Data Mining and Machine Learning
Fall 2018, Homework 6 [VERSION 2.0]
(due on Oct 23, 11.59pm EST)

Jean Honorio jhonorio@purdue.edu

The homework is based on a total of 10 points. Your code **should be in Python 2.7**. For clarity, the algorithms presented here will assume zero-based indices for arrays, vectors, matrices, etc. Please read the submission instructions at the end. **Failure to comply to the submission instructions will cause your grade to be reduced.**

Here are the questions:

1) [7 points] Implement the learning part of principal component analysis (PCA), introduced in Lecture 12. First, let define the diag function which converts a vector v to a diagonal matrix, which diagonal values are equal to the entries of v, and which every non-diagonal entry is zero. For instance, let:

$$v = \begin{bmatrix} 2.4 \\ -1.2 \\ 3.5 \\ -7.1 \end{bmatrix} \quad , \quad \text{diag}(v) = \begin{bmatrix} 2.4 & 0 & 0 & 0 \\ 0 & -1.2 & 0 & 0 \\ 0 & 0 & 3.5 & 0 \\ 0 & 0 & 0 & -7.1 \end{bmatrix}$$

Let $X \in \mathbb{R}^{n \times d}$ be the data matrix for n samples and d features. PCA maps each sample from d dimensions to $F \in \{1, \ldots, \min(n, d)\}$ dimensions, thus we can express the projection as a matrix $Z \in \mathbb{R}^{d \times F}$.

Input: number of features F, data matrix $X \in \mathbb{R}^{n \times d}$
Output: average $\mu \in \mathbb{R}^d$, principal components $Z \in \mathbb{R}^{d \times F}$
for $i = 0, \ldots, d-1$ **do**
  $\mu_i \leftarrow \frac{1}{n} \sum_{t=0}^{n-1} x_{ti}$
end for
for $t = 0, \ldots, n-1$ **do**
  for $i = 0, \ldots, d-1$ **do**
    $x_{ti} \leftarrow x_{ti} - \mu_i$
  end for
end for
Compute the singular value decomposition of X. That is, $X = U \, \text{diag}(s) \, V^{\mathrm{T}}$, where $U \in \mathbb{R}^{n \times \min(n,d)}$, $s \in \mathbb{R}^{\min(n,d)}$, $V^T \in \mathbb{R}^{\min(n,d) \times d}$, $U^{\mathrm{T}} U = I$, $V^{\mathrm{T}} V = I$

$g \leftarrow$ first F entries of vector s. That is, $g \in \mathbb{R}^F$
for $i = 0, \ldots, F-1$ **do**
  if $g_i > 0$ **then**
    $g_i \leftarrow 1/g_i$
  end if
end for
$W \leftarrow$ first F rows of matrix V^T. That is, $W \in \mathbb{R}^{F \times d}$
$Z \leftarrow W^T \, \mathrm{diag}(g)$

The header of your **Python script pcalearn.py** should be:

```
# Input: number of features F
#        numpy matrix X, with n rows (samples), d columns (features)
# Output: numpy vector mu, with d rows, 1 column
#         numpy matrix Z, with d rows, F columns
def run(F,X):
  # Your code goes here
  return (mu, Z)
```


Note: For an example of singular value decomposition in numpy, look at Slide 3 in Lecture 12. (I updated the slide on Oct 17.) In the example, you will note that numpy returns V^T, and not V.

Note: Additionally, numpy has a function "diag". For instance:


```
>>> import numpy as np
>>> v = np.array([2.4, -1.2, 3.5, -7.1])
>>> np.diag(v)
array([[ 2.4,  0. ,  0. ,  0. ],
       [ 0. , -1.2,  0. ,  0. ],
       [ 0. ,  0. ,  3.5,  0. ],
       [ 0. ,  0. ,  0. , -7.1]])
```

2) [3 points] Implement the projection part of principal component analysis (PCA), introduced in Lecture 12.

Input: data matrix $X \in \mathbb{R}^{n \times d}$, average $\mu \in \mathbb{R}^d$, principal components $Z \in \mathbb{R}^{d \times F}$
Output: projected data matrix $P \in \mathbb{R}^{n \times F}$
for $t = 0, \ldots, n-1$ **do**
  for $i = 0, \ldots, d-1$ **do**
    $x_{ti} \leftarrow x_{ti} - \mu_i$
  end for
end for
$P \leftarrow XZ$

The header of your **Python script pcaproj.py** should be:

```
# Input: number of features F
#        numpy matrix X, with n rows (samples), d columns (features)
#        numpy vector mu, with d rows, 1 column
#        numpy matrix Z, with d rows, F columns
# Output: numpy matrix P, with n rows, F columns
def run(X,mu,Z):
  # Your code goes here
  return P
```

TEST CASES.
We provide few synthetic datasets to test your Python scripts:

Test Case 1: Projecting 2-dimensional data to 1 dimension


```
>>> import pcalearn
>>> import pcaproj
>>> import numpy as np
>>> np.set_printoptions(precision=4)
>>> X = np.array([[-3, 2],
  [-2, 1.5],
  [-1, 1],
  [0, 0.5],
  [1, 0]])
>>> mu, Z = pcalearn.run(1,X)
>>> mu
array([[-1.],
       [ 1.]])
>>> Z
array([[ 0.253 ],
       [-0.1265]])
>>> P = pcaproj.run(X,mu,Z)
>>> P
array([[-0.253 ],
       [ 0.0632],
       [ 0.3795],
       [ 0.6957],
       [ 1.0119]])
```


Test Case 2: Projecting 3-dimensional data to 2 dimensions

```
>>> import pcalearn
>>> import pcaproj
>>> import numpy as np
```

```
>>> np.set_printoptions(precision=4)
>>> X = np.array([[-2, 2, 0],
  [-3, -1.5, -2],
  [-1, 1, 4],
  [1, -0.5, 5],
  [2, 0, -2]])
>>> mu, Z = pcalearn.run(2,X)
>>> mu
array([[-0.6],
       [ 0.2],
       [ 1. ]])
>>> Z
array([[ 0.0304,  0.2429],
       [ 0.0129, -0.0304],
       [ 0.1444, -0.0484]])
>>> X_test = np.array([[-4, 5, 1],
  [2, 1, 4],
  [-4, 5, 7.5],
  [-9.5, 0, 5]])
>>> P_test = pcaproj.run(X_test,mu,Z)
>>> P_test
array([[-0.0413, -0.9721],
       [ 0.3228,  0.4622],
       [ 0.8973, -1.2863],
       [-0.4172, -2.1077]])
```


Test Case 2: Projecting 7-dimensional data to 3 dimensions

```
>>> import pcalearn
>>> import pcaproj
>>> import numpy as np
>>> np.set_printoptions(precision=4)
>>> X = np.array([[-2, 2, 0, -2, 2, 0, 4],
  [-3, -1.5, -2, 6, 5, 1, 4],
  [-1, 1, 4, 0, 5, -4, 5],
  [1, -0.5, 5, -9, -9, 0, 0],
  [2, 0, -2, -4.5, 3, 3, 1]])
>>> mu, Z = pcalearn.run(3,X)
>>> mu
array([[-0.6],
       [ 0.2],
       [ 1. ],
       [-1.9],
       [ 1.2],
       [ 0. ],
       [ 2.8]])
>>> Z
```

```
array([[-0.0106,  0.026 ,  0.0614],
       [-0.0002, -0.0142,  0.0577],
       [-0.0152, -0.084 ,  0.0094],
       [ 0.0386, -0.0184, -0.1026],
       [ 0.0408,  0.008 ,  0.112 ],
       [-0.0017,  0.0824, -0.0282],
       [ 0.013 , -0.0373,  0.0118]])
>>> X_test = np.array([[-4, 5, 1, -4, -4, 0, 1],
  [2, 1, 4, -9.5, -4, 0, 1]])
>>> P_test = pcaproj.run(X_test,mu,Z)
>>> P_test
array([[-0.2813, -0.0924, -0.3198],
       [-0.6022, -0.0308,  0.4103]])
```

SOME POSSIBLY USEFUL THINGS.

Python 2.7 is available at the servers antor and data. From the terminal, you can use your Career account to start a ssh session:

```
ssh username@data.cs.purdue.edu
OR
ssh username@antor.cs.purdue.edu
```

From the terminal, to start Python:

```
python
```

Inside Python, to check whether you have **Python 2.7**:

```
import sys
print (sys.version)
```


SUBMISSION INSTRUCTIONS.

Your code **should be in Python 2.7**. We **only need** the Python scripts (.py files). We **do not need** the Python (compiled) bytecodes (.pyc files). You will get 0 points if your code does not run. You will get 0 points in you fail to include the Python scripts (.py files) even if you mistakingly include the bytecodes (.pyc files). We will deduct points, if you do not use the right name for the Python scripts (.py) as described on each question, or if the input/output matrices/vectors/scalars have a different type/size from what is described on each question. Homeworks are to be solved individually. We will run plagiarism detection software.

Please, submit a single ZIP file **through Blackboard**. Your Python scripts (**pcalearn.py**, **pcaproj.py**) should be directly inside the ZIP file. **There should not be any folder inside the ZIP file**, just Python scripts. The ZIP file should be named according to your Career account. For instance, if my Career account is jhonorio, the ZIP file should be named **jhonorio.zip**