Homework 3

CS 5785 Applied Machine Learning

Xialin Shen <xs293@cornell.edu>

An Le <aql6@cornell.edu>

9th November, 2017

PART A - PROGRAMMING EXERCISE

Question 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.

Answer:
There are 500 positive and 500 negative sentences each set, so the labels are balanced. We process it by appending them to 2 numpy arrays. One for sentence and one for its label.

(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).

Answer:
We employed the following preprocessing strategies:

(i) **Lowercase all words** for ease of word matching.
(ii) **Lemmatization** of all words helps cluster words with same meaning.
(iii) **Strip the stopwords**: remove words like 'the', 'is', 'are' which have no value in our analysis.
(iv) **Strip Punctuation:** punctuation again does not provide any value in matching. So remove it can possibly improve accuracy.

(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 i th dictionary word in the review. Implement the bag of words model and report feature vectors of any two reviews in the training set.

Answer:
We shouldn't touch the testing data during training. Doing so may lead overfit in training models.

Present any 2 reviews in training set

```
train_feature = np.array(train_feature, dtype=float)
```

```
train_data[0]
```

```
'way plug us unless go convert'
```

```
train_feature[0]
```

```
array([ 1.,  1.,  1., ...,  0.,  0.,  0.])
```

```
train_data[1]
```

```
'good case excel valu'
```

```
train_feature[1]
```

```
array([ 0.,  0.,  0., ...,  0.,  0.,  0.])
```

(e) Pick your post processing 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.

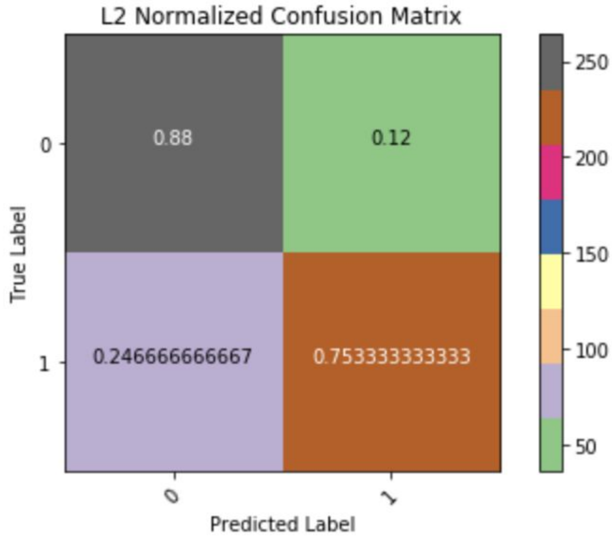
Answer:
We employ both l1 and l2 norm and compare since they are effective on sparse data. l2 pulls its elements into sphere. l1 pulls its elements into a square shaped target. Either way, they all force the elements come closer together in term of similarity.

After analyzing the scores we see that L2 does better than L1 in general.

(f) 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? Repeat this with a Naive Bayes classifier and compare performance.

<u>Logistic Regression:</u>

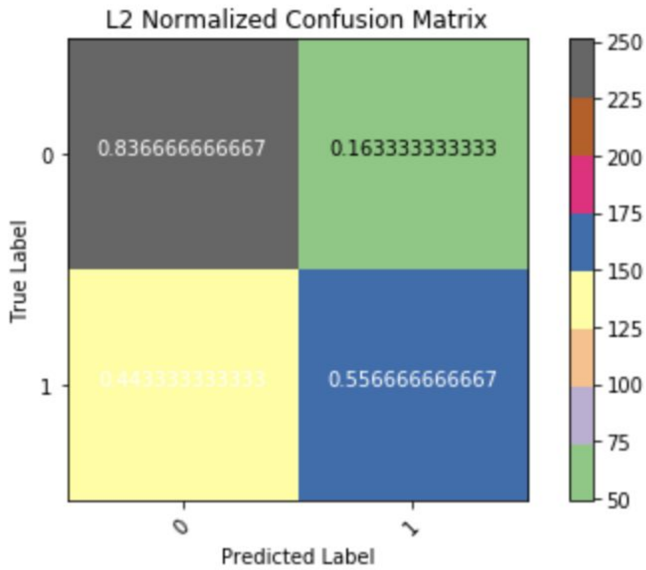
```
Accuracy: 0.8166666666666667
```




```
20 Significant Words:
great
love
bad
excel
nice
delici
amaz
poor
good
best
worst
fantast
terribl
well
wast
awesom
beauti
suck
comfort
aw
```

Naive Bayes:

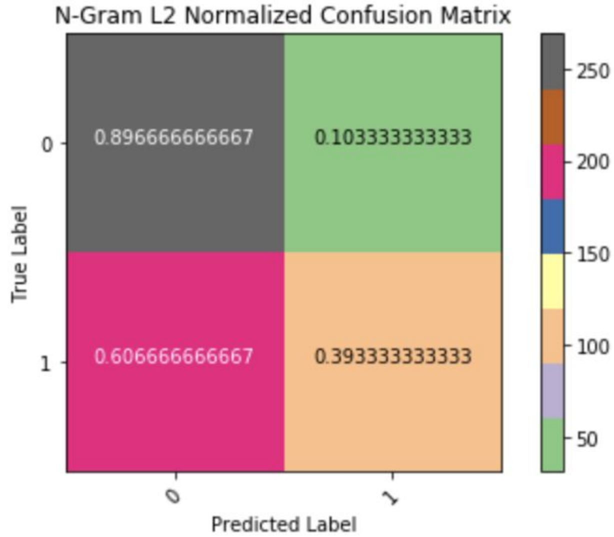
```
Naive Bayes Model with L2 norm
Accuracy: 0.6966666666666667
```



(g) N-gram model. Similar to the bag of words model, but now you build up a dictionary of n-grams, 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.

Logistic Regression:

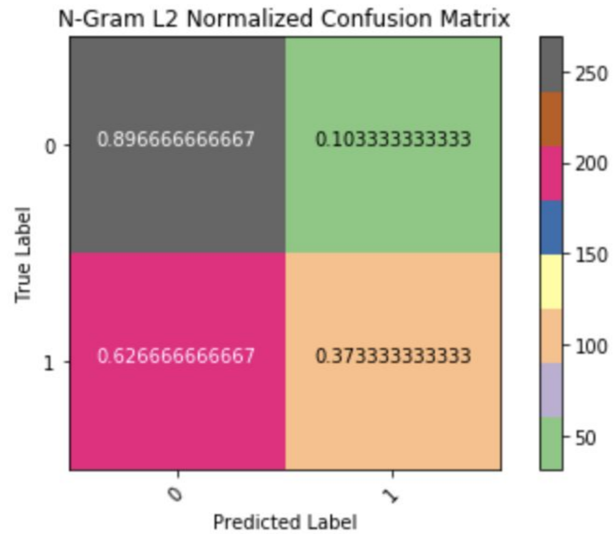
```
Regression Model with L2 norm
Accuracy: 0.645
```




```
20 Significant Words:
work great
highli recommend
wast time
one best
great phone
disappoint
wast money
great product
10 10
food good
easi use
custom servic
great food
realli good
good price
love place
love
great servic
food delici
horribl
```

Naive Bayes:

```
Naive Bayes Model with L2 norm
Accuracy: 0.635
```

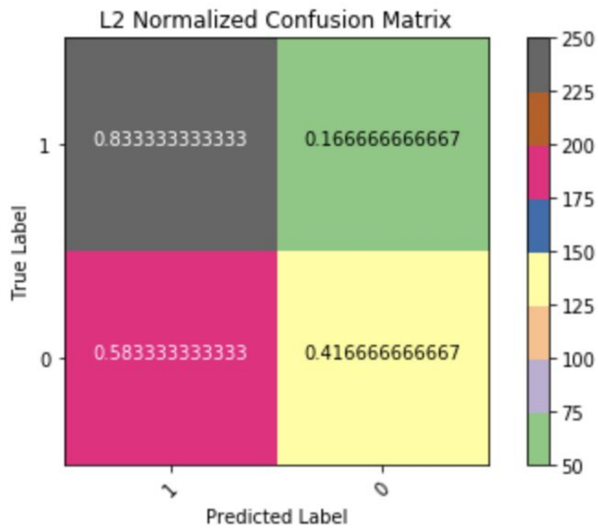


(h) PCA for bag of words model. The features in the bag of words model have large redundancy. Implement PCA to reduce the dimension of features calculated in (e) to 10, 50 and 100 respectively. Using these lower-dimensional feature vectors and repeat (f), (g). Report corresponding clustering and classification 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:

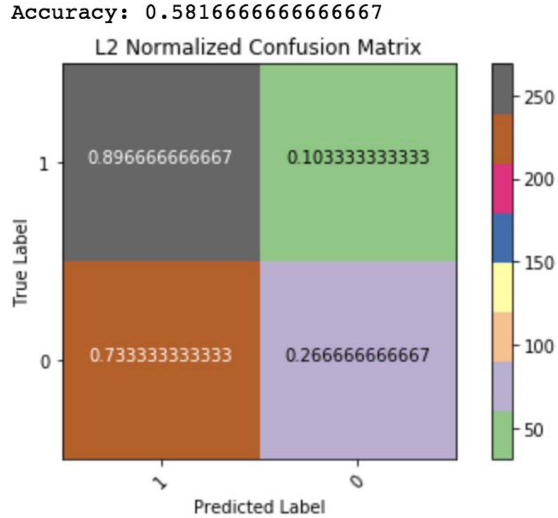
Bag of Words: Since we reduced the feature dimension to 10, there are only 10 significant words here.

```
PCA 10 Regression Model with L2 norm
Accuracy: 0.625
```



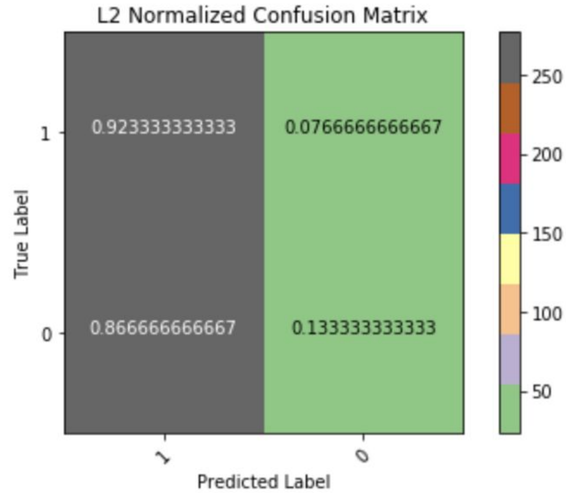

```
20 Significant Words:
way
plug
excel
case
convert
unless
valu
go
us
good
```

```
PCA 10 Naive Bayes Model with L2 norm
Accuracy: 0.5816666666666667
```



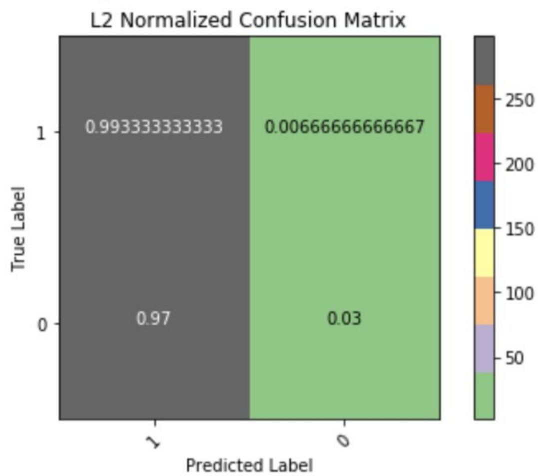
2-Gram:

```
PCA 10 Regression Model with L2 norm N-Gram
Accuracy: 0.5283333333333333
```




```
20 Significant Words:
way plug
good case
case excel
unless go
go convert
us unless
plug us
great jawbon
tie charger
excel valu
```

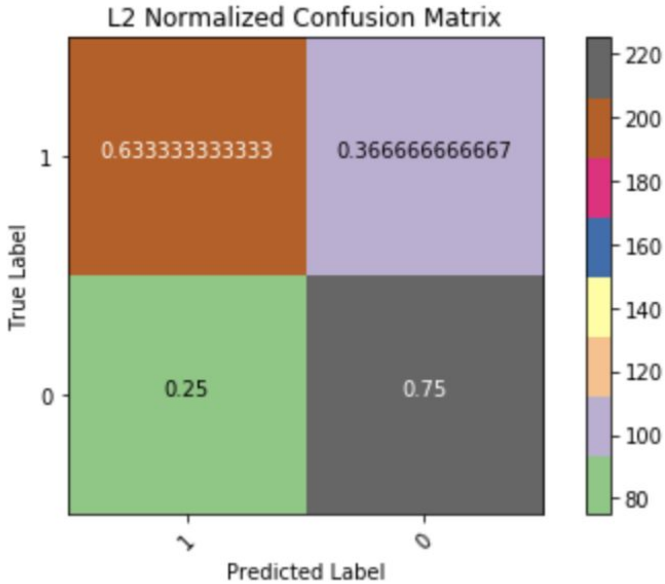
```
PCA 10 Naive Bayes Model with L2 norm N-Gram
Accuracy: 0.5116666666666667
```



PCA 50:

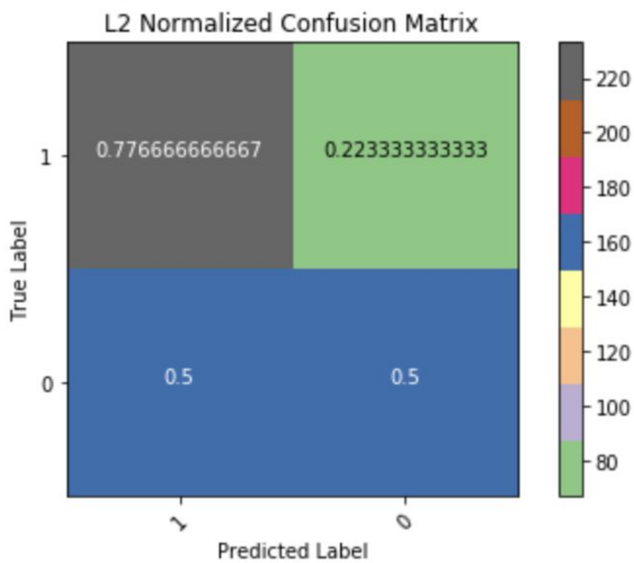
Bag of Words:

PCA 50 Regression Model with L2 norm
Accuracy: 0.6916666666666667



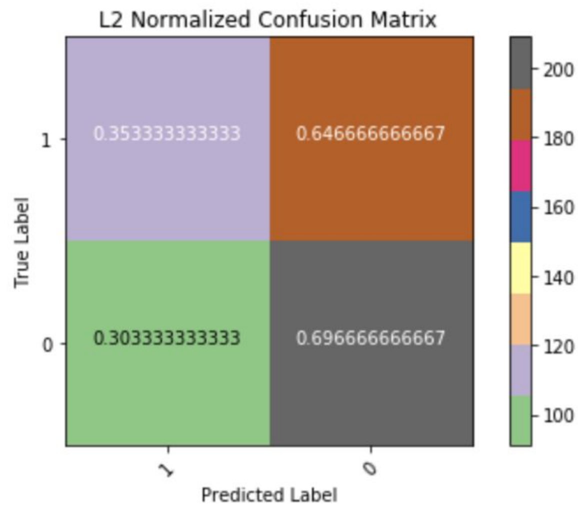
20 Significant Words:
way
decent
plug
imagin
owner
jawbon
excel
case
problem
tie
volum
mic
major
right
contact
razr
line
charger
origin
convert

PCA 50 Naive Bayes Model with L2 norm
Accuracy: 0.6383333333333333

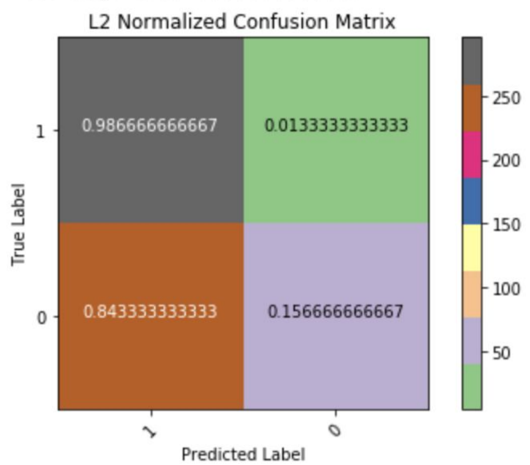


2-Gram:

PCA 50 Regression Model with L2 norm N-Gram
Accuracy: 0.525



PCA 50 Naive Bayes Model with L2 norm N-Gram
Accuracy: 0.5716666666666667

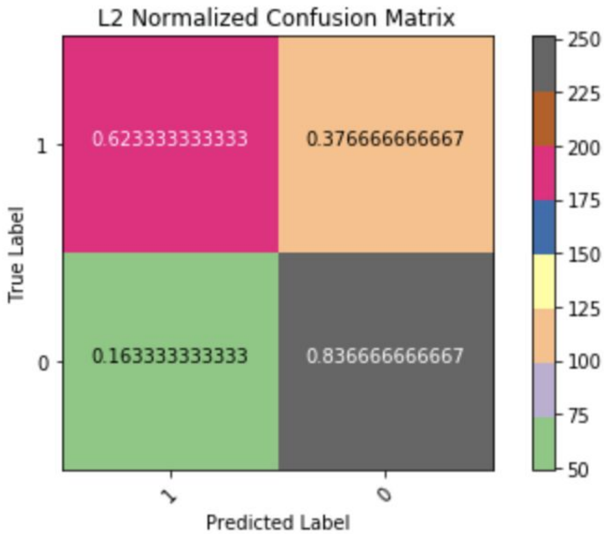


20 Significant Words:
way plug
good case
convers last
fun send
case excel
get decent
imagin fun
go origin
unless go
jiggl plug
decent volum
wast money
money time
get line
hundr contact
sever hundr
go convert
send one
dozen sever
sever dozen

PCA 100:

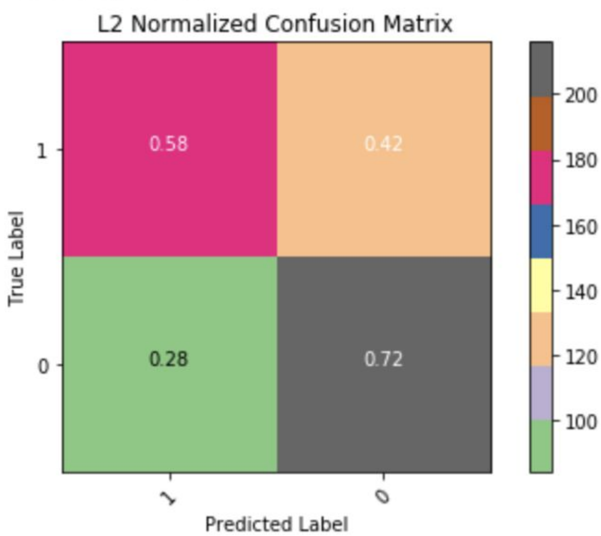
Bag of Words:

PCA 100 Regression Model with L2 norm
Accuracy: 0.73



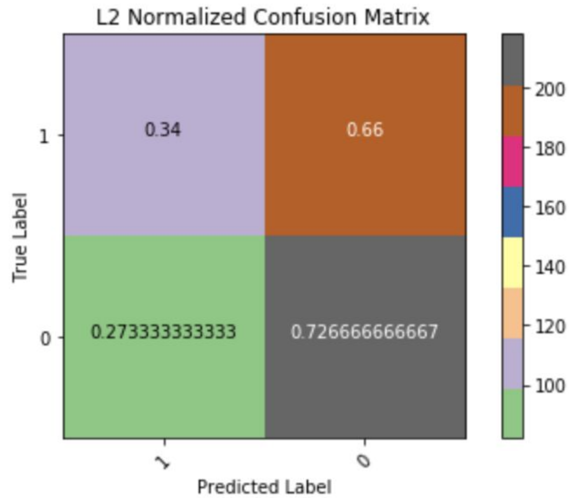
20 Significant Words:
way
decent
plug
would
blue
imagin
owner
jawbon
excel
5
case
problem
tie
advis
use
fire
mislead
commerci
mere
volum

PCA 100 Naive Bayes Model with L2 norm
Accuracy: 0.65



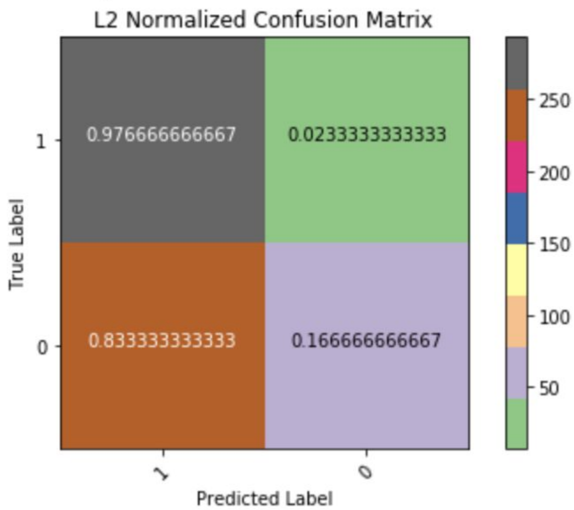
2-Gram:

PCA 100 Regression Model with L2 norm N-Gram
Accuracy: 0.5333333333333333



20 Significant Words:
way plug
good case
convers last
fun send
case excel
get decent
imagin fun
go origin
unless go
jiggl plug
decent volum
wast money
money time
get line
direct could
hundr contact
sever hundr
commerci mislead
go convert
send one

PCA 100 Naive Bayes Model with L2 norm N-Gram
Accuracy: 0.5716666666666667



(i) 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 on- line reviews (e.g., expressions that will make the posts positive/negative)? Hint: Inspect the clustering results and the weights learned from logistic regression.

Bag of Words (BOW)	2-Gram (2G)	PCA
82%	65%	PCA 10 - BOW: 63%, 2G: 53% PCA 50 - BOW: 69% , 2G: 53% PCA 100 - BOW: 73% , 2G: 53%
Best performance. It contains most number of features and thus helps Logistic Regression performs better.	N-Gram makes the data becomes sparser and thus performs worst.	PCA while reducing the dimension it actually reduces the features feeding the Logistic Regression, thus perform not as good as BOW alone.

The online review for all 3 services share the similar words to describe good service regardless the type of service.

Question 2 - Clustering for Text Analysis

(a - 1) Cluster the documents using k-means and various values of k.

Load Data

```
def readFile(fname):
    # ref: https://stackoverflow.com/questions/3277503/how-do-i-read-a-file-line-by-line-into-a-list
    with open(fname, 'r', errors='ignore') as f:
        content = f.readlines()
    content = [x.strip() for x in content] # remove whitespace characters
    return content
```

```
science2kdocword = np.load("./data/science2k-doc-word.npy")
science2kworddoc = np.load("./data/science2k-word-doc.npy")
vocabs = readFile('./data/science2k-vocab.txt')
titles = readFile('./data/science2k-titles.txt')
```

Test on K values

```
def calculateDistribution(k, labels):
    dist = [0]*k
    for label in labels:
        dist[label] += 1
    return dist
```

```
def testDifferentKValue(k_value_range, data):
    cluster_distributions = []
    inertia_array = []
    for k in k_value_range:
        kmeans = KMeans(n_clusters=k, random_state=0).fit(data)
        cluster_distributions.append(calculateDistribution(k, kmeans.labels_))
        inertia_array.append(kmeans.inertia_)
    return cluster_distributions, inertia_array
```

```
def plotDistributions(k_value_range, cluster_distributions):
    plt.figure(figsize=(8, 50))
    plt.subplots_adjust(hspace=.7)

    for k in k_value_range:
        plt.subplot(20, 1, k + 1)
        plt.bar(range(1, k + 1), cluster_distributions[k - 1])
        plt.title('K = ' + str(k))
```

Clustering Documents

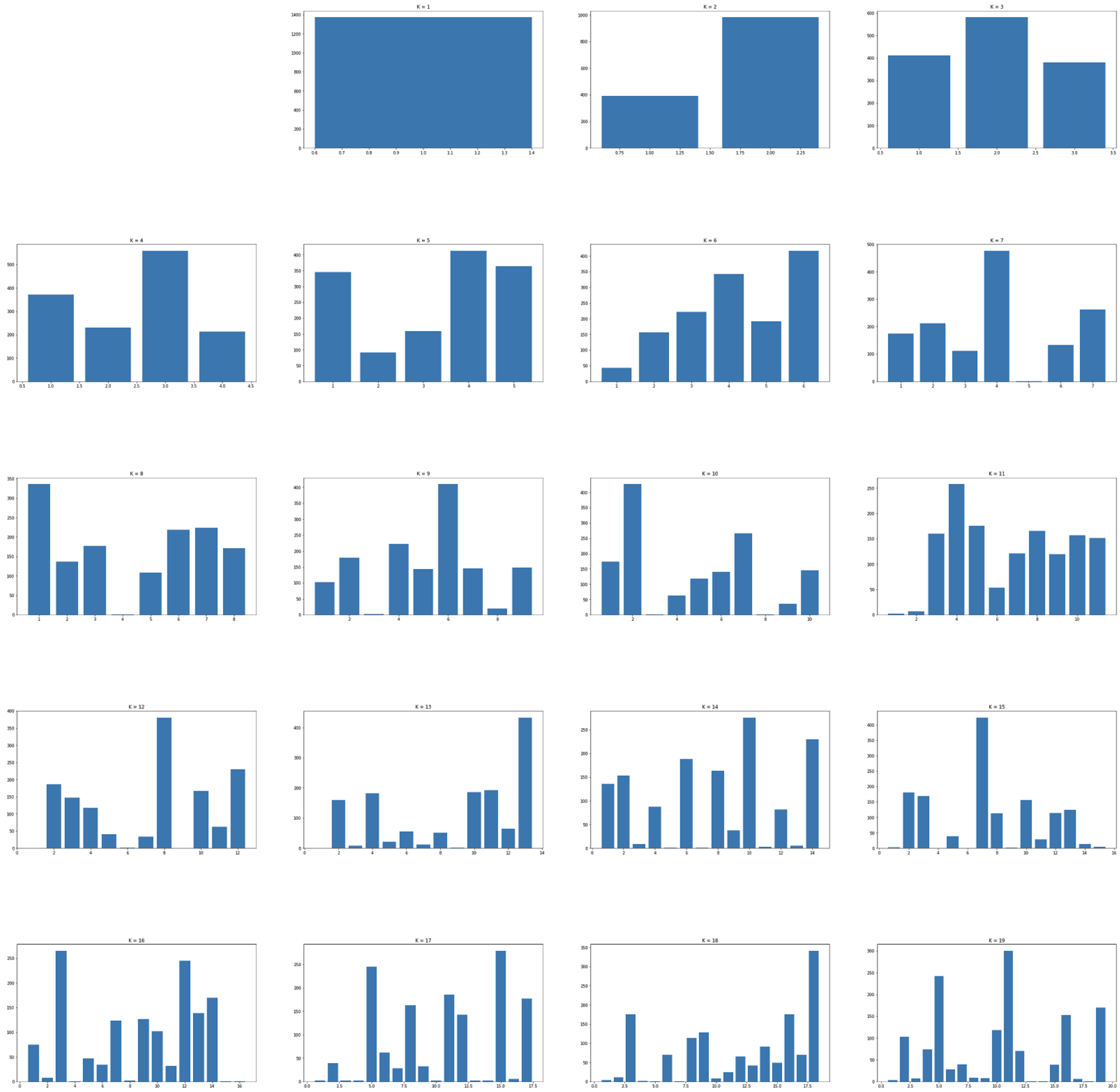
```
k_value_range = range(1, 20)
```

```
cluster_distributions, inertia_array = testDifferentKValue(k_value_range, science2kdocword)
```

```
plotDistributions(k_value_range, cluster_distributions)
```

To select the suitable k value, we evaluate the clustering performance by examine the number of data in each cluster. If for a certain K value, there are some clusters which contains only a few or even no data, we can say this K value is too big and we should try a smaller one.

(a - 2) Select a value of k



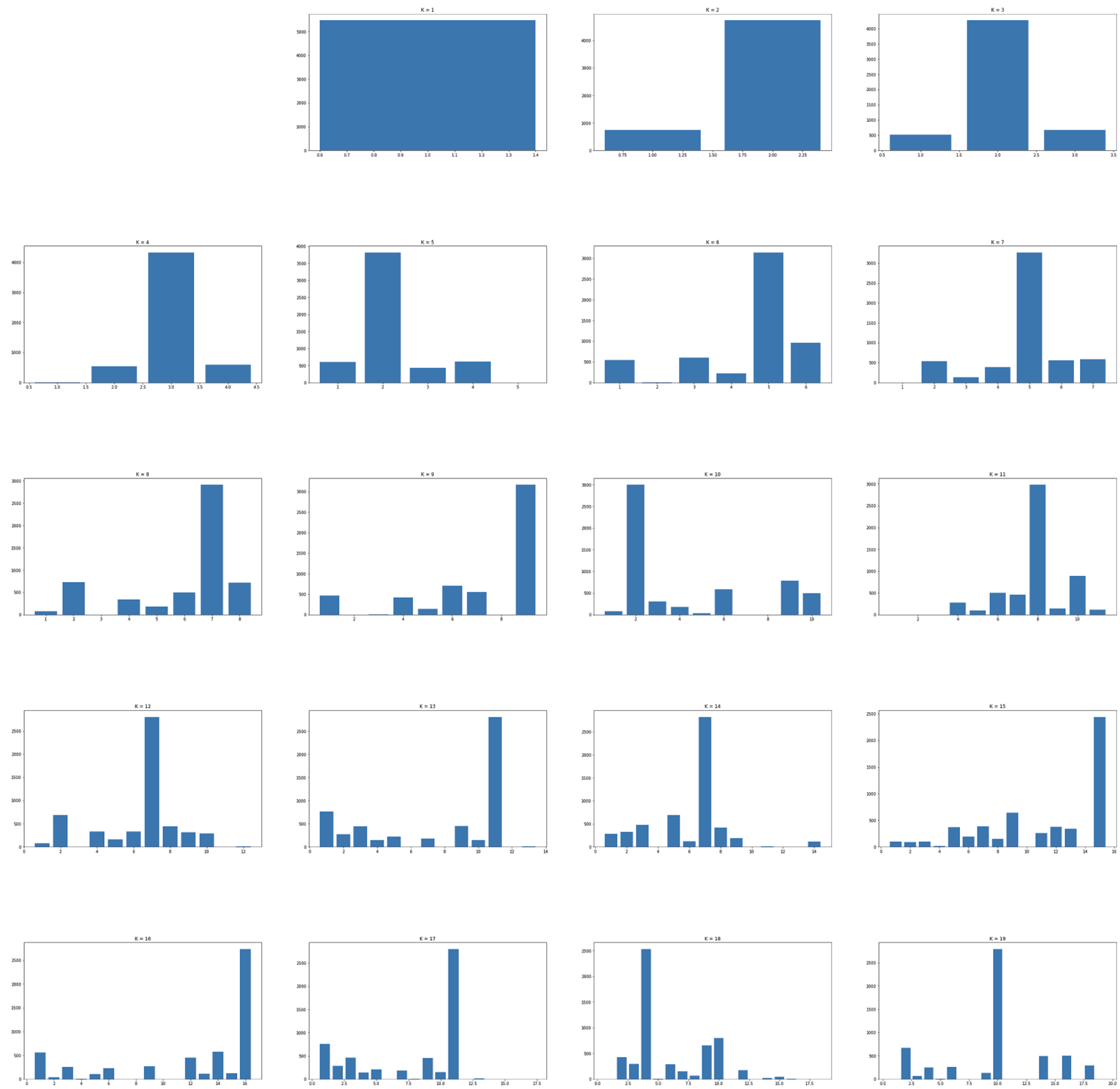
By taking a look of the distribution of data in each cluster when K varies from 1 to 20, we can find out that the distribution is ideal when **k = 5**. Each cluster have quite a lot data inside, and the distribution also makes a lot of sense.

(a - 3) For that k value, report the top 10 documents of each cluster

```
- - - - - - - - - - - - - - - -
Cluster: 1
[Top 10 closet titles]
"Distinct Classes of Yeast Promoters Revealed by Differential TAF Recruitment"
"Efficient Initiation of HCV RNA Replication in Cell Culture"
"T Cell-Independent Rescue of B Lymphocytes from Peripheral Immune Tolerance"
"Reduced Food Intake and Body Weight in Mice Treated with Fatty Acid Synthase Inhibitors"
"Patterning of the Zebrafish Retina by a Wave of Sonic Hedgehog Activity"
"Coupling of Stress in the ER to Activation of JNK Protein Kinases by Transmembrane Protein Kinase IRE1"
"Disruption of Signaling by Yersinia Effector YopJ, a Ubiquitin-like Protein Protease"
"An Anti-Apoptotic Role for the p53 Family Member, p73, during Developmental Neuron Death"
"Identification of Synergistic Signals Initiating Inner Ear Development"
"Molecular Linkage Underlying Microtubule Orientation toward Cortical Sites in Yeast"
- - - - - - - - - - - - - - - -
Cluster: 2
[Top 10 closet titles]
"Africa Boosts AIDS Vaccine R&D"
"A Renewed Assault on an Old and Deadly Foe"
"Clinton's Science Legacy: Ending on a High Note"
"Stephen Straus's Impossible Job"
"South Africa's New Enemy"
"Global AIDS Epidemic: Time to Turn the Tide"
"The Boom in Biosafety Labs"
"A Deluge of Patents Creates Legal Hassles for Research"
"Working in the Hot Zone: Galveston's Microbe Hunters"
"Against All Odds, Victories from the Front Lines"
- - - - - - - - - - - - - - - -
Cluster: 3
[Top 10 closet titles]
"Reopening the Darkest Chapter in German Science"
"Algorithmic Gladiators Vie for Digital Glory"
"Information Technology Takes a Different Tack"
"National Academy of Sciences Elects New Members"
"Heretical Idea Faces Its Sternest Test"
"Archaeology in the Holy Land"
"Divining Diet and Disease from DNA"
"Baedeker's Guide, or Just Plain 'Trouble'?"
"Vaccine Studies Stymied by Shortage of Animals"
"Science Survives in Breakthrough States"
- - - - - - - - - - - - - - - -
Cluster: 4
[Top 10 closet titles]
"Corrections and Clarifications: Charon's First Detailed Spectra Hold Many Surprises"
"Corrections and Clarifications: Unearthing Monuments of the Yarmukians"
"Corrections and Clarifications: A Short Fe-Fe Distance in Peroxodiferric Ferritin: Control of Fe Substrate ver
sus Cofactor Decay?"
"Corrections and Clarifications: One Hundred Years of Quantum Physics"
"Corrections and Clarifications: Biotech Research Proves a Draw in Canada"
"Corrections and Clarifications: Uninterrupted MCM2-7 Function Required for DNA Replication Fork Progression"
"Corrections and Clarifications: A Nuclear Solution to Climatic Change?"
"Movement Patterns in Spoken Language"
"Corrections and Clarifications: Identification of a Mating Type-like Locus in the Asexual Pathogenic Yeast Can
dida albicans"
"Corrections and Clarifications: Close Encounters: Details Veto Depth from Shadows"
- - - - - - - - - - - - - - - -
Cluster: 5
[Top 10 closet titles]
"A Stable Bicyclic Compound with Two Si=Si Double Bonds"
"Discovery of a Basaltic Asteroid in the Outer Main Belt"
"Greenland Ice Sheet: High-Elevation Balance and Peripheral Thinning"
"Viscosity Mechanisms in Accretion Disks"
"Anomalous Polarization Profiles in Sunspots: Possible Origin of Umbral Flashes"
"High-Gain Harmonic-Generation Free-Electron Laser"
"Detection of SO in Io's Exosphere"
"Discovery of a High-Energy Gamma-Ray-Emitting Persistent Microquasar"
"Isotopic Evidence for Variations in the Marine Calcium Cycle over the Cenozoic"
"Mass Balance of the Greenland Ice Sheet at High Elevations"
```

Above is the clustering result. We can find that the algorithm successfully clustering the titles which has a very similar pattern. For example, all the titles in Cluster 4 start with the sentence “Corrections and Clarifications”. For Cluster 2, all the titles have the format “XXX’s xxx”, while all the titles in Cluster 5 are in the format “XXX of xxx”. Since this algorithm can capture the format of the title, it may can be used to implement a title format checker which can check whether a given format is commonly used.

(b - 1/2) Cluster the terms using k-means and various values of k. Select a K value.



Similarly, by taking a look of the distribution of data in each cluster when K varies from 1 to 20, we can find out that the distribution is ideal when **k = 6**. Each cluster have some data inside, and the distribution also makes a lot of sense.

(b - 3) For that k value, report the top 10 terms of each cluster

```
- - - - - - - - - - - - - - - - -
Cluster: 1
[Top 10 closet vocabs]
approximation
angular
finite
coherent
nonlinear
approximate
regime
periodic
calculation
energies
- - - - - - - - - - - - - - - - -
Cluster: 2
[Top 10 closet vocabs]
protein
expression
gene
proteins
genes
cell
expressed
cells
dna
- - - - - - - - - - - - - - - - -
Cluster: 3
[Top 10 closet vocabs]
recalls
clinton
geneticist
security
prize
fight
finished
spending
campaign
rights
```

```
- - - - - - - - - - - - - - - - -
Cluster: 4
[Top 10 closet vocabs]
biochem
terminus
cooh
nh2
inhibitor
cdna
affinity
incubated
blot
specificity
- - - - - - - - - - - - - - - - -
Cluster: 5
[Top 10 closet vocabs]
lcts
aptamers
trxr
neas
dnag
proteorhodopsin
nompc
doxy
lg268
kcv
- - - - - - - - - - - - - - - - -
Cluster: 6
[Top 10 closet vocabs]
org
sciencemag
vol
thymocytes
endothelial
myeloid
caspase
agonists
immunoreactive
cd3
```

Above is the clustering result. We can find that the algorithm successfully clustering the terms which belongs to the same field. For example, all the terms in Cluster 1 are related to Math or Physics. While in cluster 2, most words are related to Biology, and the terms in Cluster 3 are more related to politics. This algorithm can be applied to select the keywords from a paper efficiently.

Clustering terms was based on the fact that the words which have similar meaning are more likely to appear in the same paper. While clustering titles focus more on comparing the pattern or format between titles.

Question 3 - EM Algorithm and Implementation

(a) 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.

K-Means algorithm is the 'hard assignment' version of the GMM algorithm where we only assign a data to a single class each time. In this case, we can have the following objective functions for E-step and M-step for K-Means:

Goal: find a partition $S = \{S_k\}_{k=1}^{K}$ so that it minimizes the objective function

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \parallel \mathbf{x_n} - \boldsymbol{\mu_k} \parallel^2$$

where $r_{nk} = 1$ if $\mathbf{x_n}$ is assigned to cluster S_k, and $r_{nj} = 0$ for $j \neq k$.

Expectation: J is linear function of r_{nk}

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \parallel \mathbf{x_n} - \boldsymbol{\mu_j} \parallel^2 \\ 0 & \text{otherwise} \end{cases}$$

Maximization: setting the derivative of J with respect to $\boldsymbol{\mu_k}$ to zero, gives:

$$\boldsymbol{\mu_k} = \frac{\Sigma_n r_{nk} \mathbf{x_n}}{\Sigma_n r_{nk}}$$

(Ref: https://www.slideshare.net/phvu/kmeans-em-and-mixture-models)

(b) Download the Old Faithful Geyser Dataset. Parse and plot all data points on a 2-D plane.

Load and Parse Data

```
raw_data = readFile('./data/OFG-data.txt')
```

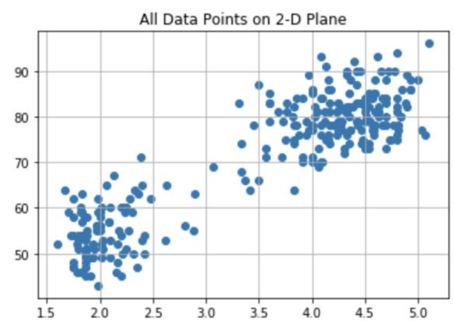
```
eruption_time = []
waiting_time = []
features = []

features = [[0 for x in range(2)] for y in range(len(raw_data))]

for data in raw_data:
    split = data.split(',')
    eruption_time.append(float(split[0]))
    waiting_time.append(float(split[1]))

for i in range(len(raw_data)):
    features[i][0] = eruption_time[i]
    features[i][1] = waiting_time[i]
```

```
featuresMat = np.array(features, dtype = 'float')
```

Plot Data Points

```
plt.grid()
plt.scatter(eruption_time, waiting_time)
plt.title('All Data Points on 2-D Plane')
```

```
<matplotlib.text.Text at 0x12cccdb00>
```



(c - 1) Implement a bimodal GMM model to fit all data point using EM algorithm. Explain the reasoning behind your termination criteria.

[GMM Model Class]

```
class GMM_bimodal:

    def __init__(self, method):
        self.method = method
        self.max_iter = 100
        self.threshold = 10**(-5)
```

[Random Initialization]

```
def randomInitialization(self, data):
    # randomly generate mean array
    cluster_centers_indexs = random.sample(range(len(data)), 2)
    cluster1_mean = data[cluster_centers_indexs[0]]
    cluster2_mean = data[cluster_centers_indexs[1]]

    # randomly generate covariance matrix
    cluster1_seed = random.uniform(0,100)
    cluster2_seed = random.uniform(0,100)

    cluster1_cov = [[cluster1_seed, 0], [0, cluster1_seed]]
    cluster2_cov = [[cluster2_seed, 0], [0, cluster2_seed]]

    # randomly generate lambda value
    lambdaValue = np.random.rand(1)[0]

    return cluster1_mean, cluster2_mean, cluster1_cov, cluster2_cov, lambdaValue
```

[E Step]

```
# E-step: compute responsibilities
def computeResponsibility(self, lambdaValue, cluster1_prob, cluster2_prob):
    numerator = lambdaValue * cluster2_prob
    denominator = lambdaValue * cluster2_prob + (1 - lambdaValue) * cluster1_prob
    resp = numerator / denominator
    return resp
```

[Termination Condition]

```
def isConverged(self, old_parameters, new_parameters):
    if np.absolute(LA.norm(new_parameters[0] - old_parameters[0])) < self.threshold
    and np.absolute(LA.norm(new_parameters[1] - old_parameters[1])) < self.threshold:
        return True
    return False
```

We will terminate the iteration when reach max iteration count (100) or we find the changes of the centers of clusters are smaller than a tolerance value (10^-5)

(c - 2) Provide the plot of trajectories of two mean vectors in 2 dimensions. Run your program for 50 times with different initial parameter guesses. Show the distribution of the total number of iterations needed for algorithm to converge.

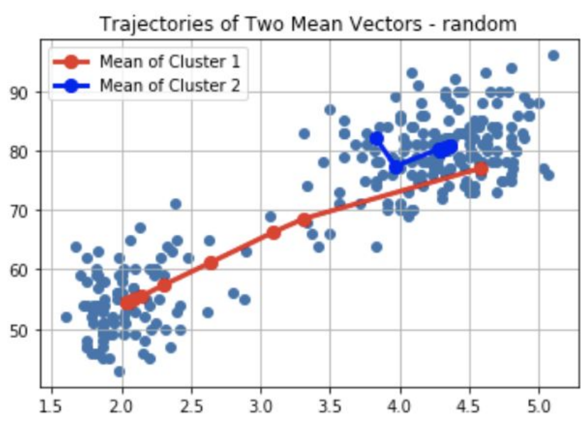
[Testing Functions]

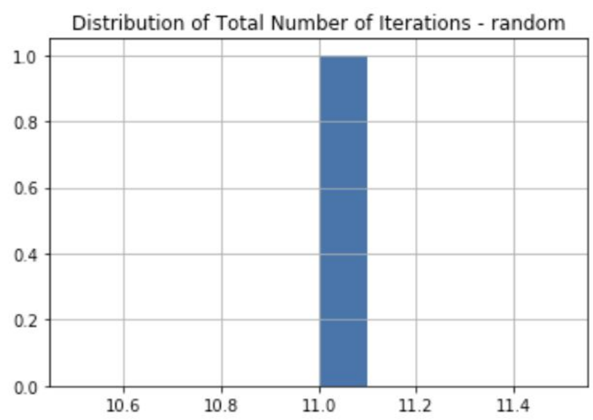
Testing Functions

```
def testDistributionOfIteration(method):
    distribution = []
    for i in range(50):
        gmm = GMM_bimodal(method)
        trajectories, iteration_count = gmm.EM(featuresMat)
        distribution.append(iteration_count)
    plt.hist(distribution)
    plt.title('Distribution of Total Number of Iterations - ' + method)
```

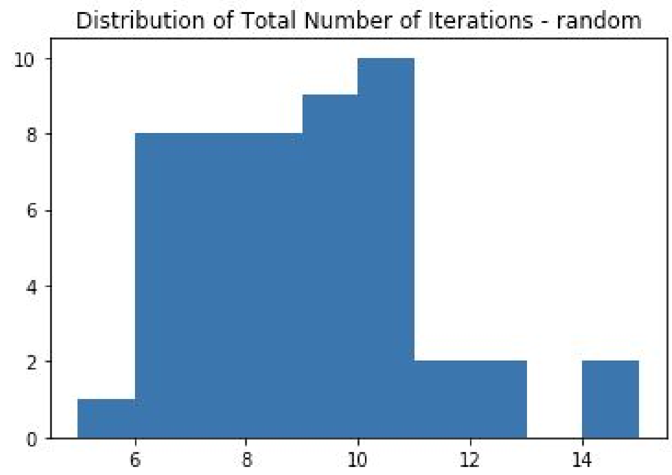
```
def testTrajectories(method):
    gmm = GMM_bimodal(method)
    trajectories, iteration_count = gmm.EM(featuresMat)
    trajectoriesArray = np.array(trajectories)

    plt.figure(1)
    plt.grid()
    plt.scatter(eruption_time, waiting_time)
    plt.plot(trajectoriesArray[:,0,0], trajectoriesArray[:,0,1],'o-', color="r", markersize=8, linewidth=3, label="
    plt.plot(trajectoriesArray[:,1,0], trajectoriesArray[:,1,1],'o-', color="b", markersize=8, linewidth=3, label="
    plt.legend(loc="best")
    plt.title('Trajectories of Two Mean Vectors - ' + method)


    plt.figure(2)
    plt.grid()
    plt.hist(iteration_count)
    plt.title('Distribution of Total Number of Iterations - ' + method)
```

[Single Trail]





[Run 50 times]

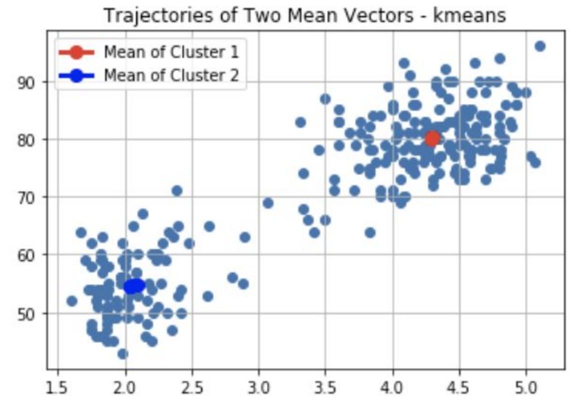


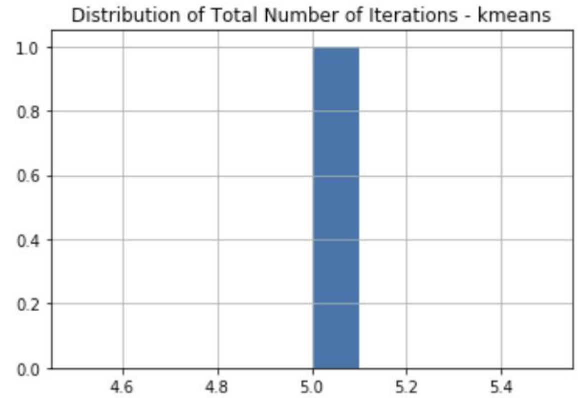
(d) Repeat the task in (c) but with the initial guess of the parameters generated from doing a k-mean clustering.

[K-Mean Initialization]

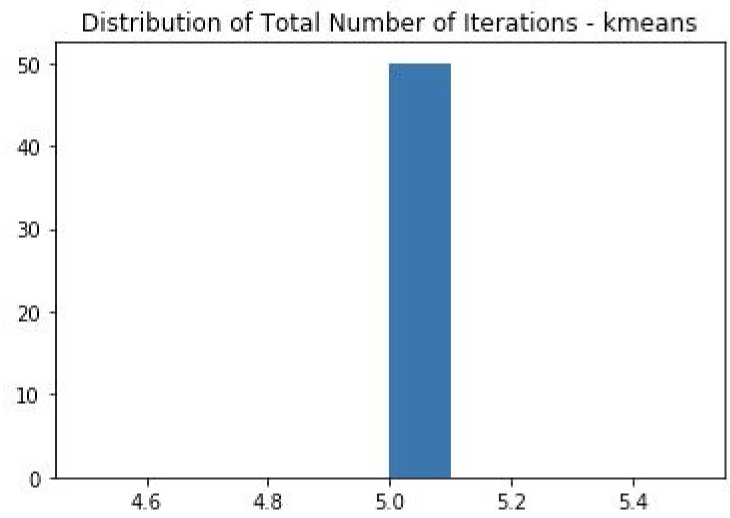
```
def kmeansInitialization(self, data):
    # kmeans clustering
    k = 2
    kmeans = KMeans(n_clusters=k, random_state=0).fit(data)
    cluster_centers = kmeans.cluster_centers_
    labels = kmeans.labels_

    cluster1_data = []
    cluster2_data = []
    # get cluster data
    for index in range(len(labels)):
        if labels[index] == 0:
            cluster1_data.append(data[index])
        else:
            cluster2_data.append(data[index])

    # assign init guess based on clustering result
    cluster1_mean = cluster_centers[0]
    cluster2_mean = cluster_centers[1]

    cluster1_sigmas = np.std(cluster1_data, axis = 0)
    cluster2_sigmas = np.std(cluster2_data, axis = 0)

    cluster1_cov = [[cluster1_sigmas[0]**2, 0], [0, cluster1_sigmas[1]**2]]
    cluster2_cov = [[cluster2_sigmas[0]**2, 0], [0, cluster2_sigmas[1]**2]]

    lambdaValue = len(cluster1_data)/len(data)

    return cluster1_mean, cluster2_mean, cluster1_cov, cluster2_cov, lambdaValue
```

[Single Trail]





[Run 50 times]



By comparing the distribution of total number of iterations required by Random and K-Means Initialization, we can find the performance of K-Means Initialization is much more better than random one. Less number of iterations are required and the cluster centers did not change a lot.

PART B - WRITTEN EXERCISES

Question 1 & 2:

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.

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 $\mathbf{E}_k \mathbf{D}_k$.

14.2

a) The log-likelihood function for $\{x_i\}_{i=1}^{N}$ is given by

$$\ell(\theta, Z) = \log\left(\prod_{i=1}^{N} g(x_i)\right) = \log\left(\prod_{i=1}^{N} \left(\sum_{k=1}^{K} \pi_k g_k(x_i)\right)\right) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k g_k(x_i)\right)$$

where $\theta = (\sigma^2, \theta_1, \ldots \theta_K) = (\sigma^2, \pi_1, \mu_1, \ldots \pi_K, \mu_K)$

b) We generalize the ideas in H7FU9. Introduce the random vector $\Delta = (\Delta_1 \ldots \Delta_K)$ satisfying $\Delta_k \in \{0,1\}$, $\sum_{k=1}^{K} \Delta_k = 1$ and $Pr(\Delta_k = 1) = \pi_k$

$$\gamma_{kn}(\theta) := Pr(\Delta_k = 1 \mid \theta, Z = x_n) = \frac{\pi_k g_k(x_n)}{\sum_{j=1}^{K} \pi_j g_j(x_n)} \qquad (1)$$

In (?) we calculate the derivatives $\frac{d\ell}{d\mu_k}$, $\frac{d\ell}{d\sigma^2}$ and $\frac{d\ell}{d\pi_k}$ we determine their zeros and find the extreme points

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk}}, \quad \sigma^2 = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{k=1}^{K} \sum_{n=1}^{N} \gamma_{nk}}$$

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma_{nk}}{N}$$

In general, the EM algorithm is given by:

1. Take initial guesses for the parameters $\sigma^2, \hat{\mu}_i, \hat{\pi}_i$ for $i = 1 \ldots K$
2. Expectation step: Compute the responsibilities
$$\hat{\gamma}_{nk} = \frac{\pi_k g_k(x_n)}{\sum_{j=1}^{K} \hat{\pi}_j g_j(x_n)} \; ; \quad i = 1 \ldots N, \quad k = 1 \ldots K$$
3. Maximization Step, compute the weighted means and variances
$$\hat{\mu}_k = \frac{\sum_{n=1}^{N} \hat{\gamma}_{nk} x_n}{\sum_{n=1}^{N} \hat{\gamma}_{nk}}, \quad \hat{\sigma}^2 = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \hat{\gamma}_{nk} (x_n - \hat{\mu}_k)(x_n - \hat{\mu}_k)^T}{\sum_{k=1}^{K} \sum_{n=1}^{N} \hat{\gamma}_{nk}}$$
$$\hat{\pi}_k = \frac{\sum_{n=1}^{N} \hat{\gamma}_{nk}}{N}$$
4. Iterate 2 & 3 until convergence

c) For each n choose j such that $(x_n-\mu_j)^T(x_n-\mu_j) \le (x_n-\mu_k)^T(x_n-\mu_k)$ for all k and provided $\pi_k \neq 0$. Note from (1) that for $k \neq j$ as $\sigma \to 0$ and $\gamma_{nj} \to 1$. Hence we have:

$$\gamma_{nk} \to r_{nk} := \begin{cases} 1 \text{ if } k = \arg\min_j (x_n-\mu_j)^T(x_n-\mu_j), \\ 0 \text{ otherwise.} \end{cases}$$

which assigns each data point to the cluster having the closest mean.

$\boxed{14.11}$

Minimize $S_C(z_1, z_2 \ldots z_N) = \sum_{i,j} (s_{ij} - \langle z_i, z_j \rangle)^2$, where

$s_{ij} \doteq \langle x_i, x_j \rangle$. To simplify we subtracted the means of x and z

$$\|x_i - x_j\|^2 = -2x_i^T x_j,$$

So we can use S to estimate x. Since S is symmetric, which indicates that its eigenvectors are orthogonal, we eigendecompose S as:

$$S = E_k D_k D_k^T E_k^T$$

indicates that the estimations of x.

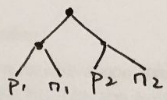
Question 3 - Decision Trees

(a)

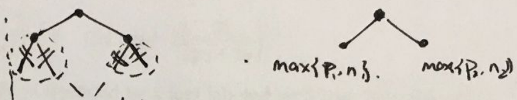
WRITTEN EXERCISES

3. Decision Trees.

(a) Consider a tree like this



After doing truncate



Replaced by a leaf labeled with more frequent class. $\Rightarrow$

• In this case, consider the p_1, n_1 branch. All the ~~[illegible]~~ data will be labled as the majority items in this split, which same as means now have (p_1+n_1) belongs to majority class.

Befor truncate, their should be only $(p_1+n_1)\cdot \max\{\frac{p_1}{p_1+n_1}, \frac{n_1}{p_1+n_1}\}$ belongs to majority class.

In this case, we can calculate the error (training mistakes) as

$$(p_1+n_1) - (p_1+n_1)\cdot \max\left\{\frac{p_1}{p_1+n_1}, \frac{n_1}{p_1+n_1}\right\}.$$

$$= (p_1+n_1)\cdot \min\left(\frac{p_1}{p_1+n_1}, \frac{n_1}{p_1+n_1}\right)$$

$$= (p_1+n_1)\cdot I\left(\frac{p_1}{p_1+n_1}\right)$$

Similarily for p_2 and n_2.

$\therefore$ The overall traning mistakes : $(p_1+n_1)\cdot I\left(\frac{p_1}{p_1+n_1}\right) + (p_2+n_2)\cdot I\left(\frac{p_2}{p_2+n_2}\right)$

(b, c)

(b) ① Gini Index

$P(+) = \frac{3}{10}$ $\quad P(-) = \frac{7}{10}$ $\quad G(\text{start}) = 1 - \left(\frac{3}{10}^2 + \frac{7}{10}^2\right) = \frac{42}{100} = 0.42$

[subset a1]

a_1	y
0	+
0	+
0	−
0	−

a_1	y
1	+
1	−
1	−
1	−
1	−
1	−

$G_{a_1=0} = 1 - \left(\frac{1}{2}^2 + \frac{1}{2}^2\right) = \frac{1}{2}$

$G_{a_1=1} = 1 - \left(\frac{1}{6}^2 + \frac{5}{6}^2\right) = \frac{10}{36} = \frac{5}{18}$

$G(\text{split by } a_1) = \frac{4}{10} \cdot \frac{1}{2} + \frac{6}{10} \cdot \frac{5}{18} = 0.3667$

$\Delta G = G(\text{start}) - G_{a_1} = 0.42 - 0.3667 = 0.0533$

[subset of a2]

a_2	y
0	+
0	−
0	−
0	−

a_2	y
1	+
1	+
1	−
1	−
1	−
1	−

$G_{a_2=0} = 1 - \left(\frac{1}{4}^2 + \frac{3}{4}^2\right) = \frac{6}{16} = \frac{3}{8}$

$G_{a_2=1} = 1 - \left(\frac{2}{6}^2 + \frac{4}{6}^2\right) = \frac{16}{36} = \frac{4}{9}$

$G(\text{split by } a_2) = \frac{4}{10} \cdot \frac{3}{8} + \frac{6}{10} \cdot \frac{4}{9} = 0.4167$

$\Delta G = 0.42 - 0.4167 = 0.0033.$

[subset of a3]

a_3	y
0	+
0	+
0	+
0	−
0	−
0	−
0	−

a_3	y
1	−
1	−
1	−

$G_{a_3=0} = 1 - \left(\frac{3}{7}^2 + \frac{4}{7}^2\right) = \frac{24}{49}$

$G_{a_3=1} = 0.$

$G(\text{split by } a_3) = \frac{7}{10} \cdot \frac{24}{49} + 0 = \frac{12}{35} = 0.3429$

$\Delta G = 0.42 - 0.3429 = 0.0771.$

Since when split by a_3, we have the maximum ΔG index value

so we will choose a_3 to split at the root

② min-error

choose a_1 $\quad 6 \cdot \frac{1}{6} + 4 \cdot \frac{1}{2} = 3$

Chose a_2 $\quad 6 \cdot \frac{2}{6} + 4 \cdot \frac{1}{4} = 3$

Choose a_3 $\quad 7 \cdot \frac{3}{7} + 3 \cdot 0 = 3$

We can choose either a_1, a_2 or a_3 at the root

Since they have the same error impurity

(C)

Before spliting, $P = P_1 + P_2$ $\quad N = n_1 + n_2$ $\quad (P+N) \cdot I\left(\frac{P}{P+N}\right)$

After spliting, $(P_1+n_1) \cdot I\left(\frac{P_1}{P_1+n_1}\right) + (P_2+n_2) \cdot I\left(\frac{P_2}{P_2+n_2}\right)$

① if $P_1 < n_1$, $P_2 < n_2$ $\quad (P_1+P_2+n_1+n_2) \cdot I\left(\frac{P_1+P_2}{P_1+P_2+n_1+n_2}\right) > (P_1+n_1)\left(\frac{P_1}{P_1+n_1}\right) + (P_2+n_2) \cdot \frac{P_2}{P_2+n_2}$

② if $P_1 > n_1$, $P_2 > n_2$ $\quad$ LEFT > Right

③ if $P_1 > n_1$, $P_2 < n_2$ or $P_1 < n_1$, $P_2 > n_2$ $\quad$ LEFT < Right

∴ The General condition is $P_1 > n_1$ && $P_2 < n_2$ or $P_1 < n_1$ && $P_2 > n_2$

(d)

(d)

As we can observe in (c) and (b)

the performance of min-error can ~~dy only be good~~ be very bad when ~~when the~~

① All the classes ~~have~~ are equally likely.

Also, this method can only produce a single Tree

Besides, the number of classes also strongly affects the degree of pruning,

In this case, min-error is only suitable for the classes who are distinct to each other and have very clear split method

PART C - APPENDIX

CODE

[Github]()

REFERENCES

[A Solution Manual and Notes for: The Elements of Statistical Learning]()

https://www.slideshare.net/phvu/kmeans-em-and-mixture-models

- END -