# DSC 140B - Homework 03

Due: Wednesday, January 28

**Instructions:**

- Write your solutions to the following problems **by hand**, either on on another piece of paper that you scan or using a tablet. Typed solutions will not be accepted for credit!
- Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer.
- Homework problems are graded pass/fail on completeness and effort, not correctness.
- Homeworks are due via Gradescope at 11:59 PM.

**Problem 1.** (1 credit)

As a data scientist, there will be many times when you will be working with massive, high dimensional data sets consisting of hundreds of thousands or even millions of points. This is not one of those times.

In this problem, we'll work with the following data set of three points:

$$\begin{aligned} x^{(1)} &= (1,3)^T \\ x^{(2)} &= (-3,-9)^T \\ x^{(3)} &= (2,6)^T \end{aligned}$$

**a)** Compute the sample covariance[1] matrix by hand. Show the calculations for each entry of the matrix.

**Solution:** We have three numbers to compute: the variance of feature 1, the variance of feature two, and the covariance.

The variance of the first feature:

$$\begin{aligned} \sigma_{11} &= \frac{1}{3}(1^2 + (-3)^2 + 2^2) \\ &= \frac{1}{3}(1+9+4) \\ &= \frac{14}{3} \end{aligned}$$

The variance of the second feature:

$$\begin{aligned} \sigma_{22} &= \frac{1}{3}(3^2 + (-9)^2 + 6^2) \\ &= \frac{1}{3}(9+81+36) \\ &= \frac{126}{3} \\ &= 42 \end{aligned}$$

[1]Use the version of the sample covariance defined in lecture, not the one that divides by $n-1$.

And the covariance:

$$\begin{aligned}\sigma_{12} &= \frac{1}{3}(1 \times 3 + (-3) \times (-9) + 2 \times 6) \\ &= \frac{1}{3}(3 + 27 + 12) \\ &= \frac{42}{3} \\ &= 14\end{aligned}$$

Therefore, the covariance matrix is

$$\begin{pmatrix} \frac{14}{3} & 14 \\ 14 & 42 \end{pmatrix}$$

**b)** What is the top eigenvector of the covariance matrix? You do not need to calculate the eigenvector explicitly, but you should justify your answer.

Hint: plot the data.

**Solution:** If we plot the data, we see that the three points are collinear – they all fall on the line of slope 3 through the origin. In other words, all three points are in the direction $(1, 3)^T$. This the direction of maximum variance, which we know is what the top eigenvector of the covariance matrix encodes.

Therefore, the top eigenvector of the covariance matrix is $(1, 3)^T$. Or, if you prefer normalized eigenvectors: $\frac{1}{\sqrt{10}}(1, 3)^T$.

**c)** What is the eigenvalue associated with the top eigenvector?

**Solution:** We know that the top eigenvector $\vec{u} = (1, 3)^T$. Since it is an eigenvector, $C\vec{u} = \lambda\vec{u}$. So we can multiply $C$ and $\vec{u}$ to find $\lambda$.

We have:

$$\begin{aligned} C\vec{u} &= \begin{pmatrix} \frac{14}{3} & 14 \\ 14 & 42 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} \\ &= \begin{pmatrix} \frac{14}{3} + 42 \\ 14 + 126 \end{pmatrix} \\ &= \begin{pmatrix} \frac{140}{3} \\ 140 \end{pmatrix} \\ &= \frac{140}{3} \begin{pmatrix} 1 \\ 3 \end{pmatrix} \end{aligned}$$

So the top eigenvalue is 140/3.

**d)** Reduce the dimensionality of each point above by carrying out PCA by hand. Be sure to use the normalized eigenvector. Show your calculations.

Hint: one of your new features should be equal to $-3\sqrt{10}$.

**Solution:** To compute the new representation $z^{(i)}$ of point $x^{(i)}$, we carry out $x^{(i)} \cdot \vec{u}$, where $\vec{u}$ is the (normalized) top eigenvector of $C$. Therefore:

$$\begin{aligned} z^{(1)} &= x^{(1)} \cdot u \\ &= (1,3)^T \cdot \frac{1}{\sqrt{10}}(1,3)^T \\ &= \frac{1}{\sqrt{10}}(1+9) \\ &= \sqrt{10} \end{aligned}$$

$$\begin{aligned} z^{(2)} &= x^{(2)} \cdot \vec{u} \\ &= (-3,-9)^T \cdot \frac{1}{\sqrt{10}}(1,3)^T \\ &= \frac{1}{\sqrt{10}}(-3-27) \\ &= \frac{-30}{\sqrt{10}} \\ &= -3\sqrt{10} \end{aligned}$$

$$\begin{aligned} z^{(3)} &= x^{(3)} \cdot \vec{u} \\ &= (2,6)^T \cdot \frac{1}{\sqrt{10}}(1,3)^T \\ &= \frac{1}{\sqrt{10}}(2+18) \\ &= \frac{20}{\sqrt{10}} \\ &= 2\sqrt{10} \end{aligned}$$

**e)** The result of PCA is a data set consisting of three numbers. Compute the variance of these three numbers.

Hint: the result should be familiar.

**Solution:** The variance of the new features is

$$\begin{aligned} \frac{1}{3}\left[(z^{(1)})^2 + (z^{(2)})^2 + (z^{(3)})^2\right] &= \frac{1}{3}\left[\left(\sqrt{10}\right)^2 + \left(-3\sqrt{10}\right)^2 + \left(2\sqrt{10}\right)^2\right] \\ &= \frac{1}{3}\left[10+90+40\right] \\ &= \frac{140}{3} \end{aligned}$$

Which is, coincidentally, the top eigenvalue of $C$.

**Problem 2.** ($\frac{1}{2}$ credit)

Let $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)}$ be a set of $n$ *centered* data vectors in $\mathbb{R}^d$. If $\vec{u}$ is a unit vector, we compute the "variance in the direction of $\vec{u}$" by 1) reducing each data vector $\vec{x}^{(i)}$ to a single number $z^{(i)}$ by setting $z^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$; and then 2) computing the variance of these new numbers, $z^{(1)}, \ldots, z^{(n)}$. That is, the "variance in the direction of $\vec{u}$" is defined to be:

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{x}^{(i)} \cdot \vec{u})^2$$

Let $C$ be the sample covariance matrix. Show that:

$$\frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}^{(i)} \cdot \vec{u}\right)^2 = \vec{u}^T C \vec{u}$$

That is, show that the variance in the direction of $\vec{u}$ is also computed by the vector-matrix-vector product $\vec{u}^T C \vec{u}$.

Hint: this is an exercise in vector and matrix algebra. It helps to remember that the covariance matrix, $C$, can be written as $\frac{1}{n} X^T X$, where $X$ is the *data matrix*. It may help to define $\vec{v} = X\vec{u}$, and to recognize that the $i$th entry of $\vec{v}$ is $\vec{x}^{(i)} \cdot \vec{u}$. It is also helpful to remember that, for any vector $\vec{a}$, $\vec{a}^T \vec{a} = \|\vec{a}\|^2$, and that for any matrices/vectors $A$ and $B$, $(AB)^T = B^T A^T$.

**Solution:** Let $\vec{v} = X\vec{u}$. The vector $\vec{v}$ has $n$ entries, the $i$th of which is given by dotting the $i$th row of $X$ with $\vec{u}$. Since the $i$th row of $X$ is $\vec{x}^{(i)}$, we have:

$$v_i = \vec{x}^{(i)} \cdot \vec{u}.$$

Noting that $\vec{v}^T = (X\vec{u})^T = \vec{u}^T X^T$, we see that $\vec{u}^T X^T X \vec{u} = \vec{v}^T \vec{v} = \|\vec{v}\|^2$. Since the squared norm of a vector is the sum of the squares of its elements, we find:

$$\begin{aligned}\vec{u}^T X^T X \vec{u} &= \|\vec{v}\|^2 \\ &= \sum_{i=1}^{n} v_i^2 \\ &= \sum_{i=1}^{n}\left(\vec{x}^{(i)} \cdot \vec{u}\right)^2.\end{aligned}$$

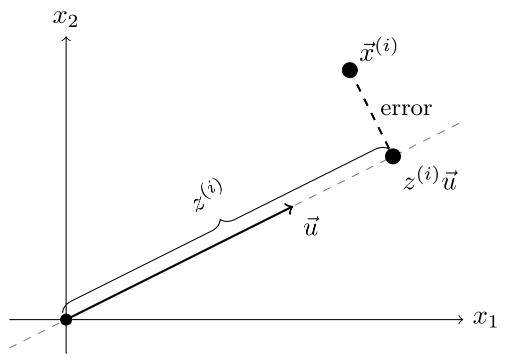
**Problem 3.** ($\frac{1}{2}$ credit)

There are several ways to derive PCA. In lecture, we've focused on maximizing variance. Another common approach (and one we will see in lecture soon) is that of minimizing "reconstruction error." In this problem, we will explore this approach in one dimension and see how PCA (like regression) finds a line of "best fit", just using a different way of measuring error.

For this problem, suppose you have a data set of $n$ two-dimensional points $\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(n)}$, where each $\vec{x}^{(i)} \in \mathbb{R}^2$. Assume that the data has already been centered.

Suppose $\vec{u}$ is a unit vector containing our "mixture coefficients" determining how much of each original feature goes into our new feature. Then, for each data point $\vec{x}^{(i)}$, we define $z^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$ to be our "new feature" for that point.

The picture below illustrates this setup, showing a single data point $\vec{x}^{(i)}$, the unit vector $\vec{u}$, and the projection of $\vec{x}^{(i)}$ onto the line spanned by $\vec{u}$. Our new feature is simply the signed distance from the origin to this projection, which is $z^{(i)}$.



If our goal is to reduce dimensionality, we keep this single new feature $z^{(i)}$ for each data point $\vec{x}^{(i)}$ and "throw away" the original two-dimensional data point. Graphically, we are replacing each point $\vec{x}^{(i)}$ with its projection onto the line spanned by $\vec{u}$. This projection is given by $z^{(i)}\vec{u} = (\vec{x}^{(i)} \cdot \vec{u})\vec{u}$, and is also shown in the figure. This point is often called the *reconstruction* of the original point $\vec{x}^{(i)}$.

The reconstruction is usually not exactly equal to the original point, so there is some *reconstruction error*, which is defined to be the squared distance between the original point and its reconstruction:

$$\text{reconstruction error of point } i = \|\vec{x}^{(i)} - (\vec{u} \cdot \vec{x}^{(i)})\vec{u}\|^2$$

Our goal is to find the unit vector $\vec{u}$ that minimizes the total reconstruction error across all data points:

$$\text{total reconstruction error} = \sum_{i=1}^{n} \|\vec{x}^{(i)} - (\vec{u} \cdot \vec{x}^{(i)})\vec{u}\|^2.$$

**a)** Show that the total reconstruction error can be written as

$$\sum_{i=1}^{n} \|\vec{x}^{(i)}\|^2 - \textit{(variance in the direction of } \vec{u}\textit{)}.$$

Hint: use the Pythagorean theorem.

**b)** Argue that minimizing the total reconstruction error is equivalent to maximizing the variance in the direction of $\vec{u}$.

**c)** The reconstruction error measures how well a line (spanned by $\vec{u}$) fits the data. Therefore, like least squares regression, PCA finds a line of best fit for the data. However, PCA and regression use different ways of measuring error.

Briefly explain why you wouldn't want to use PCA if your goal is to predict a target variable $y$ from an input variable $x$.

**Problem 4.** (1 credit)

The Olivetti faces data set contains a collection of images of peoples' faces. It can be downloaded at the following URL:

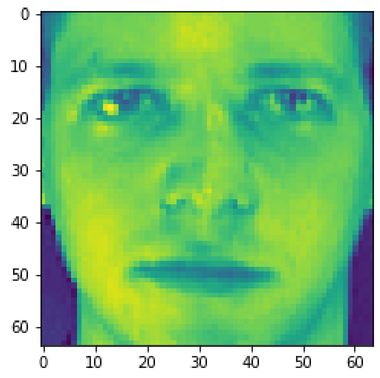
`https://f000.backblazeb2.com/file/dsc-data/faces.csv`

Each row in that file represents one face. It is a vector with 4096 entries, each entry recording the intensity of a pixel in a $64 \times 64$ image. The row can be reshaped and plotted using `matplotlib` to display it as an image. For example, the code below plots the first image in the data set.

```
import numpy as np
import matplotlib.pyplot as plt

faces = np.loadtxt("faces.csv", delimiter=',')
example_image = faces[0]
plt.imshow(example_image.reshape((64, 64)))
```

You should see this image:



The full set of images forms a point cloud in 4096-dimensional space. The directions in which this point cloud has the greatest variance are often human-interpretable, in that they tend to correspond to ways in which faces vary. For instance, one "dimension" along which faces vary is in the presence/absence of facial hair; another is in the presence/absence of a nose ring, etc.

Some of the people in this dataset are wearing glasses – but who? Since the eigenvectors of the covariance matrix correspond to directions of maximum variance, we can use them to find eyeglass wearers without using any labels whatsoever.

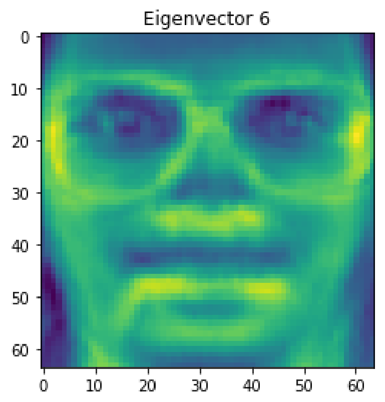
For this problem, you can use whatever programming language or package you'd like. But, as always, the tradeoff is that if you choose to use a package, you're expected to read the documentation to figure out how the package works.

Finally, since this is a programming problem, you are not expected to handwrite your solutions, but you should include a screenshot of your code and its output in your submission.

**a)** Compute the **seventh** eigenvector $\vec{u}^{(7)}$ of the data set's sample covariance matrix (that is, the eigenvector with seventh largest eigenvalue). It, too, should be a vector in $\mathbb{R}^{4096}$. Reshape this vector using code like the above, and visualize it. You should see something like a face with eyeglasses. That is, this eigenvector is an "eyeglasses detector".

In your submission, show your code and the resulting image.

**Solution:** You should see the following image:



This can reasonably be called an "eyeglass detector".

Here is the code that generated it:

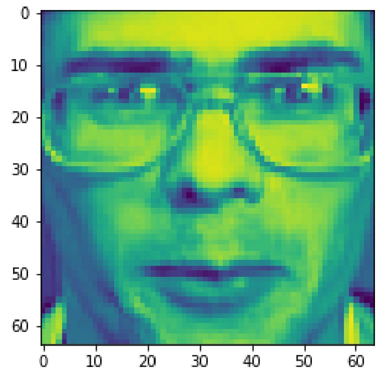
```
C = np.cov(data.T)
vals, vecs = np.linalg.eigh(C)
u_7 = vecs[-7]
```

**b)** Take the dot product of eigenvector $\vec{u}^{(7)}$ with every image in the data set. Plot the 20 images whose dot product with the eigenvector is the largest in absolute value. We will grade your answer to this problem manually by inspecting your plots.

If you did everything right, you should see a lot of eyeglasses.

Again, for this problem you should show the resulting images and your code.

**Solution:** Here is the image most aligned with $\vec{u}^{(7)}$:



This was computed by the following code:

```
top = np.argsort(np.abs(data @ u))[::-1]
for i in range(20):
    plt.figure()
    plt.imshow(data[top[i]].reshape((64, -1)))
```