# EECS 127/227AT Optimization Models in Engineering
# Spring 2019 Homework 3

**Release date:** 9/19/19.

**Due date:** 9/26/19, 23:00 (11 pm).

Please LaTeX or handwrite your homework solution and submit an electronic version.

**Submission Format**
Your homework submission should consist of a single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

## 1. Interpreting the data matrix

In several areas such as machine learning, statistics and data analysis you come across a data matrix $X$. Sometimes this matrix has dimensions $\mathbb{R}^{m\times n}$ while other times it has dimensions $\mathbb{R}^{n\times m}$ and it can get really confusing as to what exactly it represents. In this problem, we describe a way of interpreting the data matrix.

First, what exactly is a data matrix? As the name suggests, it is a collection of data points. Suppose you are collecting data about courses offered in EECS department in Fall 2019. Each course has certain attributes or features that you are interested in. Possible examples of features are the number of students in the course, the number of GSIs in the course,the number of units the course is worth, the size of the classroom that the course was taught in, the difficulty rating of the course in numerical (1-5) scale and so on. Suppose there were a total of 20 courses and for each course we have 10 features. Then we have 20 data points, with each data point being a 10-dimensional vector. We can arrange this in a matrix of size $20\times 10$, where each row corresponds to values of different features for the same point, and each column corresponds to values of same feature for different points.

Generalizing this, suppose we have $n$ data points with each point containing values for $m$ features (i.e each point lies in $m$-dimensional space) then our data matrix $X$ would be of size $n \times m$, i.e. $X \in \mathbb{R}^{n\times m}$. We can interpret $X$ in the following two ways:

(a) $X = \begin{bmatrix} \leftarrow \mathbf{x}_1^\top \rightarrow \\ \leftarrow \mathbf{x}_2^\top \rightarrow \\ \vdots \\ \leftarrow \mathbf{x}_n^\top \rightarrow \end{bmatrix}$.

Here $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, 2, \ldots, n$, and $x_i^\top$ is a row vector that contains values of different features for the $i$th data point.

(b) $X = \begin{bmatrix} \uparrow & \uparrow & \ldots & \uparrow \\ \mathbf{x}^1 & \mathbf{x}^2 & \ldots & \mathbf{x}^m \\ \downarrow & \downarrow & \ldots & \downarrow \end{bmatrix}$.

Here $\mathbf{x}^j \in \mathbb{R}^n, j = 1, 2, \ldots, m$ and $\mathbf{x}^j$ is a column vector that contains values of the the $j$th feature for different data points. Note that in several places you will encounter the case where the columns are referred to as $\mathbf{x}_1, \mathbf{x}_2, \ldots$ instead but it is important to understand the context and be clear on what the column represents.

Consider the matrix $X$ as described above. We explore how we can manipulate the data matrix to get some desirable properties.

(a) Suppose we want to compute a vector that contains the mean value for each feature. What is the length of the vector containing mean value of the features? Which of the following python commands will give us the mean value of the features:

i. feature_means = numpy.mean(X, axis = 0)

ii. feature_means = numpy.mean(X, axis = 1)

**Solution:** The vector containing mean value of features has length $m$. This can be found using the python command:
i) numpy.mean(X,axis=0)

(b) Suppose we want to compute the standard deviation of each feature. What is the dimension of the vector containing standard deviation of the features? Which of the following python commands will give us the standard deviation of the features:

i. feature_stddevs = numpy.std(X, axis = 0)

ii. feature_stddevs = numpy.std(X, axis = 1)

**Solution:** i) numpy.std(X, axis = 0)

(c) Suppose we want every feature *"centered"*, i.e. the feature is zero mean. How would you achieve this?

**Solution:** X = X - numpy.mean(X,axis=0).
Equivalent answer in text is fine.

(d) Suppose we want every feature *"standardized"*, i.e the feature is zero mean and has unit variance. How would you achieve this?

**Solution:**
X = X - numpy.mean(X,axis=0)
X = X / numpy.std(X,axis=0)
Equivalent answer in text is fine.

(e) Another metric of interest is the covariance matrix, which tells us how different features are related to each other. What is the size of the covariance matrix?:

i. $n \times n$

ii. $m \times m$.

*Hint: Is the number of features $m$ or $n$?*

**Solution:** The covariance matrix should be $m \times m$ since we have $m$ features.

(f) For rest of the problem assume that the data matrix is centred so every feature is zero mean. Let $C$ denote the covariance matrix. Show that $C$ can be represented in the following ways:

$$C = \frac{X^\top X}{n}$$

$$C = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top.$$

Recall that $\mathbf{x}_i^\top$ is the $i$th row of $X$.

**Solution:** Recall for a vector $\mathbf{x} = (x_1, ..., x_m)$ that the covariance of $\mathbf{x}$ is defined as $cov(\mathbf{x}, \mathbf{x}) = \mathbb{E}[(\mathbf{x} - \mu_x)(\mathbf{x} - \mu_x)]^\top$ where $\mu_x = \mathbb{E}[\mathbf{x}]$. If $\mathbf{x}$ is zero-mean this simplifies to $cov(\mathbf{x}, \mathbf{x}) = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$. Note for a finite sample where each feature is mean 0, we have that the covariance matrix becomes $\frac{1}{n}\sum_{i=1}^n \mathbf{x}\mathbf{x}^\top = X^\top X/n$.

(g) Let the $(i, j)$ entry of $C$ $(c_{ij})$ denote the covariance between feature $i$ and feature $j$. Then which of the following is true?

i. $c_{ij} = \frac{1}{m}(\mathbf{x}^i)^\top \mathbf{x}^j$

ii. $c_{ij} = \frac{1}{n}(\mathbf{x}^i)^\top \mathbf{x}^j$.

Recall that $\mathbf{x}^i$ is the $i$th column of $X$.

**Solution:** $c_{ij} = \frac{1}{n}(\mathbf{x}^i)^T \mathbf{x}^j$ which follows immediately from the definition of $C$ above.

(h) Recall that our data points are the rows of $X$ and these lie in a $m$-dimensional space. Suppose we are interested in taking the projection of the points onto a one-dimensional subspace in $\mathbb{R}^m$ spanned by the unit vector $u$. Sometimes this is referred to informally as "Projecting points along direction $u$". Then which of the following is true:

i. $\mathbf{u} \in \mathbb{R}^m$

ii. $\mathbf{u} \in \mathbb{R}^n$

*Hint: Think about how many points we have and what dimension a single point lies in.*

**Solution:** $\mathbf{u} \in \mathbb{R}^m$ since each of our data points lies in $\mathbb{R}^m$ and when we project along that line we take inner products and hence it must belong in the same dimension.

(i) Note there are three different interpretations of the term "projection" and these are used interchangeably with abuse of notation which can make it confusing at times.
Consider vectors $\mathbf{a}$ and $\mathbf{b}$ in $\mathbb{R}^n$. Let $\mathbf{b}$ be unit norm (i.e $\mathbf{b}^\top \mathbf{b} = 1$). Then we have:

i. The **vector projection** of $\mathbf{a}$ on $\mathbf{b}$ is given by $(\mathbf{a}^\top \mathbf{b})\mathbf{b}$. Note that is a vector in $\mathbb{R}^n$.

ii. The **scalar projection** of $\mathbf{a}$ on $\mathbf{b}$ is given $\mathbf{a}^\top \mathbf{b}$. This is a scalar but can take both positive and negative values.

iii. The **projection length** of $\mathbf{a}$ on $\mathbf{b}$ is given by $|\mathbf{a}^\top \mathbf{b}|$, and is the absolute value of the scalar projection.

Recall that our data points are the rows of $X$. Suppose we want to obtain a column vector, $\mathbf{z} \in \mathbb{R}^n$ containing scalar projections of points along the direction given by the unit vector $\mathbf{u}$. Show that this is given by,

$$z = X\mathbf{u}.$$

**Solution:** Note that since $\mathbf{u}$ is a unit vector, when we project a vector $\mathbf{y}$ onto $\mathbf{u}$, its scalar projection simply becomes $\mathbf{y}^\top \mathbf{u}$. Viewing the data matrix as each row being a data point, we have that $\mathbf{z} = X\mathbf{u}$ has the $i$ entry equal to $\mathbf{x}_i^\top \mathbf{u}$ as desired.

(j) Suppose we treat $\mathbf{z} = (z_1, z_2, \ldots . z_n)$ as samples of a random variable $Z \in \mathbb{R}$ corresponding to the scalar projection along direction $\mathbf{u}$. We are interested in calculating empirical variance of the scalar projections. Show that this can be calculated as

$$\sigma_z^2 = \frac{1}{n}\mathbf{u}^\top X^\top X\mathbf{u} = \mathbf{u}^\top C\mathbf{u}.$$

*Hint: The empirical variance is given by,* $\sigma_z^2 = \frac{1}{n}\sum_{i=1}^n (z_i - \mu_z)^2$, *where* $\mu_z = \frac{1}{n}\sum_{i=1}^n z_i$, *is the empirical mean. Recall that* $X$ *is assumed to be centered.*

**Solution:** Note first that $\mu_z = 0$ since

$$\begin{aligned}
\mu_z &= \frac{1}{n}\sum_{i=1}^{n} z_i \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{u} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} (\mathbf{x}^j)_i u_j \\
&= \sum_{j=1}^{m} \frac{1}{n} u_j \sum_{i=1}^{n} (\mathbf{x}^j)_i \\
&= \sum_{j=1}^{m} 0 \\
&= 0.
\end{aligned}$$

The second-to-last equality follows since every column of $X$ is zero-mean after centering. Here $(\mathbf{x}^j)_i$ refers to $i$th entry of column $j$ of $X$. Then we have that

$$\begin{aligned}
\sigma_z^2 &= \frac{1}{n}\sum_{i=1}^{n} (z_i - \mu_z)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{u} - 0)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{u})^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbf{u}^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{u} \\
&= \mathbf{u}^\top \Big(\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top\Big)\mathbf{u} \\
&= \frac{1}{n}\mathbf{u}^\top X^\top X \mathbf{u}
\end{aligned}$$

as desired.

**2. Computation and Geometric Interpretation of Singular Value Decomposition (SVD)**

Consider the $2 \times 2$ matrix

$$A = \frac{1}{\sqrt{10}} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & -1 \end{pmatrix} + \frac{2}{\sqrt{10}} \begin{pmatrix} -1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 1 \end{pmatrix}.$$

(a) What is an SVD of $A$? Express it as $A = USV^\top$, with $S$ the diagonal matrix of singular values ordered in decreasing fashion. Make sure to check all the properties required for $U, S, V$.

**Solution:** We have

$$A = \sigma_1 u_1 v_1^\top + \sigma_2 u_2 v_2^\top = USV^\top,$$

where $U = [u_1, u_2]$, $V = [v_1, v_2]$ and $S = \text{diag}\,((\,)\,\sigma_1, \sigma_2)$, with $\sigma_1 = 2$, $\sigma_2 = 1$, and

$$u_1 = \frac{1}{\sqrt{5}} \begin{pmatrix} -1 \\ 2 \end{pmatrix}, \quad u_2 = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad v_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad v_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

The triplet $(U, S, V)$ is an SVD of $A$, since $S$ is diagonal with non-negative elements on the diagonal, and $U$ and $V$ are orthogonal matrices ($U^\top U = V^\top V = I_2$). To check this, we first check that the Euclidean norm of $u_1, u_2, v_1, v_2$ is one. (This is why we factored the term $\sqrt{10}$ into $\sqrt{2} \cdot \sqrt{5}$.) In addition, $u_1^\top u_2 = v_1^\top v_2 = 0$. Thus, $U, V$ are orthogonal, as claimed.

(b) Find the semi-axis lengths and principal axes of the ellipsoid

$$\mathcal{E}(A) = \left\{ Ax \ : x \in \mathbb{R}^2, \ \ \|x\|_2 \leq 1 \right\}.$$

*Hint:* Use the SVD of $A$ to show that every element of $\mathcal{E}(A)$ is of the form $y = U\bar{y}$ for some element $\bar{y}$ in $\mathcal{E}(S)$. That is, $\mathcal{E}(A) = \{U\bar{y} \ : \ \bar{y} \in \mathcal{E}(S)\}$. (In other words the matrix $U$ maps $\mathcal{E}(S)$ into the set $\mathcal{E}(A)$.) Then analyze the geometry of the simpler set $\mathcal{E}(S)$.

**Solution:** We have, for every $x$, that $y := Ax = US(V^\top x)$ hence $y = U\bar{y}$, with $\bar{y} = S\bar{x}$ and $\bar{x} = V^\top x$. Since $V$ is orthogonal, $\|\bar{x}\|_2 = \|x\|_2$. In fact, when $x$ runs the unit Euclidean ball, so does $\bar{x}$. Thus every element of $\mathcal{E}(A)$ is of the form $y = U\bar{y}$ for some element $\bar{y}$ in $\mathcal{E}(S)$. To analyze $\mathcal{E}(A)$ it suffices to analyze $\mathcal{E}(S)$ and then transform the points of the latter set via the mapping $\bar{y} \to U\bar{y}$.

Since

$$\mathcal{E}(S) = \left\{ \sigma_1 \bar{x}_1 e_1 + \sigma_1 \bar{x}_2 e_2 \ : \ \bar{x}_1^2 + \bar{x}_2^2 \leq 1 \right\},$$

with $e_1, e_2$ the unit vectors, we have

$$\mathcal{E}(A) = \left\{ \sigma_1 \bar{x}_1 u_1 + \sigma_1 \bar{x}_2 u_2 \ : \ \bar{x}_1^2 + \bar{x}_2^2 \leq 1 \right\}.$$

In the coordinate system defined by the orthonormal basis $(u_1, u_2)$ the set is an ellipsoid with semi-axis lengths $(\sigma_1, \sigma_2) = (2, 1)$, and principal axes given by the coordinate axes. In the original system the principal axes are in the spans of

$$u_1 = \frac{1}{\sqrt{5}} \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

and

$$u_2 = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

respectively.

(c) What is the set $\mathcal{E}(A)$ when we append a zero vector after the last column of $A$, that is $A$ is replaced with $\tilde{A} = [A, 0] \in \mathbb{R}^{2\times 3}$?

**Solution:** When we append a zero column after the last column of $A$ we are doing nothing to $\mathcal{E}(A)$. Indeed, the condition

$$y = Ax \text{ for some } x \in \mathbb{R}^2, \ \ \|x\|_2 \leq 1$$

is the same as

$$y = \begin{pmatrix} A & 0 \end{pmatrix} z \text{ for some } z \in \mathbb{R}^3, \ \ \|z\|_2 \leq 1.$$

Geometrically, the projection of a 3-dimensional unit ball on the first two coordinates is the 2-dimensional unit ball. Hence we loose nothing if the 2D ball used to generate the points $x$ is replaced by the projection of the 3D ball.

(d) Same question when we append a row after the last row of $A$, that is, $A$ is replaced with $\tilde{A} = [A^\top, 0]^\top \in \mathbb{R}^{3\times 2}$. Interpret geometrically your result.

**Solution:** Here we append a row after the last row of $A$, replacing $A$ with

$$\tilde{A} = \begin{pmatrix} A \\ 0 \end{pmatrix} \in \mathbb{R}^{3\times 2}.$$

The set $\mathcal{E}(\tilde{A})$ is the set of points of the form $(y, 0) \in \mathbb{R}^3$ where $y \in \mathcal{E}(A)$. This means that we are simply embedding the ellipsoid $\mathcal{E}(A)$ into a 3D space, instead of the original 2D one. The set $\mathcal{E}(\tilde{A})$ is now a degenerate (flat) ellipsoid in $\mathbb{R}^3$, entirely contained in the plane orthogonal to the third coordinate.

## 3. PCA and low-rank compression

We are given a $m \times n$ matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, with $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$ being the data points. (Note that this is the transpose of the $n \times m$ data matrix seen in the previous exercise.) We assume that the data matrix is centered, in the sense that $\mathbf{x_1} + \ldots + \mathbf{x_n} = \mathbf{0}$. In lecture, it was asserted that there is equivalence between three problems:

($P_1$) Finding a line going through the origin that maximizes the variance of the points projected on the line.

($P_2$) Finding a line going through the origin that minimizes the sum of squares of the distances from the points to their projections;

($P_3$) Finding a rank-one approximation to the data matrix.

In this exercise, you are asked to show the equivalence between these three problems.

(a) Consider the problem of projecting a point $\mathbf{x}$ on a line $\mathcal{L} = \{\mathbf{x}_0 + v\mathbf{u} : v \in \mathbb{R}\}$, with $\mathbf{x}_0 \in \mathbb{R}^m$, $\mathbf{u}^T\mathbf{u} = 1$, given.
Show that the projected point $\mathbf{z}$ is given by

$$\mathbf{z} = \mathbf{x}_0 + v^*\mathbf{u},$$

where we define

$$v^* = (\mathbf{x} - \mathbf{x}_0)^\top \mathbf{u},$$

and that the minimal squared distance $\|\mathbf{z} - \mathbf{x}\|_2^2$ is equal to $\|\mathbf{x} - \mathbf{x}_0\|_2^2 - ((x - x_0)^T u)^2$.
**Solution:** The projection of point $\mathbf{x}$ on $\mathcal{L}$ corresponds to the following problem:

$$v^* = \min_v \|\mathbf{x}_0 + v\mathbf{u} - \mathbf{x}\|_2.$$

The squared objective writes

$$\|\mathbf{x}_0 + v\mathbf{u} - \mathbf{x}\|_2^2 = v^2 - 2v(\mathbf{x} - \mathbf{x}_0)^\top \mathbf{u} + \|\mathbf{x} - \mathbf{x}_0\|_2^2,$$

which proves that the optimal value of $v$ is

$$v^* = (\mathbf{x} - \mathbf{x}_0)^\top \mathbf{u}.$$

At optimum, the squared objective function, which equals the minimum squared distance $\|\mathbf{z} - \mathbf{x}\|_2^2$, takes the desired value:

$$\|\mathbf{x}_0 + v^*\mathbf{u} - \mathbf{x}\|_2^2 = \|\mathbf{x} - \mathbf{x}_0\|_2^2 - ((\mathbf{x} - \mathbf{x}_0)^\top \mathbf{u})^2.$$

(b) Show that problems $P_1, P_2$ are equivalent.
**Solution:** $P_2$ minimizes the sum of squared distances from the points to their projections on a line passing through the origin $\mathcal{L} = \{v\mathbf{u} : v \in \mathbb{R}\}$. This problem can be written as:

$$\min_{\mathbf{u} \,:\, \mathbf{u}^\top \mathbf{u} = 1} \sum_{i=1}^{n} \min_{v_i} \|\mathbf{x}_i - v_i\mathbf{u}\|_2^2 \tag{1}$$

where $v_i = \mathbf{x}_i^\top \mathbf{u}$, as defined in the previous part (note that $\mathbf{x}_0 = 0$ because $\mathcal{L}$ passes through the origin). From the previous part, we obtain the equivalent form:

$$\begin{aligned}
&\min_{\mathbf{u}\,:\,\mathbf{u}^\top\mathbf{u}=1} \sum_{i=1}^n \|\mathbf{x}_i\|_2^2 - (\mathbf{x}_i^\top \mathbf{u})^2 \\
=&\min_{\mathbf{u}\,:\,\mathbf{u}^\top\mathbf{u}=1} \sum_{i=1}^n -(\mathbf{u}^\top \mathbf{x}_i)(\mathbf{x}_i^\top \mathbf{u}) \\
=&\max_{\mathbf{u}\,:\,\mathbf{u}^\top\mathbf{u}=1} \sum_{i=1}^n \mathbf{u}^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{u}
\end{aligned}$$

This problem can be written as a variance maximization problem:

$$\max_{\mathbf{u}\,:\,\mathbf{u}^\top\mathbf{u}=1} \mathbf{u}^\top C \mathbf{u},$$

where $C := (1/n)\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$ is the covariance matrix associated with the centered data. The above is exactly $P_1$, which maximizes the variance of points projected on a line.

(c) Show that $P_3$ is equivalent to $P_1$.

*Hint:* Show that the data matrix is rank-one if and only if it can be expressed as the outer product of two vectors. Recall that $P_3$ involves minimization of a Frobenius norm, and try to use some properties of this norm.

**Solution:** Assume that all the points are on a line going through the origin: this means that there exists $u \in \mathbb{R}^m$, with $\mathbf{u}^\top \mathbf{u} = 1$, such that, for every $i$, there exists a scalar $v_i$ such that

$$\mathbf{x}_i = v_i \mathbf{u}.$$

This means that the data matrix is rank-one:

$$X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] = [v_1\mathbf{u}, \ldots, v_n\mathbf{u}] = \mathbf{u}\mathbf{v}^\top,$$

with $\mathbf{v}^\top = [v_1, \ldots, v_n]$.

Now $P_3$ is expressed as minimizing the Frobenius norm of the matrix $A = X - \mathbf{u}v^\top$, over $\mathbf{u}, \mathbf{v}$. Since only the term $\mathbf{u}\mathbf{v}^\top$ counts, we can always impose $\mathbf{u}^\top\mathbf{u} = 1$. With $\mathbf{a} = [a_1, \ldots, a_n]$, where $a_i \in \mathbb{R}^m$, $P_3$ can be written as:

$$\begin{aligned}
&\min_{A,\mathbf{v},\mathbf{u}} \sum_{i=1}^n \|a_i\|_2^2 \;:\; \mathbf{x}_i = v_i\mathbf{u} + a_i, \mathbf{u}^\top\mathbf{u} = 1 \\
=&\min_{\mathbf{v},\mathbf{u}} \sum_{i=1}^n \|\mathbf{x}_i - v_i\mathbf{u}\|_2^2 \;:\; \mathbf{u}^\top\mathbf{u} = 1
\end{aligned}$$

which is exactly $P_2$, as given by (1).

Bonus. Find the rank-one approximation of the entire EECS127/227AT material. What are the subjects that are the most likely to be at the finals?

**Solution:** A rank-one approximation of the entire EECS127/227AT material is "least-squares". But the total variance of the EECS127/227AT material explained by the first component of the principal component analysis is only 10%. So you should better know all the class material.

## 4. PCA and face analysis

We have seen how to represent images as matrices. Similarly, we may represent an image as a vector. In this exercise we will explore analyzing a dataset of images represented as vectors.

We will analyze a dataset of human faces to see if we can learn anything meaningful from it. One way to analyze a dataset is to look for directions of maximal variation. That is, in the space of the data we look for the directions in which the data samples change value the most.

Let $\mathbf{x}_i \in \mathbb{R}^m, i = 1, \ldots, n$, be the given data points to analyze, denote $\hat{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$, the mean of the data points with $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \hat{\mathbf{x}}$, the centered datapoint. Let

$$\tilde{X} = \begin{bmatrix} \leftarrow \tilde{\mathbf{x}}_1^\top \rightarrow \\ \leftarrow \tilde{\mathbf{x}}_2^\top \rightarrow \\ \vdots \\ \leftarrow \tilde{\mathbf{x}}_n^\top \rightarrow \end{bmatrix} \in \mathbb{R}^{n \times m}$$

be the matrix such that the $i$th row is $\tilde{\mathbf{x}}_i^\top$.
Then the components of the centered data along a direction (by direction, we mean unit norm vector) $\mathbf{z}$ are given by:

$$\alpha_i = \tilde{\mathbf{x}}_i^\top \mathbf{z}, \;\; i = 1, \ldots, n$$

The mean square variation of the data along direction $\mathbf{z}$ is given by

$$\frac{1}{n}\sum_{i=1}^{n} \alpha_i^2 = \frac{1}{n}\sum_{i=1}^{n} \mathbf{z}^\top \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top \mathbf{z} = \frac{1}{n}\mathbf{z}^\top \tilde{X}^\top \tilde{X}\mathbf{z}$$

The direction $\mathbf{z}$ along which the data has the largest variation can thus be found as the solution to the following optimization problem:

$$\max_{\substack{\mathbf{z}\in\mathbb{R}^m \\ \text{s.t. } ||\mathbf{z}||_2=1}} \mathbf{z}^\top \tilde{X}^\top \tilde{X}\mathbf{z}$$

(a) Show that the direction of maximal variation is the direction of the eigenvector corresponding to the largest eigenvalue of the matrix $\tilde{X}^\top \tilde{X}$.

**Solution:**

We may start by recalling that the Singular Value Decomposition (SVD) of $\tilde{X} = U\Sigma V^\top$. As such we may re-write $\tilde{X}^\top \tilde{X} = V\Sigma^2 V^\top$.

Consequently, we may re-write the problem as:

$$\max_{\substack{z\in\mathbb{R}^m \\ \text{s.t. } ||\mathbf{z}||_2=1 \\ \mathbf{z}'=V^\top\mathbf{z}}} \mathbf{z}'^\top \Sigma^2 \mathbf{z}'$$

Note that, however since $||A\mathbf{z}||_2 = ||\mathbf{z}||_2$ for any appropriately shaped orthogonal matrix $A$, then we may further re-write the problem as:

$$\max_{\substack{\mathbf{z}'\in\mathbb{R}^m \\ \text{s.t. } ||\mathbf{z}'||_2=1}} \mathbf{z}'^\top \Sigma^2 \mathbf{z}'$$

Note that $\sigma_{\min}^2||\mathbf{z}'||_2^2 \leq \mathbf{z}'^\top \Sigma^2 \mathbf{z}' = \sum_{i=1}^m \sigma_i^2 \mathbf{z}'^2{}_i \leq \sigma_{\max}^2||\mathbf{z}'||_2^2$.
In particular by restricting $||\mathbf{z}'||_2 = 1$, we can attain both sides of the bounds by choosing $\mathbf{z}$ to be the right singular vectors associated with the minimum and maximum singular values of $\tilde{X}$, respectively.
Note that this problem may also be solved using a spectral factorization of $\tilde{X}^\top \tilde{X}$.
The solution to the problem is the unit-normalized eigenvector of $\tilde{X}^\top \tilde{X}$ corresponding to the largest eigenvalue of $\tilde{X}^\top \tilde{X}$, or equivalently the right singular vector of $\tilde{X}$ corresponding to the largest singular value of $\tilde{X}$.

(b) How is this related to the singular value decomposition of $\tilde{X}$?

**Solution:**
Based on the solution to the previous problem, the optimal solution is attained by choosing $\mathbf{z}$ as the right singular vector associated with the maximum singular value of $\tilde{X}$.

(c) Suppose we wish to find the direction with the second-largest variance, and that $v_1$ is the direction of maximal variation. We can construct the following transformation to the data points:

$$\tilde{\mathbf{x}}_i^{(1)} = \tilde{\mathbf{x}}_i - \mathbf{v}_1(\mathbf{v}_1^\top \tilde{\mathbf{x}}_i) : i = 1, \ldots, n.$$

This transformation simply removes the components along the direction of maximal variation from the data points.
We can then define the matrix

$$\tilde{X}^{(1)} = \begin{bmatrix} \leftarrow \tilde{\mathbf{x}}_1^{(1)\top} \rightarrow \\ \leftarrow \tilde{\mathbf{x}}_2^{(1)\top} \rightarrow \\ \vdots \\ \leftarrow \tilde{\mathbf{x}}_n^{(1)\top} \rightarrow \end{bmatrix} \in \mathbb{R}^{n \times m}$$

The direction of second largest variation can then be found by solving the optimization problem:

$$\max_{\substack{\mathbf{z} \in \mathbb{R}^m \\ \text{s.t. } ||\mathbf{z}||_2 = 1}} \mathbf{z}\tilde{X}^{(1)\top}\tilde{X}^{(1)}\mathbf{z}$$

Show that the direction with the second largest variation is the direction of the eigenvector corresponding to the second largest eigenvalue of the matrix $\tilde{X}^\top \tilde{X}$ (Solutions that show this by doing the algebra are preffered).
We can repeat the procedure to find more directions of variation. We will use this procedure to find the directions of maximal variation in our dataset of images.

**Solution:**

Notice that $\tilde{X}^{(1)} = \tilde{X} - \tilde{X}\mathbf{v}_1\mathbf{v}_1^\top$.
We may re-write this as $\tilde{X}^{(1)} = \tilde{X}(I - \mathbf{v}_1\mathbf{v}_1^\top)$.

Given the singular value decomposition of $\tilde{X} = U\Sigma V^\top$, we may write

$$\begin{aligned}
\tilde{X}^{(1)} &= U\Sigma V^\top (I - \mathbf{v}_1\mathbf{v}_1^\top) \\
&= \sum_{i=1}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top - \sum_{i=1}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \mathbf{v}_1 \mathbf{v}_1^\top \\
&= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top - \sigma_1 \mathbf{u}_1 ||\mathbf{v}_1||_2^2 \mathbf{v}_1^\top + \sum_{i=2}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top - 0 \\
&= \sum_{i=2}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top
\end{aligned}$$

Consequently,

$$\begin{aligned}
\tilde{X}^{(1)\top}\tilde{X}^{(1)} &= (\sum_{j=2}^{m} \sigma_j \mathbf{v}_j \mathbf{u}_j^\top)(\sum_{i=2}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top) \\
&= \sum_{j=2}^{m}\sum_{i=2}^{m} \sigma_j \sigma_i \mathbf{v}_j \mathbf{u}_j^\top \mathbf{u}_i \mathbf{v}_i^\top \\
&= \sum_{j=2}^{m}\sum_{i=2}^{m} \sigma_i^2 \delta_{ij} \mathbf{v}_j \mathbf{v}_i^\top \qquad \text{where } \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases} \\
&= \sum_{i=2}^{m} \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^\top
\end{aligned}$$

Where the last two equalities are derived from the orthogonality of the singular vectors. This is similar to the decomposition for $\tilde{X}^\top\tilde{X}$, except that we no longer have $\mathbf{v}_1$. Equivalently, one can view it has $\mathbf{v}_1$ corresponding to a singular value of zero.

As a result, we have that $\mathbf{z}^\top \tilde{X}^{(1)\top}\tilde{X}^{(1)} \leq \sigma_2^2 ||\mathbf{z}||_2^2$, where the singular values are arranged in descending order. This bound can be attained be choosing $\mathbf{z}$ to be the right singular vector of $\tilde{X}$ corresponding to its second largest singular value or equivalently, the eigenvector of $\tilde{X}^\top\tilde{X}$ corresponding to the second largest eigenvalue.

(d) Loading images: In the IPython notebook for this assignment, complete the function loadImage. The function cv2.imread(filename), takes in a file name and returns the image at the file location as a matrix. Complete the function so that the matrix becomes a vector. (Hint: You may use numpy's flatten)

Remember to `conda activate ee127` before starting!

**Solution:**
See IPython notebook

(e) Finding directions of largest variation: We will not make you write the code to solve the optimization problem. Rather, this time we will make use of `sklearn.decomposition.PCA` to obtain these directions. Use the `PCA` object's `fit` method to run SVD, then complete the IPython code to obtain the mean of $X$ and the first 15 directions of largest variation in the dataset using the `PCA` object. What do you expect these directions to look like intuitively?

**Solution:**
See IPython notebook.

(f) Visualizing directions of maximal variation: We will now try to see what each of the directions looks like. To do this, we will take an image and vary it along these directions, one at a time, visualizing what it does to the image. An image has been pre-selected for you. Complete the line of code to reshape the image from a vector representation to a matrix. (Hint: You may use numpy's reshape)

**Solution:**
See IPython notebook.

(g) Each of the sliders represents a direction of variation with the first slider being the direction with the largest variation, the next being the second largest and so on. Vary the sliders one at a time, while keeping the other sliders fixed, observing what happens to the pre-selected image. What about the image does the direction of maximal variation change? Does this seem reasonable to you, and why? Take two screenshots of you varying the direction of maximal variation (first slider) only. One screenshot with the slider all the way to the left and another screenshot with the slider all the way to the right, with other sliders held fixed. You may also experiment with creating new images by varying the different sliders. Have fun creating new faces! **Solution:**
See IPython notebook. Direction of maximal variation should seem to change the lighting of the background — any solution that varies the background lighting while the other features are fixed is acceptable.

# Note: Use the escape key to close the windows, otherwise the code will get stuck in a loop

In [1]:
```
# import statements
import glob
import numpy as np
import cv2
import pdb
import time
from sklearn.decomposition import PCA
from sklearn.neighbors import KDTree
import sys
```

In [2]:
```
def read_images(img_dir):
    # read files in directory and add image to list
    all_images = []
    if not img_dir.endswith("/"):
        img_dir = img_dir + "/"
    filenames = glob.glob(img_dir + "*.jpg")
    tmp_file = filenames[0]
    tmp_img = cv2.imread(tmp_file)
    img_shape = tmp_img.shape
    data_matrix = np.empty((500, img_shape[0]*img_shape[1]*img_shape[2
]))
    count = 0
    for file in filenames[0:500]:
        tmp_img = cv2.imread(file)
        all_images.append(tmp_img)

        # d.)
        # You complete this part ----------------------------------
--------------->
        # You should reshape tmp_img to a vector and assign it to resh
aped_img
        # reshaped_img = ??
        reshaped_img = tmp_img.flatten()
        data_matrix[count, :] = reshaped_img
        count = count + 1


    return all_images, data_matrix

def create_data_matrix(images):
    # reshape each matrix and add to array
```

```
    data_matrix = []
    for image in images:
        tmp_image = image.flatten()[np.newaxis, :]
        data_matrix.append(tmp_image)


    data_matrix = np.concatenate(data_matrix, axis=0)
    return data_matrix

def get_closest_face(data_matrix, query_face):
    # data_matrix : matrix of all images, after transformed using dire
ctions of maximal variation
    # query_face : vector of query image, after transformed using dire
ctions of maximal variation
    # returns index of closest face in data set
    kdt = KDTree(data_matrix, leaf_size=30, metric='euclidean')
    idx = kdt.query(query_face, k=1, return_distance=False)

    return idx
```

In [3]:

```
IMG_DIR =  '../data/img_align_celeba/'
N_EIGEN_FACES = 15
MAX_SLIDER_VALUE = 255

all_images, data_matrix = read_images(IMG_DIR)
IMG_SHAPE = all_images[0].shape


# do PCA analysis
print("Doing PCA analysis ...")
start_time = time.time()


pca = PCA(n_components=N_EIGEN_FACES)
# e.)
# You complete this ----------------------------------------------
------>
# fit pca object with data matrix
pca.fit(data_matrix)

# You complete this ----------------------------------------------
------>
# get pca mean
mean = pca.mean_

# You complete this ----------------------------------------------
------>
# return list of directions of maximal variation (in order), Hint: loo
k at pca.components
```

```
# eigenvectors is a list of directions of maximal variation
# eigenvectors = ??
eigenvectors = pca.components_.tolist()

# we'll process this list for you
eigenvectors = [np.asarray(eigenvectors[i]) for i in range(len(eigenve
ctors))]
end_time = time.time()
print("Done!")
print("Duration: ", end_time - start_time)

mean_face = mean.reshape(IMG_SHAPE)
mean_face = np.asarray(mean_face, dtype=np.uint8)

slider_values = []
eigen_faces = []
for eigenvector in eigenvectors:
    # f.)
    # You complete this ------------------------------------------
---------------->
    # tmp_face is eigenvector reshaped to an image
    # complete code to reshape
    # tmp_face = ??
    tmp_face = eigenvector.reshape(IMG_SHAPE)

    # we'll handle the rest for you :)
    eigen_faces.append(tmp_face)

def make_face(*args):
    new_face = mean_face
    for i in range(N_EIGEN_FACES):
        slider_values[i] = cv2.getTrackbarPos("Weight" + str(i), "Trac
kbars")
        weight = slider_values[i] - MAX_SLIDER_VALUE/2
        new_face = new_face + eigen_faces[i]*weight*100
        new_face = np.maximum(np.minimum(new_face, 255),0)
        new_face = np.asarray(new_face, dtype=np.uint8)


    new_face = cv2.resize(new_face, (0,0), fx=2, fy=2)

    cv2.imshow("Demo face", new_face)
#-------------------------------------------------------------------
--------------------

cv2.namedWindow("Demo face", cv2.WINDOW_AUTOSIZE)
output = cv2.resize(mean_face, (0,0), fx=2, fy=2)
cv2.imshow( "Demo face", output)
```

```
cv2.namedWindow("Trackbars", cv2.WINDOW_AUTOSIZE)

for i in range(N_EIGEN_FACES):
    slider_values.append(int(MAX_SLIDER_VALUE/2))
    cv2.createTrackbar("Weight" + str(i), "Trackbars", int(MAX_SLIDER_
VALUE/2), MAX_SLIDER_VALUE, make_face)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Doing PCA analysis ...
Done!
Duration:  2.8001232147216797
```

In [ ]:

**5. Eigenvectors of a symmetric matrix**

Let $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$ be two linearly independent vectors, with unit norm ($\|\mathbf{p}\|_2 = \|\mathbf{q}\|_2 = 1$). Define the symmetric matrix $A \doteq \mathbf{p}\mathbf{q}^\top + \mathbf{q}\mathbf{p}^\top$. In your derivations, it may be useful to use the notation $c \doteq \mathbf{p}^\top \mathbf{q}$.

(a) Show that $\mathbf{p}+\mathbf{q}$ and $\mathbf{p}-\mathbf{q}$ are eigenvectors of $A$, and determine the corresponding eigenvalues.
**Solution:** We have

$$A\mathbf{p} = c\mathbf{p} + \mathbf{q}, \quad A\mathbf{q} = \mathbf{p} + c\mathbf{q},$$

from which we obtain

$$A(\mathbf{p}-\mathbf{q}) = (c-1)(\mathbf{p}-\mathbf{q}), \quad A(\mathbf{p}+\mathbf{q}) = (c+1)(\mathbf{p}+\mathbf{q}).$$

Thus $\mathbf{u}_\pm := \mathbf{p} \pm \mathbf{q}$ is an (un-normalized) eigenvector of $A$, with eigenvalue $c \pm 1$.

(b) Determine the nullspace and rank of $A$.
**Solution:** If $\mathbf{x} \in \mathbb{R}^n$ is in the nullspace of $A$ we must have: $A\mathbf{x} = 0$.

$$0 = A\mathbf{x} = \mathbf{p}(\mathbf{q}^\top \mathbf{x}) + \mathbf{q}(\mathbf{p}^\top \mathbf{x}).$$

Since $(\mathbf{q}^\top \mathbf{x})$ and $(\mathbf{p}^\top \mathbf{x})$ are scalars we can rewrite this as:

$$0 = A\mathbf{x} = (\mathbf{q}^\top \mathbf{x})\mathbf{p} + (\mathbf{p}^\top \mathbf{x})\mathbf{q} = 0.$$

However, since $\mathbf{p}, \mathbf{q}$ are linearly independent, the fact that a linear combination of $\mathbf{p}, \mathbf{q}$ is zero implies that $\mathbf{p}^\top \mathbf{x} = \mathbf{q}^\top \mathbf{x} = 0$. Hence, the nullspace of $A$ is the set of vectors orthogonal to $\mathbf{p}$ and $\mathbf{q}$.
Since $A\mathbf{x}$ gives us linear combinations of $A$, the range of $A$ is the span of $\mathbf{p}, \mathbf{q}$. The rank is thus 2.

(c) Find an eigenvalue decomposition of $A$, in terms of $\mathbf{p}, \mathbf{q}$. *Hint:* use the previous two parts.
**Solution:** Since the rank is 2, we need to find a total of two non-zero eigenvalues.

First, we check that that $\lambda = c \pm 1$ is not 0. Note that, since $\mathbf{p}, \mathbf{q}$ are normalized, we can interpret $c = \mathbf{p}^\top \mathbf{q}$ as projecting a unit vector onto another unit vector. This implies $|c| \neq 1$ because $\mathbf{p}$ and $\mathbf{q}$ are linearly independent, i.e., they are not aligned. Thus $c \pm 1 \neq 0$.

Thus, we have found two linearly independent eigenvectors $\mathbf{u}_\pm = \mathbf{p} \pm \mathbf{q}$ that do not belong to the nullspace. Then, the eigenvalue decomposition is

$$A = (c-1)\mathbf{v}_-\mathbf{v}_-^\top + (c+1)\mathbf{v}_+\mathbf{v}_+^\top,$$

where $\mathbf{v}_\pm$ are the normalized vectors $\mathbf{v}_\pm = \mathbf{u}_\pm / \|\mathbf{u}_\pm\|_2$. Since

$$\|\mathbf{p} \pm \mathbf{q}\|_2^2 = \mathbf{p}^\top \mathbf{p} \pm 2\mathbf{p}^\top \mathbf{q} + \mathbf{q}^\top \mathbf{q} = 2(1 \pm c),$$

we have

$$\mathbf{v}_\pm = \frac{1}{\sqrt{2(1 \pm c)}}(\mathbf{p} \pm \mathbf{q}),$$

so that the eigenvalue decomposition becomes

$$A = \frac{1}{2}\left((\mathbf{p}+\mathbf{q})(\mathbf{p}+\mathbf{q})^\top - (\mathbf{p}-\mathbf{q})(\mathbf{p}-\mathbf{q})^\top\right).$$

(d) What is the answer to the previous part if $\mathbf{p}, \mathbf{q}$ are not normalized? Write $A$ as a function $\mathbf{p}, \mathbf{q}$ and their norms and the new eigenvalues as a function of $\mathbf{p}, \mathbf{q}$ and their norms.

**Solution:** We can always scale the matrix: with $\bar{\mathbf{p}} = \mathbf{p}/\|\mathbf{p}\|_2$, $\bar{\mathbf{q}} = \mathbf{q}/\|\mathbf{q}\|_2$, and $\bar{A} = \bar{\mathbf{p}}\bar{\mathbf{q}}^\top + \bar{\mathbf{q}}\bar{\mathbf{p}}^\top$, we have

$$A = \|\mathbf{p}\|_2\|\mathbf{q}\|_2\bar{A}.$$

Since $A$ is just a scaled version of $\bar{A}$, whose eigenvalues we determined in previous parts, the eigenvalues of $A$ are also simply scaled accordingly: with $c = \bar{\mathbf{p}}^\top\bar{\mathbf{q}}$,

$$\lambda_\pm = \|\mathbf{p}\|_2\|\mathbf{q}\|_2(c \pm 1) = \mathbf{p}^\top\mathbf{q} \pm \|\mathbf{p}\|_2\|\mathbf{q}\|_2.$$

Note that, since $\mathbf{p}, \mathbf{q}$ are independent, none of these eigenvalues is zero. In fact, one is positive, and the other negative. This is due to the Cauchy-Schwartz inequality, which says that $|\mathbf{p}^\top\mathbf{q}| \leq \|\mathbf{p}\|_2\|\mathbf{q}\|_2$, with equality if and only if $\mathbf{p}, \mathbf{q}$ are linearly dependent.

The corresponding (un-normalized) eigenvectors are $\bar{\mathbf{p}} \pm \bar{\mathbf{q}}$. The eigenvalue decomposition obtained before leads to

$$A = \|\mathbf{p}\|_2\|\mathbf{q}\|_2\bar{A} = \frac{\|\mathbf{p}\|_2\|\mathbf{q}\|_2}{2}\left((\bar{\mathbf{p}} + \bar{\mathbf{q}})(\bar{\mathbf{p}} + \bar{\mathbf{q}})^\top - (\bar{\mathbf{p}} - \bar{\mathbf{q}})(\bar{\mathbf{p}} - \bar{\mathbf{q}})^\top\right).$$