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:
One simple solution is:
$\boxed{\Phi(x) = [g(x), 0, 0, ..., 0]}$
Proof:
$k(x,z) = \Phi(x)^T\Phi(z)$
$= [g(x), 0, 0, ..., 0] \cdot [g(z), 0, 0, ..., 0]$
$= g(x)g(z) + 0 * 0 + 0 * 0 + ...0 * 0$
$= g(x)g(z)$

(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:
$k(x,z) = \alpha \cdot k_1(x,z)$
$= \alpha\Phi_1(x)^T\Phi_1(z)$
Therefore
$\boxed{\Phi(x) = \sqrt{\alpha}\Phi_1(x)}$
For example if $\Phi_1(x)$ looks like:
$\Phi_1(x) = [t_1, t_2, ..., t_n]$
Then: $\Phi(x) = \sqrt{\alpha}\Phi_1(x)[t_1, t_2, ..., t_n]$
$= [\sqrt{\alpha}\Phi_1(x)t_1, \sqrt{\alpha}\Phi_1(x)t_2, ..., \sqrt{\alpha}\Phi_1(x)t_n]$
And solving the kernel is simple:
$\Phi(x)^T\Phi(x) = [\sqrt{\alpha}\Phi_1(x)t_1, \sqrt{\alpha}\Phi_1(x)t_2, ..., \sqrt{\alpha}\Phi_1(x)t_n] \cdot [\sqrt{\alpha}\Phi_1(x)t_1, \sqrt{\alpha}\Phi_1(x)t_2, ..., \sqrt{\alpha}\Phi_1(x)t_n]$
$= \alpha t_1^2 + \alpha t_2^2 + ...\alpha t_n^2$
$= \alpha[t_1^2 + t_2^2 + ...t_n^2]$
$= \alpha\Phi_1(x)^T\Phi_1(x)$
$= \alpha \cdot k_1(x,z)$
$= k(x,z)$

(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:
We want to satisfy:
$k(x,z) = k_1(x,z) + k_2(x,z)$
$k(x,z) = \Phi_1(x)^T\Phi_1(z) + \Phi_2(x)^T\Phi_2(z) = \Phi(x)^T\Phi(z)$
Assume $\Phi_1(x)$ and $\Phi_2(x)$ have the form:
$\Phi_1(x) = [t_{1,1}, t_{1,2}, ...t_{1,n}^2]$
$\Phi_2(x) = [t_{2,1}, t_{2,2}, ...t_{2,n}^2]$
Knowing:
$\Phi_1(x)^T\Phi_1(z) = t_{1,1}^2 + t_{1,2}^2 + ... + t_{1,n}^2$
and $\Phi_2(x)^T\Phi_2(z) = t_{2,1}^2 + t_{2,2}^2 + ... + t_{2,n}^2$
We can find the desired form for $\Phi(x)^T\Phi(z)$:
$\Phi(x)^T\Phi(z) = t_{1,1}^2 + t_{2,1}^2 + t_{1,2}^2 + t_{2,2}^2 + ... + t_{1,n}^2 + t_{2,n}^2$
If: $\Phi(x) = [t_{0,1}, t_{0,2}, ...t_{0,n}^2]$
Then:
$t_{0,j}^2 = t_{1,j}^2 + t_{2,j}^2$
Therefore:
$$\boxed{\Phi(x) = [\sqrt{t_{1,1}^2 + t_{2,1}^2}, \sqrt{t_{1,2}^2 + t_{2,2}^2}...\sqrt{t_{1,n}^2 + t_{2,n}^2}]}$$
This satisfies the solutions because:
$k(x,z) = \Phi(x)^T\Phi(z)$
$= (\sqrt{t_{1,1}^2 + t_{2,1}^2})^2 + (\sqrt{t_{1,2}^2 + t_{2,2}^2})^2 + ... + (\sqrt{t_{1,n}^2 + t_{2,n}^2})^2$
$= t_{1,1}^2 + t_{2,1}^2 + t_{1,2}^2 + t_{2,2}^2 + ... + t_{1,n}^2 + t_{2,n}^2$
$= \Phi_1(x)^T\Phi_1(z) + \Phi_2(x)^T\Phi_2(z)$
$= k_1(x,z) + k_2(x,z)$
$= LHS$

(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:
Let's use the same notation for the terms in $\Phi(x), \Phi_1(x), and \Phi_2(x)$ as we did in the previous part
$k(x,z) = k_1(x,z)k_2(x,z)$
$= (\Phi_1(x)^T\Phi_1(z))(\Phi_2(x)^T\Phi_2(z))$
$= (t_{1,1}^2 + t_{1,2}^2 + ... + t_{1,n}^2)(t_{2,1}^2 + t_{2,2}^2 + ... + t_{2,n}^2)$
$= (t_{1,1}^2t_{2,1}^2 + t_{1,1}^2t_{2,2}^2 + ...t_{1,1}^2t_{2,n}^2) + (t_{1,2}^2t_{2,1}^2 + t_{1,2}^2t_{2,2}^2 + ...t_{1,2}^2t_{2,n}^2) + ... + (t_{1,n}^2t_{2,1}^2 + t_{1,n}^2t_{2,2}^2 + ...t_{1,n}^2t_{2,n}^2))$
Therefore:
$t_{0,j} = \sqrt{t_{0,j}^2t_{2,1}^2 + t_{1,j}^2t_{2,2}^2 + ...t_{1,j}^2t_{2,n}^2}$
$$\boxed{\Phi(x) = [\sqrt{t_{0,1}^2t_{2,1}^2 + t_{1,1}^2t_{2,2}^2 + ...t_{1,1}^2t_{2,n}^2}, \sqrt{t_{0,2}^2t_{2,1}^2 + t_{1,2}^2t_{2,2}^2 + ...t_{1,2}^2t_{2,n}^2}...\sqrt{t_{0,n}^2t_{2,1}^2 + t_{1,n}^2t_{2,2}^2 + ...t_{1,n}^2t_{2,n}^2}]}$$
This can be shown true by computing: $k(x,z) = \Phi(x)^T\Phi(z)$
$= (t_{1,1}^2t_{2,1}^2 + t_{1,1}^2t_{2,2}^2 + ...t_{1,1}^2t_{2,n}^2) + (t_{1,2}^2t_{2,1}^2 + t_{1,2}^2t_{2,2}^2 + ...t_{1,2}^2t_{2,n}^2) + ... + (t_{1,n}^2t_{2,1}^2 + t_{1,n}^2t_{2,2}^2 + ...t_{1,n}^2t_{2,n}^2))$
$= (t_{1,1}^2 + t_{1,2}^2 + ... + t_{1,n}^2)(t_{2,1}^2 + t_{2,2}^2 + ... + t_{2,n}^2)$
$= (\Phi_1(x)^T\Phi_1(z))(\Phi_2(x)^T\Phi_2(z))$
$= k_1(x,z)k_2(x,z)$

$= LHS$

(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:

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

where $k_1^i(x, z)$ is $k_1(x, z)$ to the power of i.
Solution:
First we perform the Taylor series expansion of $\exp(k_1(x, z))$:
$= 1 + \sum_{i=1}^{i=\infty} \frac{k_1^i(x,z)}{i!}$
Using rule 2, $\frac{k_1^i(x,z)}{i!}$ is a kernel by setting $alpha = 1/i!$ Let's call this new kernel k_2
Now we have:
$= 1 + \sum_{i=1}^{i=\infty} k_2(x, z)$
Using rule 3, we know $\sum_{i=1}^{i=\infty} k_2(x, z)$ is a kernel, let's calls this k_3:
$= 1 + k_3(x, z)$
We can use rule 1 to say 1 is a kernel. Let $g(x) = 1$ for all x values. Then $k_4(x, z) = 1$ for all x, z and is a kernel.
$= k_4 + k_3$
Using rule 3 again:
$= k_1(x, z)$
It is 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:
$k(x, z) = \exp(\frac{-\|x-z\|^2}{\sigma^2})$
$= \exp(\frac{-(x-z)^T(x-z)}{\sigma^2})$
$= \exp(\frac{-(x^Tx - x^Tz - z^Tx - z^Tz)}{\sigma^2})$
$= \exp(\frac{-[\|x\|^2 - \|z\|^2 - 2x^Tz]}{\sigma^2})$
$= \exp(\frac{-[\|x\|^2 - \|z\|^2]}{\sigma^2}) \exp(\frac{2x^Tz}{\sigma^2})$
Applying the Taylor series expansion of $\exp(a)$ to the second term:
$= \exp(\frac{-\|x\|^2+\|z\|^2}{\sigma^2})[1 + \frac{1}{1!}\frac{2x^Tz}{\sigma^2} + \frac{1}{2!}(\frac{2x^Tz}{\sigma^2})^2 + \frac{1}{3!}(\frac{2x^Tz}{\sigma^2})^3 + ... + \frac{1}{\infty!}(\frac{2x^Tz}{\sigma^2})^\infty]$
$\exp(\frac{-\|x\|^2+\|z\|^2}{})$ is a constant, let's call it C
$= C \sum_{i=1} \frac{1}{i!}(\frac{2x^Tz}{\sigma^2})^i$
$= C \sum_{i=1} \frac{2^i}{i!} \frac{x^Tz^i}{\sigma^{2i}}$
Looking at the definition of a polynomial kernel:
$k(x, y) = (x^Ty + c)^d$
For every $d = i$ we can call each polynomial kernel $k_i(x, z)$
$= C \sum_{i=1} \frac{2^i}{i!} \frac{k(x,z)}{\sigma^{2i}}$
Then we can use rule 2 to know the inner term is a kernel. Let's call it k_1:
$= \sum_{i=1} k_1(x, z)$

Then we can use rule 3 to show the summation is a kernel. Let's call it $k_2 = k_2(x, z)$
Proven

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:
The support vectors are x_5 and x_2. These points are the closest together oppositely labeled points. The margin will lie in between these two points and the size of the margin will be the distance to each of these points.

(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:
We can use 2 properties to setup 2 different equations and solve the equations.
1. Firstly, the distance between the points and the line must be equal.
Using the formula: $d = \frac{|Ax_0+Bx_0+C|}{\sqrt{A^2+B^2}}$ we can create the equation:
$|3w_1 + 2w_2| = |w_1 - w_2|$
Knowing that the point on the right is labelled negative, we can set this side as negative and the other side as positive.
$3w_1 + 2w_2 = -w_1 + w_2$
We reduce to find our first equation:
$4w_1 - w_2 = 0$
2. Secondly, we know the points are $2/|w|$ apart:
$\frac{2}{\sqrt{w_1^2+w_2^2}} = \sqrt{(3-1)^2 + (2--1}^2$
This reduces to: $w_1^2 + w_2^2 = \frac{4}{13}$
I then solved the equations by graphing them and finding the intersections. This gave 2 solutions: $(w_1 = 0.135, w_2 = -0.538)$ and $(w_1 = -0.538, w_2 = 0.135)$
By testing the 2 solutions on any point, we can see the first solution is correct
$\boxed{\boldsymbol{w} = [0.135, -0.538]}$

The margin is then:
$1/||w|| = \frac{1}{\sqrt{w_1^2+w_2^2}} = \boxed{1.803}$

(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:
We can create 3 equations and solve them:
1. We know point (3,2) is the closest positive target. Therefore the prediction will be:
$1 = 3w_1 + 2w_2 + b$
2. We know point (1,-1) is the closest negative target. Therefore the prediction will be:
$-1 = w_1 - w_2 + b$
3. We know the points are a distance of $2/||W||$ apart:
$\frac{2}{||W||} = \sqrt{(3-1)^2 + (2--1)^2}$
Reduces:
$w_1^2 + w_2^2 = 4/13$
Solving the equations with a calculator yields:
$\boxed{w_1 = \frac{4}{13}, w_2 = \frac{6}{13}, b = \frac{-11}{13}}$
The classifier no longer goes through the origin.
We can easily solve for $\frac{1}{||W||}$:
$\frac{1}{||W||} = \frac{1}{\sqrt{(4/13)^2+(6/13)^2}}$:
$\boxed{\frac{1}{||W||} = 1.803}$
The margin distance did not change. Intuitively this makes sense, because the distance between the 2 points is equal to $2/||W||$, and the points did not move, so $||W||$ must remain the same.

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:
We have 4 data points and 3 clusters. Therefore, there will be 2 clusters with 1 point each and 1 cluster with 2 points. To minimize the distance, we will choose x_1 and x_2 to share a cluster, and x_3 and x_4 to have their own cluster. We can easily prove this is the optimal clustering because the total distance in any 1 point distance is 0 by definition. So we can realize our optimization problem is analogous to picking the two closest points in our data

set, and putting them in the same cluster. Therefore we pick x_1 and x_2 to go in the same cluster. We can find the value of the objective by:
1. Calculate average points per cluster:
$\mu_1 = \frac{x_1+x_2}{2} = 1.5$
$\mu_1 = \frac{x_3}{1} = 5$
$\mu_1 = \frac{x_4}{1} = 7$
2. Calculate distance squared from all points per cluster:
Cluster 1:
$(1-1.5)^2 + (2-1.5)^2 = 0.5$
Cluster 2:
$(5-5)^2 = 0$
Cluster 3:
$(7-7)^2 = 0$
3. Summing the squared distances, we have a value of $\boxed{0.5}$

(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:
The k-means algorithm first assigns data points to their closest cluster. Therefore our clusters will be:

$\boxed{Cluster1 : x_1}$ $\boxed{Cluster2 : x_2}$ $\boxed{Cluster3 : x_3, x_4}$
The algorithm then updates the value of the clusters to be the average of the data points in that cluster:
$\mu_1 = \frac{x_1}{1} = 1$
$\mu_1 = \frac{x_2}{1} = 2$
$\mu_1 = \frac{x_3+x_4}{2} = 6$
The algorithm then repeats, re-assigning the data-points to the nearest cluster. However every data point is closest to the cluster it is already assigned to.
So the algorithm does not improve the clusters at the second iteration. It converges on a local solution.

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],

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

[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. :-)

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.

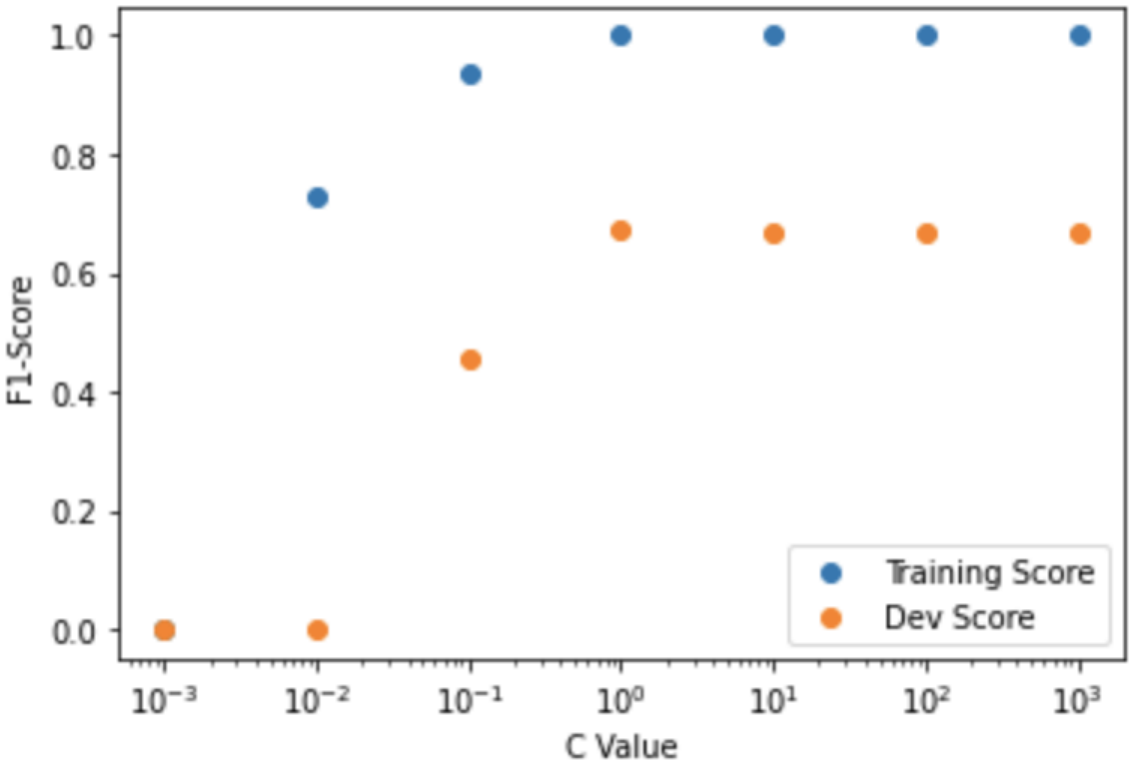
SVMs have hyperparameters that must be set by the user. For linear kernel SVM, we will select the hyper-parameter using a simple train set and a development set. In the notebook, the train,

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 reaches its best value at 1 and worst value at 0. Below is a plot of the F1-Scores for all C's:



The strength of the regularization principle is inversely proportional to C. Our data reflects that our model fits the data more when it is designed to be less general. When C increases, the model is told to be more specific, and it fits the data better. For a C¿1, the F1-Score the on the training set is 1 and 0.669 on the dev set. Interestingly we might assume that the greater the C, the worse the dev score should be because the model should suffer from overfitting but the data does not show that.

Code:

```
C_vals = 10.0 ** np.arange(-3, 4)
train_scores = []
dev_scores = []
for C_val in C_vals:
  #Create, fit, and get predictions
  model = SVC(kernel='linear',C=C_val)
  model.fit(X_train,y_train)
  y_pred = model.predict(X_train)

  #Take score measurements
  train_scores.append(metrics.f1_score(y_train,y_pred))
  y_pred = model.predict(X_dev)
  dev_scores.append(metrics.f1_score(y_dev,y_pred))

#Print information
print("C Vals: {}".format(C_vals))
print("Train scores: {}".format(train_scores))
print("Dev scores: {}".format(dev_scores))

#Plot information
plt.figure()
plt.plot(C_vals, train_scores, 'o',label="Training Score")
plt.plot(C_vals, dev_scores, 'o',label="Dev Score")
plt.xscale("log")
plt.legend()
plt.xlabel("C Value")
plt.ylabel("F1-Score")
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:

Because the last 4 values of C: $1, 10, 10^2, 10^3$ all had equally high F-1 scores, I decided to choose $C = 1$ to have the least chance of overfitting. The F1-Score on the test set ended up being: 0.8736, which is considerably higher than the dev set scores from the last part.
Code:

```
#Combine the sets
X_train_and_dev = np.concatenate((X_train, X_dev))
y_train_and_dev = np.concatenate((y_train, y_dev))

#Create and train the model, make predictions
model = SVC(kernel='linear',C=1)
model.fit(X_train_and_dev,y_train_and_dev)
y_pred = model.predict(X_test)
```

```
#Run metrics
score = metrics.f1_score(y_test,y_pred)
print("The F-1 score with C=1 on the test set is: {}".format(score))
```

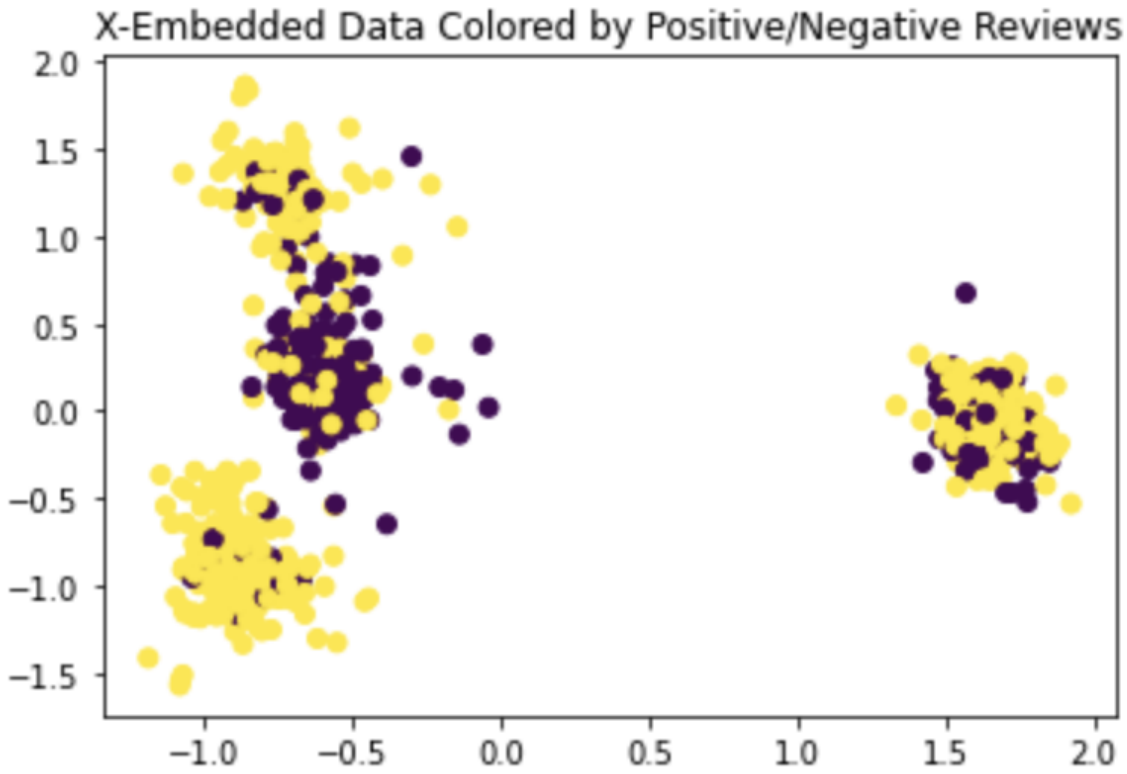
4.2 Low Dimensional Embedding Space Clustering [20 pts]

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.

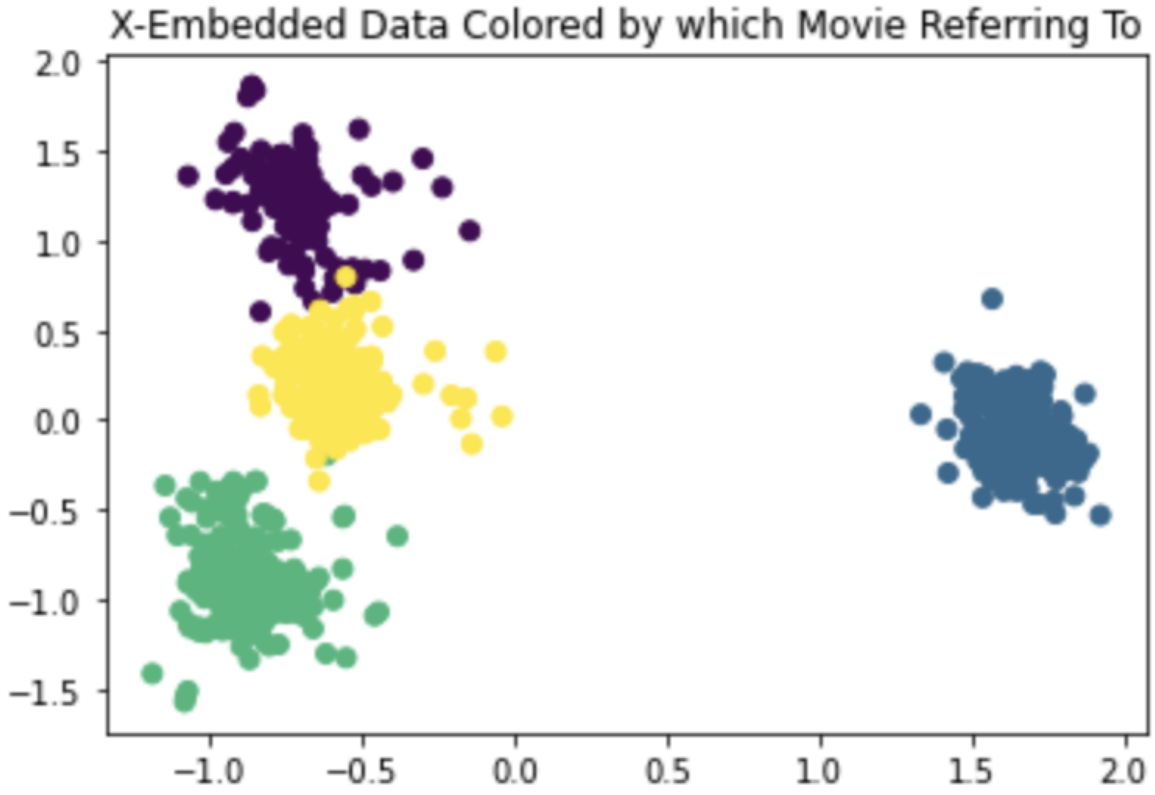
We will also evaluate the purity of the clusters returned from any unsupervised algorithms against the ground truth labels from the original `tweet_df.txt`.

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 and negative reviews do not form 2 "nice" clusters. However there are some groupings, mainly 4 areas, 3 of which are dominated by yellow dots, and 1 of which is dominated by dark dots. However, the data does not appear to be able to be separated into 2 clusters.



The 4 movies form 4 clusters very nicely. Only the yellow and dark purple clusters seem to have an intersection, but it is very small compared to the size of the clusters.

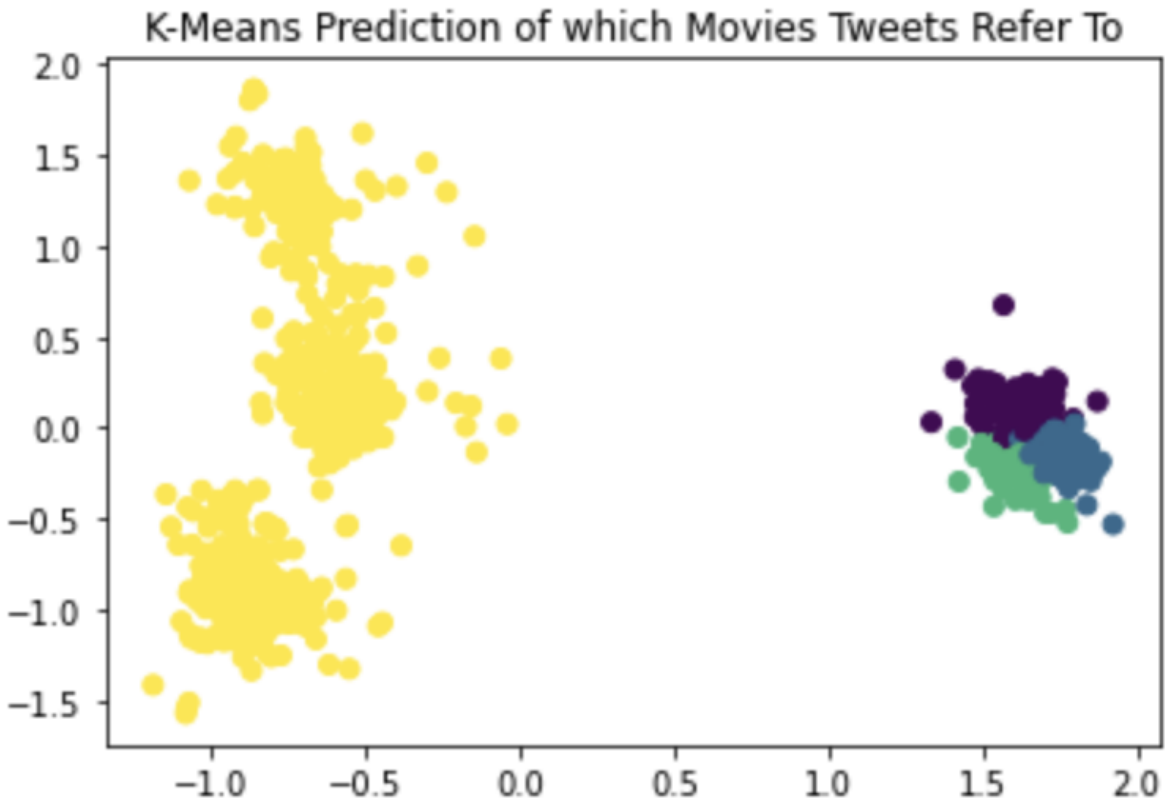
Code:

```
plot_scatter(X_embedding,y,title="X-Embedded Data Colored by Positive/Negative Reviews")
plot_scatter(X_embedding,movies,title="X-Embedded Data Colored by which Movie Referring T
```

(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

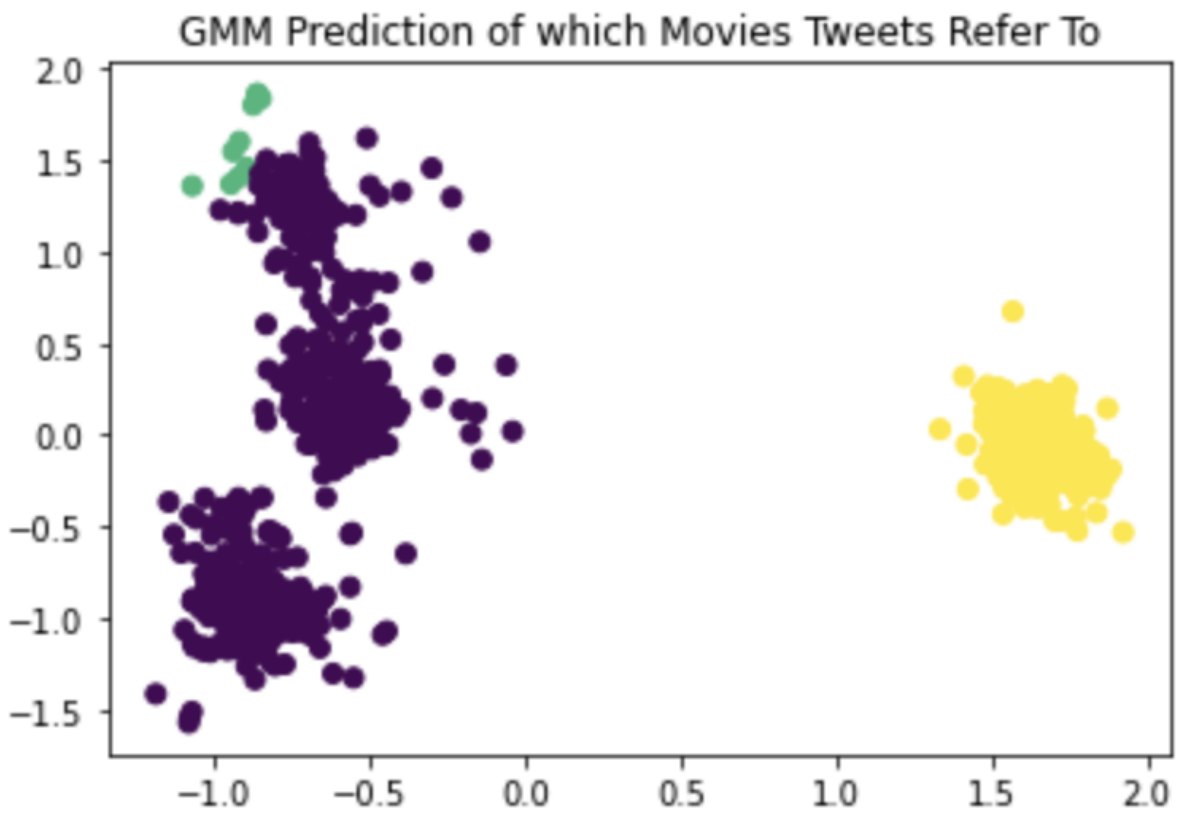
Use the `sklearn.mixture.GaussianMixture` class on `X_embedding` with `n_components` set to 4, `random_state` set to 0 `init_params` set to 'random'. Visualize the low dimensional data by labeling/coloring each tweet with GMM results. Calculate the adjusted rand score

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.**
Solution:



Adjusted Rand Score of K-Means: 0.259

The K-Means model poorly as obvious by the low adjusted rand score and by visually comparing the predictions with the actual labels. It is most likely that we started with a very bad initial clustering, given to us by random state = 2, and because n init = 1, we do not repeat with different random clusters. In the end we end up with a local solution that is not a global optimal solution.



Adjusted Rand Score of GMM: 0.419
The GMM model performed better than the K-Means model with an adjusted rand score of 0.419. However when comparing the graph with the correctly labeled graph in the previous part, it is is evident the model made a bad first guess initial clustering. If the number of initialized clusters is increased the model will surely perform better.
Code:

```
#Kmeans
#------
#Create and fit model, then make prediction
kmeans_model = KMeans(n_clusters=4,init="random",n_init=1,random_state=2)
kmeans_model.fit(X_embedding)
y_pred_kmeans = kmeans_model.labels_

#Generate Data
plot_scatter(X_embedding,y_pred_kmeans,title="K-Means Prediction of which Movies Tweets R
kmeans_score = adjusted_rand_score(movies, y_pred_kmeans)
print("The K-Means model had an adjusted rand score of: {}".format(kmeans_score))

#GMM
#---
#Create and fit model, then make prediction
gmm_model = GaussianMixture(n_components=4,init_params="random",random_state=0)
gmm_model.fit(X_embedding)
```

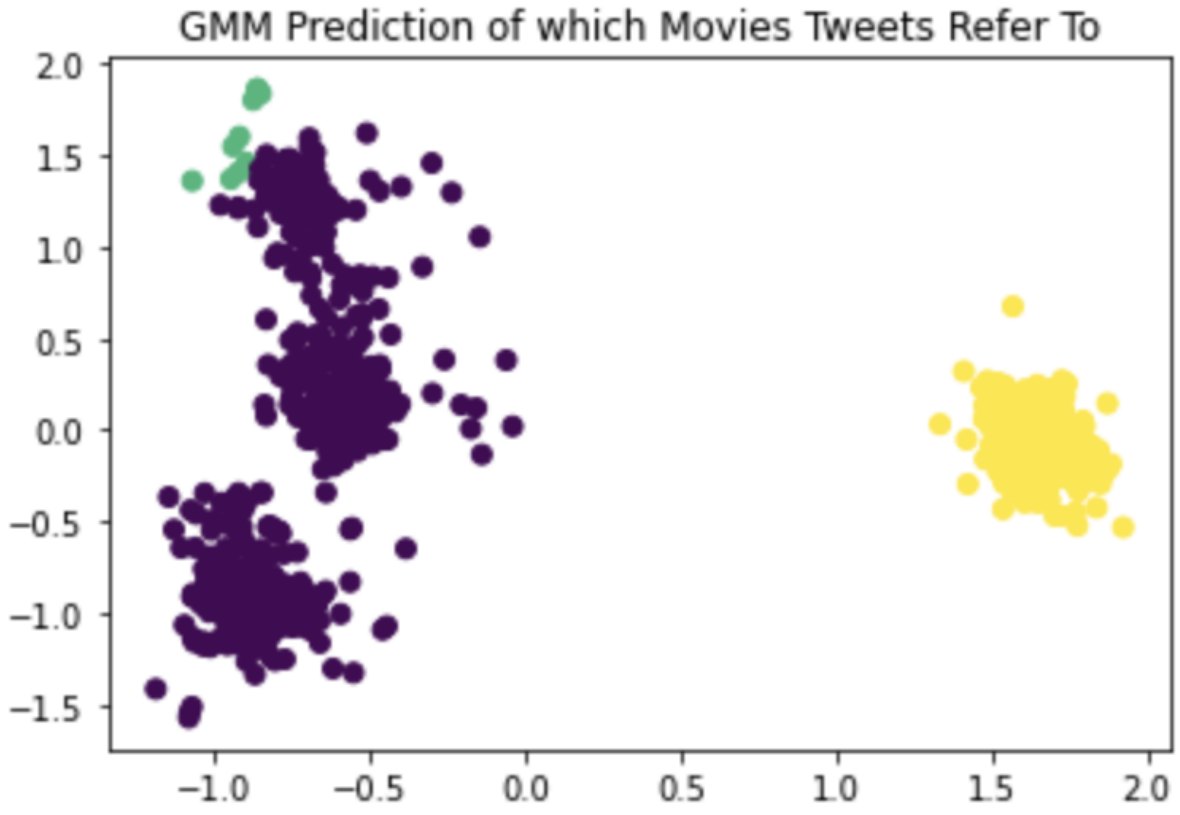
```
y_pred_kmeans = gmm_model.predict(X_embedding)

#Generate Data
plot_scatter(X_embedding,y_pred_kmeans,title="GMM Prediction of which Movies Tweets Refer
gmm_score = adjusted_rand_score(movies, y_pred_kmeans)
print("The K-Means model had an adjusted rand score of: {}".format(gmm_score))
```

(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

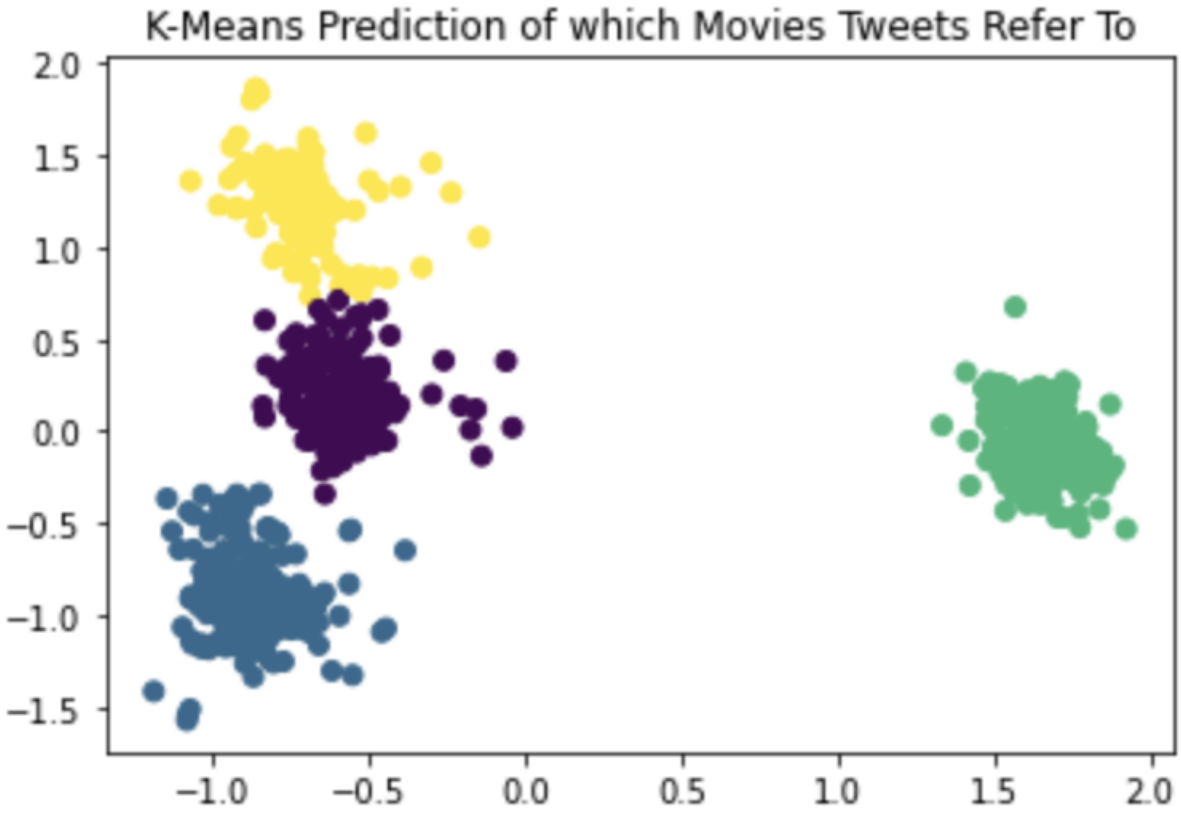
Use the `sklearn.mixture.GaussianMixture` class on `X_embedding` with `n_components` set to 4, `random_state` set to 0 `init_params` set to 'random', and `n_init` set to 100. Visualize the low dimensional data by labeling/coloring each tweet with this random initialized GMM but with 100 different starting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

Solution:



Adjusted Rand Score of K-Means: 0.982



Adjusted Rand Score of GMM: 0.982
Both models performed far better with 100 starting clusters as opposed to the 1 they had earlier. Both models now have a rand score of 0.982 vs the scores of 0.419 and 0.259 they had earlier. Also visually examining the models show the labels much closer match the actual labels.
Code:

```
#Kmeans
#------
#Create and fit model, then make prediction
kmeans_model = KMeans(n_clusters=4,init="random",n_init=100,random_state=2)
kmeans_model.fit(X_embedding)
y_pred_kmeans = kmeans_model.labels_

#Generate Data
plot_scatter(X_embedding,y_pred_kmeans,title="K-Means Prediction of which Movies Tweets R
kmeans_score = adjusted_rand_score(movies, y_pred_kmeans)
print("The K-Means model had an adjusted rand score of: {}".format(kmeans_score))

#GMM
#---
#Create and fit model, then make prediction
gmm_model = GaussianMixture(n_components=4,init_params="random",n_init=100,random_state=0
```

```
gmm_model.fit(X_embedding)
y_pred_kmeans = gmm_model.predict(X_embedding)

#Generate Data
plot_scatter(X_embedding,y_pred_kmeans,title="GMM Prediction of which Movies Tweets Refer
gmm_score = adjusted_rand_score(movies, y_pred_kmeans)
print("The K-Means model had an adjusted rand score of: {}".format(gmm_score))
```