Congratulations! You passed!

Grade received 88.89% **To pass** 80% or higher

Go to next item

1. In this quiz, we bring together all the concepts covered in this module and outline each of the steps involved in PCA.

1 / 1 point

Here is an outline of the steps we will be taking:

- Step 1: Normalize the data
- Step 2: Construct the covariance matrix of the data
- Step 3: Calculate the eigenvector and eigenvalues of the covariance matrix. We call the eigenvector with the largest eigenvalue the principal component eigenvector
- Step 4: Find the new data of reduced dimensionality

Before we go through each of the steps individually, we will first consider how to fit a line to some data. Let's consider two different ways of calculating a line of best fit.

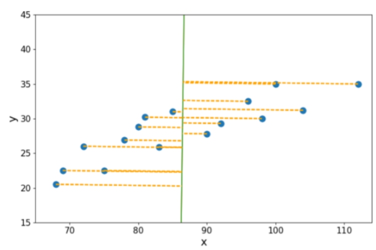
The line of best fit is either

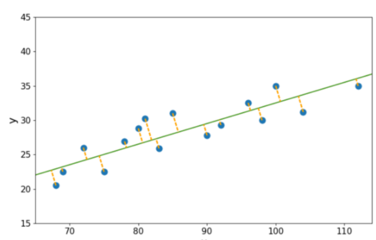
1. a line which maximises the spread of the data points when projected onto this line
or
2. a line which minimises the sum of squared distances between the original data points and the projected data points.

In fact, these two descriptions have exactly the same results.

To demonstrate this, look at the following two figures which are plots of y as a function of x. y and x represent two different variables. The above plot shows a "bad" fit since the spread of the data points when projected onto the fitted line is much less than that of the bottom plot which shows a "good" fit. At the same time, the sum of squared distances of the "bad" fit is much greater than that of the "good" fit.

The orange dotted lines in the plot below show the orthogonal projection of each of the data points onto the fitted line.





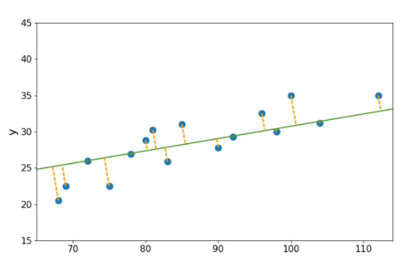
Test this out for yourself using the code cell below by changing the value for "m" which represents the gradient of the line of best fit for the data, and then run the cell.

```
plt.plot(x, y, 'o',markersize=10)
m = 0.17        #Change this value
plot_line(m)
```

Run

Reset

Sum of squared distances: 96.8396284867
Spread of data: 40.942593312197516



What value of `m` (gradient) gives a sum of squared distance value of less than 100 (give your answer to 2dp)?

0.17

Correct
Well done! This value of m gives a sum of squared distance value of less than 100.

2. Before we start the first step of PCA, let's also recall some important definitions of some of the mathematical concepts will be using throughout the process.

1 / 1 point

A measure of the correlation between different variables is given by the:

- ○ Projection matrix
- ◉ Covariance matrix
- ○ Eigenvalues

Correct
Correct!

3. The directions in which data is spread out is given by the:

1 / 1 point

- ◉ Eigenvectors
- ○ Covariance matrix
- ○ Eigenvalues

Correct
Correct!

4. The magnitude of the spread of data along the principal components is given by the:

1 / 1 point

- ○ Determinant of the covariance matrix
- ○ Inner product
- ◉ Eigenvalues

Correct
Correct!

5. Now you will normalize the data from question 1.

1 / 1 point

Normalizing data generally involves two steps:

1. subtracting the mean from the data

2. dividing the data by the standard deviation

In some implementations, both steps are essential, however in this case we will ignore this second step for reasons beyond the scope of this quiz.

However, it is still important to center the data by subtracting the mean. If this initial step is missed, the 1st principal component may not fit the data along the main direction, and will not be reflective of the correlations and spread in the data.

The data for the plots is stored in the 2D array `data`, with the first column `data[:,0]` representing the first variables x, and the second column `data[:,1]` representing the second variables y. You will calculate a new array `data_normalized`, which contains the data minus the mean of each variable.

Edit the code below to normalize (center) the data, and run the cell to check if the plot produced is what you expect. You may wish to use `np.mean`; see https://numpy.org/doc/stable/reference/generated/numpy.mean.html .

The plot shows the normalized data when it is calculated. Check that you get what you would expect before completing the rest of the exercise.

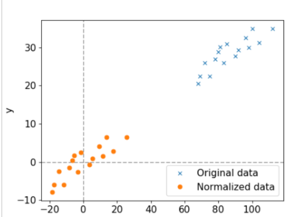
The dotted lines shown in the figure produced by the below code block mark the x and y axis.

```
# Change the line below to subtract the mean from the data
# You can use NumPy functions to help you compute the results

# `data` is a 2D NumPy array with shape (N, D)
# where N is the number of samples and D is the dimensionality.
# You should change the right hand side of the statement below to compute the
# normalized dataset.
data_normalized = data
mean = np.mean(data, axis = 0)
data_normalized = (data_normalized - mean)
plot(data_normalized)
```

Run

Reset



When you are happy that your code calculates `data_normalized` correctly, copy the code for the calculation into the cell below and run the cell to submit your answer.

```
# Copy your code from above to calculate data_normalized correctly
data_normalized = data
mean = np.mean(data, axis = 0)
data_normalized = (data_normalized - mean)
data_normalized
```

Run

Reset

```
[[ 25.5625    6.55625]
 [-17.4375   -5.94375]
 [ -3.4375   -2.54375]
 [ -8.4375   -1.54375]
 [  3.5625   -0.64375]
 [ -6.4375    0.35625]
 [  5.5625    0.85625]
 [ -5.4375    1.75625]
 [-18.4375   -7.94375]
 [ 17.5625    2.75625]
 [ 13.5625    6.55625]
 [  9.5625    4.05625]
 [ -1.4375    2.55625]
 [ 11.5625    1.55625]
 [-11.4375   -5.94375]
 [-14.4375   -2.44375]]
```

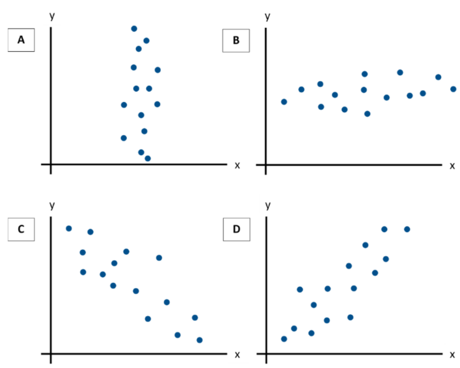
Correct

Good job!

6. Before we move onto the next step where we construct the covariance matrix, let's recall the main features of a covariance matrix.

1 / 1 point

Match the plots with the corresponding covariance matrix - note that you do not need to calculate anything; use what you know about features of a covariance matrix and what it represents to match the following plots to the corresponding covariance matrix.



- (selected) $\mathbf{A}=\begin{bmatrix}2 & 0\\ 0 & 7\end{bmatrix}, \mathbf{B}=\begin{bmatrix}6 & 0\\ 0 & 3\end{bmatrix}, \mathbf{C}=\begin{bmatrix}4 & -2\\ -2 & 5\end{bmatrix}, \mathbf{D}=\begin{bmatrix}1 & 0.5\\ 0.5 & 3\end{bmatrix}$
- $\mathbf{A}=\begin{bmatrix}4 & -2\\ -2 & 5\end{bmatrix}, \mathbf{B}=\begin{bmatrix}1 & 0.5\\ 0.5 & 3\end{bmatrix}, \mathbf{C}=\begin{bmatrix}6 & 0\\ 0 & 3\end{bmatrix}, \mathbf{D}=\begin{bmatrix}2 & 0\\ 0 & 7\end{bmatrix}$
- $\mathbf{A}=\begin{bmatrix}2 & 0\\ 0 & 7\end{bmatrix}, \mathbf{B}=\begin{bmatrix}6 & 0\\ 0 & 3\end{bmatrix}, \mathbf{C}=\begin{bmatrix}1 & 0.5\\ 0.5 & 3\end{bmatrix}, \mathbf{D}=\begin{bmatrix}4 & -2\\ -2 & 5\end{bmatrix}$
- $\mathbf{A}=\begin{bmatrix}6 & 0\\ 0 & 3\end{bmatrix}, \mathbf{B}=\begin{bmatrix}1 & 0.5\\ 0.5 & 3\end{bmatrix}, \mathbf{C}=\begin{bmatrix}2 & 0\\ 0 & 7\end{bmatrix}, \mathbf{D}=\begin{bmatrix}4 & -2\\ -2 & 5\end{bmatrix}$
- $\mathbf{A}=\begin{bmatrix}6 & 0\\ 0 & 3\end{bmatrix}, \mathbf{B}=\begin{bmatrix}2 & 0\\ 0 & 7\end{bmatrix}, \mathbf{C}=\begin{bmatrix}4 & -2\\ -2 & 5\end{bmatrix}, \mathbf{D}=\begin{bmatrix}1 & 0.5\\ 0.5 & 3\end{bmatrix}$

Correct

Well done!

7. Now let's move onto the next step, where you will construct the covariance matrix for our data. Recall the following formula for an element of the covariance matrix:

0 / 1 point

$$\mathrm{cov}[x,y] = E[(x-\mu_x)(y-\mu_y)] \approx \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)$$

Here, the subscript i refers to the ith entry of a given variable. μ_x represents the mean of x and μ_y represents the mean of y. N represents the number of data points which for our data set is 16.

The covariance matrix itself is given by:

$$A = \begin{bmatrix}\mathrm{cov}[x,x] & \mathrm{cov}[x,y]\\ \mathrm{cov}[y,x] & \mathrm{cov}[y,y]\end{bmatrix}.$$

You will calculate the covariance matrix for the data in the previous questions in the code cell below. Remember that `data_normalized[i,0]` refers to the normalized ith element in x, given by $(x_i-\mu_x)$. Similarly, `data_normalized[i,1]` refers to the normalized ith element in y, given by $(y_i-\mu_y)$. Hence `data_normalized[:,0]` is a 2D array (column vector) containing all the normalized data for variable x, and similarly `data_normalized[:,1]` is a 2D array (column vector) containing all the normalized data for variable y. The variable `cov_xx` is equivalent to $\mathrm{cov}[x,x]$.

Complete the code cell by calculating the remaining elements of the covariance matrix. `cov_xy` has been provided as an example. When you are done, **directly** copy & paste the output of the code cell into the answer box below.

```python
# Complete the code below to compute cov[x, x] cov[y, x] and cov[y, y].
# cov_xy has been provided as an example
cov_xx = (1/N) * np.sum(data_normalized[:, 0] * data_normalized[:, 0])
cov_xy = (1/N) * np.sum(data_normalized[:, 0] * data_normalized[:, 1])
cov_yx = (1/N) * np.sum(data_normalized[:, 1] * data_normalized[:, 0])
cov_yy = (1/N) * np.sum(data_normalized[:, 1] * data_normalized[:, 1])

# This prints the final covariance matrix
# using the elements computed above
# Once you have finished writing the code above,
# DIRECTLY copy the output of the following function
# to the answer cell.
print_covariance_matrix()
```

Run

Reset

```
[[ 159.62109375   45.23710938]
 [  45.23710938   16.99496094]]
```

[[159.62109375, 45.23710938], [45.23710938, 16.99496094]]

Incorrect

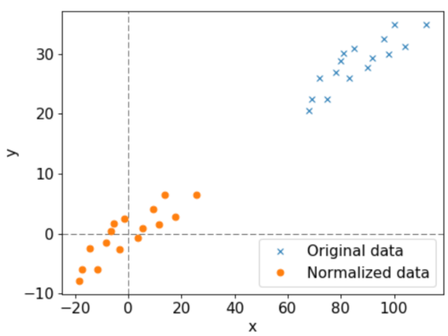
Try again! Use the `cov_xy` which has been done for you, as an example. Here, we essentially are taking the dot product of the centered data for x with the centered data for y and then divide by N. Similarly for the first element (`cov_xx`), we would need to take to dot product of the centered data for x with itself and the divide by N. We repeat this procedure for the remaining two elements. Remember that `cov_xy` is the same as `cov_yx`.

---

8. In the next step, we will be calculating the eigenvectors and eigenvalues of the covariance matrix that we found in the previous question.

1 / 1 point

Here is a plot of the normalized data which shows the correlation between x and y



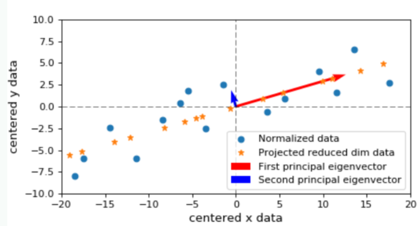
Using this, in which direction do you think the first principal component eigenvector will point?

- (•) Upwards and towards the right
- ( ) Upwards and towards the left
- ( ) Downwards and towards the right

Correct

Well done! The first principal component eigenvector points in the direction where the data is spread most.

Note that the length of the eigenvectors are the square root of the corresponding eigenvalue



9. Finally, we want to find the new lower dimensional representation of the data. Recall from lectures that the coordinates ($\mathbf{X}$) of the projection from 2D to 1D with respect to the orthonormal basis of the principal subspace is given by:

$$\mathbf{X} = \mathbf{B}^T \mathbf{x}_*,$$

1 / 1 point

where $\mathbf{B}$ is the basis vectors arranged as the columns of a matrix, and $\mathbf{x}_*$ is vector containing the normalized original data. In this case $\mathbf{B}$ is just the first principal eigenvector.

To project the reduced dimension coordinates back into the original 2D space, we use

$$\mathbf{B}\mathbf{B}^T \mathbf{x}_*$$

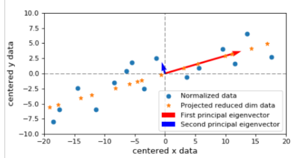
This final step has been done for you, where we combine everything we have found in the previous steps to plot the final projected coordinates.

Run the code cell below (you do not need to edit the code) to convince yourself that we have indeed reduced the dimensionality of the original data set from two to one. Note that the length of the eigenvectors are the square root of the corresponding eigenvalue.

```
plot_projected_coords()
```

Run

Reset



Congratulations, you have performed PCA for a small 2d dataset! In this week's assessment you will write PCA code more formally and apply it to higher dimensional datasets!

To wrap-up, which of the following do you think are real-world applications of PCA?

- [x] PCA saves memory on devices when storing data

Correct

Correct! Since PCA reduces the dimensionality of the dataset, we can compress the data since we only need to store the data of reduced dimensionality and can throw away the original data.

- [x] PCA can be used for image compression and facial recognition

Correct

Correct! PCA reduces dimensionality and hence the complexity of the data but highlights and summarises the main trends and patterns - in this way PCA picks up the main features of the data used to represent the picture

- [x] PCA makes it easy to spot patterns and trends in data

Correct

Correct! PCA reduces dimensionality and hence the complexity of the data but highlights and summarises the main trends and patterns

- [ ] PCA can capture non-linear relationships between variables