Advanced Analytics with PCA, Eigenvalues, Covariance matrix, MLE, ISOMAP, & Image recognition

Margaret Catherman

Spring 2024

Contents

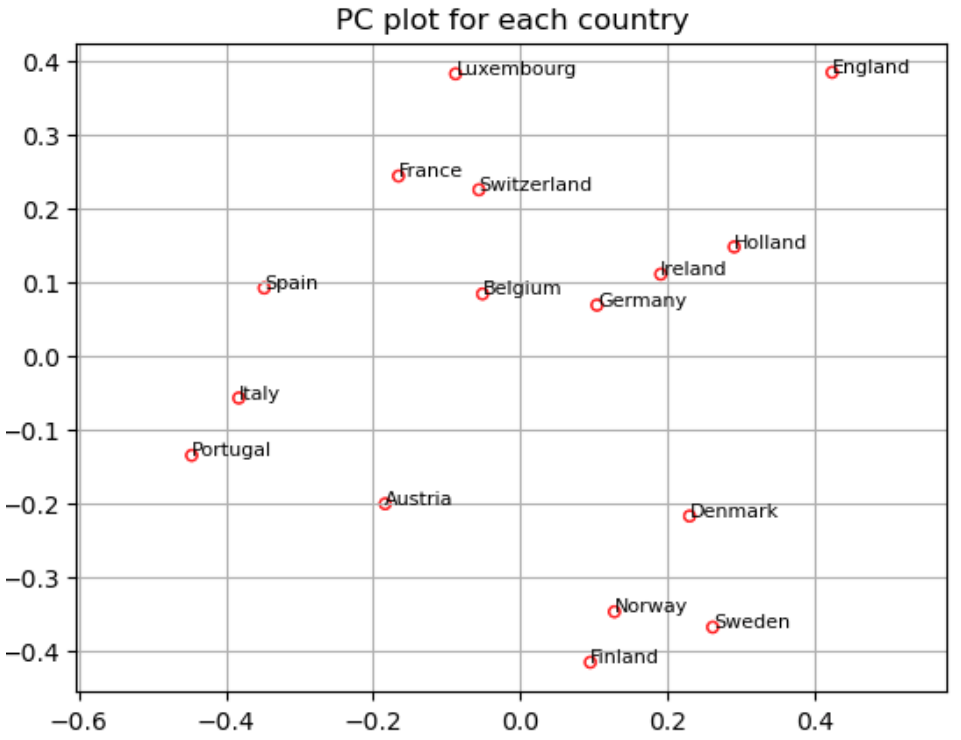
Part 1. Application of Concepts in Advanced Analytics

1. PCA: Food consumption in European countries

The data "food-consumption.csv" contains 16 countries in Europe and their consumption for 20 food items, such as tea, jam, coffee, yogurt, and others. We will perform principal component analysis, in two different ways, to explore the data. *Source code for this section, See /models/pca_of_food_consumption.py*

(a) PCA, approach A: For the first approach, we perform PCA on countries by treating *each country's food consumption as their "feature" vectors*. We will explain how the data matrix is set-up in this case (e.g., the columns and the rows of the matrix correspond to what). We will extract the first two principal components for each data point (thus, this means we will represent each data point using a two-dimensional vector), & draw a scatter plot of two-dimensional representations of the countries using their two principal components.

Observations: The data matrix is set-up in this case such that the matrix is of the form m x n, with the 16 countries as m or rows, and the 20 foods as n, or columns. Relative food consumption styles of each country are illustrated in the scatter plot chart, right.



Interestingly, food preferences by country appear to follow patterns somewhat related to a country's geographical proximity to another country, although there are some exceptions.

The map, above, helps to illustrate this theory.

Examples of countries with similar food preferences to their geographic neighbors are the countries Denmark, Sweden, Norway and Finland; Holland, Germany and Belgium; Switzerland and France, with Luxembourg not far away. England's distance from other countries on this chart follows the geographic proximity logic, as it is an island, physically separated by water from the continent.

As mentioned, there are exceptions that do not follow the theory that countries' food preferences proximity on this chart corresponds to geographic proximity. For example, on the chart, Ireland is close to Holland and Germany; Italy and Portugal are close, with Spain more distant from Portugal. Austria is as isolated as England on the chart, despite the fact that a large part of its borders are shared with other countries in this analysis.

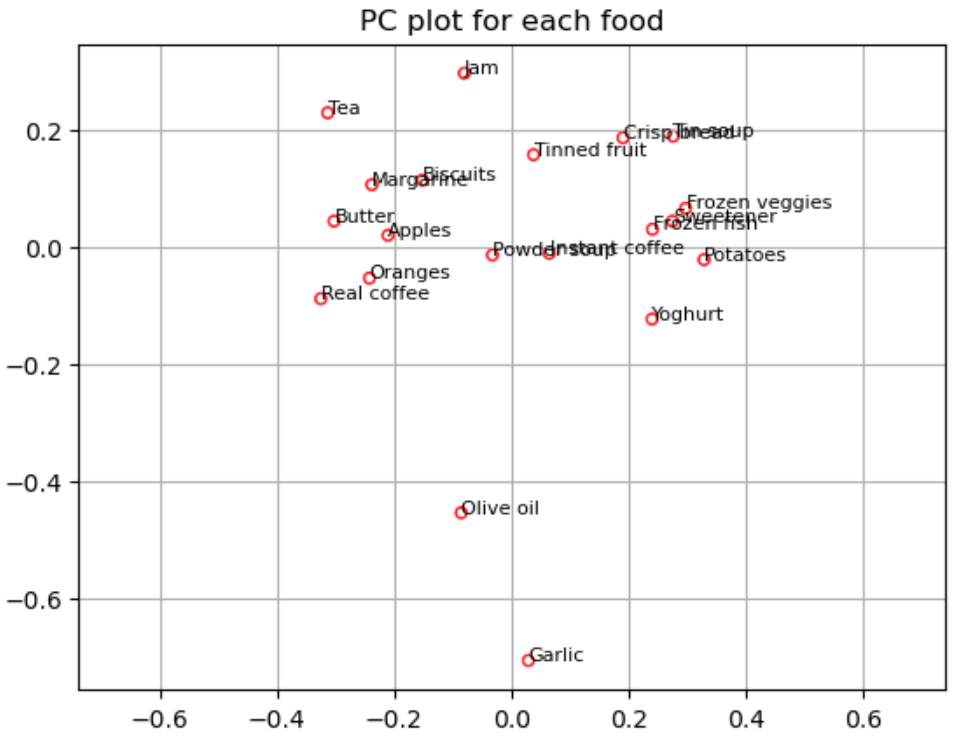
It is possible a study of ancient European history regarding who settled different areas at various times, as we as old maps showing countries that were once one, might offer insight into some of these discrepancies. For example, Austria, while it does not have similar food preferences with countries it borders to the West, might share food preferences with those to its East. Climate, proximity to the ocean and economic factors might impact these groupings as well.

2. PCA: Food consumption in European countries (cont.)

b. PCA approach B: way: Now, we will perform PCA analysis on the data by treating *country consumptions as "feature" vectors for each food item*. In other words, we will now find *weight vectors to combine country consumptions for each food item to perform PCA another way.* Project data to obtain their two principle components (thus, again each data point – for each food item – can be represented using a two-dimensional vector). Draw a scatter plot of food items. Mark the food items on the plot (you can do this by hand if you want). Please explain any pattern you observe in the scatter plot.

Observations: The groupings of food in the "Food Principal Component Plot" is somewhat similar to groupings of Market Basket analysis. In both cases, items are grouped with other items a person would eat together, for the most part. For example, it's logical to buy butter, margarine and biscuits together; apples and oranges together; crisp bread and soup; instant coffee and powdered soup; frozen vegetables and frozen fish.

In some cases, the frequency with which some items are bought, may relate to the food's shelf life. For example: olive oil. I would think this would be used for cooking throughout all countries in this analysis. Yet because it is only needed in small amounts, comes in a large bottle and has a long shelf life, it does not need to be bought as frequently as many of the other items, such as apples and oranges. Items bought less frequently should have a negative X and Y value in this analysis.

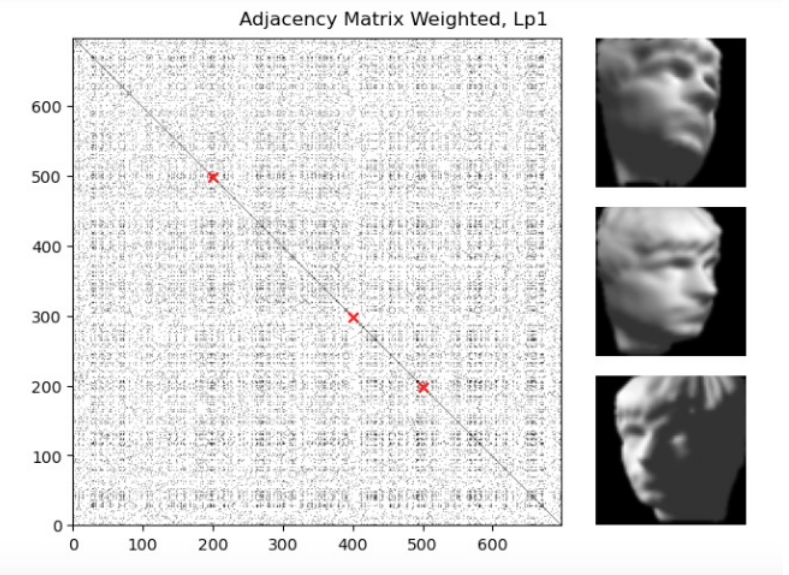


2. Order of faces using ISOMAP

•This analysis attempts to reproduce the ISOMAP algorithm from the original paper for ISOMAP, by J.B. Tenenbaum, V. de Silva, and J.C. Langford, Science 290 (2000) 2319-2323. The file isomap.mat (or isomap.dat) contains 698 images, corresponding to different poses of the same face. Each image is given as a 64 × 64 luminosity map, hence represented as a vector. *Source code for this section: See /model/ ISOMAP.py*

(a) First we visualize the nearest neighbor graph and illustrate a few images corresponds to nodes at different parts of the graph, marked in red.

Weighted adjacency matrix, right. The distance to the nearest neighbor graph is represented by each entry. The randomly selected images, pictured to the right, have their locations marked with the red crosses.



ISOMAP theoretical key ideas for manifold learning and non-linear dimensionality reduction.

– Keep "walking distance" over the data cloud (manifold) by creating low dimensional representation.

– Let A be the adjacency matrix recording neighbor Euclidean distance, & find neighbors $N(i)$ of each data point, x^i, within distance E.

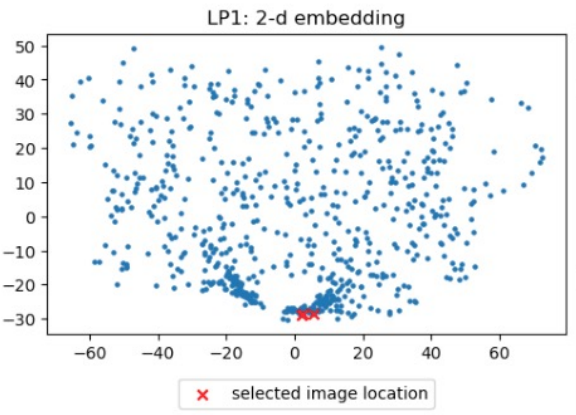
– Based on A, determine the shortest path distance matrix D between pairs of points, $x^i and x^j$.

– This preserves the distances information in D by finding low dimensional representation.

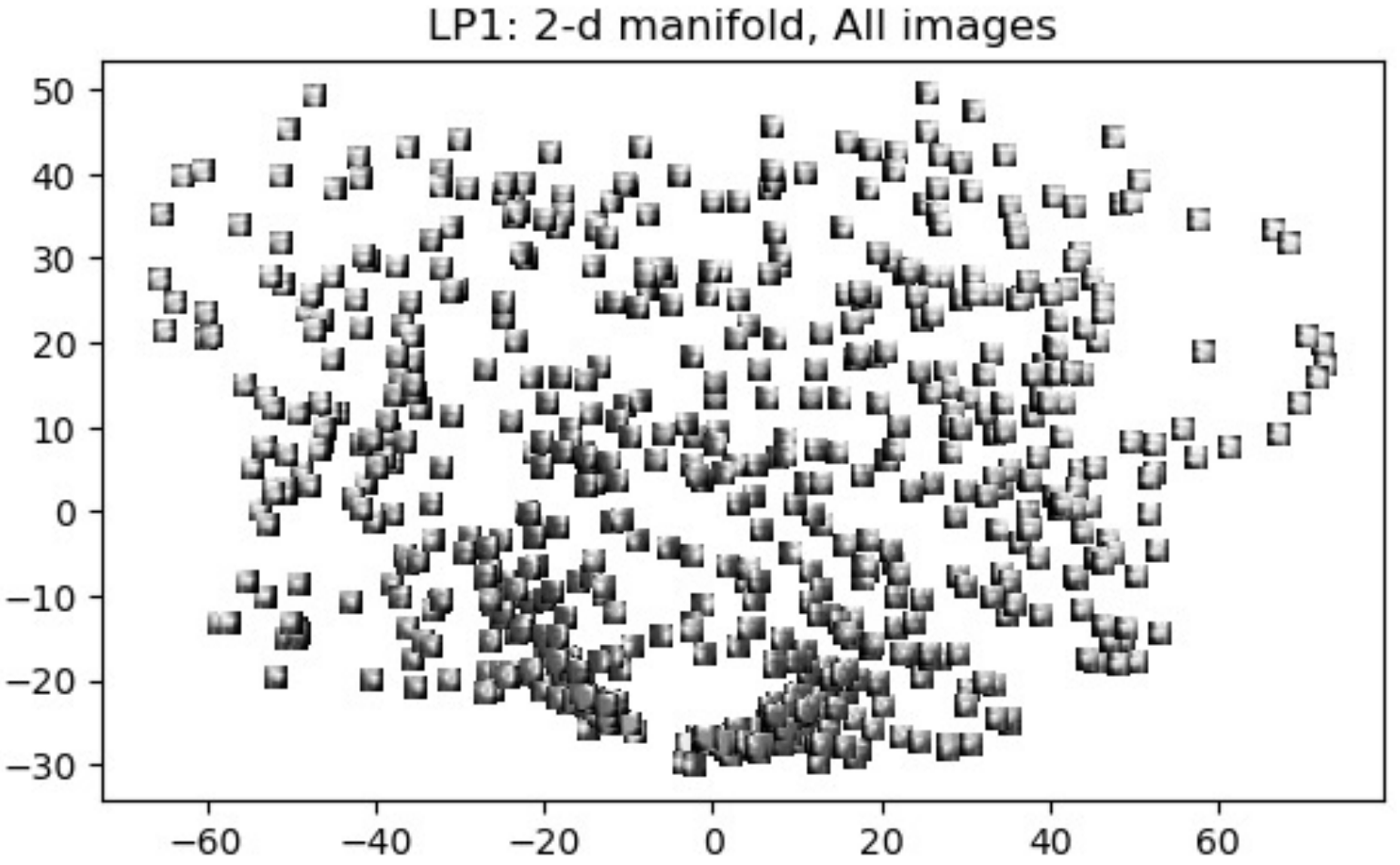
Source: Nonlinear Dimensionality Reduction lecture, Xie, Yao, Ph.D., Associate Professor, Georgia Institute of Technology

2. Order of faces using ISOMAP (cont.)

(b) Next, we implement the ISOMAP algorithm by hand to obtain a two-dimensional, low-dimensional embedding. We plot the embeddings using a scatter plot, mark a few images in the embedding space, enlarge these images, & specify the face locations on the scatter plot, *right*. The randomly selected images, pictured to the right, have their locations marked with the red crosses.

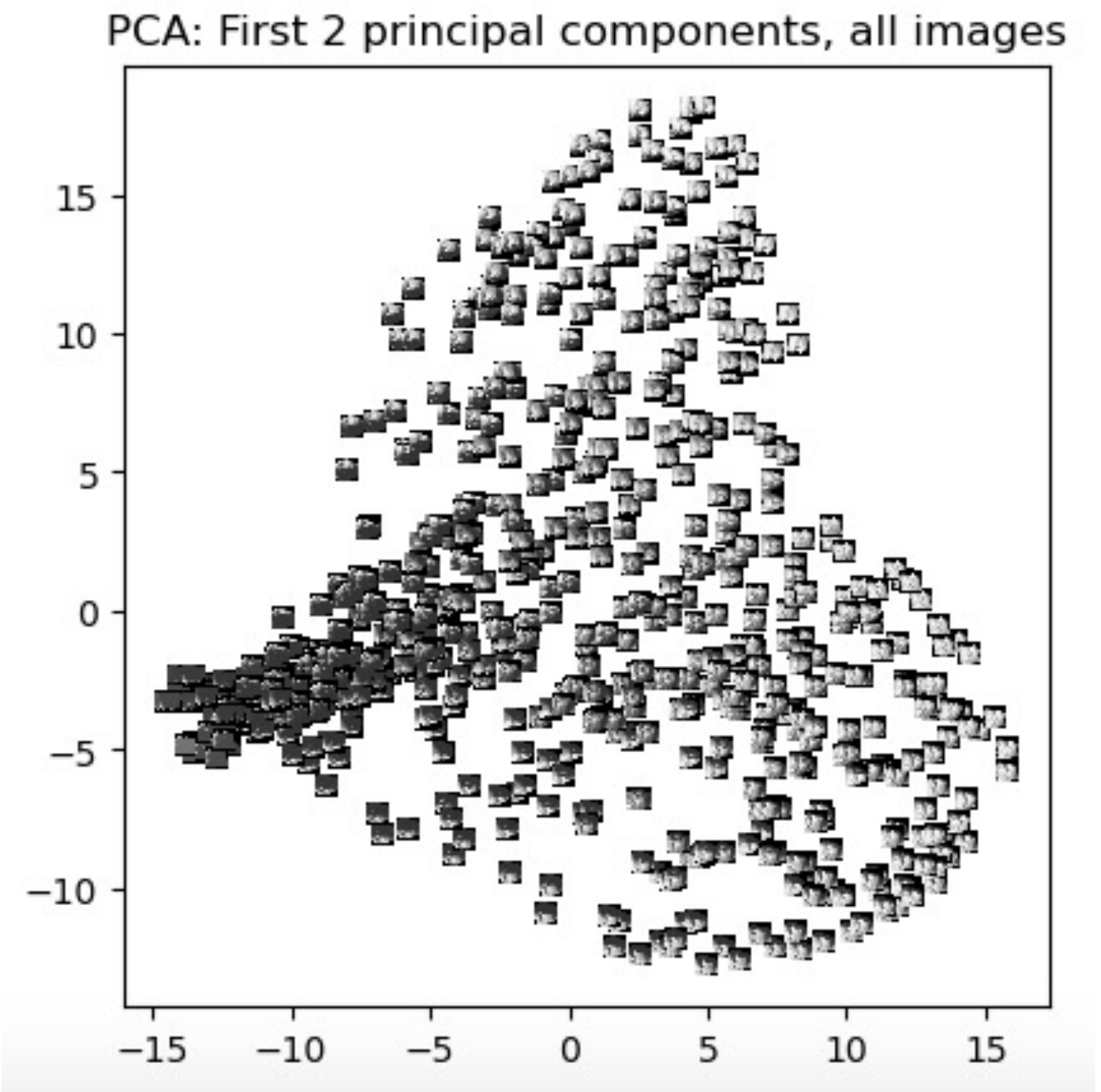


Observations: There are similarities among the images and their arrangements, to observations noted in the original paper. For example, we notice a gradual change in direction that the poses are facing as we move along the manifold. There is a 'drifting' change in these faces globally, both here, and as was noted in the paper.



2. Order of faces using ISOMAP [25 points] (cont.)

(c) Finally, we perform PCA on the images and project them into the top 2 principal components.
Observations: Notice the more meaningful projection with ISOMAP, compared to PCA. Locally, the PCA embedded faces do show similarity. However, PCA does not capture the global 'drifting' change of the faces in the manifold.



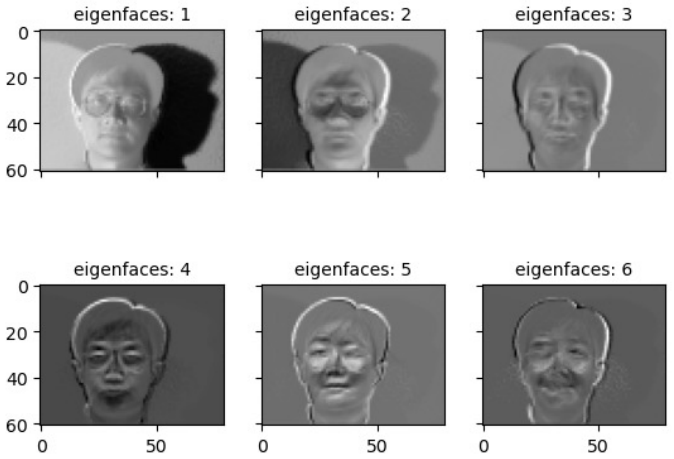
3. PCA & Eigenfaces and simple face recognition

This analysis is a simplified illustration of using PCA for face recognition. We will use a subset of data from the famous Yale Face dataset. *Source code for this section: See /model/ pca_and_image_recognition.py*

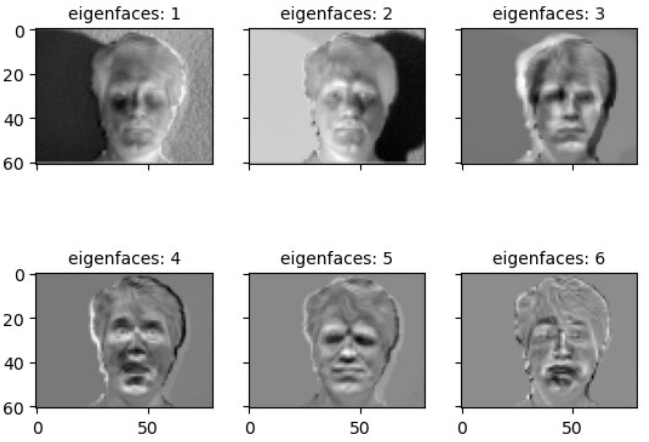
(a) Here, we analyze the Yale face dataset for Subject 1 and Subject 2, respectively, using all the images EXCEPT for the two pictures named subject01-test.gif and subject02-test.gif. We plot the first 6 eigenfaces for each subject. When visualizing, we reshape the eigenvectors into proper images.

Observations: Following are the first six eigenfaces for subject 1 and 2. We noticed images associated with higher eigenvalues show a greater clarity of the features, and thus allow for greater recognition.

Eigenfaces for subject 1



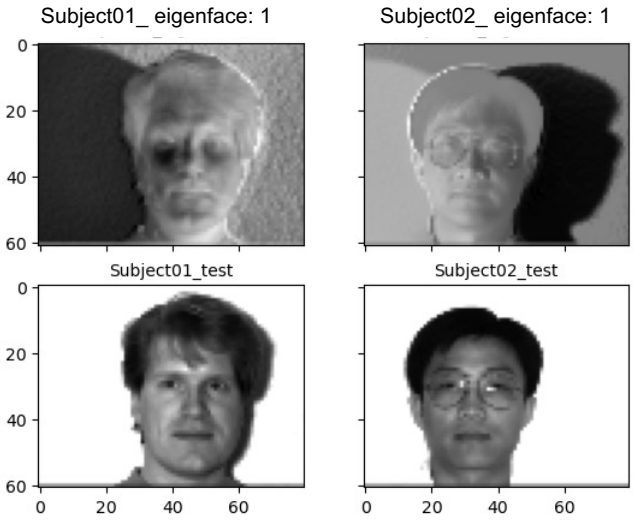
Eigenfaces for subject 2



(b) Face recognition using PCA proceeds as follows: Given the test image subject01-test.gif and subject02-test.gif, first, we downsize the image by a factor of 4 (as before), and vectorize each image. Next, we take the top eigenfaces of Subject 1 and Subject 2, respectively. Then we calculate the projection residual of the two vectorized test images with the vectorized eigenfaces:

$$sij = \|(\text{test image})j - (\text{eigenfacei})(\text{eigenface})\text{Ti}(\text{testmage})j\|22$$

We report all four scores: sij , i = 1, 2, j = 1, 2, *left*.



We calculated the recognition by choosing the eigenface with a lower residual or errors with the test image. For this analysis, that was true for subject 1, but not subject 2.

	test 1	test 2
eg_face_1	9.700357e+11	1.048548e+12
eg_face_2	9.703570e+11	1.048716e+12

(c) As noted above, this face recognition algorithm did not work well, as it was correct only 50% of the time. To improve this, I would like to try the normalized inner product score.

Part 2: Statistical Theory & Derivation of Concepts

1. Principal Component Analysis (PCA):

a. Here we show that the first principle component direction "w", corresponds to the largest eigenvector of the sample covariance matrix, written as:

$$v = \arg\max_{w:||w||<=1} \frac{1}{m} \sum_{i=1}^{m} (w^T x^i - w^T \mu)^2$$

The formula above states we are looking for the direction of w (or weight vector) such that the variance of the data along direction w is maximized & less than or equal to 1. Here $w^T x^i$ is the projected data, relative to the mean: $w^T \mu$

First, we use linear algebra to manipulate the objective:

$$\frac{1}{m} \sum_{i=1}^{m} (w^T x^i - w^T \mu)^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (w^T (x^i - \mu))^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (w^T (x^i - \mu) w^T (x^i - \mu))$$

$$= \frac{1}{m} \sum_{i=1}^{m} w^T (x^i - \mu)(x^i - \mu)^T w$$

$$= w^T \underbrace{\left(\frac{1}{m} \sum_{i=1}^{m} (x^i - \mu)(x^i - \mu)^T \right)}_{C} w$$

We see the covariance matrix, C, is part of the equation, above, as noted. Therefore we can rewrite this equation as:

$$\max\ w^T C w$$
$$||w|| <= 1$$

The direction is represented by w, which should maximize variance. To solve the PCA problem, it becomes a constrained optimization problem. The general approach for solving optimization problems is using the Lagrangian function, which we apply to the equation from above:

$$L(w, \lambda) = w^T C w + \lambda(1 - ||w||^2)$$

We use the Lagrangian multiplier λ to create some violation. What is the corresponding w that we will achieve by solving this optimization problem? We maximize the Lagrangian function with respect to w, while minimizing it with respect to λ.

If w is a maximum of the original optimization problem, then there exists a λ, where (w, λ) is a stationary point of $L(w, \lambda)$.

How to do this: Use multivariate calculus to get the derivatives of parts of the previous equation:

$$\delta w^T C w = 2Cw$$
$$\delta ||w||^2 = 2w$$

Take the derivative of the Lagrangian function, set it equal to zero. This implies that:

$$\frac{\delta L}{\delta w} = 0 = 2Cw - 2\lambda w <=> Cw = \lambda w$$

We see we end up with an eigendecomposition problem, with w the eigenvector and λ the eigenvalue for C.

Objective function becomes λ or eigen-value (associated with w)

$$w^T C w = \lambda w^T w = \lambda ||w||^2$$

Furthermore, it is apparent that the largest eigenvector is the weight vector required for the first principle component direction, as Cw = λw. The reason for this is that the λ value which maximizes the objective function, or variance, is the largest eigenvalue.

The problem becomes finding the largest eigenvalue of C.

The optimal solution w should be an eigen-vector of C. By choosing w_1 or the largest eigen-vector of C, the corresponding largest eigen-value, or variance, is λ_1.

Thus, we compute the reduced representations or principle components, of a data point as follows:

$$z^i = w^{i^T}(x^i - \mu)/\sqrt{\lambda_i}$$

The first principle component direction v corresponds to the largest eigenvector of the the sample covariance matrix, written as z^i:

$$z^1 = w^{1^T}(x^1 - \mu)/\sqrt{\lambda_1}$$

Source: pca.pdf, Xie, Yao, Ph.D., Associate Professor, Georgia Institute of Technology

b. Next, we show how to find the third largest principle component, based on the proof above.

To get the third largest principle component direction, we simply take the third largest eigen-vector & eigen-value pair of C, or w_3 eigen-vector or direction, & corresponding next largest eigen-value or variance of λ_3, with the constraints that the third principle component is orthogonal to the second.

Building on the equation above, the equation for the third largest principle component is:

$$z^3 = w^{3^T}(x^3 - \mu)/\sqrt{\lambda_3}$$

We recall that in solving an eigenvalue problem, such as this one, there will be multiple solution: $w_1, w_2, w_3, \ldots$, the eigenvectors or directions of C, with corresponding eigenvalues or variance of $\lambda_1, \lambda_2, \lambda_3 \ldots$ The eigenvectors are ortho-normal.

To visualize this concept we reference the slide from the course lecture series. Here we see the first principal component w1 has the largest variance, and the second principal

2. Maximum Likelihood Estimation (MLE) for Gaussian Mean & Variance. Here we show that the maximum likelihood estimate (MLE) for Gaussian mean and variance parameters are given by:

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^i, \hat{\sigma}^2 = \frac{1}{m}\sum_{i=1}^{m}(x^i - \hat{\mu})^2$$

,

respectively.

First, we begin with an overview of the approach to reaching the estimation of parametric models.

We begin by assuming m data points $D = \{x^1, x^2, \ldots x^m\}$, iid & from an unknown distribution $P^*(x)$. We want to fit the data with the model $P(x|\theta)$, w/ parameters θ

$$\theta = argmax_\theta log P(D|\theta)$$

θ is the maximizer to solve this optimization problem. Alter θ to increase the probability of seeing the data.

Take the log, which is written as the product of the distribution.

$$\hat{\theta} = argmax_{\theta} log \prod_{i=1}^{m} P(x^i|\theta)$$

After introducing the log the product becomes the sum, which simplifies finding the gradient. (Using the sum instead of the product simplifies the problem)

$$\hat{\theta} = argmax_{\theta} \sum_{i=1}^{m} logP(x^i|\theta)$$

In a similar fashion, we begin the step by step process to show that the maximum likelihood estimate (MLE) for Gaussian mean and variance parameters are given by the equation presented in question 3, above.

We start with the equation for the Gaussian distribution in R

$$p(x|\mu, \sigma) = \frac{1}{(2\pi)^{\frac{1}{2}}\sigma} exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2 \right)$$

We apply the log, which gives the following:

$$\mu, \sigma = argmax \sum_{i=1}^{N} logp(x_i|\mu, \sigma) = argmax \sum_{i=1}^{N} log\frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x_1-\mu)^2}{2\sigma^2} \right)$$

Simplify the expression inside the log, (property 1)

$$\mu, \sigma = argmax \sum_{i=1}^{N} log(1) - log(\sqrt{2\pi}) - log(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2}$$

Remove constants not influencing parameters μ, σ :

$$\mu, \sigma = argmax \sum_{i=1}^{N} -log(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2}$$

Change the sign & turn the maximization problem into a minimization problem (standard notation):

$$\mu, \sigma = argmin \sum_{i=1}^{N} log(\sigma) + \frac{(x_1-\mu)^2}{2\sigma^2}$$

Substitute the objective function:

$$J(\mu, \sigma) = \sum_{i=1}^{N} log(\sigma) + \frac{(x_1-\mu)^2}{2\sigma^2}$$

Or:

$$\mu, \sigma = argmin \sum_{i=1}^{N} J(\mu, \sigma)$$

Find the parameters μ, σ: calculate the partial derivatives and set to zero:

$$\frac{\partial}{\partial \mu} J(\mu, \sigma) = 0 -> \mu$$

$$\frac{\partial}{\partial \sigma} J(\mu, \sigma) = 0 -> \sigma$$

A. Partial derivative for μ:

$$\frac{\partial}{\partial \mu} J(\mu, \sigma) = 0$$

$$\frac{\partial}{\partial \mu} \sum_{i=1}^{N} log(\sigma) + \frac{(x_1 - \mu)^2}{2\sigma^2} = 0$$

$$\sum_{i=1}^{N} \frac{2(x_1 - \mu)(-1)}{2\sigma^2} = 0$$

$$2\left(\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \mu \right) = 0$$

$$\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} \mu$$

$$\sum_{i=1}^{N} x_i = N\mu$$

$$\frac{1}{N} \sum_{i=1}^{N} x_i = \mu$$

B. Partial derivative for σ (variance):

$$\frac{\partial}{\partial \sigma} J(\mu, \sigma) = 0 -> \sigma$$

$$\frac{\partial}{\partial \sigma} (-\frac{n}{2} log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i}^{n} (x_i - \mu)^2)$$

Set = to 0:

$$-\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_{i}^{n}(x_i - \mu)^2 = 0$$

Subtract the first term from each side & multiply by σ^3:

$$\sum_{i}^{n}(x_i - \mu)^2 = n\sigma^2$$

Divide each side by n:

$$\frac{1}{N}\sum_{i=1}^{N}(x^i - \mu)^2 = \hat{\sigma}^2$$

Put the partial derivatives for σ^2 & μ, found above, next to each other, add hats, so we have: $\hat{\sigma}^2$ & $\hat{\mu}$, switch equations from one side of the equal sign to the other, replace N with m, and we have proved that the maximum likelihood estimate (MLE) for Gaussian mean and variance parameters are given by

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^i, \hat{\sigma}^2 = \frac{1}{m}\sum_{i=1}^{m}(x^i - \hat{\mu})^2$$

,

which is what question 1.3 asked us to prove. Source: Lecture 6: Density estimation: Estimation of parametric models, Xie, Yao, Ph.D., Associate Professor, Georgia Institute of Technology