CS 5875 Applied Machine Learning

Homework 3

Weiming Zhang, Yan Jiang

I. Objective

In this homework we are going to cover sentiment analysis of online reviews using K-means, bag-of-words and PCA in question 1. We will be covering EM algorithm in question 2.

II. Problem 1: Sentiment analysis of online reviews.

a) *Download Sentiment Labelled Sentences Data Set. There are three data files under the root folder. yelp_labelled.txt, amazon_cells_labelled.txt and imdb_labelled.txt. Parse each file with the specifications in readme.txt. Are the labels balanced? If not, what's the ratio between the two labels? Explain how you process these files.*

The labels are **balanced** according to the readme.txt, with 500 positive and 500 negative in each file. During the process, all 3 files are combined and parse into a set with first column being sentences, second being labels.

b) *Pick your preprocessing strategy. Since these sentences are online reviews, they may contain significant amounts of noise and garbage. You may or may not want to do one or all of the following. Explain the reasons for each of your decision (why or why not).*

Lowercase all of the words:
Yes, lowercasing all words will help with keyword matching.
Lemmatization of all the words:
Yes, this will help generalize words which help with matching accuracy.
Strip punctuation:
Yes, removing punctuation avoids interruption of punctuations to the words matching, which can affect accuracy.
Strip the stop words:
Yes, words like these are common but have minimal value. Removing stop words can help extract more important information.

c) *Split training and testing set. In this assignment, for each file, please use the first 400 instances for each label as the training set and the remaining 100 instances as testing set. In total, there are 2400 reviews for training and 600 reviews for testing.*

d) *Bag of Words model. Extract features and then represent each review using bag of words model, i.e., every word in the review becomes its own element in a feature vector. In order to do this, first, make one pass through all the reviews in the training set (Explain why we can't use testing set at this point) and build a dictionary of unique words. Then, make another pass through the review in both the training set and testing set and count up the occurrences of each word in your dictionary. The ith element of a review's feature vector is the number of occurrences of the ith dictionary word in the review. Implement the bag of words model and report feature vectors of any two reviews in the training set.*

During the training step we must not touch the test data. Using test data will cause overfitting.
Here are two reviews in the training set:

```
Review 1:
So there is no way for me to plug it in here in the US unless I go by a converter.
After post processing:
way plug us unless go convert
Feature vector:
[1 1 1 ..., 0 0 0]

Review 2:
Good case, Excellent value.
After post processing:
good case excel valu
Feature vector:
[0 0 0 ..., 0 0 0]
```

e) *Pick your postprocessing strategy. Since the vast majority of English words will not appear in most of the reviews, most of the feature vector elements will be 0. This suggests that we need a postprocessing or normalization strategy that combats the huge variance of the elements in the feature vector. You may want to use one of the following strategies. Whatever choices you make, explain why you made the decision.*

We use **Tf–idf term weighting** for post processing. This "re-weight" post process is great for dealing with sparse words feature set with most of entries being 0 and not adding much information. More info can be found at: http://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting

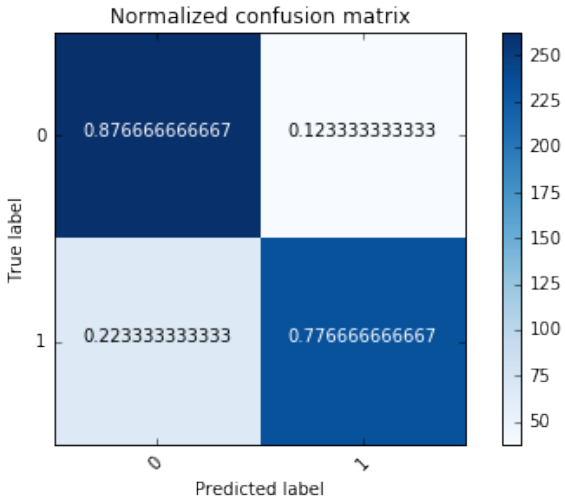
f) *Training set clustering. For the feature vectors you computed from (e). Implement K-means algorithm to divide the training set into 2 clusters (i.e., K = 2). Report the centers of both clusters. Inspect the clustering results and the labels associated with each instance, and evaluate the performance of your K-means algorithm and bag of words feature representation.*

Two clusters with entries, 1101: 604 pos + 497 neg, 1299: 596 pos + 604 neg
Average accuracy: **53.58%**

g) *Sentiment prediction. Train a logistic regression model (you can use existing packages here) on the training set and test on the testing set. Report the classification accuracy and confusion matrix. Inspecting the weight vector of the logistic regression, what are the words that play the most important roles in deciding the sentiment of the reviews?*

Accuracy: **82.67%**
Confusion Matrix:

```
      0[0.88   0.12]
      1[0.22   0.78]
        0       1
```



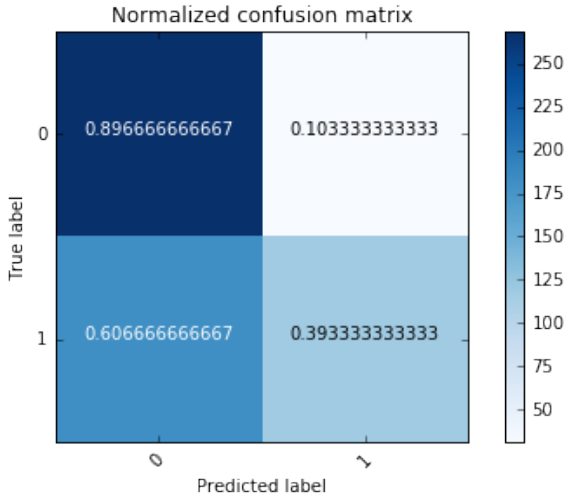
Most significant words:

```
great love bad excel good nice poor
delici amaz best
```

h) *N-gram model. Similar to the bag of words model, but now you build up a dictionary of ngrams, which are contiguous sequences of words. For example, "Alice fell down the rabbit hole" would then map to the 2-grams sequence: ["Alice fell", "fell down", "down the", "the rabbit", "rabbit hole"], and all five of those symbols would be members of the n-gram dictionary. Try n = 2, repeat (d)-(g) and report your results.*

Two clusters with entries, 2399: 1200 pos + 1199 neg, 1 neg
K-Means Average Accuracy: **50.04%**
Logistic Regression Accuracy: **64.5%**
Confusion Matrix:

```
      0[0.9    0.1]
      1[0.61   0.39]
        0       1
```



Most significant 2-grams:

```
work great; highli recommend; wast
time; great phone; disappoint;
wast money; one best; great product
10 10; food good
```

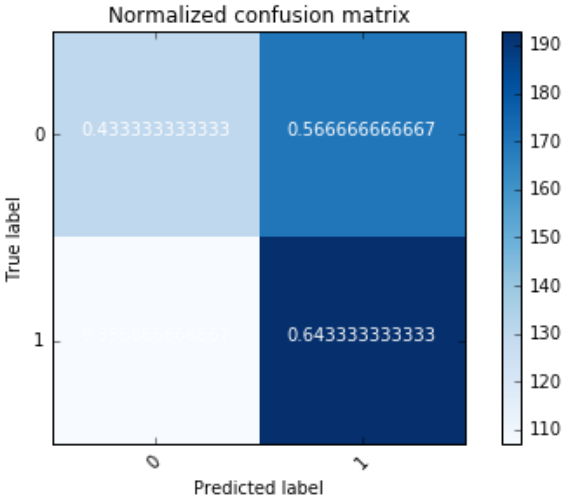
Note that there is a single word in the top 10 2-grams because after processing some sentence became just one word.

i) *PCA for bag of words model. The features in the bag of words model have large redundancy. Implement PCA to reduce the dimension of features calculated in (e) to 10, 50 and 100 respectively. Using these lower-dimensional feature vectors and repeat (f), (g). Report corresponding clustering and classification results. (Note: You should implement PCA yourself, but you can use numpy.svd or some other SVD package. Feel free to double-check your PCA implementation against an existing one)*

PCA-10
Two clusters with entries, 780: 318 pos + 462 neg, 1620: 882 pos + 738 neg
K-Means Average Accuracy: **56%**
Logistic Regression Accuracy: **53.83%**

Confusion Matrix:

0[0.43 0.57]
1[0.36 0.64]



PCA-50

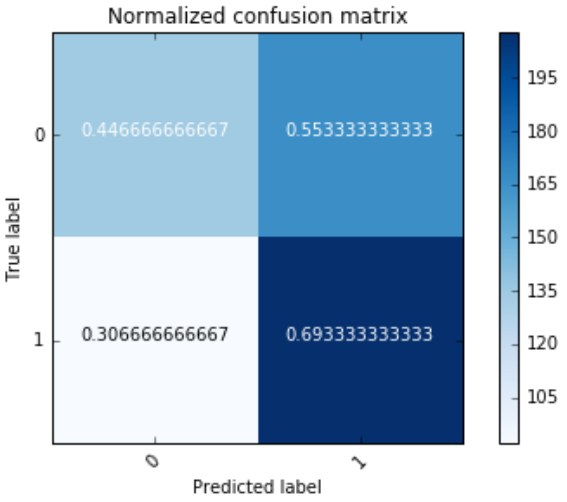
Two clusters with entries, 707: 217 pos + 490 neg, 1683: 993 pos + 710 neg

K-Means Average Accuracy: **61.38%**

Logistic Regression Accuracy: **57%**

Confusion Matrix:

0[0.45 0.55]
1[0.31 0.69]



PCA-100

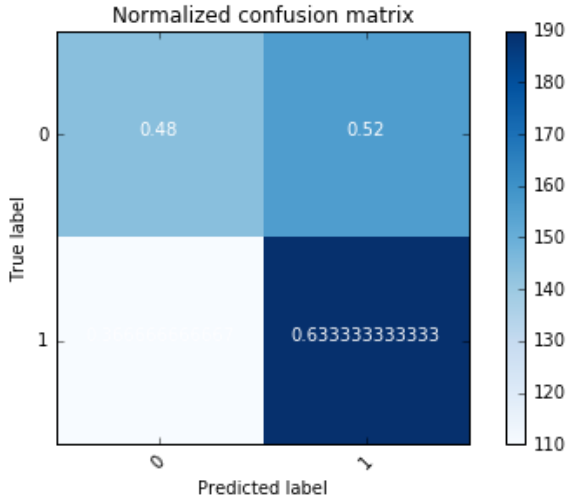
Two clusters with entries, 239: 116 pos + 123 neg, 1861: 1084 pos + 1077 neg

K-Means Average Accuracy: **50.83%**

Logistic Regression Accuracy: **55.67%**

Confusion Matrix:

0[0.48 0.52]
1[0.37 0.63]



j) Algorithms comparison and analysis. According to the above results, compare the performances of bag of words, 2-gram and PCA for bag of words. Which method performs best in the prediction task and why? What do you learn about the language that people use in online reviews (e.g., expressions that will make the posts positive/negative)? Hint: Inspect the clustering results and the weights learned from logistic regression.

Looking back, we find:

	Bag of words	2-gram	PCA
KM	Around 50%, not great. Using KM for reviews is not very optimal, as there are so many features and the vectors are so sparse. KM is not sensitive to the subtle difference between feature vectors.	Even worse. With more than 10,000 features the different between feature vectors become even harder to detect. Also 2-gram creates more bias.	Better than 2-gram and bag of words, as we are reducing the less significant info.
Log	Best of all. It took good use of all features and is sensitive to all the changes.	Worse than bag of words. With 2-gram we introduce a lot of unnecessary features than just words. Thus more bias.	Worse than bag of words as PCA reduces features for log regression

Regarding language people tend to use similar key words for online reviews whether it is a product or a piece of movie.

Expression for positive: `great love excel good nice delici amaz best`
Expression for positive: `bad poor`
With this knowledge, we can build tools to identify positive and negative reviews easily by just looking for keywords instead of everything.

III. Problem 2: EM algorithmandimplementation

(a) The parameters of Gaussian Mixture Model (GMM) can be estimated via the EM algorithm. Show that the alternating algorithm for K-means (in Lec. 11) is a special case of the EM algorithm and show the corresponding objective functions for E-step and M-step.

If we take σ to 0, π_k converges to 1 for the most probable cla ss and 0 for the rest. $\gamma_k$converges to a step function between 0 and 1.

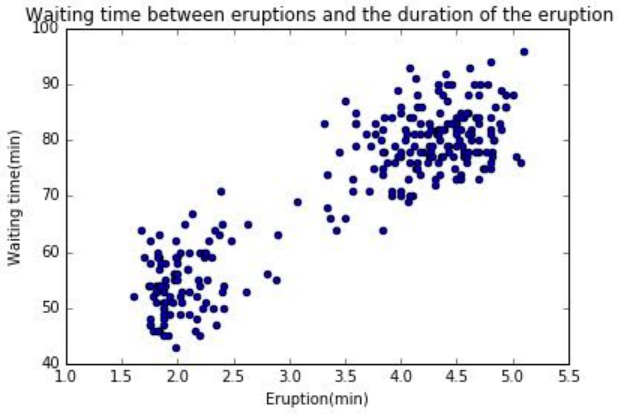
- E-step: compute responsibilities

$$\gamma_k = I(argmin_{1\le k\le K}\|x_i - x_k^-\|^2 = k), for\ i = 1,2,\dots N$$

- M-step: compute means

$$\hat{\mu_k} = \frac{\sum_{i=1}^{N}\gamma_{ik}y_i}{\sum_{i=1}^{N}\gamma_{ik}}$$

(b) Download the Old Faithful Geyser Dataset. The data file contains 272 observations of (eruption time, waiting time). Treat each entry as a 2 dimensional feature vector. Parse and plot all data points on 2-D plane.



(c) Implement a bimodal GMM model to fit all data points usin g EM algorithm. Explain the reasoning behind your termina tion criteria. For this problem, we assume the covariance m atrix is spherical (i.e., it has the form of σ2 I) and you can r andomly initialize Gaussian parameters. For evaluation pur poses, please submit the following figures:

- *Plot the trajectories of two mean vectors in 2 dimensions (i.e. coordinates vs. iteration).*
- *Run your program for 50 times with different initial par ameter guesses. Show the distribution of the total numbe r of iterations needed for algorithm to converge.*

i. **Data normalization**

For each feature of the training data, we do the following:
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
thus, each feature is linearly scaled to be within [0,1]

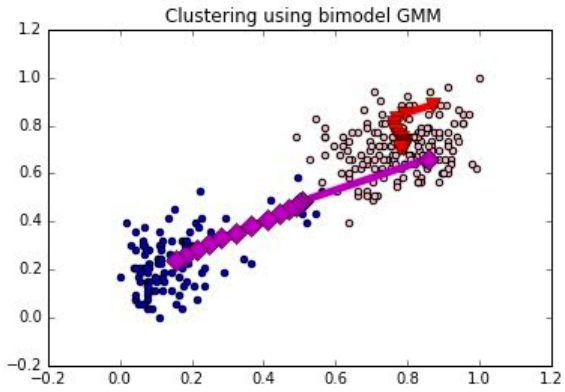
ii. **Termination Criteria**

We set a very small threshold $th = 1\times10^{-15}$
When the difference of the old and new log-likelihood is s maller than the threshold,

$$L_{new} - L_{old} < th$$

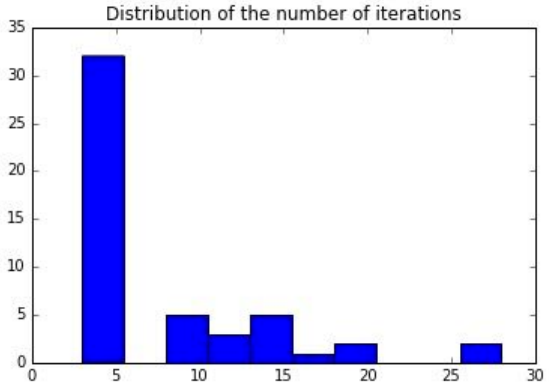
we consider it as a convergence.

iii. **The trajectories of two mean vectors**



The purple ■ indicates the mean vectors of cluster 1, and th e red ▲ indicates the mean vectors of the other cluster.

iv. **The distribution of the total number of iterations**



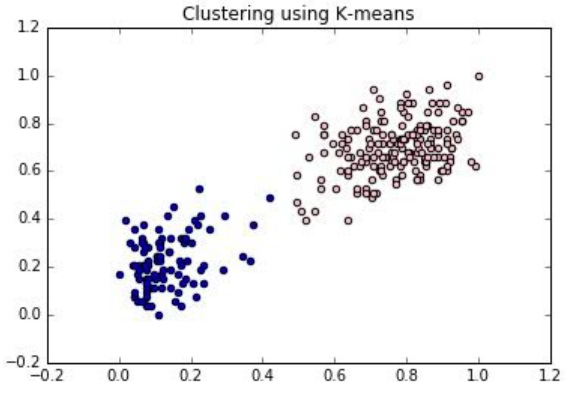
(X-axis: the number of iterations)
As we can see here, the algorithm usually needs less than 5 times to reach a convergence.

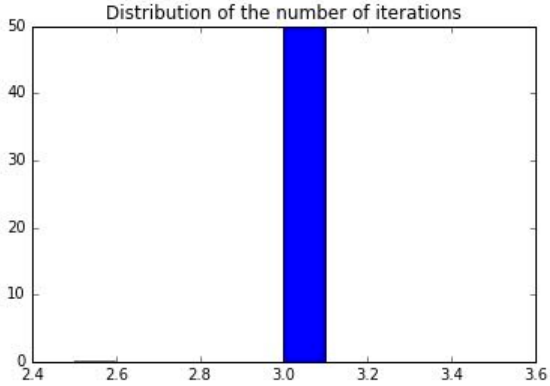
(d) Repeat the task in (c) but with the initial guesses of the par ameters generated from the following process:

- *Run a K-means algorithm over all the data points with K = 2 and label each point with one of the two clusters.*
- *Estimate the first guess of the mean and covariance ma trices using maximum likelihood over the labeled data*

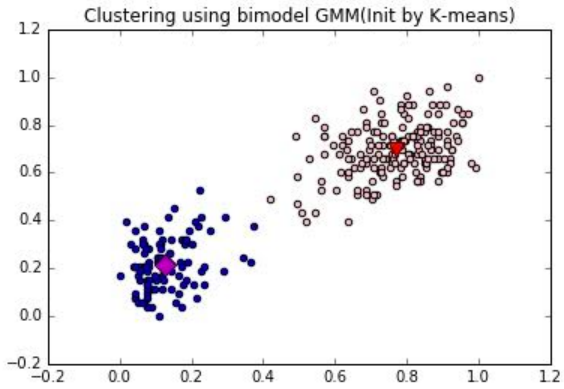
points. Compare the algorithm performances of (c) and (d).

i. **Clustering by k-means**





ii. **Clustering by GMM (Initialization by k-means)**



Similarly as in problem (c), we recorded the trajectories of the two mean vectors and ran the problem 50 times to observe the distribution of the numbers of iteration needed. As we can see from the above graphs, that the GMM algorithm initialized by k-means needs less iterations to reach convergence and its performance is more stable than random initialized GMM clustering.

CS 5875 Applied Machine Learning

Homework 3

Weiming Zhang, Yan Jiang

I. Objective

In this homework we are going to cover sentiment analysis of online reviews using K-means, bag-of-words and PCA in question 1. We will be covering EM algorithm in question 2.

II. Problem 1: Sentiment analysis of online reviews.

a) *Download Sentiment Labelled Sentences Data Set. There are three data files under the root folder. yelp_labelled.txt, amazon_cells_labelled.txt and imdb_labelled.txt. Parse each file with the specifications in readme.txt. Are the labels balanced? If not, what's the ratio between the two labels? Explain how you process these files.*

The labels are **balanced** according to the readme.txt, with 500 positive and 500 negative in each file. During the process, all 3 files are combined and parse into a set with first column being sentences, second being labels.

b) *Pick your preprocessing strategy. Since these sentences are online reviews, they may contain significant amounts of noise and garbage. You may or may not want to do one or all of the following. Explain the reasons for each of your decision (why or why not).*

Lowercase all of the words:
Yes, lowercasing all words will help with keyword matching.
Lemmatization of all the words:
Yes, this will help generalize words which help with matching accuracy.
Strip punctuation:
Yes, removing punctuation avoids interruption of punctuations to the words matching, which can affect accuracy.
Strip the stop words:
Yes, words like these are common but have minimal value. Removing stop words can help extract more important information.

c) *Split training and testing set. In this assignment, for each file, please use the first 400 instances for each label as the training set and the remaining 100 instances as testing set. In total, there are 2400 reviews for training and 600 reviews for testing.*

d) *Bag of Words model. Extract features and then represent each review using bag of words model, i.e., every word in the review becomes its own element in a feature vector. In order to do this, first, make one pass through all the reviews in the training set (Explain why we can't use testing set at this point) and build a dictionary of unique words. Then, make another pass through the review in both the training set and testing set and count up the occurrences of each word in your dictionary. The ith element of a review's feature vector is the number of occurrences of the ith dictionary word in the review. Implement the bag of words model and report feature vectors of any two reviews in the training set.*

During the training step we must not touch the test data. Using test data will cause overfitting.
Here are two reviews in the training set:

```
Review 1:
So there is no way for me to plug it in here in the US unless I go by a converter.
After post processing:
way plug us unless go convert
Feature vector:
[1 1 1 ..., 0 0 0]

Review 2:
Good case, Excellent value.
After post processing:
good case excel valu
Feature vector:
[0 0 0 ..., 0 0 0]
```

e) *Pick your postprocessing strategy. Since the vast majority of English words will not appear in most of the reviews, most of the feature vector elements will be 0. This suggests that we need a postprocessing or normalization strategy that combats the huge variance of the elements in the feature vector. You may want to use one of the following strategies. Whatever choices you make, explain why you made the decision.*

We use **Tf–idf term weighting** for post processing. This "re-weight" post process is great for dealing with sparse words feature set with most of entries being 0 and not adding much information. More info can be found at: http://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting

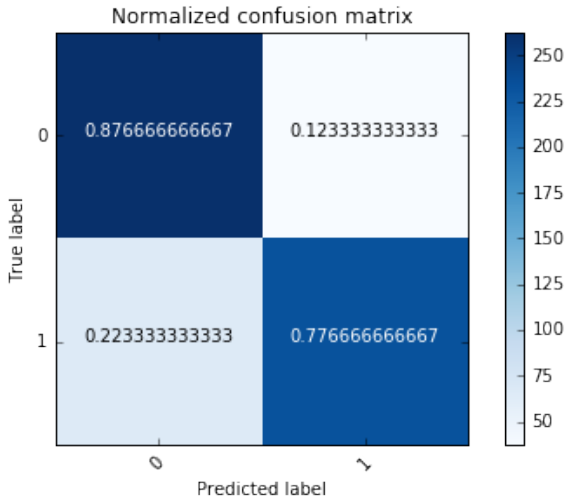
f) *Training set clustering. For the feature vectors you computed from (e). Implement K-means algorithm to divide the training set into 2 clusters (i.e., K = 2). Report the centers of both clusters. Inspect the clustering results and the labels associated with each instance, and evaluate the performance of your K-means algorithm and bag of words feature representation.*

Two clusters with entries, 1101: 604 pos + 497 neg, 1299: 596 pos + 604 neg
Average accuracy: **53.58%**

g) *Sentiment prediction. Train a logistic regression model (you can use existing packages here) on the training set and test on the testing set. Report the classification accuracy and confusion matrix. Inspecting the weight vector of the logistic regression, what are the words that play the most important roles in deciding the sentiment of the reviews?*

Accuracy: **82.67%**
Confusion Matrix:

```
     0[0.88   0.12]
     1[0.22   0.78]
       0      1
```



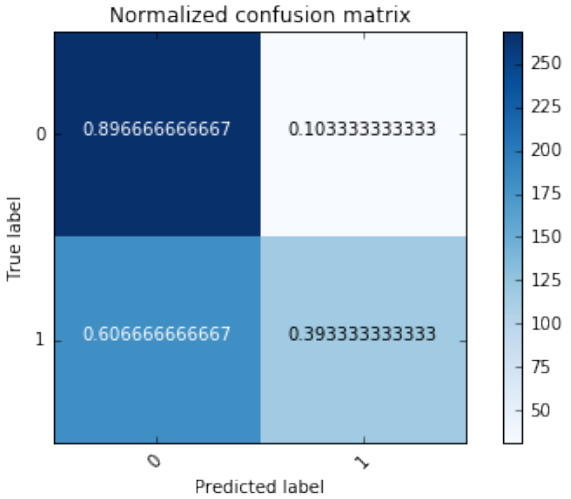
Most significant words:

```
great love bad excel good nice poor
delici amaz best
```

h) *N-gram model. Similar to the bag of words model, but now you build up a dictionary of ngrams, which are contiguous sequences of words. For example, "Alice fell down the rabbit hole" would then map to the 2-grams sequence: ["Alice fell", "fell down", "down the", "the rabbit", "rabbit hole"], and all five of those symbols would be members of the n-gram dictionary. Try n = 2, repeat (d)-(g) and report your results.*

Two clusters with entries, 2399: 1200 pos + 1199 neg, 1 neg
K-Means Average Accuracy: **50.04%**
Logistic Regression Accuracy: **64.5%**
Confusion Matrix:

```
     0[0.9    0.1]
     1[0.61   0.39]
       0      1
```



Most significant 2-grams:

```
work great; highli recommend; wast
time; great phone; disappoint;
wast money; one best; great product
10 10; food good
```

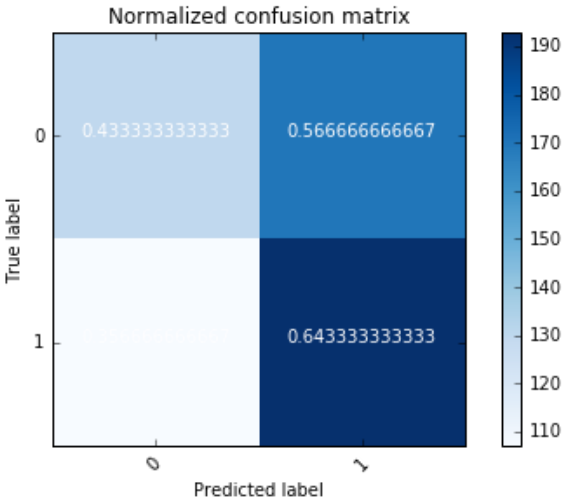
Note that there is a single word in the top 10 2-grams because after processing some sentence became just one word.

i) *PCA for bag of words model. The features in the bag of words model have large redundancy. Implement PCA to reduce the dimension of features calculated in (e) to 10, 50 and 100 respectively. Using these lower-dimensional feature vectors and repeat (f), (g). Report corresponding clustering and classification results. (Note: You should implement PCA yourself, but you can use numpy.svd or some other SVD package. Feel free to double-check your PCA implementation against an existing one)*

PCA-10
Two clusters with entries, 780: 318 pos + 462 neg, 1620: 882 pos + 738 neg
K-Means Average Accuracy: **56%**
Logistic Regression Accuracy: **53.83%**

Confusion Matrix:

```
0[0.43   0.57]
1[0.36   0.64]
```



PCA-50

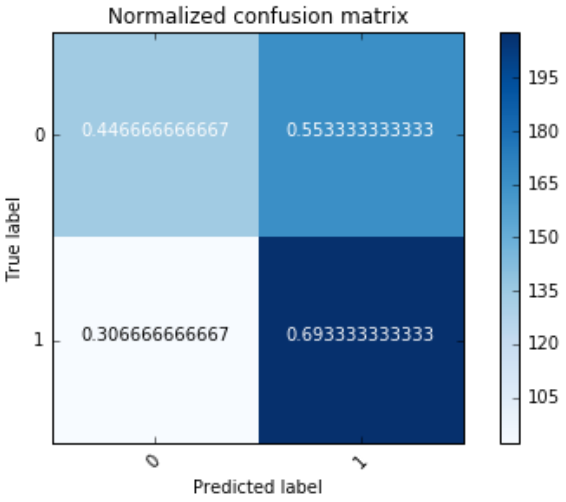
Two clusters with entries, 707: 217 pos + 490 neg, 1683: 993 pos + 710 neg

K-Means Average Accuracy: **61.38%**

Logistic Regression Accuracy: **57%**

Confusion Matrix:

```
0[0.45   0.55]
1[0.31   0.69]
```



PCA-100

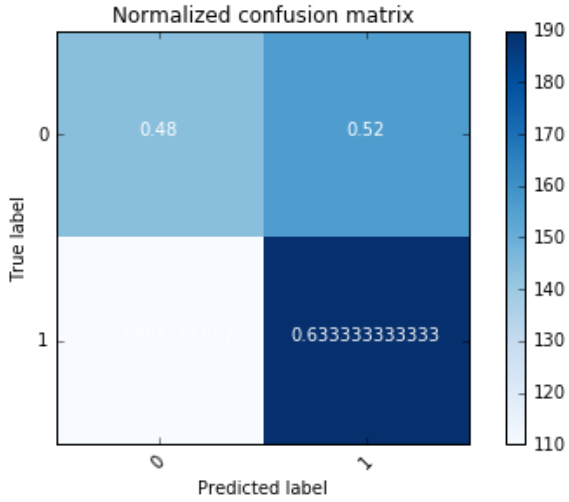
Two clusters with entries, 239: 116 pos + 123 neg, 1861: 1084 pos + 1077 neg

K-Means Average Accuracy: **50.83%**

Logistic Regression Accuracy: **55.67%**

Confusion Matrix:

```
0[0.48   0.52]
1[0.37   0.63]
```



j) Algorithms comparison and analysis. According to the above results, compare the performances of bag of words, 2-gram and PCA for bag of words. Which method performs best in the prediction task and why? What do you learn about the language that people use in online reviews (e.g., expressions that will make the posts positive/negative)? Hint: Inspect the clustering results and the weights learned from logistic regression.

Looking back, we find:

	Bag of words	2-gram	PCA
KM	Around 50%, not great. Using KM for reviews is not very optimal, as there are so many features and the vectors are so sparse. KM is not sensitive to the subtle difference between feature vectors.	Even worse. With more than 10,000 features the different between feature vectors become even harder to detect. Also 2-gram creates more bias.	Better than 2-gram and bag of words, as we are reducing the less significant info.
Log	Best of all. It took good use of all features and is sensitive to all the changes.	Worse than bag of words. With 2-gram we introduce a lot of unnecessary features than just words. Thus more bias.	Worse than bag of words as PCA reduces features for log regression.

Regarding language people tend to use similar key words for online reviews whether it is a product or a piece of movie.

Expression for positive: `great love excel good nice delici amaz best`
Expression for positive: `bad poor`
With this knowledge, we can build tools to identify positive and negative reviews easily by just looking for keywords instead of everything.

III. Problem 2: EM algorithmandimplementation

(a) The parameters of Gaussian Mixture Model (GMM) can be estimated via the EM algorithm. Show that the alternating algorithm for K-means (in Lec. 11) is a special case of the EM algorithm and show the corresponding objective functions for E-step and M-step.

If we take σ to 0, π_k converges to 1 for the most probable cla ss and 0 for the rest. $\gamma_k$converges to a step function between 0 and 1.

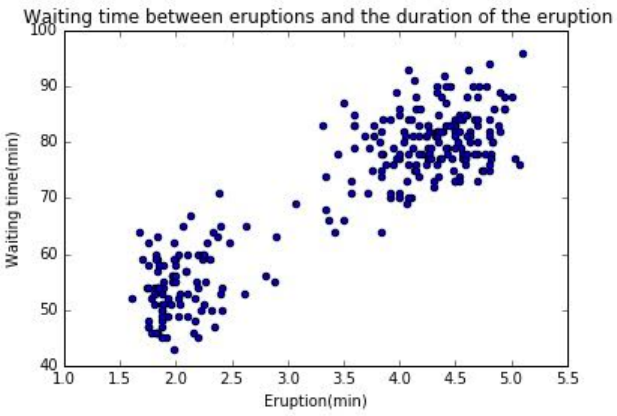
- E-step: compute responsibilities

$$\gamma_k = I(argmin_{1\le k\le K}\|x_i - x_k^-\|^2 = k), for\ i = 1,2,\dots N$$

- M-step: compute means

$$\hat{\mu_k} = \frac{\sum_{i=1}^{N}\gamma_{ik}y_i}{\sum_{i=1}^{N}\gamma_{ik}}$$

(b) Download the Old Faithful Geyser Dataset. The data file contains 272 observations of (eruption time, waiting time). Treat each entry as a 2 dimensional feature vector. Parse and plot all data points on 2-D plane.



(c) Implement a bimodal GMM model to fit all data points usin g EM algorithm. Explain the reasoning behind your termina tion criteria. For this problem, we assume the covariance m atrix is spherical (i.e., it has the form of σ2 I) and you can r andomly initialize Gaussian parameters. For evaluation pur poses, please submit the following figures:

- *Plot the trajectories of two mean vectors in 2 dimensions (i.e. coordinates vs. iteration).*
- *Run your program for 50 times with different initial par ameter guesses. Show the distribution of the total numbe r of iterations needed for algorithm to converge.*

i. **Data normalization**

For each feature of the training data, we do the following:
`X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))`
thus, each feature is linearly scaled to be within [0,1]

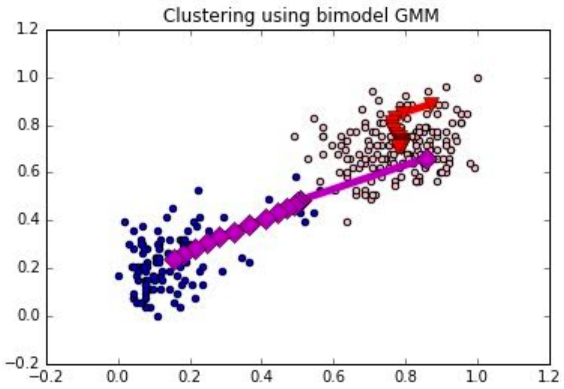
ii. **Termination Criteria**

We set a very small threshold $th = 1\times10^{-15}$
When the difference of the old and new log-likelihood is s maller than the threshold,

$$L_{new} - L_{old} < th$$

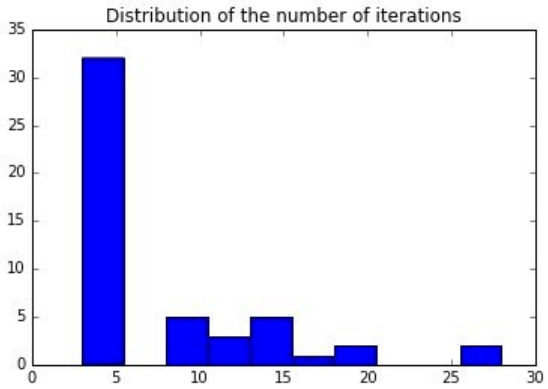
we consider it as a convergence.

iii. **The trajectories of two mean vectors**



The purple ■ indicates the mean vectors of cluster 1, and th e red ▲ indicates the mean vectors of the other cluster.

iv. **The distribution of the total number of iterations**



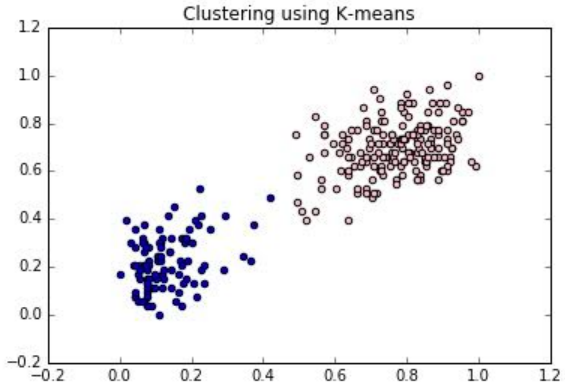
(X-axis: the number of iterations)
As we can see here, the algorithm usually needs less than 5 times to reach a convergence.

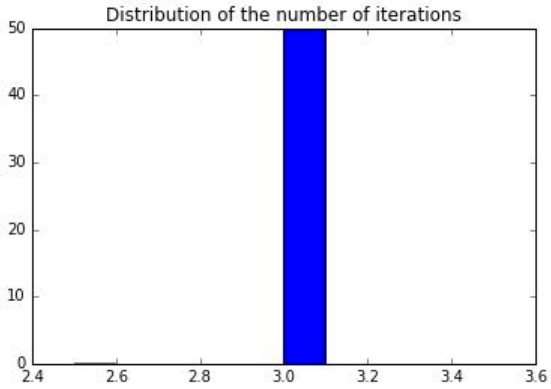
(d) Repeat the task in (c) but with the initial guesses of the par ameters generated from the following process:

- *Run a K-means algorithm over all the data points with K = 2 and label each point with one of the two clusters.*
- *Estimate the first guess of the mean and covariance ma trices using maximum likelihood over the labeled data*

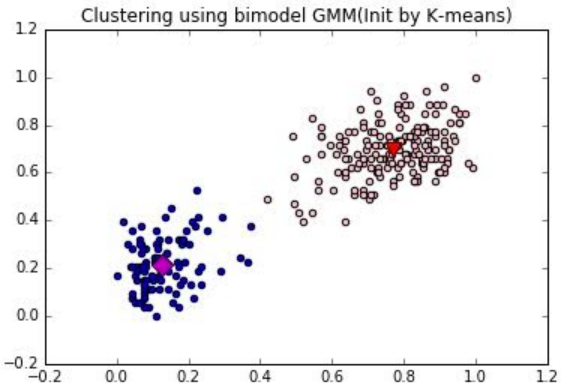
points. Compare the algorithm performances of (c) an d (d).

i. **Clustering by k-means**





ii. **Clustering by GMM (Initialization by k-means)**



Similarly as in problem (c), we recorded the trajectories of th e two mean vectors and ran the problem 50 times to observe the distribution of the numbers of iteration needed. As we ca n see from the above graphs, that the GMM algorithm initiali zed by k-means needs less iterations to reach convergence an d its performance is more stable than random initialized GM M clustering.

Applied Machine Learning

Homework3 Written Exercises

1. HTF Exercise14.2 (Gaussian mixture model and EM algorithm)

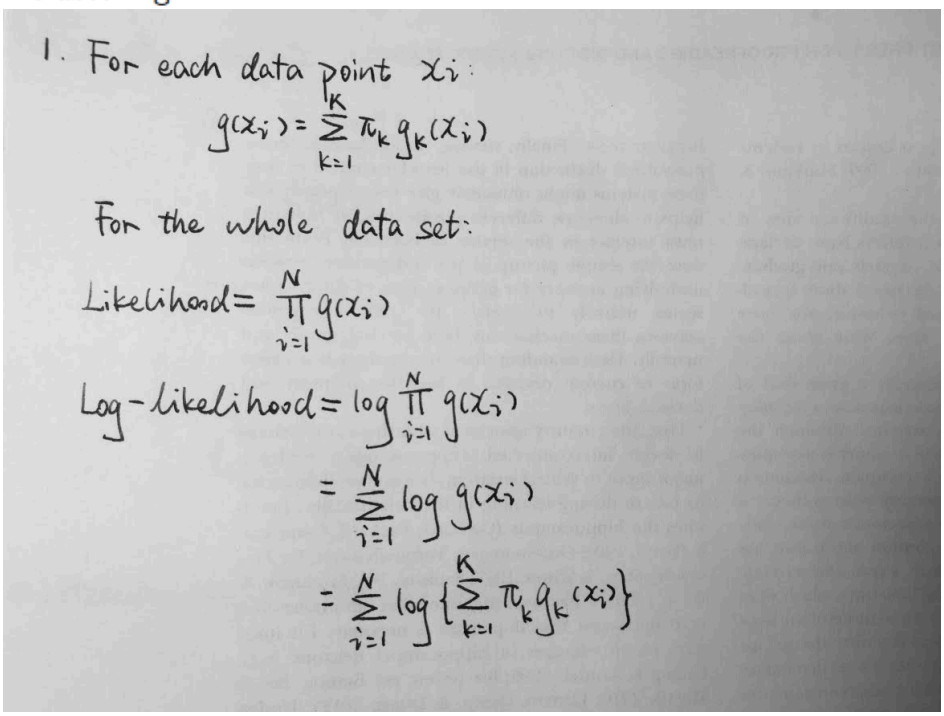
Ex. 14.2 Consider a mixture model density in p-dimensional feature space,

$$g(x) = \sum_{k=1}^{K} \pi_k g_k(x), \tag{14.114}$$

where $g_k = N(\mu_k, \mathbf{L}\cdot\sigma^2)$ and $\pi_k \geq 0\ \forall k$ with $\sum_k \pi_k = 1$. Here $\{\mu_k, \pi_k\}, k = 1, \ldots, K$ and σ^2 are unknown parameters.

Suppose we have data $x_1, x_2, \ldots, x_N \sim g(x)$ and we wish to fit the mixture model.

1. Write down the log-likelihood of the data
2. Derive an EM algorithm for computing the maximum likelihood estimates (see Section 8.1).
3. Show that if σ has a known value in the mixture model and we take $\sigma \to 0$, then in a sense this EM algorithm coincides with K-means clustering.



2(1) Take initial guesses for unknown parameters

$(\mu_k, \pi_k, \sigma_k^2)$ $k = 1, 2, \ldots K$

(2) E-step: compute responsibilities.

Define $\hat{\gamma}_{jk}$ as the probability that the jth observation is generated by component k.

$$\hat{\gamma}_{jk} = P(z=k \mid y_j, \theta)$$

$$= \frac{P(z=k, y_j \mid \theta)}{\sum_{k=1}^{K} P(z=k, y_j \mid \theta)}$$

$$= \frac{P(y_j \mid z=k, \theta) P(z=k \mid \theta)}{\sum_{k=1}^{K} P(y_j \mid z=k, \theta) P(z=k \mid \theta)}$$

$$= \frac{\pi_k \phi(y_j \mid \theta_k)}{\sum_{k=1}^{K} \pi_k \phi(y_j \mid \theta_k)}$$

(3) M-step: compute weighted means and variances

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_{ik} y_i}{\sum_{i=1}^{N} \gamma_{ik}}$$

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

$$\hat{\pi}_k = \frac{\sum_{i=1}^{N} \gamma_{ik}}{N}$$

(4) Iterate steps (2) and (3) until convergence.

3. If we take σ to 0.

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \gamma_{ik}(y_i - \hat{\mu}_i)^2}{\sum_{i=1}^{N} \gamma_{ik}} = 0$$

$$\sum_{i=1}^{N} \gamma_{ik}(y_i - \hat{\mu}_i)^2 = 0$$

$$\sum_{i=1}^{N} \pi_k g_k(x_i)(y_i - \hat{\mu}_i)^2 = 0$$

π_k converges to 1 for the most probable class and 0 for the rest. γ_{ik} converges to a step function between 0 and 1.

Resulting steps:

1) Take initial guesses for unknown parameters μ_k, $k = 1, 2, \ldots K$

2) E-step: compute responsibilities

$$\gamma_{ik} = I(\operatorname{argmin}_{1 \le k \le K} \|x_i - \bar{x}_k\|^2 = k) \text{ for } i = 1, 2, \ldots N$$

3) M-step: compute means

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_{ik} y_i}{\sum_{i=1}^{N} \gamma_{ik}}$$

4) Iterate steps 2 and 3 until convergence.

2. HTF Exercise14.8 (Procrustes problem.)

Ex. 14.8 Derive the solution (14.57) to the Procrustes problem (14.56). Derive also the solution to the Procrustes problem with scaling (14.58).

2. Since we want to evaluate

$$\operatorname{argmin}_{\mu,R} \|X_2 - (X_1 R + \vec{1}\mu^T)\|_F$$

which has the same solution as

$$\operatorname{argmin}_{\mu,R} \|X_2 - (X_1 R + \vec{1}\mu^T)\|_F^2$$

Since $\|X\|_F^2 = \operatorname{tr}(X^T X)$

$$\|X_2 - (X_1 R + \vec{1}\mu^T)\|_F^2 = \operatorname{tr}((X_2 - X_1 R - \vec{1}\mu^T)^T (X_2 - X_1 R - \vec{1}\mu^T))$$

$$((X_2 - X_1 R) - \vec{1}\mu^T)^T ((X_2 - X_1 R) - \vec{1}\mu^T)$$

$$= ((X_2 - X_1 R)^T - \mu \vec{1}^T)((X_2 - X_1 R) - \vec{1}\mu^T)$$

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

$$\triangleq (*)$$

Take the derivative of $\operatorname{tr}(*)$ and set it equal to zero.

$$-(X_2 - X_1 R)^T 1 - (X_2 - X_1 R)^T 1 + 2N\mu = 0$$

$$\Rightarrow \mu = \frac{1}{N}(X_2^T - R^T X_1^T) 1 = \bar{x}_2 - R^T \bar{x}_1$$

$$X_2 - X_1 R - 1\mu^T = X_2 - X_1 R - 1(\bar{x}_2^T - \bar{x}_1^T R)$$

$$= \tilde{X}_2 - \tilde{X}_1 R$$

To solve

$$\operatorname{argmin}_R \operatorname{tr}(\tilde{X}_2^T \tilde{X}_2 - \tilde{X}_2^T \tilde{X}_1 R - R^T \tilde{X}_1^T \tilde{X}_2 + R^T \tilde{X}_1^T \tilde{X}_1 R)$$

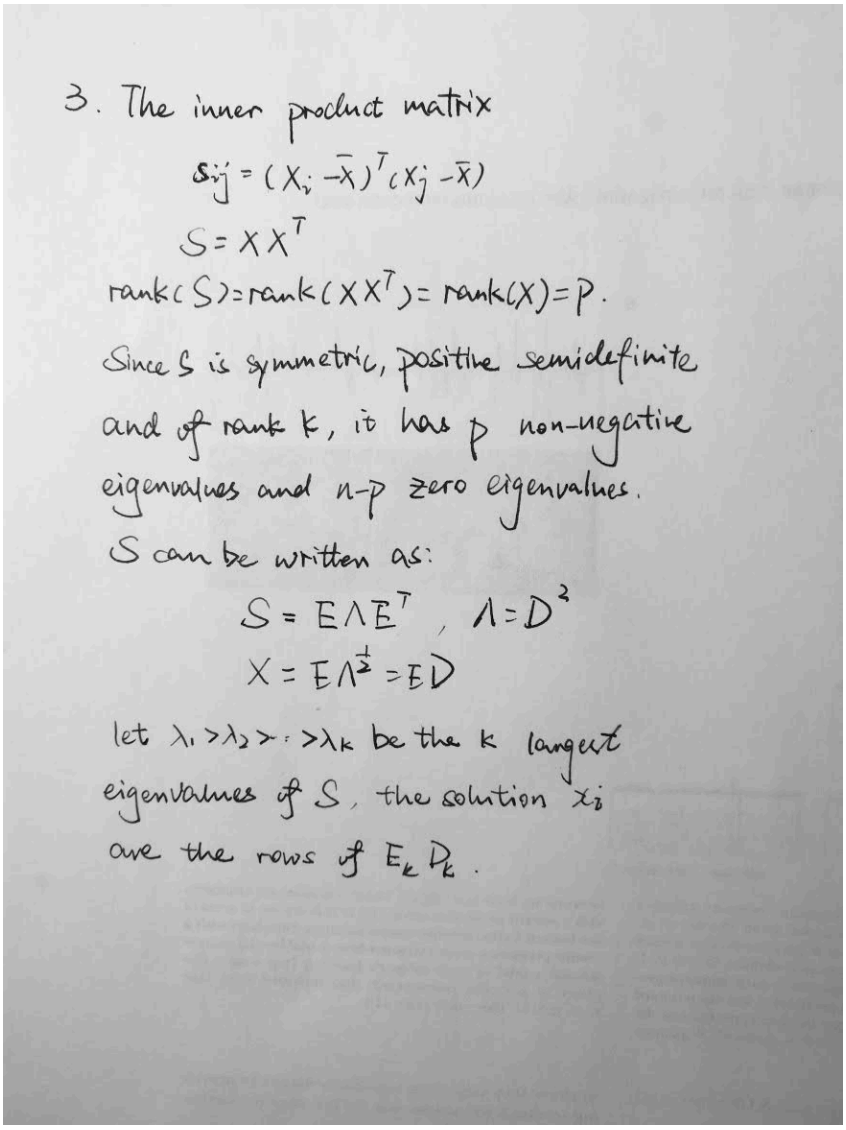
take the derivative and set it equal to zero:

$$-\tilde{X}^T \tilde{X}_2 - \tilde{X}_1^T \tilde{X}_2 + 2\tilde{X}_1^T \tilde{X}_1 R = 0$$

$$\Rightarrow R = (\tilde{X}_1^T X_1)^{-1} (\tilde{X}_1^T \tilde{X}_2)$$

3. HTF Exercise14.11 (Multidimensional Scaling.)

Ex. 14.11 *Classical multidimensional scaling.* Let $\mathbf{S}$ be the centered inner product matrix with elements $\langle x_i - \bar{x}, x_j - \bar{x} \rangle$. Let $\lambda_1 > \lambda_2 > \cdots > \lambda_k$ be the k largest eigenvalues of $\mathbf{S}$, with associated eigenvectors $\mathbf{E}_k = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k)$. Let $\mathbf{D}_k$ be a diagonal matrix with diagonal entries $\sqrt{\lambda_1}$, $\sqrt{\lambda_2}, \ldots, \sqrt{\lambda_k}$. Show that the solutions z_i to the classical scaling problem (14.100) are the *rows* of $\mathbf{E}_k \mathbf{D}_k$.



3. The inner product matrix

$$S_{ij} = (X_i - \bar{X})^T (X_j - \bar{X})$$

$$S = XX^T$$

$\text{rank}(S) = \text{rank}(XX^T) = \text{rank}(X) = p$.

Since S is symmetric, positive semidefinite and of rank k, it has p non-negative eigenvalues and $n-p$ zero eigenvalues.

S can be written as:

$$S = E \Lambda E^T, \quad \Lambda = D^2$$

$$X = E \Lambda^{\frac{1}{2}} = ED$$

let $\lambda_1 > \lambda_2 > \cdots > \lambda_k$ be the k largest eigenvalues of S, the solution z_i are the rows of $E_k D_k$.