Introduction to Machine Learning
Problems: Principal Component Analysis

Profs. Sundeep Rangan and Yao Wang

1. Assume that you have 4 samples each with dimension 3, described in the data matrix X,

$$X = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 5 \\ 1 & 2 & 3 \\ 0 & 2 & 5 \end{bmatrix}$$

   For the problems below, you may do the calculations in python. Describe the commands you used.

   (a) Find the sample mean.

   (b) Find the sample covariance matrix Q.

   (c) Find the eigenvalues and eigenvectors. You can use the `numpy.linalg.eig` to compute eigenvalues and eigenvectors from Q.

   (d) Find the PCA coefficients corresponding to samples in X.

   (e) Reconstruct the original samples from the PCA coefficients.

   (f) Approximate the samples using principle components corresponding to the two largest eigenvalues.

   (g) Verify the sum of reconstruction error squares = sum of squares of skipped PCA coefficients.

2. *PC bases.* Suppose that the mean and first two PCs in a basis are,

$$\boldsymbol{\mu} = [1, 0, 2], \quad \mathbf{v}_1 = \frac{1}{\sqrt{2}}[1, 1, 0], \quad \mathbf{v}_2 = \frac{1}{\sqrt{2}}[1, -1, 0],$$

   (a) What are the two PC coefficients of the vector $\mathbf{x} = [2, 3, 4]$?

   (b) What is $\widehat{\mathbf{x}}$, the approximation of $\mathbf{x}$ using two PCs?

   (c) What is the approximation error $\|\mathbf{x} - \widehat{\mathbf{x}}\|^2$?

3. *Fitting data with PCs.* You are given python functions for PCA transform and a binary classifier:

```
mu, V = PCA(X)  # Finds the mean and PCs for a data matrix X

clf = Classifier()
clf.fit(Z,y)    # Fits a binary classifier from features Z and labels y
yhat = clf.predict(Z)  # Predicts labels from Z
```

Given a data matrix `X` with binary labels `y`, you wish to build a classifier that uses a PCA transform followed by the `Classifier`. Write python code that uses model validation to select the optimal number of PCA components to use. You may assume:

- You use simple cross validation with one training and test split. Not You do not need to do K-fold validation.
- You should use roughly 25% of the samples for test. Data shuffling before splitting is not needed.

4. *PC with images.* A dataset has 1000 images of size 28 × 28, represented as python array `X` with shape `(1000,28,28)`. You are given the following python functions:

```
Y = reshape(X, shape)         # Reshapes X to a shape
pca = PCA(n_components = nc)  # Creates a PCA object
pca.fit(Y)   # Finds the mean and PCs from training data Y
Z = pca.transform(Y)   # Find coefficients in the PC basis
Yhat = pca.inverse_transform(Z)  # Invert the PC transform
```

Write a few lines of python code to (i) fit a PC model from the first 500 images with 5 components; and (ii) create an array of approximations of the remaining 500 images.

5. *PCs using SVDs.* You are given a python function:

```
U,s, Vtr = svd(Z, full_matrices = False)
```

which computes an “economy” SVD. Given a data matrix `X`, write python code that:

(i) Finds the PCs and mean of the data.

(ii) Finds the minimum number of PCs in order that the proportion of variance is at least 90%.

(iii) Create an approximation `Xhat` of `X` using the number of PCs in part (ii).