High-Dimensional Data Analysis

Lecture 4 - Principal Component Analysis

Fall semester - 2024

Dr. Eng. Valentin Leplat
Innopolis University
September 18, 2024

Outline

Key notions and Toy example

Outline

Principal Component Analysis

- **Principal components analysis** (PCA) is one of a family of techniques for taking high-dimensional data, and using the dependencies between the variables to represent it in a more *tractable*, *lower-dimensional* form, without losing too much information.
- a versatile statistical method for reducing a cases-by-variables data matrix/table to its essential features, called *principal components*.
- Principal components are a few linear combinations of the original variables that *maximally explain the variance* of all the variables. [1]
- In the process, the method provides an approximation of the original data matrix using only these few major components.

[1] this sentence may sound enigmatic, but it will become clear in the second section

Principal Component Analysis

- **Therefore**: PCA is one of the simplest and most robust ways of doing such *dimensionality reduction*.
- It is also one of the oldest, and has been rediscovered many times in many fields: *aka* the Karhunen-Loeve transformation, the Hotelling transformation, the method of empirical orthogonal function, and singular value decomposition [2]

Ok, we will call it PCA:).

[2] Strictly speaking, singular value decomposition is a matrix algebra trick which is used in the most common algorithm for PCA.

Outline

Example

- The following toy example gives a sense of the problem solved by PCA, and
- many of the reasons why you might want to apply it to a data set:
  1. to visualize the data in a lower-dimensional space,
  2. to understand the sources of variability in the data,
  3. and to understand correlations between different coordinates of the data points.

Example

- The following toy example gives a sense of the problem solved by PCA, and
- many of the reasons why you might want to apply it to a data set:
  1. to visualize the data in a lower-dimensional space,
  2. to understand the sources of variability in the data,
  3. and to understand correlations between different coordinates of the data points.
- Suppose you ask some friends to rate, on a scale of 1 to 10, four different foods: salad, Vkusno I Tochka, sashimi, and "Jubilee" cookies, with the results shown in the Table here-under

	salad	V. I T.	sashimi	J. cookies
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

Example

- The following toy example gives a sense of the problem solved by PCA, and
- many of the reasons why you might want to apply it to a data set:
  1. to visualize the data in a lower-dimensional space,
  2. to understand the sources of variability in the data,
  3. and to understand correlations between different coordinates of the data points.
- Suppose you ask some friends to rate, on a scale of 1 to 10, four different foods: salad, Vkusno I Tochka, sashimi, and "Jubilee" cookies, with the results shown in the Table here-under

	salad	V. I T.	sashimi	J. cookies
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

- In our usual notations: m (the number of data points) is 4, and n (the number of dimensions) is also 4.
- Note that: in contrast to the linear regression setting, the data points do not have "labels" of any sort (beyond the names of the people).

Example

- Can we usefully visualize this data set in fewer than 4 dimensions?
- Can we understand the forces at work behind the differences in opinion of the various foods?

Example

- Can we usefully visualize this data set in fewer than 4 dimensions?
- Can we understand the forces at work behind the differences in opinion of the various foods?
- The key observation is that each row (data point) can be approximately expressed as

$$x^{(i)} \approx \bar{x} + a_1^{(i)} v_1 + a_2^{(i)} v_2, \text{ for } 1 \leqslant i \leqslant m = 4$$

where

$$\bar{x} = \begin{pmatrix} 5.5 \\ 4.5 \\ 5 \\ 5.5 \end{pmatrix}$$

is the average of the data points.

Example

- The key observation is that each row (data point) can be approximately expressed as

$$x^{(i)} \approx \bar{x} + a_1^{(i)} v_1 + a_2^{(i)} v_2, \text{ for } 1 \leqslant i \leqslant m = 4$$

where

$$\bar{x} = \begin{pmatrix} 5.5 \\ 4.5 \\ 5 \\ 5.5 \end{pmatrix}$$

is the average of the data points.

Example

- The key observation is that each row (data point) can be approximately expressed as

$$x^{(i)} \approx \bar{x} + a_1^{(i)} v_1 + a_2^{(i)} v_2, \text{ for } 1 \leqslant i \leqslant m = 4$$

where

$$\bar{x} = \begin{pmatrix} 5.5 \\ 4.5 \\ 5 \\ 5.5 \end{pmatrix}$$

is the average of the data points.

$$v_1 = \begin{pmatrix} 3 \\ -3 \\ -3 \\ 3 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

and $(a_1^{(1)}, a_2^{(1)})^T = (1, 1)^T$ for Alice, $(a_1^{(2)}, a_2^{(2)})^T = (1, -1)^T$ for Bob, $(a_1^{(3)}, a_2^{(3)})^T = (-1, -1)^T$ for Carolyn, and $(a_1^{(4)}, a_2^{(4)})^T = (-1, 1)^T$ for Dave.

Example

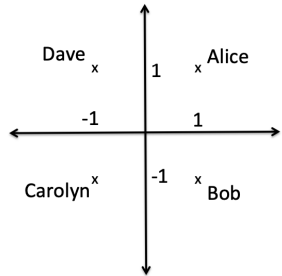


Figure: Visualizing 4-dimensional data in the plane.

For example:

- $\bar{x} + v_1 + v_2 = (9.5, 0.5, 3, 7.5)^T$, approximately equal to Alice's scores $(10, 1, 2, 7)^T$.
- $\bar{x} - v_1 - v_2 = (1.5, 8.5, 7, 3.5)^T$, approximately equal to Carolyn's scores $(2, 9, 7, 3)^T$.

Question: What use is such an approximation?

Example - answers

1. One answer is that it provides a way to visualize the data points, as in previous Figure : with more data points, we can imagine inspecting the resulting figure for clusters of similar data points.

[3] Conversely, if we knew which friends were vegetarian and which were not, we could deduce which foods are most likely to be vegetarian.

Example - answers

1. One answer is that it provides a way to visualize the data points, as in previous Figure : with more data points, we can imagine inspecting the resulting figure for clusters of similar data points.
2. A second answer is that it helps interpret the data:
   - We think of each data point as having a "v_1-coordinate" (i.e., $a_1^{(i)}$) and a "v_2-coordinate" ($a_2^{(i)}$).
   - What does the vector $v_1 = \big(3, -3, -3, 3\big)^T$ "mean" ?
     2.1 The first and fourth coordinates both have a large positive value (and so are positively correlated)
     2.2 while the second and third coordinates have a large negative value (and so are positively correlated with each other and negatively correlated with the first and fourth coordinates).

[3] Conversely, if we knew which friends were vegetarian and which were not, we could deduce which foods are most likely to be vegetarian.

Example - answers

1. One answer is that it provides a way to visualize the data points, as in previous Figure : with more data points, we can imagine inspecting the resulting figure for clusters of similar data points.
2. A second answer is that it helps interpret the data:
   - We think of each data point as having a "v_1-coordinate" (i.e., $a_1^{(i)}$) and a "v_2-coordinate" ($a_2^{(i)}$).
   - What does the vector $v_1 = \left(3, -3, -3, 3\right)^T$ "mean" ?
     2.1 The first and fourth coordinates both have a large positive value (and so are positively correlated)
     2.2 while the second and third coordinates have a large negative value (and so are positively correlated with each other and negatively correlated with the first and fourth coordinates).
   - Noting that salad and J. Cookies are vegetarian, while Vkusno I Tochka and sashimi are not, we can interpret the v_1 coordinate as indicating the extent to which someone has vegetarian preferences [3]

[3] Conversely, if we knew which friends were vegetarian and which were not, we could deduce which foods are most likely to be vegetarian.

Example - answers

1. One answer is that it provides a way to visualize the data points, as in previous Figure : with more data points, we can imagine inspecting the resulting figure for clusters of similar data points.
2. A second answer is that it helps interpret the data:
   - We think of each data point as having a "v_1-coordinate" (i.e., $a_1^{(i)}$) and a "v_2-coordinate" ($a_2^{(i)}$).
   - What does the vector $v_1 = \left(3, -3, -3, 3\right)^T$ "mean" ?
     2.1 The first and fourth coordinates both have a large positive value (and so are positively correlated)
     2.2 while the second and third coordinates have a large negative value (and so are positively correlated with each other and negatively correlated with the first and fourth coordinates).
   - Noting that salad and J. Cookies are vegetarian, while Vkusno I Tochka and sashimi are not, we can interpret the v_1 coordinate as indicating the extent to which someone has vegetarian preferences [3]
   - Similarly: we can interpret the second vector v_2 as indicating the extent to which someone is health-conscious.
   - The fact that v_1 has larger coordinates than v_2 indicates that it is the stronger of the two effects.

[3] Conversely, if we knew which friends were vegetarian and which were not, we could deduce which foods are most likely to be vegetarian.

Example - end

- We'll see that the vectors v_1 and v_2, once normalized, correspond to the "top two principal components" of the data.
- The point of PCA is to compute approximations of the type

$$x^{(i)} \approx \bar{x} + \sum_{j}^{k} a_j^{(i)} v_j, \text{ for } 1 \leqslant i \leqslant m$$

  automatically, including for large data sets.
- I.e., express each of m n-dimensional vectors[4] $x^{(1)}, ..., x^{(m)} \in \mathbb{R}^n$ as linear combinations of k n-dimensional vectors $v_1, ..., v_k \in \mathbb{R}^n$.

[4] Representing, for example: images (dimensions = pixels); measurements (dimensions = sensors); documents (dimensions = words); and so on.

Relation to Linear Regression

Both PCA and linear regression concern fitting the "best" k-dimensional subspace to a point set.
First difference : the interpretation of the input data

- In linear regression:
  1. each data point has a real-valued label, which can be interpreted as a special "label coordinate." **Goal**: learn the relationship between this special coordinate and the other coordinates.
  2. This makes sense when the "label" is a dependent variable that is approximately a linear combination of all of the other coordinates (the independent variables).
- In PCA:
  1. all coordinates are treated equally, and they are not assumed to be independent from one another.
  2. This makes sense when there is a set of latent (i.e., hidden/underlying) variables, and all of the coordinates of your data are (approximately) linear combinations of those variables.

Relation to Linear Regression

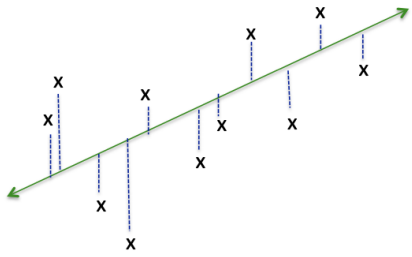
Second difference: PCA and linear regression use different definitions of "best fit."

- In least squares linear regression: the goal is to minimize the total squared error, where the error on a data point is the difference *between* the linear function's prediction and the actual label of the data point.[5]
- We will define the PCA objective function formally later, but in a nutshell: for two-dimensional data, the goal is to compute the line that minimizes the sum of the squared *perpendicular* distances between the line and the data points [6]

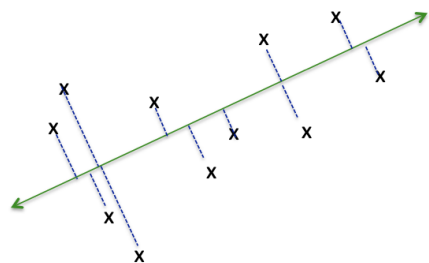
[5] This reflects the assumption that the coordinate corresponding to labels is the important one.

[6] This reflects the assumption that all coordinates play a symmetric role and so the usual Euclidean distance is most appropriate (e.g., rotating the point set just results in the "best-fit" line being rotated accordingly).

Relation to Linear Regression



(a) Linear regression



(b) PCA

Figure: Linear regression minimizes the sum of squared *vertical* distances, while PCA minimizes the sum of squared *perpendicular* distances.

Preprocessing - centering

Before using PCA, it's crucial to *preprocess* the data.
First:

- the points $x^{(1)}, ..., x^{(m)}$ should be centered around the origin, in the sense that $\sum_{i=1}^{m} x^{(i)}$ is the all-zero vector.
- This is easy to enforce by subtracting (i.e., shifting) each point by the "sample mean" $\bar{x} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$ [7]
- After finding the best-fit line for the shifted point set, one simply shifts the line back by the original sample mean to get the best-fit line for the original uncentered data set.
- This shifting trick makes the necessary linear algebra simpler and clearer.

[7] in the toy example, we had $\bar{x} = \left(5.5, 4.5, 5, 5.5\right)^T$

Preprocessing - scaling

Second:

- In many applications it is important to *scale* each coordinate appropriately.
- The most common approach to this is:
  1. Let $x^{(1)}, ..., x^{(m)}$ be the point set centered at the origin,
  2. then for each coordinate $j = 1, 2, ..., n$, divide the jth coordinate of every point by the "sample deviation" in that coordinate,

$$\sqrt{\sum_{i=1}^{m} (x_j^{(i)})^2}$$

- **Motivation** for this coordinate scaling: without it, the result of PCA would be *highly sensitive* to the *units* in which each coordinate is measured.

Preprocessing - scaling

Example: changing units from miles to kilometers in some coordinate yields the "same" data set in some sense, and yet this change would scale up all values in this coordinate, which in turn would cause a different "best-fit" line to be computed.[8]

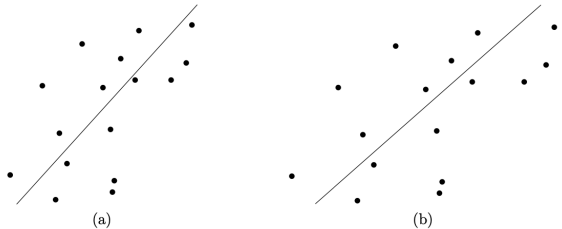


Figure: Scaling the x-axis yields a different best-fit line.

In some applications, like with images — where all coordinates are in the same units, namely pixel intensities — there is no need to do such coordinate scaling.

[8] Already in two dimensions, that stretching out one coordinate affects the best line through the points.

Mathematics of Principal Components

Outline

Problem formulation

For clarity, we first discuss the special case of $k = 1$, where the goal is to fit the "best" line to a data set. [9]

[9] Everything in this section generalizes easily to the case of general k

[10] Note that we're identifying a line (through the origin) with a unit vector v in the same direction.

Problem formulation

For clarity, we first discuss the special case of $k = 1$, where the goal is to fit the "best" line to a data set. [9]

- As foreshadowed above, PCA defines the "best-fit line" as the one that minimizes the average squared Euclidean distance between the line and the data points[10]:

$$\text{argmin}_{v:\|v\|=1} \frac{1}{m} \sum_{i=1}^{m} (\text{distance between } x^{(i)} \text{ and line spanned by } v)^2 \tag{1}$$

- Minimizing the Euclidean distances between the points and the chosen line should seem natural enough; the one thing you might be wondering about is why we square these distances before adding them up ?

[9] Everything in this section generalizes easily to the case of general k

[10] Note that we're identifying a line (through the origin) with a unit vector v in the same direction.

Problem formulation

For clarity, we first discuss the special case of $k = 1$, where the goal is to fit the "best" line to a data set. [9]

- As foreshadowed above, PCA defines the "best-fit line" as the one that minimizes the average squared Euclidean distance between the line and the data points[10]:

$$\text{argmin}_{v:\|v\|=1} \frac{1}{m} \sum_{i=1}^{m} (\text{distance between } x^{(i)} \text{ and line spanned by } v)^2 \qquad (1)$$

- Minimizing the Euclidean distances between the points and the chosen line should seem natural enough; the one thing you might be wondering about is why we square these distances before adding them up ?
- One reason is that it ensures that the best-fit line passes through the origin. Another is a tight connection to variance maximization, discussed next.

[9] Everything in this section generalizes easily to the case of general k

[10] Note that we're identifying a line (through the origin) with a unit vector v in the same direction.

Problem formulation

- Recall the geometry of projections and the inner product

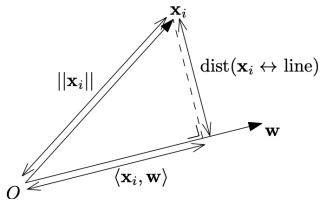


Figure: The geometry of the inner product with a unit length vector, w.

- $\langle x^{(i)}, v \rangle$ is the (signed) length of the projection of $x^{(i)}$ onto the line spanned by v.
- By Pythagorean Theorem, for any norm $\|.\|$ on $\mathbb{R}^n$ induced by an inner product:

$$(\langle x^{(i)}, v \rangle)^2 + (\text{dist}(x^{(i)} \leftrightarrow \text{line}))^2 = \|x^{(i)}\|^2 \tag{2}$$

Problem formulation

- The right-hand side of Equation (2) (recalled here-under for convenience)

$$(\langle x^{(i)}, v\rangle)^2 + (\text{dist}(x^{(i)} \leftrightarrow \text{line}))^2 = \|x^{(i)}\|^2$$

  is a constant, independent of the choice of line v.
- Thus, there is a zero-sum game here :).
- This implies that the objective function of maximizing the squared projections — the *variance* of the projection of the point set — is equivalent to the original objective in Problem (1):

$$\text{argmax}_{v:\|v\|=1} \frac{1}{m} \sum_{i=1}^{m} (\langle x^{(i)}, v\rangle)^2 \tag{3}$$

Problem formulation

- The right-hand side of Equation (2) (recalled here-under for convenience)

$$(\langle x^{(i)}, v\rangle)^2 + (\text{dist}(x^{(i)} \leftrightarrow \text{line}))^2 = \|x^{(i)}\|^2$$

  is a constant, independent of the choice of line v.
- Thus, there is a zero-sum game here :).
- This implies that the objective function of maximizing the squared projections — the *variance* of the projection of the point set — is equivalent to the original objective in Problem (1):

$$\text{argmax}_{v:\|v\|=1} \frac{1}{m} \sum_{i=1}^{m} (\langle x^{(i)}, v\rangle)^2 \tag{3}$$

Puzzle: try to convince yourselves that $\frac{1}{m}\sum_{i=1}^{m}(\langle x^{(i)}, v\rangle)^2$ is indeed the variance of the projection of point set onto v.
Hints: See $x^{(i)}$ as samples of a random variable x, and pose $y = \langle x, v\rangle$ for a fixed v.

Problem formulation - *intermezzo*

- The objective function of maximizing variance seems *natural/intuitive*.
- Actually: PCA's objective function admits multiple interpretations, this means that it's performing a *fundamental* operation on the data.
- **Example** imagine the data points fall into two well-separated clusters

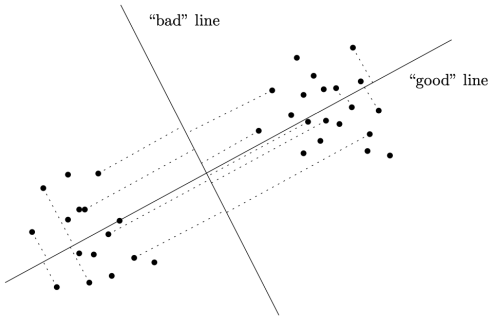


Figure: For the good line, the projection of the points onto the line keeps the two clusters separated, while the projection onto the bad line merges the two clusters.

Problem formulation - *intermezzo*

- **Here**: maximizing variance corresponds to preserving the separation between the two clusters post-projection.
- With a poorly chosen line, the two clusters would project on top of one another, obscuring the structure in the data.
- PCA effectively assumes that variance in the data corresponds to interesting information.
- Caution: one can imagine scenarios where such variance corresponds to noise rather than signal, in which case PCA may not produce illuminating results.

Problem formulation - Larger Values of k

- The discussion so far focused on the $k = 1$ case [11], but the case of larger k is almost the same.
- **Key observation**: for general k, the objective functions of minimizing the squares of distances to a k-dimensional subspace and of maximizing the variance of the projections onto a k-dimensional subspace are again equivalent !
- So the PCA objective is now:

$$\underset{k\text{-dimensional subspaces } S}{\operatorname{argmax}} \frac{1}{m} \sum_{i=1}^{m} (\text{length of } x^{(i)}\text{'s projection on } S)^2 \tag{4}$$

- To rephrase this objective in a more convenient form, let us recall the basics from the Linear Algebra Boot Camp !

[11] fitting a line to the data

Problem formulation - Larger Values of k

- Let $v_1, ..., v_k \in \mathbb{R}^n$ be an orthonormal basis of S. I.e. $S = \text{span}(v_1, ..., v_k)$.
  - If $k = 1$, then this span is a line through the origin;
  - if $k = 2$, then the span is a plane (through the origin); and so on.

Problem formulation - Larger Values of k

- Let $v_1, ..., v_k \in \mathbb{R}^n$ be an orthonormal basis of S. I.e. $S = \text{span}(v_1, ..., v_k)$.
  - If $k = 1$, then this span is a line through the origin;
  - if $k = 2$, then the span is a plane (through the origin); and so on.
- One nice fact about orthonormal vectors $v_1, ..., v_k \in \mathbb{R}^n$ is that the squared projection length of a point onto the subspace spanned by the vectors is just the sum of the squares of its projections onto each of the vectors:

$$(\text{length of } x^{(i)}\text{'s projection on span}(v_1, ..., v_k))^2 = \sum_{j=1}^{k} (\langle x^{(i)}, v_j \rangle)^2 \tag{5}$$

Exercise: prove the previous statement :).
Hints: given $x \in \mathbb{R}^n$, and $v_1, ..., v_k \in \mathbb{R}^n$ an orthonormal basis of S, then we know that : $x_{\text{proj}_S} = \sum_{j=1}^{k} \langle x, v_j \rangle v_j$. Compute $\|x_{\text{proj}_S}\|^2$.

Problem formulation - Larger Values of k

- Combining Equations (4) and (5), we can state the objective of PCA for general k in its standard form:
  compute orthonormal vectors $v_1, ..., v_k$ to maximize

$$\frac{1}{m}\sum_{i=1}^{m} \underbrace{\sum_{j=1}^{k}(\langle x^{(i)}, v_j\rangle)^2}_{\text{squared projection length}} \tag{6}$$

- The resulting k orthonormal vectors are called the *top k principal components* of the data.
- To recap

Definition of the problem solved by PCA

Given $x^{(1)}, ..., x^{(m)} \in \mathbb{R}^n$ and parameter $k \geqslant 1$, compute orthonormal vectors $v_1, ..., v_k \in \mathbb{R}^n$ to maximize (6).

Outline

Methods for finding the top k principal components.

- To develop the solution, we first consider only the $k = 1$ case. We'll see that the case of general k reduces to a sequence of this case !
- With $k = 1$, the optimisation problems is:

$$\underset{v:\|v\|=1}{\operatorname{argmax}} \frac{1}{m} \sum_{i=1}^{m} (\langle x^{(i)}, v \rangle)^2 \tag{7}$$

Methods for finding the top k principal components.

- To develop the solution, we first consider only the $k = 1$ case. We'll see that the case of general k reduces to a sequence of this case !
- With $k = 1$, the optimisation problems is:

$$\underset{v:\|v\|=1}{\operatorname{argmax}} \frac{1}{m} \sum_{i=1}^{m} (\langle x^{(i)}, v \rangle)^2 \tag{7}$$

- Let's see how to rewrite variance-maximization Problem (7) using linear algebra. First, we take the data points $x^{(1)}, ..., x^{(m)}$ — remember these are in d-dimensional space — and write them as the rows of an $m \times n$ matrix X:

$$X = \begin{pmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{pmatrix}$$

Methods for finding the top k principal components.

- Thus, for a unit vector $v \in \mathbb{R}^n$, we have

$$Xv = \begin{pmatrix} \langle x^{(1)}, v \rangle \\ \langle x^{(2)}, v \rangle \\ \vdots \\ \langle x^{(m)}, v \rangle \end{pmatrix}$$

Methods for finding the top k principal components.

- Thus, for a unit vector $v \in \mathbb{R}^n$, we have

$$Xv = \begin{pmatrix} \langle x^{(1)}, v \rangle \\ \langle x^{(2)}, v \rangle \\ \vdots \\ \langle x^{(m)}, v \rangle \end{pmatrix}$$

- so Xv is just a column vector populated with all the projection lengths of the $x^{(i)}$'s onto the line spanned by v.

Methods for finding the top k principal components.

- Thus, for a unit vector $v \in \mathbb{R}^n$, we have

$$Xv = \begin{pmatrix} \langle x^{(1)}, v \rangle \\ \langle x^{(2)}, v \rangle \\ \vdots \\ \langle x^{(m)}, v \rangle \end{pmatrix}$$

- so Xv is just a column vector populated with all the projection lengths of the $x^{(i)}$'s onto the line spanned by v.
- We care about the sum of the squares of these (recall Problem (7)), which motivates taking the inner product of Xv with itself:

$$\frac{1}{m} v^T X^T X v = \frac{1}{m} (Xv)^T (Xv) = \frac{1}{m} \sum_{i=1}^{m} (\langle x^{(i)}, v \rangle)^2$$

Methods for finding the top k principal components.

- Summarizing, our variance-maximization problem can be rephrased as

$$\underset{v:\|v\|=1}{\operatorname{argmax}} \quad v^T A v \tag{8}$$

where $A = \frac{1}{m} X^T X \in \mathbb{R}^{n \times n}$, symmetric and positive semi-definite.

Methods for finding the top k principal components.

- Summarizing, our variance-maximization problem can be rephrased as

$$\underset{v:\|v\|=1}{\text{argmax}} \quad v^T A v \tag{8}$$

  where $A = \frac{1}{m} X^T X \in \mathbb{R}^{n \times n}$, symmetric and positive semi-definite.
- This problem is called "maximizing a quadratic form."

Interpretation of $A = \frac{1}{m} X^T X$

The matrix $A = \frac{1}{m} X^T X$ has a natural interpretation[12].

- The (i, j) entry of this matrix is the inner product of the ith row of X^T and the jth column of X — i.e., of the ith and jth columns of X.
- So $X^T X$ just collects the inner products of columns of X,

[12] You might recall that this matrix also appeared in the closed-form solution to the ordinary least squares problem

[13] So after centering such an X, frequently co-occurring pairs of words correspond to positive entries of $X^T X$ and pairs of words that almost never appear together are negative entries.

Interpretation of $A = \frac{1}{m} X^T X$

The matrix $A = \frac{1}{m} X^T X$ has a natural interpretation[12].

- The (i, j) entry of this matrix is the inner product of the ith row of X^T and the jth column of X — i.e., of the ith and jth columns of X.
- So $X^T X$ just collects the inner products of columns of X,
- **Example**:
  - suppose the $x^{(i)}$'s represent documents, with dimensions (i.e., columns of X) corresponding to words.
  - Then the inner product of two columns of X measures how frequently the corresponding pair of words co-occur in a document.[13]
  - The matrix $\frac{1}{m} X^T X$ is called the *sample covariance* or *correlation* matrix of the $x^{(i)}$'s, depending on whether or not each of the coordinates was normalized so as to have unit variance in a preprocessing step.

[12] You might recall that this matrix also appeared in the closed-form solution to the ordinary least squares problem

[13] So after centering such an X, frequently co-occurring pairs of words correspond to positive entries of $X^T X$ and pairs of words that almost never appear together are negative entries.

Solving Problem (8)

- We simply use the basics from the LA Boot camp :). $X^T X$ is symmetric and positive semi-definite, hence by the *Spectral Theorem* we have:

$$\frac{1}{m} X^T X = A = Q \Lambda Q^T$$

where Q is an orthogonal matrix of size $n \times n$, and Λ real diagonal matrix with diagonal entries $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n \geqslant 0$.

Solving Problem (8)

- We simply use the basics from the LA Boot camp :). $X^T X$ is symmetric and positive semi-definite, hence by the *Spectral Theorem* we have:

$$\frac{1}{m} X^T X = A = Q \Lambda Q^T$$

where Q is an orthogonal matrix of size $n \times n$, and Λ real diagonal matrix with diagonal entries $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n \geqslant 0$.

- Problem (8) becomes:

$$\underset{v:\|v\|=1}{\operatorname{argmax}} \quad v^T Q \Lambda Q^T v \tag{9}$$

- Let $y = Q^T v$, we have $\|y\| = \|Q^T v\| = \|v\|$ for any *unitarily invariant* norm $\|.\|$ on $\mathbb{R}^n$.
- Therefore, we may equivalently solve:

$$\underset{y:\|y\|=1}{\operatorname{argmax}} \quad y^T \Lambda y := \sum_{i=1}^{n} \lambda_i y_i^2 \tag{10}$$

Solving Problem (8)

From this point: the results presented are only valid for $\|.\| := \|.\|_2$.

The optimal solution for Problem (10) can be obtained in two steps

1. Find an upper-bound for the objective function:

$$\begin{aligned} & \max_{y:\|y\|_2=1} && (\lambda_1 y_1^2 + \cdots + \lambda_n y_n^2) \\ \Leftrightarrow \quad & \max_{y:\|y\|_2=1} && \lambda_1 (y_1^2 + \frac{\lambda_2}{\lambda_1} y_2^2 + \cdots + \frac{\lambda_n}{\lambda_1} y_n^2) \\ \leqslant \quad & \max_{y:\|y\|_2=1} && \lambda_1 (y_1^2 + y_2^2 + \cdots + y_n^2) = \lambda_1 \end{aligned}$$

2. Find a **feasible** $y^\star$[14] that reaches this upper-bound: easy, for instance $y^\star = e_1$:).

[14] that is $\|y^\star\| = 1$

Solving Problem (8)

From this point: the results presented are only valid for $\|.\| := \|.\|_2$.

The optimal solution for Problem (10) can be obtained in two steps

1. Find an upper-bound for the objective function:

$$\begin{aligned} & \max_{y:\|y\|_2=1} \quad (\lambda_1 y_1^2 + \cdots + \lambda_n y_n^2) \\ \Leftrightarrow \quad & \max_{y:\|y\|_2=1} \quad \lambda_1 (y_1^2 + \frac{\lambda_2}{\lambda_1} y_2^2 + \cdots + \frac{\lambda_n}{\lambda_1} y_n^2) \\ \leqslant \quad & \max_{y:\|y\|_2=1} \quad \lambda_1 (y_1^2 + y_2^2 + \cdots + y_n^2) = \lambda_1 \end{aligned}$$

2. Find a **feasible** $y^\star$[14] that reaches this upper-bound: easy, for instance $y^\star = e_1$:).

Finally, the optimal solution of (8) is $v^\star = Qy^\star = q_1$, that is the first eigenvector of A.

[14] that is $\|y^\star\| = 1$

Larger Values of k

- To solve Problem (6), that is PCA for general k follows easily. All the mathematical derivations are provided in Appendix-1 :).
- **The solution**: the first k columns of Q where $\frac{1}{m}X^TX = Q\Lambda Q^T$ with Q orthonormal and Λ diagonal with sorted diagonal entries.
- Note that since Q is orthonormal, the first k columns of Q are indeed a set of k orthonormal vectors, as required !
- these are called the *top k principal components of X*.

Larger Values of k

- To solve Problem (6), that is PCA for general k follows easily. All the mathematical derivations are provided in Appendix-1 :).
- **The solution**: the first k columns of Q where $\frac{1}{m}X^TX = Q\Lambda Q^T$ with Q orthonormal and Λ diagonal with sorted diagonal entries.
- Note that since Q is orthonormal, the first k columns of Q are indeed a set of k orthonormal vectors, as required !
- these are called the *top k principal components of X*.

In summary

PCA boils down to computing the k eigenvectors of the sample covariance matrix $\frac{1}{m}X^TX$ that have the largest eigenvalues.

Larger Values of k

Simple but powerful interpretations:

1. Variance and Covariance are measurements of the *spreading* of the data.
2. Eigenvectors are just the "axes of spreading" of the covariance matrix.
3. with eigenvalues giving the spreading factors.

Outline

How does one compute the top k principal components?

1. The first method comes from the previous developments: build $A = \frac{1}{m} X^T X$, compute its EigenValue Decomposition (EVD), and keep the k first eigenvectors of A.

[15] to get the strict equivalence with EVD, we have to set $X \leftarrow \frac{X}{\sqrt{m}}$ after the centering

[16] for those interested: randomized SVD can be used instead.

How does one compute the top k principal components?

1. The first method comes from the previous developments: build $A = \frac{1}{m} X^T X$, compute its EigenValue Decomposition (EVD), and keep the k first eigenvectors of A.
2. A second method uses the "singular value decomposition (SVD); indeed
   - let $X = U \Sigma V^T$ the SVD of X[15].
   - By definition of the SVD, our top k principal components correspond to the first k columns of V.
   - The Time Complexity of the SVD is roughly cubic, that is $O(\min(mn^2, m^2 n))$, which is fine for medium-size but not for large-scale problems. [16]

[15] to get the strict equivalence with EVD, we have to set $X \leftarrow \frac{X}{\sqrt{m}}$ after the centering

[16] for those interested: randomized SVD can be used instead.

How does one compute the top k principal components?

1. The first method comes from the previous developments: build $A = \frac{1}{m} X^T X$, compute its EigenValue Decomposition (EVD), and keep the k first eigenvectors of A.
2. A second method uses the "singular value decomposition (SVD); indeed
   - let $X = U \Sigma V^T$ the SVD of X[15].
   - By definition of the SVD, our top k principal components correspond to the first k columns of V.
   - The Time Complexity of the SVD is roughly cubic, that is $O(\min(mn^2, m^2 n))$, which is fine for medium-size but not for large-scale problems. [16]
3. The *power iteration*: popular in practice, especially in settings where one only wants the top few eigenvectors (as is often the case in PCA applications). See Appendix-2 for the description of the algorithm.

In ***Matlab*** and ***SciPy*** there are optimized and ready-to-use implementations of both the SVD and power iteration methods

[15] to get the strict equivalence with EVD, we have to set $X \leftarrow \frac{X}{\sqrt{m}}$ after the centering

[16] for those interested: randomized SVD can be used instead.

Compute projection onto the span of the top k components?

For simplicity, consider $k = 2$.

- In the toy example, after centering, each data point (row of X) can be approximately expressed as $x^{(i)} \approx x^{(i)}_{\text{proj}_S} = a^{(i)}_1 v_1 + a^{(i)}_2 v_2$
- how to compute the $(a^{(i)}_1, a^{(i)}_2)$? Very easy ! v_1, v_2 are an orthonormal basis of S, hence

Compute projection onto the span of the top k components?

For simplicity, consider $k = 2$.

- In the toy example, after centering, each data point (row of X) can be approximately expressed as $x^{(i)} \approx x^{(i)}_{\text{proj}_S} = a_1^{(i)} v_1 + a_2^{(i)} v_2$
- how to compute the $(a_1^{(i)}, a_2^{(i)})$? Very easy ! v_1, v_2 are an orthonormal basis of S, hence

$$\begin{pmatrix} a_1^{(i)} \\ a_2^{(i)} \end{pmatrix} = \begin{pmatrix} \langle x^{(i)}, v_1 \rangle \\ \langle x^{(i)}, v_2 \rangle \end{pmatrix}$$

  for all i.

- **Remarkably**, all these components can be computed at once for all i with a simple matrix-matrix product:

$$L = \begin{pmatrix} a_1^{(1)} & a_2^{(1)} \\ \vdots & \vdots \\ a_1^{(m)} & a_2^{(m)} \end{pmatrix} = \begin{pmatrix} \langle x^{(1)}, v_1 \rangle & \langle x^{(1)}, v_2 \rangle \\ \vdots & \vdots \\ \langle x^{(m)}, v_1 \rangle & \langle x^{(m)}, v_2 \rangle \end{pmatrix} = X \begin{pmatrix} v_1 & v_2 \end{pmatrix} = X V_{1:2} \tag{11}$$

  L is sometimes referred to as the *loadings* matrix, each row gives the components of the data points within the principal components basis.

Compute projection onto the span of the top k components?

- This was used in our toy example to "draw" Dave, Alice, Carolyn and Bob in the plane :) (although the scaling was different because our vectors v_1 and v_2 were not unit vectors).
- For general k, it is super easy to get the "loadings":

$$L_{1:k} = X \begin{pmatrix} v_1 & \cdots & v_k \end{pmatrix} = XV_{1:k}$$

- And what about the $x^{(i)}_{\text{proj}_S}$'s ? Just need to do this:

$$X_{\text{proj}_S} = XV_{1:k}V_{1:k}^T = \begin{pmatrix} — & (a_1^{(1)}v_1 + a_2^{(1)}v_2 + \cdots + a_k^{(1)}v_k)^T & — \\ \vdots & \vdots & \vdots \\ — & (a_1^{(m)}v_1 + a_2^{(m)}v_2 + \cdots + a_k^{(m)}v_k)^T & — \end{pmatrix}$$

  which is just a use case of orthogonal projector $P = V_{1:k}V_{1:k}^T$:).
- **Great...** but do we really need all these operations ? Do we have already "something" that includes all these information ??

Compute projection onto the span of the top k components?

- This was used in our toy example to "draw" Dave, Alice, Carolyn and Bob in the plane :) (although the scaling was different because our vectors v_1 and v_2 were not unit vectors).
- For general k, it is super easy to get the "loadings":

$$L_{1:k} = X \begin{pmatrix} v_1 & \cdots & v_k \end{pmatrix} = XV_{1:k}$$

- And what about the $x^{(i)}_{\text{proj}_S}$'s ? Just need to do this:

$$X_{\text{proj}_S} = XV_{1:k}V_{1:k}^T = \begin{pmatrix} — & (a_1^{(1)}v_1 + a_2^{(1)}v_2 + \cdots + a_k^{(1)}v_k)^T & — \\ \vdots & \vdots & \vdots \\ — & (a_1^{(m)}v_1 + a_2^{(m)}v_2 + \cdots + a_k^{(m)}v_k)^T & — \end{pmatrix}$$

  which is just a use case of orthogonal projector $P = V_{1:k}V_{1:k}^T$:).

- **Great...** but do we really need all these operations ? Do we have already "something" that includes all these information ??
  SVD contains it all ! (V, $L_{1:k} = U_{1:k}\Sigma_{1:k}$, and $X_{\text{proj}_S} = \sum_{j=1}^{k} u_j\sigma_j v_j^T$)

PCA workflow: EVD and SVD for $k = 2$

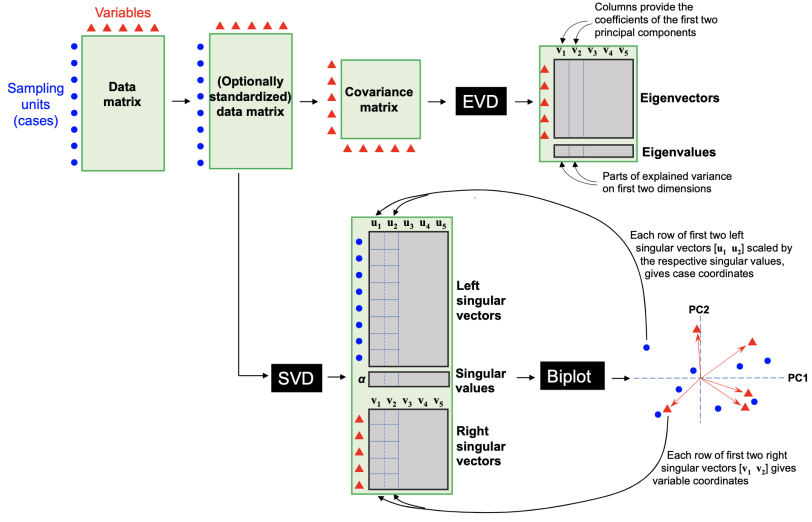


Figure: Extracted from Ref[1]

PCA workflow: EVD and SVD for $k = 2$

Comments on previous figure:

- The definition of the principal components (PCs) can be obtained using the eigenvalue decomposition (EVD) of the covariance matrix of the variables, scaling is optional, but centring is mandatory, and if the variables are divided by their standard deviations, then the covariance matrix is the correlation matrix and the analysis is sometimes referred to as "correlation PCA".
- The lower pathway is a more efficient one, using the singular value decomposition (SVD). The eigenvectors are identical to the right singular vectors. For the lower pathway to be exactly equivalent to the upper one, the (optionally standardized) data matrix should be divided by $\sqrt{m}$.

Results

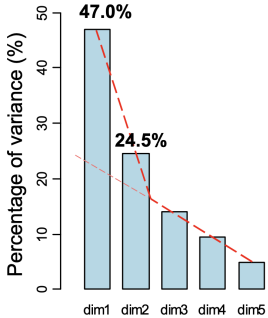
Dimensionality of a PCA solution

The first question of interest: how much of the data variance is explained by the consecutive dimensions of the solution ?

- PCA sorts the data variance into the major features in the data on the leading dimensions, and
- what is considered random noise on the minor dimensions.
- The sequence of percentages of variance explained suggests how many non-random major dimensions there are.

Dimensionality of a PCA solution : choose k

Choose $k \Leftrightarrow$ where is the *elbow* ? (Figure Extracted from Ref[1])



- shows the bar chart of the five percentages in a PCA of five variables, where: $\text{percentage}_j = \frac{\lambda_j(X^TX)}{\sum_{i=i}^{n}\lambda_i(X^TX)} \times 100$
- the percentages on the first two dimensions, 47.0% and 24.5%, can be seen to stand out from the last three.
- showing the "elbow"[a] that suggests that the first two dimensions are signal, whereas the last three dimensions are random noise.

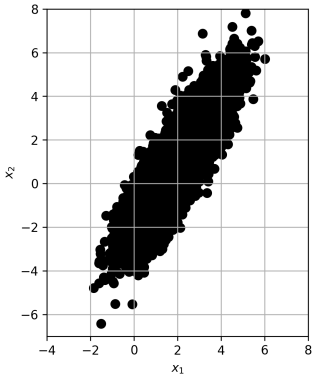
[a] by drawing a line (the red dashed line) through the last three, showing that the first two are above that approximate linear descending pattern.

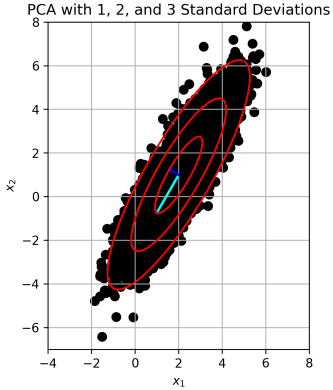
Examples

Noisy Gaussian data

- The data is generated by selecting 10.000 vectors from a two-dimensional normal distribution with zero mean and unit variance.
- These vectors are then scaled in the x_1 and x_2 directions by the values in Table after and rotated by $\pi/3$.
- Finally, the entire point set is translated so that it has a nonzero center $x_c = (2, 1)^T$.
- Using the code from ▸ Colab - Section 1, the PCA is performed and used to plot confidence intervals using multiple standard deviations.

Noisy Gaussian data





Left: Principal components capture the variance of mean-subtracted Gaussian data
Right: The first three standard deviation ellipsoids (red), and the two right singular vectors, scaled by singular values ($\sigma_1 v_1 + x_c$, cyan, and $\sigma_2 v_2 + x_c$, blue).

Noisy Gaussian data

The singular values, shown in next Table, match the data scaling.

	σ_1	σ_1
Data	2	0.5
SVD	2.01	0.50

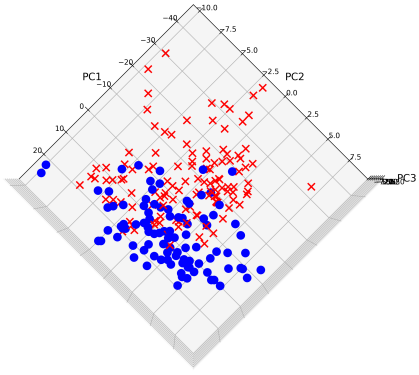
The matrix V from the SVD also closely matches the rotation matrix, up to a sign on the columns:

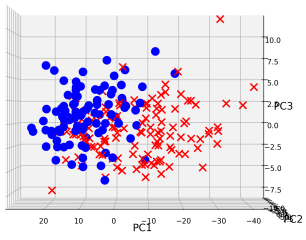
$$R_{\pi/3} = \begin{pmatrix} 0.5 & -0.8660 \\ 0.8660 & 0.5 \end{pmatrix}, \qquad U = \begin{pmatrix} 0.499 & 0.8665 \\ 0.8665 & -0.499 \end{pmatrix} \tag{12}$$

Ovarian Cancer data set

- The ovarian cancer data set, which is built into Matlab, provides a more realistic example to illustrate the benefits of PCA. .mat file
- This example consists of gene data for 216 patients, 121 of whom have ovarian cancer, and 95 of whom do not. For each patient, there is a vector of data containing the expression of 4000 genes.
- There are multiple challenges with this type of data, namely the high dimension of the data features.

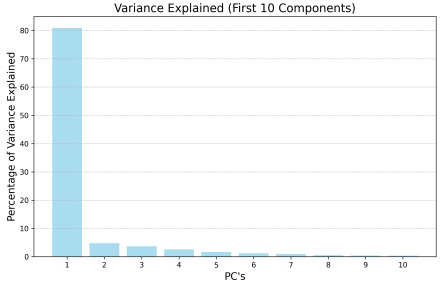
Ovarian Cancer data set

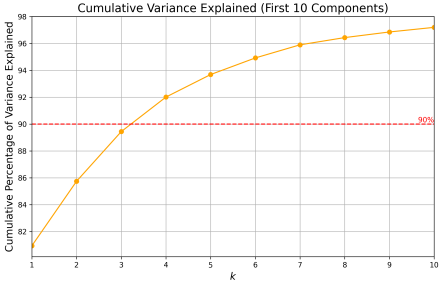




Patients with ovarian cancer appear to cluster separately from patients without cancer when plotted in the space spanned by the first three PCA modes. ▸ Colab file - Section 2

Ovarian Cancer data set





Colab file - Section 2

- we see that there is significant variance captured in the first few PCA modes. I.e., the gene data is highly correlated, so that many patients have significant overlap in their gene expression.
- The ability to visualize patterns and correlations in high-dimensional data is an important reason to use PCA, and PCA has been widely used to find patterns in high-dimensional biological and genetic data (see the Appendix and the lab :)).

Eigenfaces example

- One of the most striking demonstrations of SVD/PCA is the so-called *eigenfaces* example.
- In this problem, PCA (i.e. SVD on mean-subtracted data) is applied to a large library of facial images to extract the most dominant correlations between images.
- The result of this decomposition is a set of *eigenfaces* that define a new coordinate system. Images may be represented in these coordinates by taking the dot product with each of the principal components.
- Images of the same person tend to cluster in the eigenface space, making this a useful transformation for facial recognition and classification

Eigenfaces example

- Here, we demonstrate this algorithm using the *YALE face database*[17]
- It consisting of 15 subjects in 11 slightly different poses, for a total of 165 64x64 pixel images conditions.
- Facial expressions or configurations: center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink.
- Each of the facial images in our library have been reshaped into a large column vector with $64 \times 64 = 4096$ elements, then arranged in the rows of matrix X (size 165x4096) so that the first 11 rows correspond to the 11 images for the first face, the next 11 rows to the second face, and so on.
- **Demo**: Colab file - Section 4

[17] The database can be downloaded at link or here

Eigenfaces example

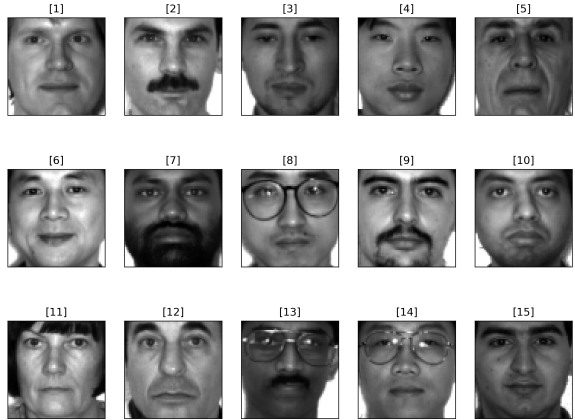


Figure: A single image for each person in the Yale database.

Eigenfaces example

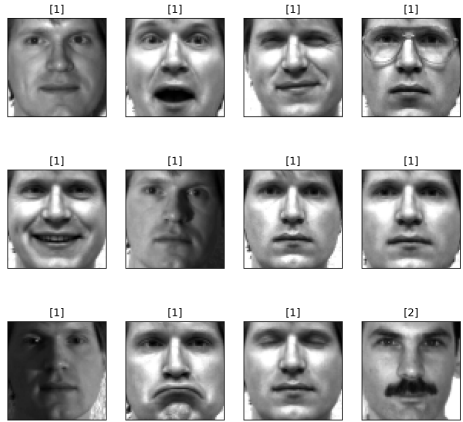
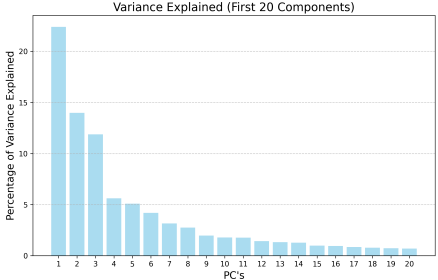


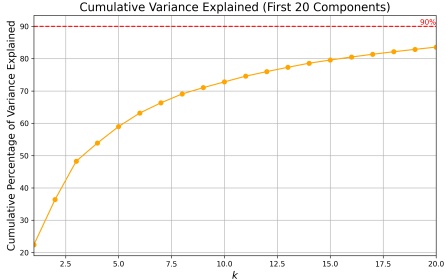
Figure: All images for a specific person in the Yale database.

Eigenfaces example

Figure: Eigenfaces: reshaped columns of $V_{1:k}$ with $k = 20$.

Eigenfaces example





Eigenfaces example

Figure: Approximate representation of test image using eigenfaces basis of various order k.

Interpretation of a PCA biplot

Both the v_j's and the projections of the $x^{(i)}$'s onto them can be interesting. Here are some things to look at:

1. Look at the data points with the largest (most positive) and smallest (most negative) projections $\langle x^{(i)}, v_1 \rangle$ on the first principal component. Does this suggest a potential "meaning" for the component? Are there data points clustered together at either of the two ends, and if so, what do these have in common?
2. Plot all points according to their k coordinate values (i.e., projection lengths onto the top k principal components). Do any interesting clusters pop out (e.g., with $k = 2$, in any of the four corners)? For example, by looking at pairs that have similar second coordinates — both pairs with similar first coordinates, and pairs with very different first coordinates — it is often possible to obtain a rough interpretation of the second principal component.
3. Looking at the coordinates of a principal component — the linear combination of the original data point attributes that it uses — can also be helpful. (E.g., see the Eigenfaces application.)

Failure cases

When and why does PCA fail?

1. You messed up the scaling/normalizing. PCA is sensitive to different scalings/normalizations of the coordinates. Much of the artistry of getting good results from PCA involves choosing an appropriate scaling for the different data coordinates, and perhaps additional preprocessing of the data (like outlier removal).
2. Non-linear structure. PCA is all about finding linear structure in your data. If your data has some low-dimensional, but non-linear structure, e.g., you have two data coordinates, x, y, with $y \approx x^2$, then PCA will not find this. [18]Similarly for GPS data from someone on a ferris wheel (Figure 7) — the data is one dimensional (each point can be specified by an angle) but this dimension will not be identified by PCA.
3. Non-orthogonal structure. It is nice to have coordinates that are interpretable, like in our toy example. Even if your data is essentially linear, the fact that the principal components are all orthogonal will often mean that after the top few components, it will be almost impossible to interpret the meanings of the components.

[18]As "puzzled" in the context of linear regression last lecture, you can always add extra non-linear dimensions to your data, like coordinates x^2, y^2, xy, etc., and then PCA this higher-dimensional data. This idea doesn't always work well with PCA, however.

When and why does PCA fail?

- GPS Data from a ferris wheel

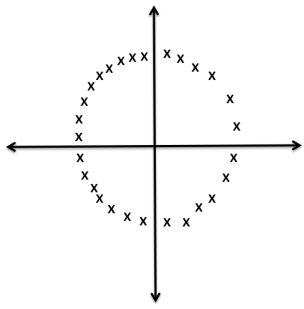


Figure: PCA is not designed to discover nonlinear structure.

- Dataset that is nonlinear - Colab file - Section 3

Summary

Summary

We have seen :

- General definition of the *Principal Component Analysis*: represent high-dimensional data in lower-dimensional form without losing too much info.
- PCA vs *linear regression*, and the important steps of *preprocessing*.
- The mathematics of *Principal Components*: the formulation, the methods and algorithms for finding the top k principal components.
  In summary: *PCA boils down to computing the k eigenvectors of the sample covariance matrix* $A = \frac{1}{m} X^T X$ *that have the largest eigenvalues.*
- PCA workflow: *Eigenvalue Decomposition* and *Singular Value Decomposition*.
- *Dimensionality* of a PCA solution: Choose $k \Leftrightarrow$ where is the *elbow* ?
- Some introductory examples

Preparation for the lab

- Review the lecture :).
- **Strongly** recommended to read the following recent paper published in **Nature**:
  Ref[1]: M. Greenacre et al. *Principal Component Analysis.* In Nature Reviews Methods Primers, 2022. ‣ Nature website, or ‣ pdf
  You will find:
  1. additional insights on PCA,
  2. more examples and experiments,
  3. variants of PCA including *constraints*, for instance *sparsity*.
  4. ***Python*** and ***R*** packages and functions implementing PCA and its variants.
  5. Limitations and optimizations: this part may be very interesting for your project.
  6. Outlook of PCA: in a nutshell, PCA is still an active field of research despite its age :).
- Additional reading *Do Genomes Encode Geography?*
  Ref[2]: J. Novembre et al. *Genes mirror geography within Europe.* Nature, 456:98–101, 2008. ‣ Nature website ‣ pdf

  Motivation: Can you infer where someone grew up from their DNA? In a remarkable study, Novembre et al. used PCA to show that in some cases the answer is "yes." This is a case study in how PCA can reveal that the data "knows" things that you wouldn't expect.

Appendices

Larger Values of k

- Let us first note that the Problem

$$\underset{\{v_j\}_j^k \text{orthonormal}}{\text{argmax}} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{k} (\langle x^{(i)}, v_j \rangle)^2 \tag{13}$$

  can be reformulated equivalently as follows

$$\underset{\{v_j\}_j^k \text{orthonormal}}{\text{argmax}} \sum_{j=1}^{k} \sum_{i=1}^{m} (\langle x^{(i)}, v_j \rangle)^2. \tag{14}$$

- Following the methodology for $k = 1$, Problem (14) can be written as

$$\underset{\{v_j\}_j^k \text{orthonormal}}{\text{argmax}} \sum_{j=1}^{k} v_j^T Q \Lambda Q^T v_j. \tag{15}$$

Larger Values of k

- Posing $y_j = Q^T v_j$ for $1 \leqslant j \leqslant k$, Problem (15) becomes:

$$\underset{\{y_j\}_j^k \text{orthonormal}}{\operatorname{argmax}} \sum_{j=1}^{k} y_j^T \Lambda y_j. \tag{16}$$

- **Key observation**: the objective function of Problem (16) is *separable*, that is each jth term is function of only y_j !
- It is then allowed to maximize each $y_j^T \Lambda y_j$ over each y_j separately.
- **Attention**: ok but the y_j must orthonormal ![19]
- This is not so problematic actually, "all stars can be aligned".

[19] $\|y_j\|_2 = 1$, and $\langle y_i, y_j \rangle = 0$ for all $i \neq j$.

Larger Values of k

- Without loss of generality, assuming the diagonal entries of Λ are ordered such that $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \lambda_n \geqslant 0$, one may start by maximizing the first term.
- **Result ?**: well, exactly the one from the case $k = 1$, that is $y_1^\star = e_1$, and $v_1^\star = q_1$.
- Let us move to $j = 2$, we have to solve:

$$\underset{\|y_2\|_2=1, \langle y_1^\star, y_2 \rangle = 0}{\operatorname{argmax}} \quad y_2^T \Lambda y_2$$

- What do we learn from the constraint $\langle y_1^\star, y_2 \rangle = 0$ for all y_2?
  This: $y_{2_1} = 0$. Hence, we can reformulate the problem as follows:

$$\underset{\|y_2\|_2=1, y_{2_1} = 0}{\operatorname{argmax}} \quad y_2^T \Lambda y_2$$

- Following the same methodology for the case $k = 1$, we find $y_2^\star = e_2$ and $v_2^\star = q_2$. [20]
- Continuing step by step, we find $y_j^\star = e_j$ and $v_j^\star = q_j$ for $1 \leqslant j \leqslant k$.

[20] Do it to convince yourselves.

The Algorithm

The power iteration algorithm

Given matrix $A = X^T X$:

- Select random unit vector v_0
- For $i = 1, 2, ...$:
  1. Set $v_i = \frac{Av_{i-1}}{\|Av_{i-1}\|}$
  2. If $v_i \approx u_{i-1}$, then return v_i.

- To understand the geometric intuition behind the method, recall that if one views the matrix $A = X^T X$ as a function that maps the unit sphere to an ellipsoid, then the longest axis of the ellipsoid corresponds to the top eigenvector of A.
- Given that the top eigenvector corresponds to the direction in which multiplication by A stretches the vector the most, it is natural to hope that if we start with a random vector v, and keep applying A over and over, then we will end up having stretched v so much in the direction of A's top eigenvector that the image of v will lie almost entirely in this same direction.

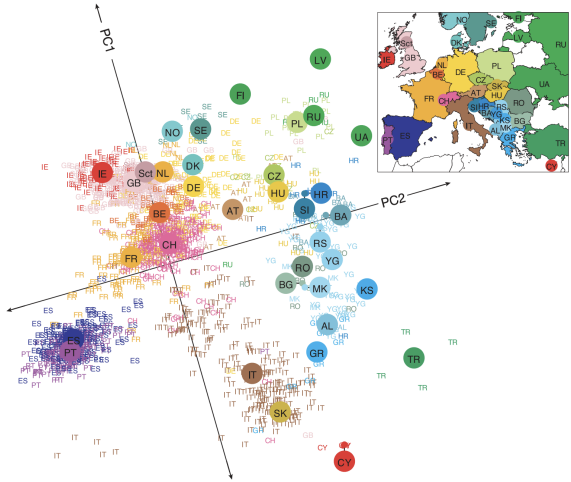
Infer where someone grew up from their DNA

- Novembre et al. considered a data set comprising 1387 Europeans (the rows). Each person's genome was examined in the same 200000 "SNPs" (the columns) — positions that tend to exhibit gene mutation. So in one of these positions, maybe 90% of all people have a "C," while 10% have an "A."
- In a nutshell, Novembre et al. apply PCA to this data set (with $m \approx 1400$, $n \approx 200000$, and $k = 2$) and plot the data according to the top two principal components v_1 and v_2 (using the exact same recipe as the PCA workflow, with the PC1- and PC2-axes corresponding to v_1 and v_2)[21]

[21] As usual with PCA, for best results non-trivial preprocessing of the data was necessary.

Infer where someone grew up from their DNA

Extracted from Ref[2]:



Infer where someone grew up from their DNA

- This plot depends on genomic data **only**.
- To **interpret** it: authors then color-coded each plotted point according to the country of origin of the corresponding person (this information was available in the meta-data).
- Check out the resulting plot: it's essentially a map of Europe!
  1. The first principal component v_1 corresponds roughly to the latitude of where someone's from.
  2. v_2 to the longitude (rotated 16 degrees counter-clockwise).
- That is, their genome "knows" where they're from! This suggests that many Europeans have bred locally for long enough for geographic information to be reflected in their DNA.
- Caution when analyzing data (especially genomic data) via PCA:
  1. the top principal components often only explain a small fraction of the overall variance in the data.
  2. true despite the fact that the top principal components do often correspond to clearly identifiable characteristics (such as geographic location in the above example).

Goodbye, So Soon

THANKS FOR THE ATTENTION

- v.leplat@innopolis.ru
- sites.google.com/view/valentinleplat/