CM146, Fall 2020
Problem Set 4: Kernel, SVM, K-Means, GMM
Due Dec 10, 2020 at 11:59 pm

1 Kernels [30 pts]

One way to construct complex kernels is to build from simple ones. In the following, we will prove a list of properties of kernels (Rule 1 – Rule 5) and then use these rules to show the Gaussian kernel $k(x,z) = \exp(\frac{-\|x-z\|^2}{\sigma^2})$ is a valid kernel.

(a) **(5 pts)** Rule 1: Suppose we have $x \in \mathbb{X}, z \in \mathbb{X}$, $g : \mathbb{X} \to \mathbb{R}$. Prove that $k(x,z) = g(x) \times g(z)$ is a valid kernel by constructing a feature map $\Phi(\cdot)$ and show that $k(x,z) = \Phi(x)^T\Phi(z)$.
Solution: $\Phi(x) = g(x)$ and $\Phi(x)^T\Phi(z) = g(x) \times g(z) = k(x,z)$

(b) **(5 pts)** Rule 2: Suppose we have a valid kernel $k_1(x,z) = \Phi_1(x)^T\Phi_1(z)$. Prove that $k(x,z) = \alpha \cdot k_1(x,z) \quad \forall \alpha \geq 0$ is also a valid kernel by constructing a new feature map $\Phi(\cdot)$ using $\Phi_1(\cdot)$ and show that $k(x,z) = \Phi(x)^T\Phi(z)$.
Solution: $\Phi(x) = \sqrt{\alpha}\Phi_1(x)$ and $\Phi(x)^T\Phi(z) = (\sqrt{\alpha}\Phi_1(x))^T(\sqrt{\alpha}\Phi_1(z)) = \sqrt{\alpha}\sqrt{\alpha}\Phi_1(x)^T\Phi_1(z) = \alpha\Phi_1(x)^T\Phi_1(z) = \alpha \cdot k_1(x,z) = k(x,z)$

(c) **(5 pts)** Rule 3: Suppose we have two valid kernels $k_1(x,z) = \Phi_1(x)^T\Phi_1(z)$ and $k_2(x,z) = \Phi_2(x)^T\Phi_2(z)$. Prove that $k(x,z) = k_1(x,z) + k_2(x,z)$ is also a valid kernel by constructing a new feature map $\Phi(\cdot)$ using $\Phi_1(\cdot)$ and $\Phi_2(\cdot)$ and show that $k(x,z) = \Phi(x)^T\Phi(z)$.
Solution: let $\Phi(x)$ be a vector of concatenated form of $\Phi_1(x)$ and $\Phi_2(x)$
such that $\Phi(x) = \begin{pmatrix}\Phi_1(x)\\ \Phi_2(x)\end{pmatrix}$ Hence $\Phi(x)^T\Phi(z) = \begin{pmatrix}\Phi_1(x)\\ \Phi_2(x)\end{pmatrix}^T \cdot \begin{pmatrix}\Phi_1(z)\\ \Phi_2(z)\end{pmatrix} = \Phi_1(x)^T\Phi_1(z) + \Phi_2(x)^T\Phi_2(z) = k_1(x,z) + k_2(x,z) = k(x,z)$

(d) **(5 pts)** Rule 4: Suppose we have two valid kernels $k_1(x,z) = \Phi_1(x)^T\Phi_1(z)$ and $k_2(x,z) = \Phi_2(x)^T\Phi_2(z)$. Prove that $k(x,z) = k_1(x,z) \times k_2(x,z)$ is a valid kernel by constructing a new feature map $\Phi(\cdot)$ using $\Phi_1(\cdot)$ and $\Phi_2(\cdot)$ and show that $k(x,z) = \Phi(x)^T\Phi(z)$.
Solution: $\Phi(x) = \begin{pmatrix}\Phi_1(x_1)\Phi_2(x_1)\\ \Phi_1(x_1)\Phi_2(x_2)\\ \dots\\ \Phi_1(x_i)\Phi_2(x_j)\end{pmatrix}$, which includes all possible combinations
$\Phi(x)^T\Phi(z) = \sum_i\sum_j \Phi_1(x_i)\Phi_2(x_j)\Phi_1(z_i)\Phi_2(z_j) = \sum_i\sum_j \Phi_1(x_i)\Phi_1(z_i)\Phi_2(x_j)\Phi_2(z_j)$
$= \Phi_1(x)^T\Phi_1(z)\Phi_2(x)^T\Phi_2(z) = k_1(x,z) \times k_2(x,z) = k_1(x,z)$

(e) **(5 pts)** Rule 5: Suppose we have a valid kernel $k_1(x,z) = \Phi_1(x)^T\Phi_1(z)$. Prove that $\exp(k_1(x,z))$ is also a valid kernel by applying Rules 1-4 in problem 1(a)-1(d).
Hint:

$$\exp(k_1(x,z)) = \lim_{i\to\infty} 1 + k_1(x,z) + \dots + \frac{k_1^i(x,z)}{i!}$$
$$= 1 + \sum_{i=1}^{i=\infty} \frac{k_1^i(x,z)}{i!},$$

where $k_1^i(x,z)$ is $k_1(x,z)$ to the power of i.
Solution: The constant 1 is obviously a kernel and from Rule 2, we know that every single term of $\lim_{i\to\infty} 1 + k_1(x,z) + ... + \frac{k_1^i(x,z)}{i!}$ is a kernel. From Rule 3, we know that the sum of valid kernels is also a kernel. Hence $\exp(k_1(x,z))$ is also a valid kernel.

(f) **(5 pts)** Prove the Gaussian Kernel $k(x,z) = \exp(\frac{-\|x-z\|^2}{\sigma^2})$ is a valid kernel by applying Rules 1-5 in problem 1(a)-1(e).
Solution: $\exp(\frac{-\|x-z\|^2}{\sigma^2}) = \exp(\frac{-\|x\|^2-\|z\|^2+2x^Tz}{\sigma^2}) = \exp(\frac{-\|x\|^2}{\sigma^2})\exp(\frac{-\|z\|^2}{\sigma^2})\exp(\frac{2x^Tz}{\sigma^2})$
$= g(x)g(z)\exp(\frac{2x^Tz}{\sigma^2})$. From Rule 1, $g(x)g(z)$ is a valid kernel. Obviously, x^Tz is a valid kernel with $\Phi(x) = x$. Hence, from Rule 2, $\frac{2x^Tz}{\sigma^2}$ is also a valid kernel. From Rule 5, we can further get that $\exp(\frac{2x^Tz}{\sigma^2})$ is a valid kernel. Finally, from Rule 4, we can conclude that $g(x)g(z)\exp(\frac{2x^Tz}{\sigma^2}) = \exp(\frac{-\|x-z\|^2}{\sigma^2}) = k(x,z)$ is a valid kernel.

2 SVM [25 pts]

Suppose we have the following six training examples. x_1, x_2, x_3 are positive instances and x_4, x_5, x_6 are negative instances. Note: we expect you to use a simple geometric argument to narrow down the search and derive the same solution SVM optimization would result in for the following two questions. You don't need to write a program to solve this problem.

$Example$	$feature_1$	$feature_2$	y
x_1	1	7	1
x_2	3	2	1
x_3	4	10	1
x_4	−1	−7	−1
x_5	1	−1	−1
x_6	3	−6	−1

(a) **(5 pts)** Suppose we are looking for a hard-SVM decision boundary $\boldsymbol{w}^T\boldsymbol{x}_n + b = 0$ passing through origin (i.e., $b = 0$). In other words, we minimize $||\boldsymbol{w}||_2$ subject to $y_n\boldsymbol{w}^T\boldsymbol{x}_n \geq 1, n = 1, \ldots, N$. Identify the `support vectors` (data points that are actually used in the calculation of w and margin) in this training dataset.
Solution: x_2 and x_5

(b) **(5 pts)** Following part (a), what is $\boldsymbol{w}^* \in \mathbb{R}^2$ in this case and what is the margin: $\frac{1}{\|w^*\|_2}$?
Solution:

$$3w_1 + 2w_2 = 1$$
$$w_1 - w_2 = -1$$

$w_1 = -0.2$ and $w_2 = 0.8$, $\boldsymbol{w}^* = \begin{pmatrix} -0.2 \\ 0.8 \end{pmatrix}$ and $\frac{1}{\|w^*\|_2} = \frac{1}{\sqrt{0.2^2+0.8^2}} = \frac{5\sqrt{17}}{17} = 1.212678125$

(c) **(15 pts)** Suppose we now allow the offset parameter b to be non-zero. In other words, we minimize $||\boldsymbol{w}||_2$ subject to $y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1, n = 1, \ldots, N$. How would the classifier and the actual margin change in the previous question? What are $\boldsymbol{w}^*, b^*, \frac{1}{\|\boldsymbol{w}^*\|_2}$,? Compare your

solutions with problem 2(b).
Solution: support vectors remain the same (still x_2 and x_5)
by computing the gradient of the line crossing x_2 and x_5 we get $w_1 = \frac{2}{3}w_2$

$$\begin{aligned} 3w_1 + 2w_2 + b &= 1 \\ w_1 - w_2 + b &= -1 \\ w_1 &= \frac{2}{3}w_2 \end{aligned}$$

$w_1 = \frac{4}{13}$ $w_2 = \frac{6}{13}$ $b^* = \frac{-11}{13}$ $\boldsymbol{w}^* = \begin{pmatrix} \frac{4}{13} \\ \frac{6}{13} \end{pmatrix}$ $\frac{1}{\|w^*\|_2} = \frac{1}{\sqrt{\frac{4}{13}^2 + \frac{6}{13}^2}} = \frac{\sqrt{13}}{2} = 1.802775638$
Now we get a greater margin.$1.802775638 > 1.212678125$ The offset parameter b gives a better performance.

3 K-means [10 pts]

In this problem, we will first work through a numerical K-means example for a one-dimensional data and show that K-Means does not guarantee global optimal solution.

(a) **(5 pts)** Consider the case where $K = 3$ and we have 4 data points $x_1 = 1, x_2 = 2, x_3 = 5, x_4 = 7$. What is the optimal clustering for this data ? What is the corresponding value of the objective $\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\|x_n - \mu_k\|_2^2$? ($x_n$ denotes the n^{th} data point, μ_k denotes the cluster mean of the k^{th} cluster and γ_{nk} is a Boolean indicator variable and is set to 1 only if n^{th} data point belongs to k^{th} cluster.)
Solution: $\mu_1 = 1.5$ $\mu_2 = 5$ $\mu_3 = 7$
Hence x_1 and x_2 belong to μ_1, x_3 belongs to μ_2, x_4 belongs to μ_3.
$\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\|x_n - \mu_k\|_2^2 = 0.5^2 + 0.5^2 = 0.5$

(b) **(5 pts)** In K-Means, if we initialize the cluster centers as

$$c_1 = 1, c_2 = 2, c_3 = 6$$

what is the corresponding cluster assignment and the objective $\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\|x_n - \mu_k\|_2^2$ at the first iteration? Does k-Mean algorithm improve the clusters at the second iteration?
Solution: At first iteration, x_1 belongs to c_1, x_2 belongs to c_2, x_3 and x_4 belong to c_3, $\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\|x_n - \mu_k\|_2^2 = 1^2 + 1^2 = 2$. At second iteration, $c_1 = 1, c_2 = 2, c_3 = \frac{5+7}{2} = 6$, cluster centers remain the same and the objective sum also remains the same. No improvement at the second iteration.

4 Twitter analysis using SVMs, KMeans, GMM [35 pts]

In this project, you will be working with Twitter data. Specifically, we have supplied you with a number of tweets that are reviews/reactions to 4 different movies[1],

[1]Please note that these data were selected at random and thus the content of these tweets do not reflect the views of the course staff. :-)

e.g., *"@nickjfrost just saw The Boat That Rocked/Pirate Radio and I thought it was brilliant! You and the rest of the cast were fantastic! < 3"*.

The original data can be found here if you are interested in skimming through the tweets to get a sense of the data. We have preprocessed them for you and export them to a tweet_df.txt file

Specifically, we use a bag-of-words model to convert each tweet into a feature vector. A bag-of-words model treats a text file as a collection of words, disregarding word order. The first step in building a bag-of-words model involves building a "dictionary". A dictionary contains all of the unique words in the text file. For this project, we will be including punctuations in the dictionary too. For example, a text file containing *"John likes movies. Mary likes movies2!!"* will have a dictionary `{'John':0, 'Mary':1, 'likes':2, 'movies':3, 'movies2':4, '.':5, '!':6}`. Note that the `(key,value)` pairs are `(word, index)`, where the index keeps track of the number of unique words (size of the dictionary).

Given a dictionary containing d unique words, we can transform the n variable-length tweets into n feature vectors of length d by setting the i^{th} element of the j^{th} feature vector to 1 if the i^{th} dictionary word is in the j^{th} tweet, and 0 otherwise.

The preprocessed data contains 628 tweets on 4 different movies. Each line in the file contains exactly one tweet, so there are 628 lines in total. If a tweet praises or recommends a movie, it is classified as a positive review and labeled $+1$; otherwise it is classified as a negative review and labeled -1. The labels for positive or negative reviews as well as the labels for which movie is this tweet referring to are also included in the file.

You will learn to automatically classify such tweets as either positive or negative reviews. To do this, you will employ Support Vector Machines (SVMs), a popular choice for a large number of classification problems. You will also learn to automatically cluster such tweets in low dimensional space with Gaussian Mixture Model (GMM) and Kmeans.

Next, please download the preprocessed version of the tweets data tweet_df.txt and upload them to your google drive. For all the coding, please refer to the following colab notebook Fall2020-CS146-HW4.ipynb. Please save a local copy to your own drive first and only edit the TODO blocks in the notebook.

Documentation

- Support Vector Machine: link
- F1 score: link
- PCA: link
- KMeans: link
- Gaussian Micture Model: link
- Adjusted Rand Index: link

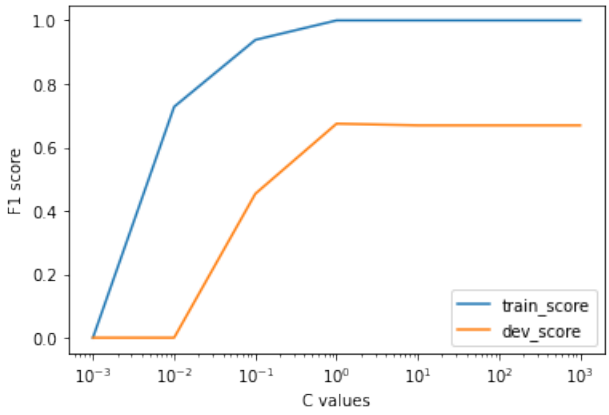
4.1 Hyper-parameter Selection for a Linear-Kernel SVM [15 pts]

We will learn a classifier to separate the training data into positive and negative tweets. For the classifier, we will use SVMs with the linear kernel. We will use the `sklearn.svm.SVC` class and explicitly set only two of the initialization parameters: `kernel` set to 'linear', and `C`. As usual, we will use `SVC.fit(X,y)` to train our SVM,and `SVC.predict(X)` to make predictions.

SVMs have hyperparameters that must be set by the user. For linear kernel SVM, we will select the hyper-parameter using a simple train set and a development set. In the notebook, the train, dev, test split is already done for you. Please do NOT change that split on your own as we will evaluate all answers under the split we provided in the Notebook. We will train several different models with different hyper-parameters using the train set and select the hyper-parameter that leads to the 'best' performance in the dev set.

(a) **(10 pts)** Please use **F1-Score** as the performance measure. Train the linear SVM with different values of `C`, `C` $= 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3$. Plot the train f1 score and the dev f1 score against different choices of `C`. Include this plot in your writeup, and provide a 1-2 sentence description of your observations. What is the effect of C? What is the best value of `C`? What is the dev set f1-score when using the best `C`? **Please also copy-paste your code as part of the solution for this problem.**

Solution:



F1 score for training data is always higher than the dev data and the score increases as the C value increases at the beginning of the graph then become flat when C is large enough. C works as the Regularization parameter that penalizes over-complex hypothesis.. The strength of the regularization is inversely proportional to C.From the graph, we can see that the best value of C is 1. The dev set f1-score when using the best C is 0.6745098039215687.

```
C_value=[0.001,0.01,0.1,1,10,100,1000]
f1_train=[]
f1_dev=[]
for i in C_value:
  clf = SVC(C=i, kernel='linear')
  clf.fit(X_train, y_train)
  y_pred_train = clf.predict(X_train)
```

```
  y_pred_dev = clf.predict(X_dev)
  f1_train.append(metrics.f1_score(y_train,y_pred_train))
  f1_dev.append(metrics.f1_score(y_dev,y_pred_dev))

plt.figure()
plt.semilogx(C_value, f1_train, label='train_score')
plt.semilogx(C_value, f1_dev, label='dev_score')
plt.xlabel('C values')
plt.ylabel('F1 score')
plt.legend()
plt.show()
```

(b) **(5 pts)** Retraining the model with the best `C` on the train and dev set together. Report the F1-score on the test set. **Please also copy-paste your code as part of the solution for this problem.**

Solution: F1 score:0.8735632183908046.

```
new_train_index= np.where((movies == 1) | (movies == 3)|(movies == 2))[0]
X_newtrain=X[new_train_index,]
y_newtrain=y[new_train_index,]
# print(X_newtrain.shape)
clf = SVC(C=1, kernel='linear')
clf.fit(X_newtrain,y_newtrain)
y_pred_test = clf.predict(X_test)
print(metrics.f1_score(y_test,y_pred_test))
```

4.2 Low Dimensional Embedding Space Clustering [20 pts]

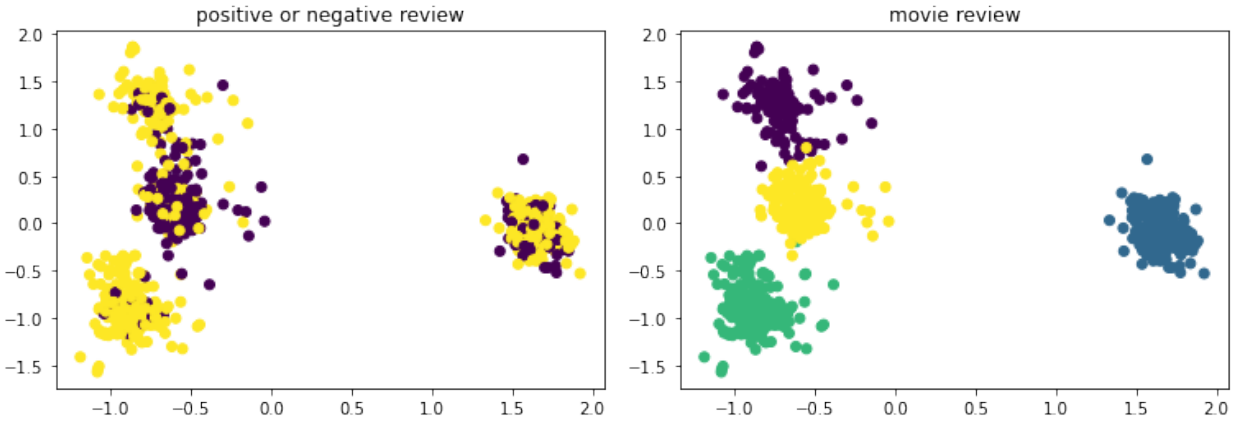
In this section, we will explore two unsupervised clustering algorithms: KMeans and GMM. The preprocessed twitter data lives in high (1811) dimensional space but it is hard for us to visualize more than three dimensional objects. Let's project the data into a low dimensional embedding space with a commonly used algorithm: Principal Component Analysis (PCA). We have already provided the code to do that. Please run it and from now on work with `X_embedding` variable which is 628 by 2 instead of 628 by 1811. If you are interested in the math behind PCA here are some good reading materials: A One-Stop Shop for Principal Component Analysis, Wikipedia page for PCA. However, for this project you are not required to know the math behind it. Intuitively, this algorithm is extracting the most useful linear combination of the features such that it reduces the amount of information loss occurred when projecting from high (1811) dimensions to low (2) dimensions.

We will also evaluate the purity of the clusters returned from any unsupervised algorithms against the ground truth labels from the original `tweet_df.txt`.

Here we will use `sklearn.metrics.adjusted_rand_score`. On a high level, this metric is measuring what fraction of the examples are labeled correctly but normalized so that if the cluster assignment is near random then the adjusted rand score will be close to 0. If the assignment is

near perfect, the adjusted rand score should be close to 1.

(a) **(5 pts)** Visualize the low dimensional data by calling function `plot_scatter`. First label/color the scatter plot by positive or negative reviews. Then label/color each tweet/dot by which movie is that review referring to. Do positive reviews and negative reviews form two nice clusters? Do 4 different movies form nice clusters? Include both plots in your writeup. **Please also copy-paste your code as part of the solution for this problem.**
Solution:



Positive reviews and negative reviews failed to form two nice clusters, while 4 different movies form nice clusters.

```
plt.figure()
plot_scatter(X_embedding, y, show = True, save_as = "PN.png", title =
↪  "positive or negative review")
plot_scatter(X_embedding, movies, show = True, save_as = "movie.png", title
↪  = "movie review")
```

(b) **(7.5 pts)** Use the `sklearn.cluster.KMeans` class on `X_embedding` with `n_clusters` set to 4, `init` set to 'random', `n_init` set to 1, and `random_state` set to 2. Visualize the low dimensional data by labeling/coloring each tweet with Kmeans results. Calculate the adjusted rand score

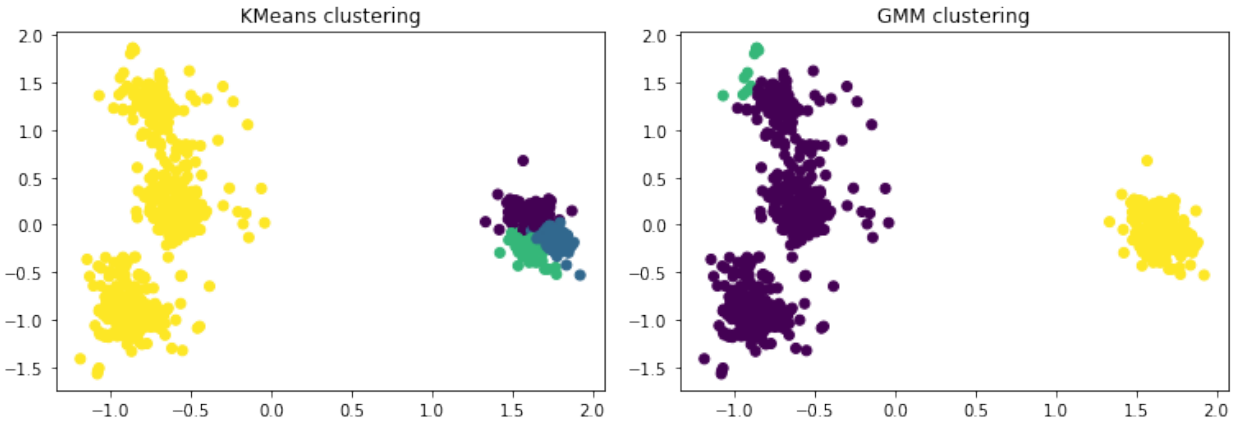
Use the `sklearn.mixture.GaussianMixture` class on `X_embedding` with `n_components` set to 4, `random_state` set to 0 `init_params` set to 'random'. Visualize the low dimensional data by labeling/coloring each tweet with GMM results. Calculate the adjusted rand score

Visually, do you think they perform poorly or great? What are the adjusted rand scores for both Kmeans labels and GMM labels? Why might that be the case? Include both plots, the performance results, and a 1-2 sentence description/justification of what you saw in your writeup. **Please also copy-paste your code as part of the solution for this problem.**

```
clt_KM = KMeans(n_clusters=4, init='random', n_init=1, random_state=2)
clt_KM.fit(X_embedding)
```

```
y_KM_pred = clt_KM.predict(X_embedding)
plot_scatter(X_embedding, y_KM_pred, save_as='KMeans.png', title='KMeans clustering')
print('KMean adjusted random score:', adjusted_rand_score(movies,y_KM_pred ))
clt_GMM = GaussianMixture(n_components=4, random_state=0, init_params='random')
clt_GMM.fit(X_embedding)
y_GMM_pred = clt_GMM.predict(X_embedding)
plot_scatter(X_embedding, y_GMM_pred, save_as='GMM.png', title='GMM clustering')
print('GMM adjusted random score:', adjusted_rand_score(movies, y_GMM_pred))
```

Solution:



KMean adjusted random score: 0.25935622456493296
GMM adjusted random score: 0.4189198834394529
Both algorithm performed poorly on the clustering because they did not identify 4 nice clusters. This is probably because the algorithm does not guarantee to achieve the global optimal solution, the initial state may lead to convergence on a local minimum instead of the optimal one.

(c) **(7.5 pts)** Use the `sklearn.cluster.KMeans` class on `X_embedding` with `n_clusters` set to 4, `init` set to 'random', `n_init` set to 100, and `random_state` set to 2. Visualize the low dimensional data by labeling/coloring each tweet with Kmeans over 100 different starting points results. Calculate the adjusted rand score

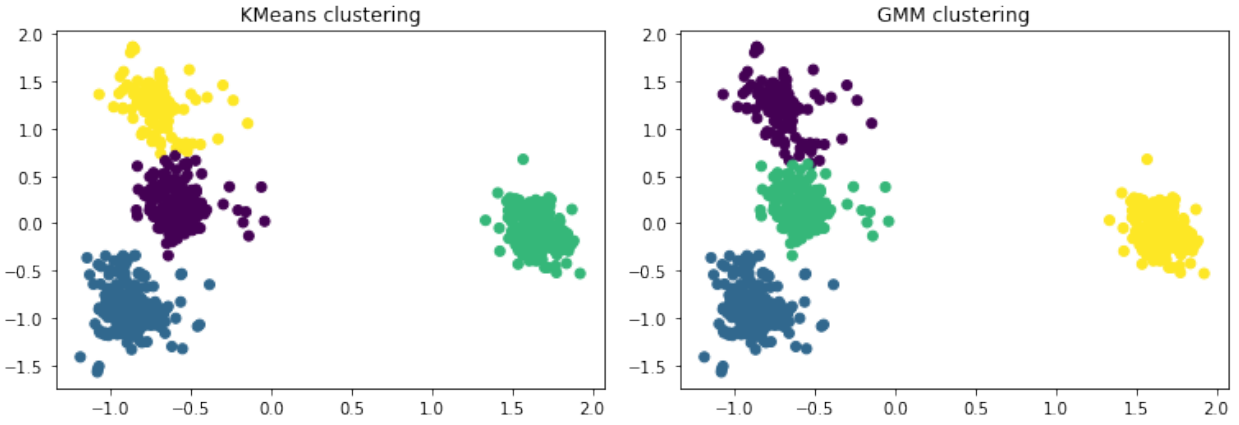
Use the `sklearn.mixture.GaussianMixture` class on `X_embedding` with `n_components` set to 4, `random_state` set to 0 `init_params` set to 'random', and `n_init` set to 100. Visualize the low dimensional data by labeling/coloring each tweet with this random initialized GMM but with 100 different starting points results. Calculate the adjusted rand score

Do you see a huge performance gain over what we got from part (b)? Include both plots, the performance results, and a 1-2 sentence description/justification of what you saw in your

writeup. **Please also copy-paste your code as part of the solution for this problem.**

```
clt_KM = KMeans(n_clusters=4, init='random', n_init=100, random_state=2)
clt_KM.fit(X_embedding)
y_KM_pred = clt_KM.predict(X_embedding)
plot_scatter(X_embedding, y_KM_pred, save_as='KMeans.png', title='KMeans clustering')
print('KMean adjusted random score:', adjusted_rand_score(movies,y_KM_pred ))
clt_GMM = GaussianMixture(n_components=4, random_state=0, init_params='random', n_init=100)
clt_GMM.fit(X_embedding)
y_GMM_pred = clt_GMM.predict(X_embedding)
plot_scatter(X_embedding, y_GMM_pred, save_as='GMM.png', title='GMM clustering')
print('GMM adjusted random score:', adjusted_rand_score(movies, y_GMM_pred))
```

Solution:



KMean adjusted random score: 0.9824442429232366
GMM adjusted random score: 0.982486480501798
Now both plots show 4 nice clusters and their adjusted random scores also improve a lot. Hence we now see a huge performance gain over the ones from part (b). By changing n_init to a larger value like 100, it now runs the algorithm with 100 different positions of the starting centers and the final results will be the best output of n_init consecutive runs in terms of inertia. Hence, it increases the probability of achieving the global optimal.