PCA Review

Week 10 -- 1/11/17

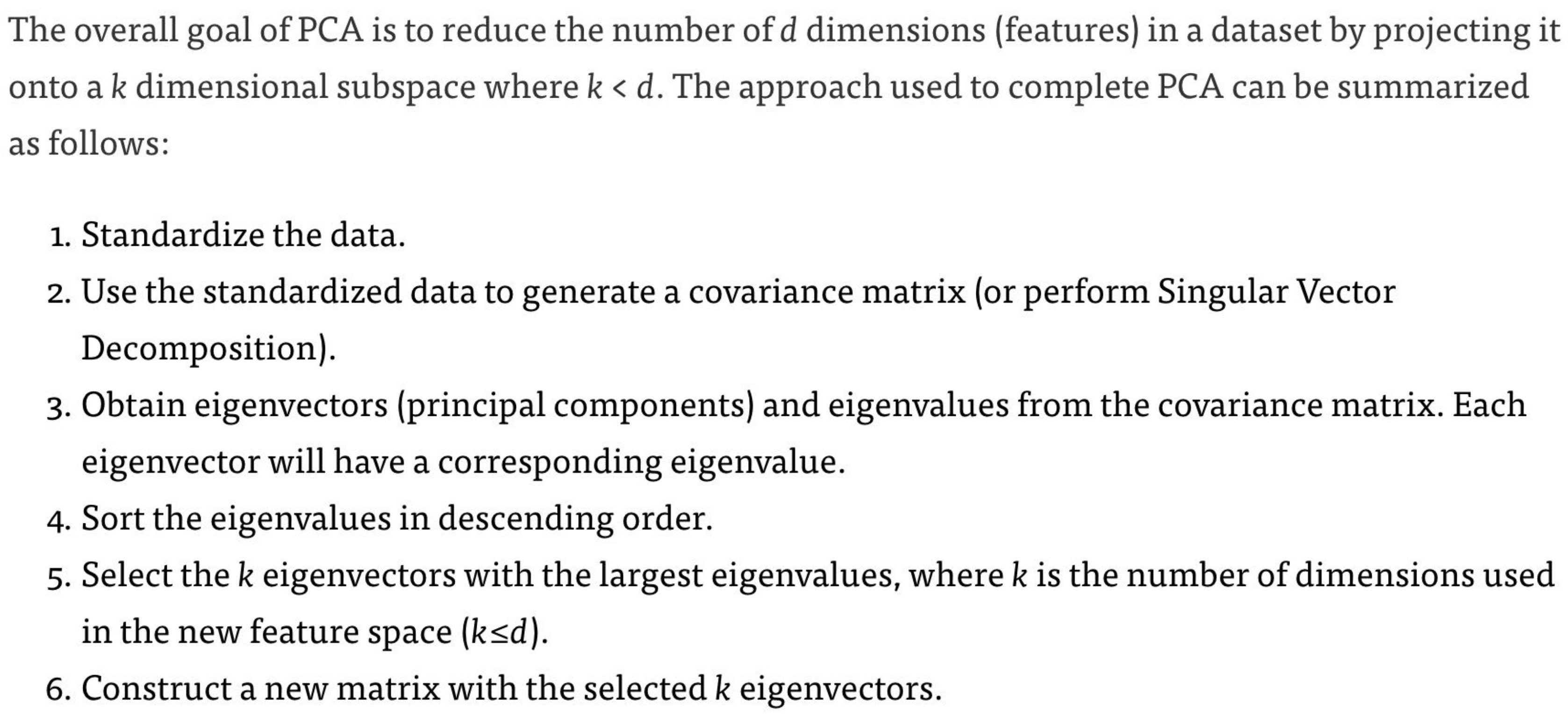
Update on PCA Resources:

- Best for an **easy introduction to PCA**:
  - https://districtdatalabs.silvrback.com/principal-component-analysis-with-python
- Best for a **detailed step-by-step PCA walkthrough** (steps and explanations for both manual PCA and scikit-learn PCA):
  - http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html
- Best for a **Kilian-esque style walkthrough for 3 approaches to PCA** (manual PCA; matplotlib.mlab.PCA(); scikit-learn PCA)
  - http://sebastianraschka.com/Articles/2014_pca_step_by_step.html

Easy introduction to PCA

Source:
https://districtdatalabs.silvrback.com/principal-component-analysis-with-python

PCA Overall Steps



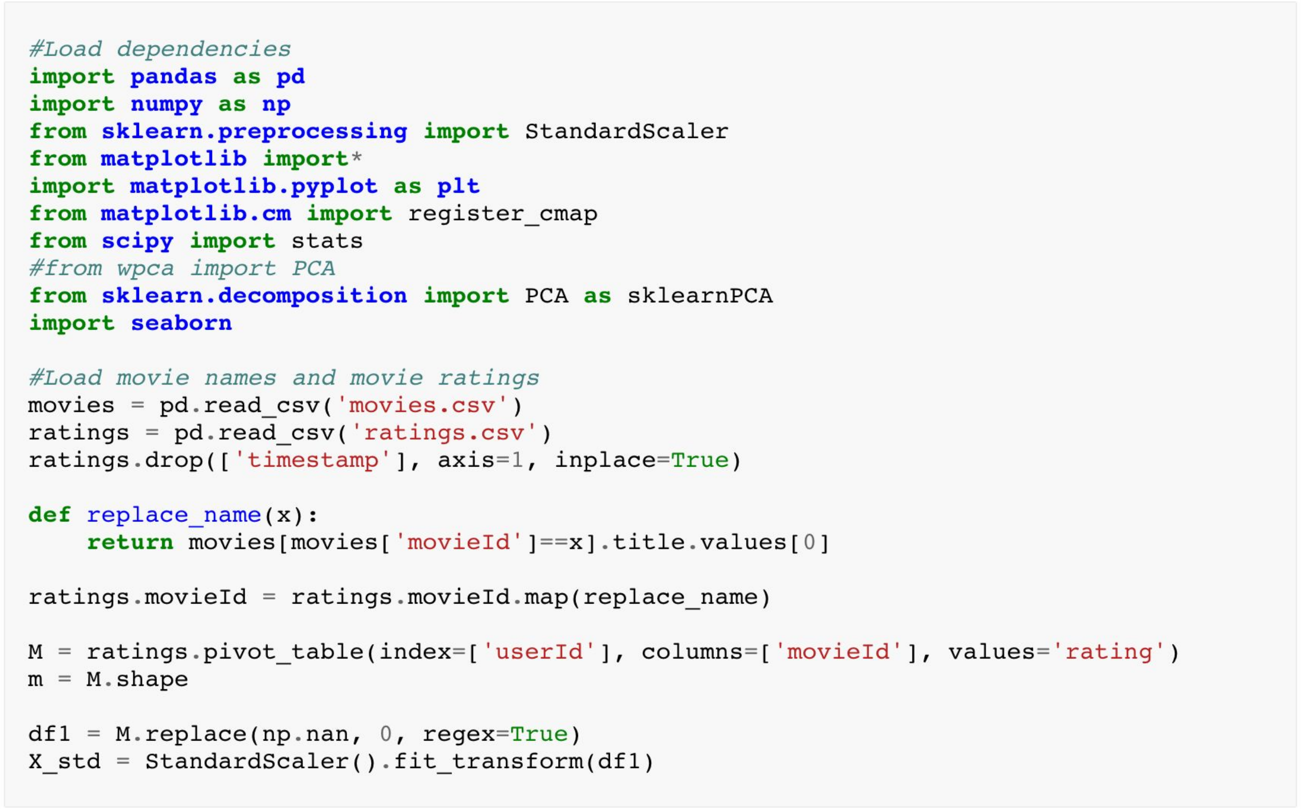
The overall goal of PCA is to reduce the number of d dimensions (features) in a dataset by projecting it onto a k dimensional subspace where $k < d$. The approach used to complete PCA can be summarized as follows:

1. Standardize the data.
2. Use the standardized data to generate a covariance matrix (or perform Singular Vector Decomposition).
3. Obtain eigenvectors (principal components) and eigenvalues from the covariance matrix. Each eigenvector will have a corresponding eigenvalue.
4. Sort the eigenvalues in descending order.
5. Select the k eigenvectors with the largest eigenvalues, where k is the number of dimensions used in the new feature space ($k \leq d$).
6. Construct a new matrix with the selected k eigenvectors.

Step 1: Load and Standardize Data

First we'll load the data and store it in a pandas dataframe. The data set contains ratings from 718 users (instances) for 8,913 movies (features). Even though all of the features in the dataset are measured on the same scale (a 0 through 5 rating), we must make sure that we standardize the data by transforming it onto a unit scale (mean=0 and variance=1). Also, all null (NaN) values were converted to 0. It is necessary to transform data because PCA can only be applied on numerical data.

Step 1: Load and Standardize Data (cont’d)



```python
#Load dependencies
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from matplotlib import*
import matplotlib.pyplot as plt
from matplotlib.cm import register_cmap
from scipy import stats
#from wpca import PCA
from sklearn.decomposition import PCA as sklearnPCA
import seaborn

#Load movie names and movie ratings
movies = pd.read_csv('movies.csv')
ratings = pd.read_csv('ratings.csv')
ratings.drop(['timestamp'], axis=1, inplace=True)

def replace_name(x):
    return movies[movies['movieId']==x].title.values[0]

ratings.movieId = ratings.movieId.map(replace_name)

M = ratings.pivot_table(index=['userId'], columns=['movieId'], values='rating')
m = M.shape

df1 = M.replace(np.nan, 0, regex=True)
X_std = StandardScaler().fit_transform(df1)
```

Step 2: Covariance Matrix and Eigendecomposition

Next, a covariance matrix is created based on the standardized data. The covariance matrix is a representation of the covariance between each feature in the original dataset.

The covariance matrix can be found as follows:

```
mean_vec = np.mean(X_std, axis=0)
cov_mat = (X_std - mean_vec).T.dot((X_std - mean_vec)) / (X_std.shape[0]-1)
print('Covariance matrix \n%s' %cov_mat)
```

Alternatively, you can also create the same covariance matrix with one line of code.

```
print('NumPy covariance matrix: \n%s' %np.cov(X_std.T))
```

Step 2: Covariance Matrix and Eigendecomposition (cont’d)

After the covariance matrix is generated, eigendecomposition is performed on the covariance matrix. Eigenvectors and eigenvalues are found as a result of the eigendceomposition. Each eigenvector has a corresponding eigenvalue, and the sum of the eigenvalues represents all of the variance within the entire dataset.

The eigendecomposition can be performed as follows:

```
#Perform eigendecomposition on covariance matrix
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

Step 3: Selecting Principal Components

Eigenvectors, or principal components, are a normalized linear combination of the features in the original dataset. The first principal component captures the most variance in the original variables, and the second component is a representation of the second highest variance within the dataset.

Step 3: Selecting Principal Components (cont'd)

The eigenvectors with the lowest eigenvalues describe the least amount of variation within the dataset. Therefore, these values can be dropped. First, lets order the eigenvalues in descending order:

```
# Visually confirm that the list is correctly sorted by decreasing eigenvalues
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]
print('Eigenvalues in descending order:')
for i in eig_pairs:
    print(i[0])
```

To get a better idea of how principal components describe the variance in the data, we will look at the explained variance ratio of the first two principal components.

```
pca = PCA(n_components=2)
pca.fit_transform(df1)
print pca.explained_variance_ratio_
```

Step 3: Selecting Principal Components (cont’d)

The first two principal components describe approximately 14% of the variance in the data. In order gain a more comprehensive view of how each principal component explains the variance within the data, we will construct a scree plot. A scree plot displays the variance explained by each principal component within the analysis.

```
#Explained variance
pca = PCA().fit(X_std)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```

SQL Knowledge Check

Source:
https://www.analyticsvidhya.com/blog/2017/01/46-questions-on-sql-to-test-a-data-science-professional-skilltest-solution/?utm_source=feedburner&utm_medium=email&utm_campaign=Feed%3A+AnalyticsVidhya+%28Analytics+Vidhya%29

Quizlet Link:
https://quizlet.com/_2ywonv