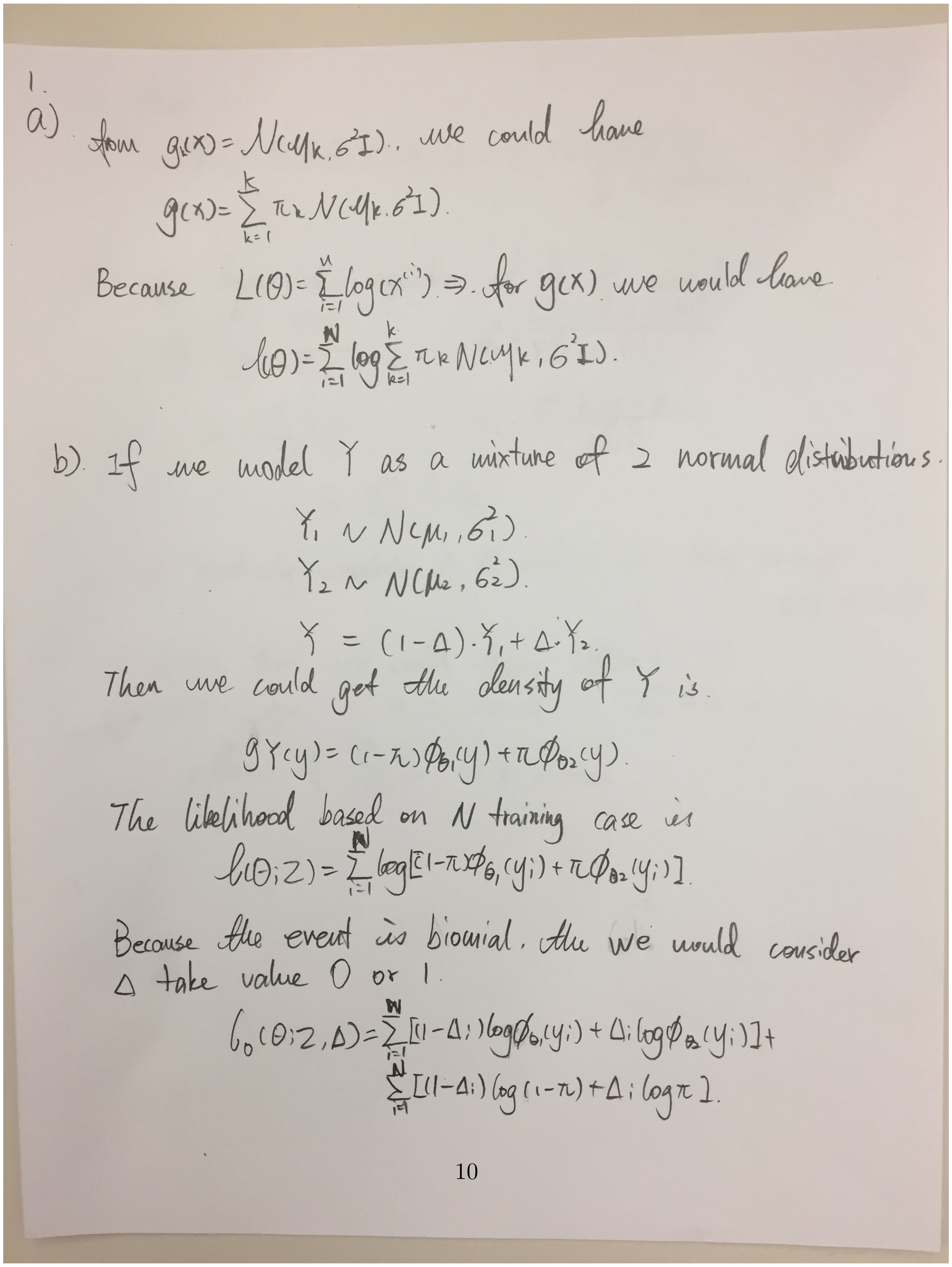
Since according the labeled result of K-means, each time the initialization of parameter are the same, so the distribution of the total iteration till converge are the same.

```
(array([  0.,   0.,   0.,   0.,   0.,  50.,   0.,   0.,   0.,
 array([ 6.5,  6.6,  6.7,  6.8,  6.9,  7. ,  7.1,  7.2,  7.3,
 <a list of 10 Patch objects>)
```



3 Written exercise

3.1 HTF EX 14.2(Gaussian mixture model and EM algorithm



1.

a) from $g_k(x) = N(\mu_k, \sigma^2 I)$.. we could have

$$g(x) = \sum_{k=1}^{k} \pi_k N(\mu_k, \sigma^2 I).$$

Because $L(\theta) = \sum_{i=1}^{n} \log(x^{(i)})$ $\Rightarrow$. for $g(x)$ we would have

$$l(\theta) = \sum_{i=1}^{N} \log \sum_{k=1}^{k} \pi_k N(\mu_k, \sigma^2 I).$$

b). If we model Y as a mixture of 2 normal distributions.

$$Y_1 \sim N(\mu_1, \sigma_1^2).$$
$$Y_2 \sim N(\mu_2, \sigma_2^2).$$
$$Y = (1-\Delta)\cdot Y_1 + \Delta \cdot Y_2.$$

Then we could get the density of Y is.

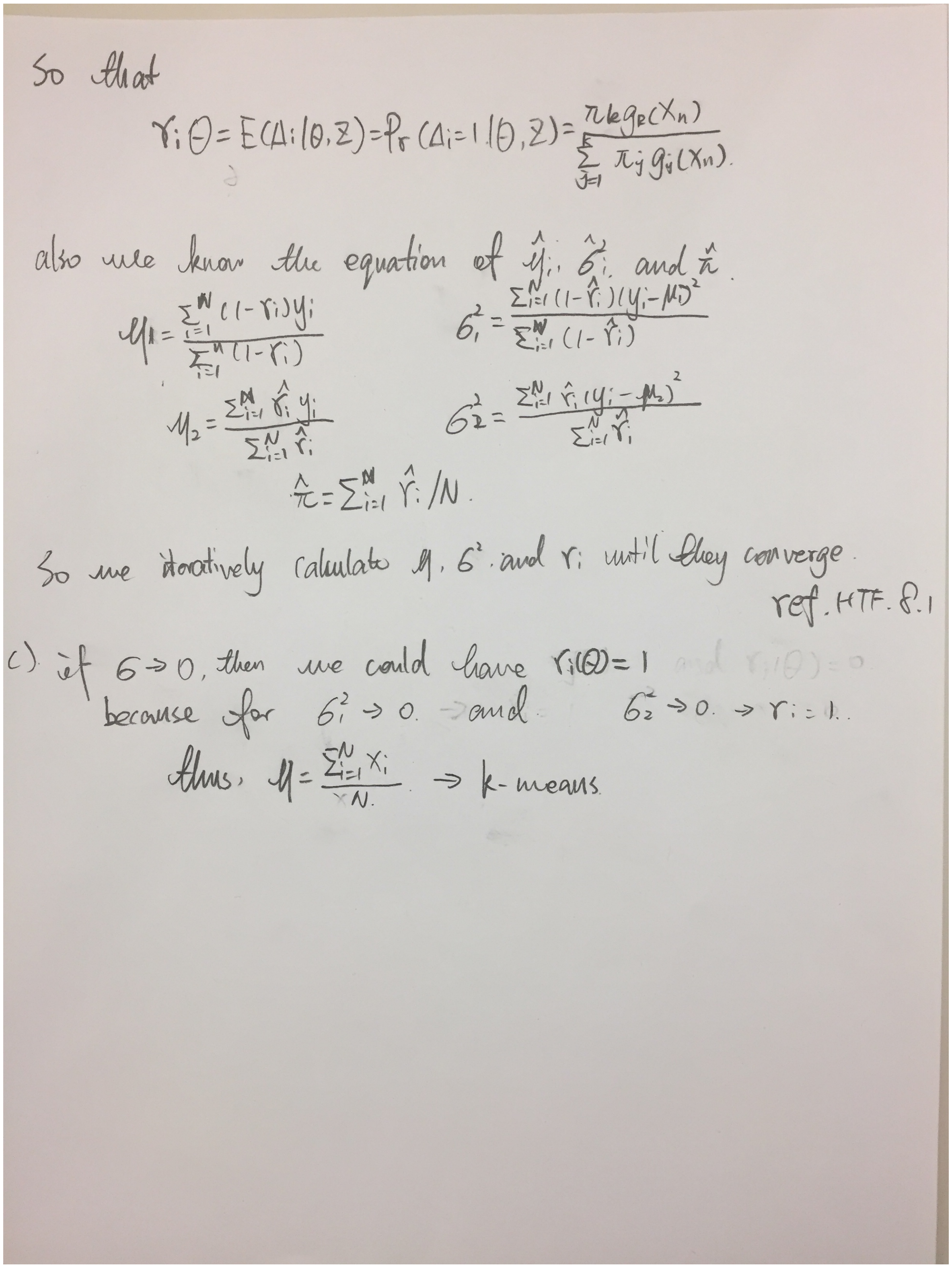
$$g_Y(y) = (1-\pi)\phi_{\theta_1}(y) + \pi \phi_{\theta_2}(y).$$

The likelihood based on N training case is

$$l(\theta; Z) = \sum_{i=1}^{N} \log[(1-\pi)\phi_{\theta_1}(y_i) + \pi \phi_{\theta_2}(y_i)].$$

Because the event is biomial. the we would consider Δ take value 0 or 1.

$$l_0(\theta; Z, \Delta) = \sum_{i=1}^{N} [(1-\Delta_i)\log\phi_{\theta_1}(y_i) + \Delta_i \log\phi_{\theta_2}(y_i)] + \sum_{i=1}^{N} [(1-\Delta_i)\log(1-\pi) + \Delta_i \log \pi].$$



So that

$$\gamma_i \Theta = E(\Delta_i|\theta, Z) = P_r(\Delta_i = 1|\theta, Z) = \frac{\pi_k g_k(x_n)}{\sum_{j=1}^{k} \pi_j g_j(x_n)}.$$

also we know the equation of $\hat{y}_i$, $\hat{\sigma}_i$ and $\hat{\pi}$.

$$\mu_1 = \frac{\sum_{i=1}^{N} (1-\gamma_i) y_i}{\sum_{i=1}^{N} (1-\gamma_i)} \qquad \sigma_1^2 = \frac{\sum_{i=1}^{N} (1-\hat{\gamma}_i)(y_i-\mu_1)^2}{\sum_{i=1}^{N} (1-\hat{\gamma}_i)}$$

$$\mu_2 = \frac{\sum_{i=1}^{N} \hat{\gamma}_i y_i}{\sum_{i=1}^{N} \hat{\gamma}_i} \qquad \sigma_2^2 = \frac{\sum_{i=1}^{N} \hat{\gamma}_i (y_i-\mu_2)^2}{\sum_{i=1}^{N} \hat{\gamma}_i}$$

$$\hat{\pi} = \sum_{i=1}^{N} \hat{\gamma}_i / N.$$

So we iteratively calculate μ, σ^2, and γ_i until they converge.

ref. HTF. 8.1

c). if $\sigma \to 0$, then we could have $\gamma_i(\Theta) = 1$

because for $\sigma_1^2 \to 0$. $\to$ and $\sigma_2^2 \to 0$. $\to \gamma_i = 1$.

thus, $\mu = \frac{\sum_{i=1}^{N} x_i}{N}$. $\to$ k-means.

3.2 HTF EX 14.8(Procrustes problem)

The problem is:

$$\min_{\mu,R} ||X_2 - (X_1 R + 1\mu^T)||_F$$

Here

$$||X||_F^2 = tr(X^T X)$$

Then the problem becomes

$$\min_{\mu,R} tr(X_2 - (X_1 R + \vec{1}\mu^T))^T (X_2 - (X_1 R + \vec{1}\mu^T))$$

open the quadratic expression:

$$\min_{\mu,R} tr((X_2 - X_1 R)^T (X_2 - X_1 R) - (X_2 - X_1 R)^T \vec{1}\mu^T - \mu\vec{1}^T (X_2 - X_1 R) + \mu\vec{1}^T \vec{1}\mu^T)$$

take derivative with respect to mu and get:

$$\vec{\mu} = \vec{x_2} - R\vec{x_1}$$

$$\begin{aligned} R &= arg\min ||X_2 - X_1 R||_F^2 \\ &= arg\min \langle X_2 - RX_1, X_2 - RX_1 \rangle \\ &= arg\min ||A||_F^2 + ||B||_F^2 - 2\langle X_1 R, X_2 \rangle \\ &= arg\max \left\langle R, BA^T \right\rangle \\ &= arg\max \left\langle V^T R^R U, \Sigma \right\rangle \\ &= UV^T \end{aligned}$$

3.3 HTF EX 14.11(Multidimensional Scaling)

The MDS problem is to minimize:

$$S_C(z_1, z_2, ..., z_N) = \sum_{i,i'} (S_{ii'} - \langle z_i - \bar{z}, z_{i'} - \bar{z} \rangle)^2$$

let $z_i - \bar{z} = m_i$

$$\begin{aligned} S_C(z_1, z_2, ..., z_N) &= tr((S - MM^T)^T (S - MM^T)) \\ &= tr((S^T - MM^T)(S - MM^T)) \\ &= tr(S^T S - S^T MM^T - MM^T S + MM^T MM^T) \end{aligned}$$

take derivative of MM^T

$$\frac{\partial S_C}{\partial MM^T} = -S - S + 2MM^T$$

$$S_C = MM^T$$

According to the eigenvalue decomposition of S_c

$$S_C = EDD^TE^T = (ED)(ED)^T = MM^T$$

so z_i are the rows of E_kD_k