PA04 Unsupervised Learning: dimensionality reduction

In this short programming assignment you will apply dimensionality reduction on the movie rating data from PA03.

Part I: Collaborative Filtering ratings revisited

In PA03 you used user-user and item-item similarity to generate expected ratings by collaborative filtering. We defined similarity by representing each user (or item) as sparse vectors in a high-dimensional space. For example, in the user-user similarity case, we represented each user as a vector in a space of dimensions equal to the number of movies in the dataset. We then computed cosine similarities in this high-dimensional space.

In this assignment you will perform a dimensionality reduction method before computing similarities.

Q1: Implement the principal component analysis method. To compute a k-dimensional embedding (where k is much lower than the data-dimensionality D), you need to calculate the first k eigenvectors of the sparse ratings matrix. You should get acquainted with the `scipy.sparse` library, and its linear algebra facilities `numpy.sparse.linalg`. You should implement a function of signature

```
def pca(df, axis=0, k):
"""
Compute a k-dimensional embedding of matrix mat using PCA


Arguments:
df: a two-dimensional pandas DataFrame with columns userid, itemid and rating
axis: which axis to treat as examples (0 or 1)
k: number of dimensions in embedding


Returns:
numpy matrix of shape (m,k) where m is the number of unique userids in df when axis=0
and m is the number of unique itemids in df when axis=1
"""
```

Q2: Plot a two-dimensional embedding of movies labeled by genre. Discuss.

Q3: Plot a two-dimensional embedding of users. Is there any salient structure explainable by the user features in the database?

Q4: Implement a method that uses low dimensional embeddings to calculate expected ratings using collaborative filtering. This requires that you define similarity on the low-dimensional space. Compare collaborative filtering expected ratings from low-dimensional embedding to those we got originally from the high-dimensional space. Discuss the effect of the cardinality of the low-dimensional space on the accuracy of these ratings.

Part II: Classification using new ratings

Q5: Add the new collaborative filtering ratings from Part I to your best classifier from PA03. Discuss accuracy performance.

Part III: Ratings challenge

Q6: Submit a classifier for a final semester contest. The goal is to predict ratings (1-5, not isgood). You are free to use any method or code you choose. The only constriaint is that your classifier has the following signature:

```
def rating_class(df, ratings):
"""
Build a movie rating predictor


Arguments:
df: a pandas DataFrame with the same features as the data in PA03 (it may not contain the isgood column so you shouldn't refer to it) w
ratings: a pandas Series with ratings between 1-5


Returns:
An object with a predict method as described below
"""
```

The function should return an object with a `predict` method with the following signature:

```
def predict(df):
"""
Predict movie ratings for dataframe df

Arguments:
df: a pandas DataFrame with the same features as the data in PA03 (it may not contain the isgood column so you shouldn't refer to it)

Returns:
Predictions (between 1-5) for each of the user-movie pairs in df
"""
```

We will evaluate your classifier using mean squared error (your classifier can return any real number between 1-5). Please describe how your method predicts ratings.

In [ ]: