Principal component analysis

Jianxin Wu

LAMDA Group
National Key Lab for Novel Software Technology
Nanjing University, China
wujx2001@gmail.com

February 11, 2020

Contents

1 Motivation **2**
1.1 Dimensionality and inherent dimensionality 2
1.2 Dimensionality reduction 4
1.3 PCA and subspace methods 5

2 PCA to zero-dimensional subspace **5**
2.1 The idea–formalization–optimization approach 6
2.2 A simple optimization 6
2.3 A few comments 7

3 PCA to one-dimensional subspace **7**
3.1 New formalization 8
3.2 Optimality condition and simplification 8
3.3 The eigen-decomposition connection 10
3.4 The solution 10

4 PCA for more dimensions **11**

5 The complete PCA algorithm **12**

6 Variance analysis **13**
6.1 PCA from maximization of variance 14
6.2 A simpler derivation 15
6.3 How many dimensions do we need? 15

7 When to use or not to use PCA? **16**
7.1 PCA for Gaussian data 16
7.2 PCA for non-Gaussian data 17
7.3 PCA for data with outliers 17

8 The whitening transform **18**

9 Eigen-decomposition vs. SVD **18**

Exercises **21**

In this chapter, we will introduce the Principal Component Analysis (PCA) technique. The key focus of this chapter is comprehensibility. We focus on the motivation and ideas behind PCA, rather than the mathematical details. However, we will maintain correctness of the equations. The goal is that an average non-math-major student with undergraduate mathematical background about linear algebra and some basic probability will read through this chapter without any difficulty. And, we will start from the motivation. 本章侧重理解

1 Motivation

PCA is a linear feature extraction method, and at the same time a linear *dimensionality reduction* technique, too. Hence, let us start from an examination of various aspects of the dimensionality of our data. PCA是一个线性特征提取方法 也是线性降维技术

1.1 Dimensionality and inherent dimensionality

潜在的

Let us consider some data which are two dimensional, and we will use (x, y) to denote one such data point. Hence, the (natural) dimensionality of our data is 2. In different scenarios 情景, we will also encounter data that exhibit 展示 different properties, which we illustrate in Figure 1. We discuss these examples one by one.

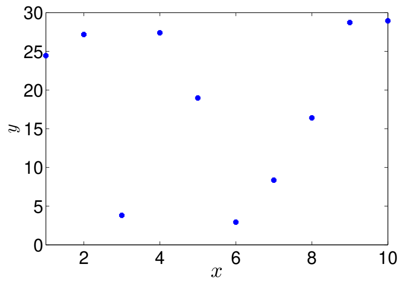
- The data set has two degrees of freedom. As illustrated by those points in Figure 1a, we need exactly two values (x, y) to specify any single data point in this case. Hence, we say that the data set has an *inherent dimensionality* of 2, which is the same as its natural dimensionality.

  In fact, the data points in this figure are generated in a way that makes y independent of x, as shown in Table 1, if we consider x and y as random variables. We cannot find a relationship among the ten 2-dimensional data points in this figure.

- The world, however, usually generates data that are not independent. In other words, x and y are dependent in most cases, especially considering those data we need to deal with in machine learning or pattern recognition. In our examples, we expect that x and y are usually correlated with each

自然维度=内在维度=2

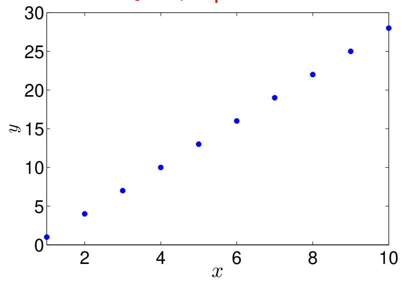
X,Y相互独立



(a) Random points in 2D

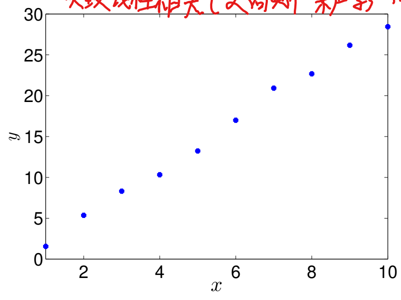
内在维度=1

X,y线性相关



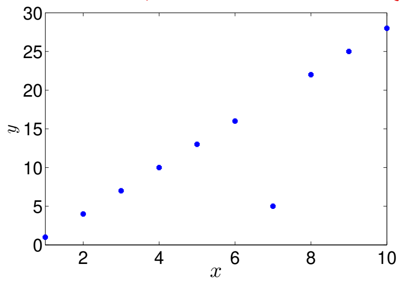
(b) A linear relationship

大致线性相关(受高斯噪声影响)



(c) A linear relationship with noise

由于离群点存在,无法说内在维度=1



(d) A linear relationship with outlier

内在维度=1

Figure 1: Illustration of various types of relationships between dimensions.

other. Figure 1b exhibits a special type of correlation: linear correlation. As shown by the data generation command in Table 1, y is a linear function of x.

Hence, in this figure, there is only one degree of freedom—once we know x, we can immediately determine the value of y; and vice versa. We only need one value to *completely* specify a pair (x, y), which could be x, y, or even *a linear combination of* x *and* y. Hence, we say that the inherent dimensionality of the data in this figure is 1, which is obviously smaller than its natural dimensionality. In the figure, the 10 points are aligned 排列 on one line, $y = 3x - 2$. In fact, as we will soon see, a linear combination of the original dimensions is the central part of PCA.

- The world, once again, is not as benign as what is shown in Figure 1b. In many cases, there exist obvious correlations among dimensions of the original data. However, the correlation is rarely a perfect linear relationship. As shown in Figure 1c, many people will agree that x and y are roughly related by a linear relationship (i.e., the 10 points are roughly aligned in a line), but with noticeable deviations.

  As shown in Table 1, there is indeed the same linear relationship between x and y as the one in Figure 1b, but in addition this is affected by Gaussian

Table 1: Matlab / Octave code to generate the data points in Figure 1.

Generating x	`x=1:10;`
Generating y for Fig. 1a	`y=rand(1,10)*30;`
Generating y for Fig. 1b	`y=3*x-2;`
Generating y for Fig. 1c	`y=3*x-2; y=y+randn(size(y))*2;`
Generating y for Fig. 1d	`y=3*x-2; y(7) = 5;`

noise. The noise is usually not welcome and we want to get rid of it. Hence, it is still reasonable to say that the data in Figure 1c has an inherent dimensionality of 1, because the number of meaningful degrees of freedom is still one.

- The world could be even more 敌对的 hostile to us, as shown by the example in Figure 1d. If we remove the 7th data point, we get a perfect linear relationship. This point is significantly different than the others, and we call it an *outlier*. It is difficult to say that the dataset in Figure 1d has an inherent dimensionality of 1, given the existence of the outlier.

  In other words, outliers seem more difficult to handle than noise. However, if we are able to remove the outlier using some 复杂的 sophisticated techniques, the inherent dimensionality of the remaining points is still 1.

1.2 Dimensionality reduction

In these four examples, we may safely conclude that the inherent dimensionality in both Figures 1b and 1c is 1. In other words, we require only one variable z to represent such data points. Finding a lower dimensional representation of the original (relatively) higher dimensional vector is called *dimensionality reduction*. In spite of the fact that fewer dimensions are used, we expect the dimensionality reduction process to keep the useful information in the original data.

The potential benefits of dimensionality reduction can be many-fold, with a few listed below.

- Lower resource requirements. A direct consequence of lower dimensionality is that less memory is needed to store the data, either in the main memory or on disk. An equally obvious benefit is that fewer CPU cycles are required.

- Removal of noise. As shown in Figure 1c, we can recover the linear relationship and reduce the dimensionality to 1. However, a benign side effect is that the noise may well be removed from the data in this process, which happens in PCA in many problems. Less noisy data usually leads to higher accuracy.

- Explanation and understanding. If the outcomes of dimensionality reduction happen to coincide well with the underlying hidden factors that generate the original data, the new lower-dimensional representation will be helpful in explaining how the data is generated and in understanding its properties.

  In addition, when we come across a new dataset, we can reduce it to two or three dimensions (by PCA or other methods). A visualization of the dimensionality reduced data will give us some hints on the properties of that dataset.

1.3 PCA and subspace methods

As far as we know, there is no precise definition for the inherent dimensionality of a dataset or a data generation process/distribution in general. In the above examples, we use the degrees of freedom of the data generation processes to describe their inherent dimensionality. Data generation, however, could be non-linear, which will be more complex than the linear relationship shown in Figure 1. But, since PCA only considers linear relationships, we will ignore non-linear relationships or non-linear dimensionality reduction methods in this chapter.

Now consider a vector $\boldsymbol{x} \in \mathbb{R}^D$. A linear relationship among its components can be expressed as a simple equation

$$\boldsymbol{x}^T \boldsymbol{w} + b = 0\,. \tag{1}$$

From basic linear algebra, we know that any $\boldsymbol{x}$ that satisfies Equation 1 resides in a subspace of $\mathbb{R}^D$ whose dimensionality is $D-1$. If there are more linear constraints on $\boldsymbol{x}$, $\boldsymbol{x}$ will reside in a subspace with even lower dimensionality.

Hence, the problem of linear dimensionality reduction can be seen as finding the linear constraints or finding the lower dimensional subspace of $\mathbb{R}^D$, which are also called subspace methods. Subspace methods differ from each other in how they find the constraints or subspaces. They have different evaluation metrics (e.g., which subspace is considered the best) or assumptions (e.g., do we know the category label of $\boldsymbol{x}$?).

PCA is possibly the simplest subspace method, which we will introduce in the coming sections.

2 PCA to zero-dimensional subspace

Let us start from an extreme case. What if the lower dimensional subspace is only a single point? In other words, what if its dimensionality is 0?

Suppose we are given a set of instantiations of $\boldsymbol{x}$,

$$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}\,,$$

which form the training set for us to learn PCA parameters. Note that there is a slight abuse of notation here. The symbol $\boldsymbol{x}$ may refer to a single data point.

However, it may be used to refer to the underlying distribution (or random variable) that generates the training set.

If there is no noise and a zero-dimensional subspace exists to represent this set, the only possibility is that

$$\boldsymbol{x}_1 = \boldsymbol{x}_2 = \cdots = \boldsymbol{x}_N \,.$$

We can use $\boldsymbol{x}_1$ to represent any example in X without requiring any additional information. Storing $\boldsymbol{x}_1$ requires D dimensions. However, the average number of dimensions needed for every $\boldsymbol{x}_i$ is only $\frac{D}{N}$. Because $\lim_{N\to\infty} \frac{D}{N} = 0$, it is reasonable to say that this is a zero-dimensional representation.

But, if noise exists, there will be $1 \le i < j \le N$ such that $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. How shall we find the *best* zero-dimensional representation in the presence of noise? We still need to find a vector $\boldsymbol{m}$ that represents every element in X. The key issue is: how shall we decide the optimality?

2.1 The idea–formalization–optimization approach

The *idea* can be inspired by the noise-free case. If we assume that the noise scale is small, we want to find an $\boldsymbol{m}$ that is *close to all the elements in* X. This choice has two nice properties. First, it coincides well with our intuition. Second, when the noise scale is 0, "close to" can be changed to "equal to," and degenerates to the noise free case.

The next step is to *formalize* this idea. It is natural to translate "close to" as "small distance," and translate "distance to all elements in X is small" to "the sum of distances to all elements in X is small." If the sum is small, of course every individual distance must be small.

Hence, we can write our idea precisely in mathematical language as

$$\boldsymbol{m}^\star = \operatorname*{arg\,min}_{\boldsymbol{m}} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{m}\|^2 \,, \tag{2}$$

where the right hand side is an optimization problem, and $\boldsymbol{m}^\star$ is the parameter that solves the optimization problem (which is the best $\boldsymbol{m}$ we seek).

The final step is: how can we get $\boldsymbol{m}^\star$, or, how to optimize Equation 2?

2.2 A simple optimization

With some background in vector calculus, it is easy to solve Equation 2. We can denote

$$J = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{m}\|^2 = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{m})^T (\boldsymbol{x}_i - \boldsymbol{m}) \,, \tag{3}$$

and get

$$\frac{\partial J}{\partial \boldsymbol{m}} = \frac{2}{N} \sum_{i=1}^{N} (\boldsymbol{m} - \boldsymbol{x}_i) \,, \tag{4}$$

where the partial derivative rules can be found in *The Matrix Cookbook*. Setting this term to 0 gives us the following optimality condition:

$$\boldsymbol{m} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \triangleq \bar{\boldsymbol{x}}\,. \tag{5}$$

That is, the best zero-dimensional representation is the average of all the training examples, which we denote by $\bar{\boldsymbol{x}}$.

2.3 A few comments

In spite of its simplicity, we have a few comments on the zero-dimensional reduction.

- When we encounter a new problem, the *idea–formalization–optimization* process can be very helpful. We first inspect the problem (maybe with some initial trials or visualization to understand the data property), which gives us some ideas to its solution.

  We then define proper notations and convert our ideas into a precise mathematical form, which often appears in an optimization problem format. The final step is to solve it, either by ourselves or by using the abundant tools that are available.

- It is also worth noting that Equation 2 is in fact slightly different from our idea. First, it is the squared distances that are summed, rather than the distances. Second, the term $\frac{1}{N}$ converts the sum into an average.

  The term $\frac{1}{N}$ will not change the optimization's solution. It is introduced following the tradition in the literature, and sometimes it helps simplify the optimization or reduce numerical difficulties. The change from distance to squared distance, as shown in the optimization, makes the optimization much easier. In addition, a small squared distance implies a small distance. Hence, our idea is still valid.

  It is always good to *tune the mathematical translation* so long as the tuning still matches our idea and makes the optimization easier to do.

- Although our idea starts from the assumption that the elements in X are the same data point subject to different noise, this assumption is *not* used at all in the formalization or optimization step.

  Thus, for *any* data set X, we can *generalize* and say that its average is the best zero-dimensional representation (under the sum of squared distance evaluation metric).

3 PCA to one-dimensional subspace

Now we are ready to move one step further to the one-dimensional subspace, where we can use one additional value to represent each $\boldsymbol{x}_i$ in addition to $\bar{\boldsymbol{x}}$.

3.1 New formalization

Any element $\boldsymbol{x} \in \mathbb{R}^D$ in a one-dimensional subspace can be represented as

$$\boldsymbol{x} = \boldsymbol{x}_0 + a\boldsymbol{w}$$

for some $a \in \mathbb{R}$, $\boldsymbol{x}_0 \in \mathbb{R}^D$, and $\boldsymbol{w} \in \mathbb{R}^D$, and vice versa, in which $\boldsymbol{x}_0$ and $\boldsymbol{w}$ are determined by the subspace and a is determined by the element $\boldsymbol{x}$ (recall your linear algebra and note that a one-dimensional subspace means a line!).

Now that we already have the zero-dimensional representation, we should set

$$\boldsymbol{x}_0 = \bar{\boldsymbol{x}} \,.$$

Hence, for any $\boldsymbol{x}_i$, the new one dimensional representation is a_i. Using this new representation, we can find an approximation of $\boldsymbol{x}_i$ as

$$\boldsymbol{x}_i \approx \bar{\boldsymbol{x}} + a_i\boldsymbol{w} \,,$$

and the difference (or residue),

$$\boldsymbol{x}_i - (\bar{\boldsymbol{x}} + a_i\boldsymbol{w}) \,,$$

is considered to be caused by noise, which we want to minimize. Note that now we do *not* need to require that $\boldsymbol{x}_i$ resides in a one-dimensional subspace.

The parameters we need to find are a_i $(1 \le i \le N)$ and $\boldsymbol{w}$. We denote

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_N)^T \,,$$

and define an objective J to minimize the average squared distance:

$$\begin{aligned} J(\boldsymbol{w}, \boldsymbol{a}) &= \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - (\bar{\boldsymbol{x}} + a_i\boldsymbol{w})\|^2 && (6) \\ &= \frac{1}{N}\sum_{i=1}^{N} \|a_i\boldsymbol{w} - (\boldsymbol{x}_i - \bar{\boldsymbol{x}})\|^2 && (7) \\ &= \sum_{i=1}^{N} \frac{a_i^2\|\boldsymbol{w}\|^2 + \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2 - 2a_i\boldsymbol{w}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})}{N} \,. && (8) \end{aligned}$$

3.2 Optimality condition and simplification

Now we calculate the partial derivatives, and set them to 0, as

$$\begin{aligned} \frac{\partial J}{\partial a_i} &= \frac{2}{N}\left(a_i\|\boldsymbol{w}\|^2 - \boldsymbol{w}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})\right) = 0 \quad \forall i \,, && (9) \\ \frac{\partial J}{\partial \boldsymbol{w}} &= \frac{2}{N}\sum_{i=1}^{N}\left(a_i^2\boldsymbol{w} - a_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}})\right) = 0 \,. && (10) \end{aligned}$$

Equation 9 gives us the solution for a_i, as

$$a_i = \frac{\boldsymbol{w}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})}{\|\boldsymbol{w}\|^2} = \frac{(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\boldsymbol{w}}{\|\boldsymbol{w}\|^2} \,. \tag{11}$$

Note that the projection of $\boldsymbol{x}_i - \bar{\boldsymbol{x}}$ onto $\boldsymbol{w}$ is $\frac{(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\boldsymbol{w}}{\|\boldsymbol{w}\|^2}\boldsymbol{w}$; we conclude that the optimal a_i can be viewed as the signed length of $\boldsymbol{x}_i - \bar{\boldsymbol{x}}$ projected onto $\boldsymbol{w}$. When the angle between $\boldsymbol{x}_i - \bar{\boldsymbol{x}}$ and $\boldsymbol{w}$ is larger than $90°$, $a_i \le 0$.

Before proceeding to process Equation 10, a further examination of $a_i\boldsymbol{w}$ shows that

$$a_i\boldsymbol{w} = \frac{\boldsymbol{w}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})\boldsymbol{w}}{\|\boldsymbol{w}\|^2} = \frac{(c\boldsymbol{w})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(c\boldsymbol{w})}{\|c\boldsymbol{w}\|^2} \,, \tag{12}$$

for any non-zero scalar value $c \in \mathbb{R}$. In other words, we have the freedom to specify that

$$\|\boldsymbol{w}\| = 1 \,, \tag{13}$$

and this additional constraint will not change the optimization problem's solution!

We choose $\|\boldsymbol{w}\| = 1$ because this choice greatly simplifies our optimization problem. Now we have

$$a_i = \boldsymbol{w}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) = (\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\boldsymbol{w} \,. \tag{14}$$

Plugging it back into the optimization objective, we get a much simplified version

$$J(\boldsymbol{w}, \boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N}\left[\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2 - a_i^2\right] \,, \tag{15}$$

by noting that $a_i\boldsymbol{w}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) = a_i^2$ and $a_i^2\|\boldsymbol{w}\|^2 = a_i^2$.

Hence, we know the optimal parameters are obtained via *maximizing*

$$\frac{1}{N}\sum_{i=1}^{N} a_i^2 \tag{16}$$

because $\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2$ does not depend on either $\boldsymbol{w}$ or $\boldsymbol{a}$.

One note we want to add is that various transformations can greatly simplify our optimization problem, and it is worthwhile to pay attention to such simplification opportunities. In fact, in this derivation, we could specify the constraint $\|\boldsymbol{w}\| = 1$ *before* finding the optimality conditions.

It is easy to observe that

$$J(\boldsymbol{w}, \boldsymbol{a}) = J\left(c\boldsymbol{w}, \frac{1}{c}\boldsymbol{a}\right) \tag{17}$$

for any $c \neq 0$, and there is no constraint on $\boldsymbol{w}$ or $\boldsymbol{a}$ in the original formulation. Hence, if $(\boldsymbol{w}^\star, \boldsymbol{a}^\star)$ is an optimal solution that minimizes J, so will be $(c\boldsymbol{w}^\star, \frac{1}{c}\boldsymbol{a}^\star)$

for any $c \neq 0$. That is, for an optimal solution $(\boldsymbol{w}^\star, \boldsymbol{a}^\star)$, $\left(\frac{1}{\|\boldsymbol{w}^\star\|}\boldsymbol{w}^\star, \|\boldsymbol{w}^\star\|\boldsymbol{a}^\star\right)$ will also be an optimal solution.

Obviously the norm of $\frac{1}{\|\boldsymbol{w}^\star\|}\boldsymbol{w}^\star$ is 1. Hence, we can specify $\|\boldsymbol{w}\| = 1$ without changing the optimization objective, but this will greatly simplify the optimization from the very beginning. It is always beneficial to find such simplifications and transformations before we attempt to solve the optimization task.

3.3 The eigen-decomposition connection

Now we turn our attention to Equation 10, which tells us that

$$\frac{1}{N}\left(\sum_{i=1}^{N} a_i^2\right)\boldsymbol{w} = \frac{1}{N}\sum_{i=1}^{N} a_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}})\,. \tag{18}$$

And, plugging a_i into Equation 18, we can simplify its right hand side as

$$\begin{aligned} \frac{1}{N}\sum_{i=1}^{N} a_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) &= \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})a_i \qquad (19)\\ &= \frac{\sum_{i=1}^{N}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\boldsymbol{w}}{N} \qquad (20)\\ &= \operatorname{Cov}(\boldsymbol{x})\boldsymbol{w}\,, \qquad (21) \end{aligned}$$

in which

$$\operatorname{Cov}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$$

is the covariance matrix of $\boldsymbol{x}$ computed from the training set X.

Hence, Equation 18 now gives us

$$\operatorname{Cov}(\boldsymbol{x})\boldsymbol{w} = \frac{\sum_{i=1}^{N} a_i^2}{N}\boldsymbol{w}\,, \tag{22}$$

which immediately reminds us of the eigen-decomposition equation—this equation tells us that the optimal $\boldsymbol{w}$ must be an eigenvector of $\operatorname{Cov}(\boldsymbol{x})$, and $\frac{\sum_{i=1}^{N} a_i^2}{N}$ is the corresponding eigenvalue!

The constraint in Equation 13 also fits in this eigen-decomposition interpretation, because eigenvectors are also constrained to have unit ℓ_2 norm.

3.4 The solution

$\operatorname{Cov}(\boldsymbol{x})$ has many eigenvectors and their corresponding eigenvalues. However, Equation 16 reminds us that we want to maximize $\frac{\sum_{i=1}^{N} a_i^2}{N}$, while Equation 22 tells us that $\frac{\sum_{i=1}^{N} a_i^2}{N}$ is the eigenvalue corresponding to $\boldsymbol{w}$. Hence, it is trivial to choose among all the eigenvectors—choose the one that corresponds to the largest eigenvalue!

Now we have everything to compute the one-dimensional reduction. From X we can compute the covariance matrix $\mathrm{Cov}(\boldsymbol{x})$, and we set $\boldsymbol{w}^\star = \boldsymbol{\xi}_1$, where $\boldsymbol{\xi}_1$ is the eigenvector of $\mathrm{Cov}(\boldsymbol{x})$ that corresponds to the largest eigenvalue. The optimal new one-dimensional representation for $\boldsymbol{x}_i$ is then

$$a_i^\star = \boldsymbol{\xi}_1^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})\,. \tag{23}$$

Given the one-dimensional representation, the original input $\boldsymbol{x}$ is approximated as

$$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + (\boldsymbol{\xi}_1^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_1\,. \tag{24}$$

Because $(\boldsymbol{\xi}_1^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_1$ equals the projection of $\boldsymbol{x}_i - \bar{\boldsymbol{x}}$ onto $\boldsymbol{\xi}_1$,[1] we also call $\boldsymbol{\xi}_1$ the first *projection direction*, and call $\boldsymbol{\xi}_1^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$ the projected value of $\boldsymbol{x}_i$ onto this direction.

4 PCA for more dimensions

Now we generalize PCA to two or more dimensions, thanks to the spectral decomposition of the covariance matrix.

It is obvious that $\mathrm{Cov}(\boldsymbol{x})$ is a real symmetric matrix, and furthermore it is a positive semi-definite matrix. According to matrix analysis theory, $\mathrm{Cov}(\boldsymbol{x})$ has D eigenvectors $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_D$ whose elements are all real numbers. And, the eigenvalues corresponding to them are $\lambda_1, \lambda_2, \ldots, \lambda_D$, which are all real numbers satisfying $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$. The spectral decomposition states that

$$\mathrm{Cov}(\boldsymbol{x}) = \sum_{i=1}^{D} \lambda_i \boldsymbol{\xi}_i \boldsymbol{\xi}_i^T\,. \tag{25}$$

The eigenvectors of real symmetric matrices satisfy that for any $i \neq j$, $\boldsymbol{\xi}_i^T \boldsymbol{\xi}_j = 0$ and $\|\boldsymbol{\xi}_i\| = 1$ for $1 \leq i \leq D, 1 \leq j \leq D$. Hence, if we construct a $D \times D$ matrix E, whose i-th column is formed by $\boldsymbol{\xi}_i$, we have

$$EE^T = E^T E = I\,. \tag{26}$$

Then, we can show that

$$\begin{align} \boldsymbol{x} &= \bar{\boldsymbol{x}} + (\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{27}\\ &= \bar{\boldsymbol{x}} + EE^T(\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{28}\\ &= \bar{\boldsymbol{x}} + (\boldsymbol{\xi}_1^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_1 + (\boldsymbol{\xi}_2^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_2 + \cdots + (\boldsymbol{\xi}_D^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_D\,, \tag{29} \end{align}$$

for any $\boldsymbol{x} \in \mathbb{R}^D$, even if $\boldsymbol{x}$ does not follow the same relationships inside the training set X (in other words, being an outlier!).

Comparing Equation 24 with Equation 29, a guess of PCA with more dimensions naturally comes to us: $\boldsymbol{\xi}_i$ should be the i-th projection direction, and the coefficient is $\boldsymbol{\xi}_i^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$.

This conjecture is correct, and is easy to prove following the procedure in Section 3. We will omit the details here, but leave that to the reader.

[1]Note that the projection of $\boldsymbol{x}$ onto $\boldsymbol{y}$ is $\frac{\boldsymbol{x}^T\boldsymbol{y}}{\|\boldsymbol{y}\|^2}\boldsymbol{y}$, and $\|\boldsymbol{\xi}_1\| = 1$.

5 The complete PCA algorithm

The complete principal component analysis algorithm is described in Algorithm 1.

Algorithm 1 The PCA algorithm

```
1: Input: a $D$-dimensional training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ and the new (lower) dimensionality $d$ (with $d < D$).
2: Compute the mean
     $$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \,.$$
3: Compute the covariance matrix
     $$\operatorname{Cov}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \,.$$
4: Find the spectral decomposition of $\operatorname{Cov}(\boldsymbol{x})$, obtaining the eigenvectors $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \dots, \boldsymbol{\xi}_D$ and their corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_D$. Note that the eigenvalues are sorted, such that $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_D \ge 0$.
5: For any $\boldsymbol{x} \in \mathbb{R}^D$, its new lower dimensional representation is
     $$\boldsymbol{y} = \left(\boldsymbol{\xi}_1^T(\boldsymbol{x} - \bar{\boldsymbol{x}}), \boldsymbol{\xi}_2^T(\boldsymbol{x} - \bar{\boldsymbol{x}}), \dots, \boldsymbol{\xi}_d^T(\boldsymbol{x} - \bar{\boldsymbol{x}})\right)^T \in \mathbb{R}^d \,, \quad (30)$$
   and the original $\boldsymbol{x}$ can be approximated by
     $$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + (\boldsymbol{\xi}_1^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_1 + (\boldsymbol{\xi}_2^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_2 + \cdots + (\boldsymbol{\xi}_d^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_d \,. \quad (31)$$
```

Let E_d be the $D \times d$ matrix which consists of the first d columns in E; the new lower dimensional representation can be succinctly written as

$$\boldsymbol{y} = E_d^T(\boldsymbol{x} - \bar{\boldsymbol{x}}) \,, \tag{32}$$

and the approximation is

$$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + E_d E_d^T(\boldsymbol{x} - \bar{\boldsymbol{x}}) \,. \tag{33}$$

Dimensions of the new representation are called the *principal components*, hence the name Principal Component Analysis (PCA). Sometimes a typo will happen, which spells PCA as Princip$\underline{\text{le}}$ Component Analysis, but that is not correct.

6 Variance analysis

Note that we have used $\boldsymbol{y}$ to denote the new lower dimensional representation in Algorithm 1. Let y_i be the i-th dimension of $\boldsymbol{y}$; we can compute its expectation:

$$\mathbb{E}[y_i] = \mathbb{E}\left[\boldsymbol{\xi}_i^T(\boldsymbol{x} - \bar{\boldsymbol{x}})\right] = \boldsymbol{\xi}_i^T \mathbb{E}\left[\boldsymbol{x} - \bar{\boldsymbol{x}}\right] = \boldsymbol{\xi}_i^T \mathbf{0} = 0\,, \tag{34}$$

where $\mathbf{0}$ is a vector whose elements are all zero.

We can further calculate its variance:

$$\begin{aligned}
\operatorname{Var}(y_i) &= \mathbb{E}[y_i^2] - (\mathbb{E}[y_i])^2 && (35)\\
&= \mathbb{E}[y_i^2] && (36)\\
&= \mathbb{E}\left[\boldsymbol{\xi}_i^T(\boldsymbol{x} - \bar{\boldsymbol{x}})\boldsymbol{\xi}_i^T(\boldsymbol{x} - \bar{\boldsymbol{x}})\right] && (37)\\
&= \mathbb{E}\left[\boldsymbol{\xi}_i^T(\boldsymbol{x} - \bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^T\boldsymbol{\xi}_i\right] && (38)\\
&= \boldsymbol{\xi}_i^T \operatorname{Cov}(\boldsymbol{x})\boldsymbol{\xi}_i && (39)\\
&= \boldsymbol{\xi}_i^T \left[\operatorname{Cov}(\boldsymbol{x})\boldsymbol{\xi}_i\right] && (40)\\
&= \boldsymbol{\xi}_i^T \left(\lambda_i\boldsymbol{\xi}_i\right) && (41)\\
&= \lambda_i\boldsymbol{\xi}_i^T\boldsymbol{\xi}_i && (42)\\
&= \lambda_i\,. && (43)
\end{aligned}$$

Hence, y_i is zero-mean and its variance is λ_i.

Equation 22 tells us that for the first new dimension,

$$\frac{\sum_{i=1}^N a_i^2}{N} = \lambda_1\,. \tag{44}$$

And, Equation 15 tells us that

$$J(\boldsymbol{\xi}_1) = \frac{1}{N}\sum_{i=1}^N \left[\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2 - a_i^2\right] = \frac{\sum_{i=1}^N \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2}{N} - \lambda_1\,. \tag{45}$$

That is, $\frac{\sum_{i=1}^N \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2}{N}$ is the average squared distance for the zero dimensional representation, and λ_1 is the part of cost that is reduced by introducing the $\boldsymbol{\xi}_1$ projection direction as a new dimension. It is also easy to prove that

$$J(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_k) = \frac{\sum_{i=1}^N \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2}{N} - \lambda_1 - \lambda_2 - \cdots - \lambda_k \tag{46}$$

for $1 \le k \le D$ and

$$J(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_D) = 0\,. \tag{47}$$

Hence, every new dimension is responsible for reducing the reconstruction distance between any point $\boldsymbol{x}$ and its approximation $\bar{\boldsymbol{x}} + \sum_{i=1}^k(\boldsymbol{\xi}_i^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_i$. And, we know that the i-th new dimension reduces the expected squared distance by λ_i, and that distance will reduce to 0 if all eigenvectors are used. From these observations, we get

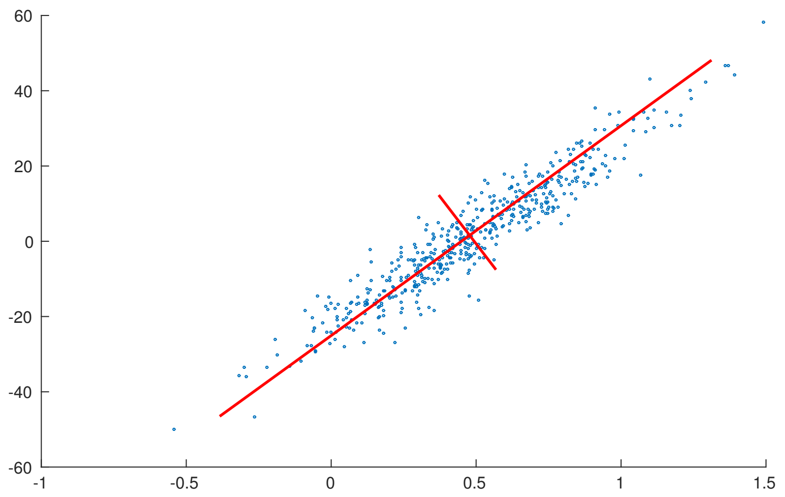


Figure 2: Variance of projected values. ()

- If all eigenvectors are used, PCA is simply a shift plus a *rotation*, because

  $$\boldsymbol{y} = E^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

  for any $\boldsymbol{x}$, and E is an orthogonal matrix. We also know that in this case the norm is unchanged:

  $$\|\boldsymbol{y}\| = \|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \,.$$

- We know that the larger an eigenvalue, the more its associated eigenvector (projection direction) will reduce the approximation error.

- We also know that the eigenvalue λ_i is the expectation of the square of the i-th new dimension (cf. Equation 43). Hence, we will expect the average scale of y_i (the i-th dimension in $\boldsymbol{y}$) to be larger than that of y_j if $i < j$.

6.1 PCA from maximization of variance

Based on the above observations, we can also interpret PCA as maximizing the variance of the projected values onto a certain direction based on Equation 43. This perspective is illustrated in Figure 2.

The two red lines show the two eigenvectors of the covariance matrix. It is not surprising that the long red line has the highest variance of its projected values. In other words, PCA can also be derived by maximizing the variance of the projected values onto a projection direction, and the optimal solution of this formulation must be the same as what we obtain from the minimization of average squared distances.

Hence, we can use the following terms interchangeably because they are *equivalent in the context of PCA*: variance (of projected values), eigenvalue, and reduction in approximation error.

6.2 A simpler derivation

Let us just work out the first projection direction that maximizes the variance of the projected values. Given any projection direction $\boldsymbol{w}$, the projected point of a data point $\boldsymbol{x}$ onto $\boldsymbol{w}$ will be (cf. footnote 1 in page 11)

$$\frac{\boldsymbol{x}^T\boldsymbol{w}}{\|\boldsymbol{w}\|^2}\boldsymbol{w} = \boldsymbol{x}^T\boldsymbol{w}\boldsymbol{w}\,. \tag{48}$$

In the above formula, we assume $\|\boldsymbol{w}\| = 1$. In the minimal reconstruction error formulation, we add this restriction because the norm of $\boldsymbol{w}$ does not change our optimization.

Here, $\boldsymbol{w}$ is a projection *direction*, and hence its length is irrelevant. We can then add the same constraint $\|\boldsymbol{w}\| = 1$. Hence, the projected value for $\boldsymbol{x}$ is $\boldsymbol{x}^T\boldsymbol{w}$. The mean of all projected values is

$$\mathbb{E}[\boldsymbol{x}^T\boldsymbol{w}] = \bar{\boldsymbol{x}}^T\boldsymbol{w}\,.$$

Next, we compute the variance of all projected values, as

$$\begin{aligned}
\operatorname{Var}(\boldsymbol{x}^T\boldsymbol{w}) &= \mathbb{E}\left[(\boldsymbol{x}^T\boldsymbol{w} - \bar{\boldsymbol{x}}^T\boldsymbol{w})^2\right] && (49)\\
&= \boldsymbol{w}^T\mathbb{E}\left[(\boldsymbol{x}-\bar{\boldsymbol{x}})(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\right]\boldsymbol{w} && (50)\\
&= \boldsymbol{w}^T\operatorname{Cov}(\boldsymbol{x})\boldsymbol{w}\,. && (51)
\end{aligned}$$

This is the second time we have seen this optimization problem:

$$\begin{aligned}
\max_{\boldsymbol{w}} \quad & \boldsymbol{w}^T\operatorname{Cov}(\boldsymbol{x})\boldsymbol{w} && (52)\\
\text{subject to} \quad & \|\boldsymbol{w}\| = 1\,. && (53)
\end{aligned}$$

This is sufficient evidence that the variance maximization perspective leads to exactly the same PCA solution as the one we just obtained by minimizing the approximation error. The variance perspective, however, is much easier for deriving the PCA solution. The approximation perspective, although requiring a little more effort, reveals more properties of PCA, such as the relationship between the eigenvalue, approximation error, and variance. Hence, we started from the approximation perspective in this chapter.

6.3 How many dimensions do we need?

The equivalence of these terms also gives us hints on how to choose d, the number of dimensions in the new representation.

If one eigenvalue is 0, then its corresponding eigenvector (projection direction) is not useful in keeping information of the original data distribution at all. All data points that have the same characteristic as the training set will have a constant projected value for this eigenvector because the variance of the projected values equals the eigenvalue, which is 0. Hence, this eigenvalue and its associated eigenvector can be safely removed.

When an eigenvalue is quite small, there is good reason to conjecture that this particular projection direction does not contain useful information either. Rather, it could be there due to white (or other types of) noise, as illustrated in the example of Figure 1c. Hence, removing such directions is usually encouraged and in many cases will increase the accuracy of our new representation in subsequent classification systems (or other tasks).

We want to keep a reasonably large portion of the variance, such that the rest of the eigenvalues/variance are small enough and likely caused by noise. Hence, the rule of thumb is often to choose to cut off if the accumulated eigenvalues has exceeded 90% of the sum of all eigenvalues. In other words, we choose d to be the first integer that satisfies

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_d}{\lambda_1 + \lambda_2 + \cdots + \lambda_D} > 0.9\,. \tag{54}$$

Although 0.9 seems to be the widely used cutoff threshold, other values (such as 0.85 or 0.95) can also be used.

7 When to use or not to use PCA?

Discussions on this question will end this chapter's presentation on PCA. And, before we tackle this tough question, we start with a simpler one: How will PCA affect $\boldsymbol{x}$ if the latter follows a normal distribution?

7.1 PCA for Gaussian data

Let us suppose $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \Sigma)$. Usually we do not know the exact value of the mean $\boldsymbol{\mu}$ or the covariance matrix Σ. However, the maximum likelihood estimation of these terms can be obtained as $\bar{\boldsymbol{x}}$ and $\operatorname{Cov}(\boldsymbol{x})$, respectively.[2]

Let Λ denote the diagonal matrix consisting of the eigenvalues of $\operatorname{Cov}(\boldsymbol{x})$, i.e.,

$$\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)\,.$$

Following the properties of normal distributions,[3] it is easy to verify that the new PCA representation $\boldsymbol{y}$ is also normally distributed, with parameters estimated as

$$\boldsymbol{y} \sim N(\mathbf{0}, \Lambda)\,, \tag{55}$$

if all projection directions are used. That is, PCA performs a translation with respect to $\bar{\boldsymbol{x}}$, followed by a rotation such that the axes of the normal distribution are parallel to the coordinate system's axes.

[2]We will discuss maximum likelihood estimation in Chapter 8.

[3]See Chapter 13 for more details.

Table 2: PCA outcome for data in Figure 1b, 1c, and 1d.

Data source	λ_1	λ_2	projection direction
Figure 1b	825	0	(3.00,1)
Figure 1c	770	0.75	(2.90,1)
Figure 1d	859	16.43	(3.43,1)

A direct consequence of this result is that different components of $\boldsymbol{y}$ are *independent* of each other because different dimensions of an ellipsoidal normal distribution are independent.

If only the first d eigenvectors are used, then we can define a $d \times d$ matrix Λ_d such that $\Lambda_d = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$, and

$$\boldsymbol{y}_d \sim N(\mathbf{0}, \Lambda_d) \,. \tag{56}$$

The projected dimensions are independent of each other, too. Figure 3 shows an example, in which Figure 3a contains 2000 normally distributed two-dimensional data points, generated in Matlab/Octave by `x=randn(2000,2)*[2 1;1 2]`. After the PCA operation, Figure 3b shows that these data points are rotated back to follow an ellipsoidal normal distribution (i.e., one whose covariance matrix is diagonal).

7.2 PCA for non-Gaussian data

We will, however, expect many non-Gaussian datasets. Figure 4 shows the eigenvalue distribution of a non-Gaussian dataset.

We can observe an *exponential* trend in how the eigenvalues decrease. Because of this exponential decay trend, the first few eigenvalues may quickly accumulate a high proportion of the total variance (or, sum of eigenvalues). Hence, when the eigenvalues show an exponential decay trend, the last few dimensions may be noise, and it is reasonable to apply PCA to such data.

In Figure 4, the last three eigenvalues only account for 6% of the total variance. We can safely set $d = 7$ in this example (where $D = 10$).

If the data are not Gaussian, $\boldsymbol{y}$ will have mean $\mathbf{0}$ and covariance matrix Λ (or Λ_d) after PCA. Thus, we know the dimensions in $\boldsymbol{y}$ are not correlated. However, they are not necessarily independent because $\boldsymbol{y}$ is not a multivariate normal distribution if $\boldsymbol{x}$ is non-Gaussian.

7.3 PCA for data with outliers

Outliers can cause serious trouble for PCA. We compute PCA for the data used in Figure 1. In Table 2, we listed the PCA results for the data in the last three subfigures of Figure 1.

When there is no noise, PCA successfully estimates the major projection direction as $(3, 1)$, which fits Figure 1b. The white noise is applied to every data point in Figure 1c. However, it only slightly increases λ_2, from 0 to 0.75, and slightly changes the projection direction to $(2.9, 1)$. But, one single outlier in the data of Figure 1d significantly changes the projection direction to $(3.43, 1)$ and leads to a large λ_2 (16.43). Overall, PCA is not effective in the existence of outliers.

8 The whitening transform

Sometimes we have reasons to require that the dimensions in $\boldsymbol{y}$ have roughly the same scale. However, PCA ensures that

$$\mathbb{E}[y_1^2] \geq \mathbb{E}[y_2^2] \geq \cdots \geq \mathbb{E}[y_d^2] \,.$$

The whitening transform is a simple variation of PCA, and can achieve this goal.

The whitening transform derives the new lower dimensional representation as

$$\boldsymbol{y} = (E_d \Lambda_d^{-1/2})^T (\boldsymbol{x} - \bar{\boldsymbol{x}}) \,. \tag{57}$$

This equation differs from Equation 32 by an additional term $\Lambda_d^{-1/2}$, which guarantees that

$$\mathbb{E}[y_1^2] = \mathbb{E}[y_2^2] = \cdots = \mathbb{E}[y_d^2]$$

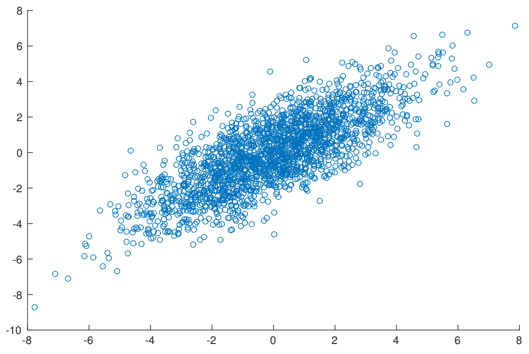
after the whitening transform. However, we have to remove any projection direction whose corresponding eigenvalue is 0 in the whitening transform.

As shown in Figure 3c, after the whitening transform the dataset follows a spherical normal distribution.

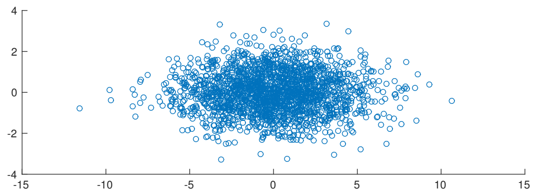
9 Eigen-decomposition vs. SVD

When either the number of data points N or the dimensionality D is large, especially when D is large, eigen-decomposition could be computationally very expensive. Singular Value Decomposition (SVD) is usually used to replace eigen-decomposition in this scenario.

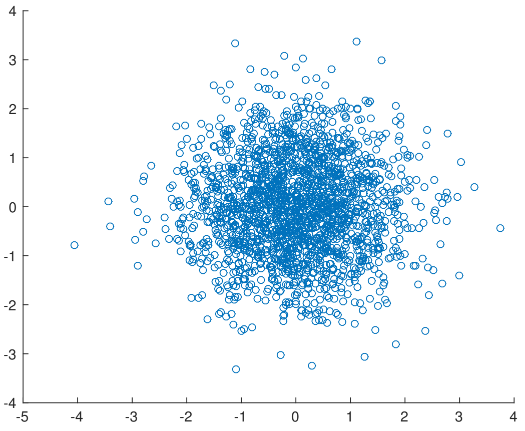
The covariance matrix $\text{Cov}(\boldsymbol{x})$ does not need to be explicitly computed. Simply using the data matrix X, SVD can compute the (left and right) singular vectors and singular values. Depending on whether $N > D$ or $D > N$, the eigenvectors of $\text{Cov}(\boldsymbol{x})$ will match either the left or right singular vectors. And, the singular values, when squared, will match the eigenvalues.



(a) Input data



(b) PCA



(c) Whitening

Figure 3: PCA and the whitening transform applied to Gaussian data. Figure 3a is the 2-dimensional input data. After PCA, the data are rotated such that two major axes of the data are parallel to the coordinate system's axes in Figure 3b (i.e., the normal distribution becomes an ellipsoidal one). After the whitening transform, the data has the same scale in the two major axes in Figure 3c (i.e., the normal distribution becomes a spherical one). Note that the x- and y-axes in the different subfigures have different scales.

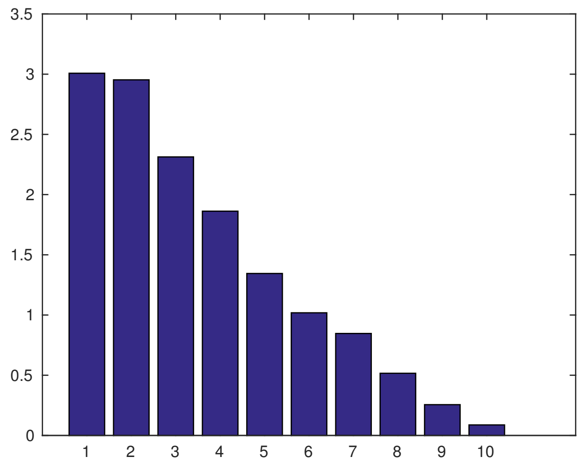


Figure 4: Eigenvalues shown in decreasing order.

Exercises

1. Let X be an $m \times n$ matrix with singular value decomposition

$$X = U\Sigma V^T\,,$$

where $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)})^T$ contains the singular values of X.

(a) What are the eigenvalues and eigenvectors of XX^T?

(b) What are the eigenvalues and eigenvectors of X^TX?

(c) What is the relationship between the eigenvalues of XX^T and X^TX?

(d) What is the relationship between the singular values of X and the eigenvalues of XX^T (X^TX)?

(e) If $m = 2$ and $n = 100\,000$, how will you compute the eigenvalues of X^TX?

2. We study the effect of the average vector in PCA in this problem. Use the following Matlab/Octave code to generate a dataset with 5000 examples and compute their eigenvectors. If we forget to transform the data by minus the average vector from every example, is there a relationship between the first eigenvector (i.e., the one associated with the largest eigenvalue) and the average vector?

Observe these vectors while changing the value of `scale` in the set

$$\{1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}\,.$$

What is the correct eigenvector (in which the average vector is removed from every example) if `scale` changes?

```
% set the random number seed to 0 for reproducibility
rand('seed',0);
avg = [1 2 3 4 5 6 7 8 9 10];
scale = 0.001;
% generate 5000 examples, each 10 dim
data = randn(5000,10)+repmat(avg*scale,5000,1);

m = mean(data);  % average
m1 = m / norm(m); % normalized avearge

% do PCA, but without centering
[~, S, V] = svd(data);
S = diag(S);
e1 = V(:,1); % first eigenvector, not minus mean vector
% do correct PCA with centering
newdata = data - repmat(m,5000,1);
[U, S, V] = svd(newdata);
S = diag(S);
new_e1 = V(:,1); % first eigenvector, minus mean vector

% correlation between first eigenvector (new & old) and mean
avg = avg - mean(avg);
avg = avg / norm(avg);
e1 = e1 - mean(e1);
e1 = e1 / norm(e1);
new_e1 = new_e1 - mean(new_e1);
new_e1 = new_e1 / norm(new_e1);
corr1 = avg*e1
corr2 = e1'*new_e1
```

3. Complete the following experiments using Matlab or GNU Octave. Write your own code to implement both PCA and whitening—you can use functions such as `eig` or `svd`, but do not use functions that directly complete the task for you (e.g., the `princomp` function).

   (a) Generate 2000 examples using `x=randn(2000,2)*[2 1;1 2]`, in which the examples are two-dimensional. Use the `scatter` function to plot these 2000 examples.

   (b) Perform the PCA operation on these examples and keep both dimensions. Use the `scatter` function to plot the examples after PCA.

   (c) Perform the whitening operation on these examples and keep both dimensions. Use the `scatter` function to plot the examples after PCA.

   (d) Why is PCA a rotation (after the translation operation) of the data if all dimensions are kept in the PCA operation? Why is this operation useful?

4. (Givens rotations) Givens rotations (which are named after James Wallace Givens, Jr., a US mathematician) are useful in setting certain elements in a vector to 0. A Givens rotation involves two indexes i, j and an angle θ, which generates a matrix of the form:

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{58}$$

   in which $c = \cos\theta$ and $s = \sin\theta$. The diagonal entries in the matrix $G(i,j,\theta)$ are all 1, except $G(i,j,\theta)_{i,i} = G(i,j,\theta)_{j,j} = c$. Most of the off-diagonal entries are 0, except $G(i,j,\theta)_{i,j} = s$ and $G(i,j,\theta)_{j,i} = -s$. Let $G(i,j,\theta)$ be of size $m \times m$ and $\boldsymbol{x} \in \mathbb{R}^m$.

   (a) What is the effect of left multiplying $\boldsymbol{x}$ by $G(i,j,\theta)^T$? That is, what is the difference between $\boldsymbol{x}$ and $\boldsymbol{y} = G(i,j,\theta)^T\boldsymbol{x}$?

   (b) If we want to enforce $y_j = 0$, what is your choice of θ (or equivalently, of c and s values)?

   (c) Evaluation of trigonometric functions or their inverse functions is expensive. Without using trigonometric functions, how could you determine the matrix $G(i,j,\theta)$? That is, how to determine the values c and s?

(d) If $G(i,j,\theta)^T$ left multiplies a matrix A of size $m \times n$, how does it alter A? How can we use a Givens rotation to change one entry of a matrix A to 0? What is the computational complexity of applying this transformation?

(e) (QR decomposition) Let A be a real matrix of size $m \times n$. Then, there exists an orthogonal real matrix Q (of size $m \times m$) and an upper triangular real matrix R (of size $m \times n$),[4] such that

$$A = QR\,.$$

This decomposition for any real matrix A is called the *QR decomposition*. How would you make use of Givens rotations to produce a QR decomposition?

5. (Jacobi method) One method to calculate the principal components is Jacobi's approach, invented by and named after Carl Gustav Jacob Jacobi, a German mathematician.

   Let X be a real symmetric matrix of size $n \times n$.

   (a) If G is an orthogonal $n \times n$ real matrix and $\boldsymbol{x} \in \mathbb{R}^n$, prove that $\|\boldsymbol{x}\| = \|G\boldsymbol{x}\|$ and $\|\boldsymbol{x}\| = \|G^T\boldsymbol{x}\|$ (i.e., a rotation will not change the length of a vector).

   (b) The Frobenius norm of an $m \times n$ matrix A is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j}^2} = \sqrt{\operatorname{tr}(AA^T)}\,.$$

   If G is an orthogonal $n \times n$ real matrix, and X is any real matrix of size $n \times n$, prove that

$$\|X\|_F = \|G^T X G\|_F\,.$$

   (c) For a real symmetric matrix X, Jacobi defined a loss function for it as the Frobenius norm of the off-diagonal elements in X—i.e.,

$$\texttt{off}(X) = \sqrt{\sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} x_{i,j}^2}\,.$$

   The basic building block in the Jacobi method for eigen-decomposition is to find an orthogonal matrix J such that

$$\texttt{off}(J^T X J) < \texttt{off}(X)\,.$$

[4] R is an upper triangular matrix if any entry below the main diagonal is 0, i.e., $R_{ij} = 0$ so long as $i > j$.

Explain why this basic step is useful in finding the eigenvectors and eigenvalues of X?

(d) How do you choose an orthogonal matrix J such that the (i, j)-th and (j, i)-th entries in $J^T XJ$ are both zeros $(i \neq j)$? (Hint: Let $J(i, j, \theta)$ be a Givens rotation matrix)

(e) The classical Jacobi method iterates between the following steps:

i. Find one off-diagonal entry in X that has the largest absolute value—i.e.,

$$|x_{pq}| = \max_{i \neq j} |x_{ij}|\,;$$

ii.

$$X \leftarrow J(p, q, \theta)^T XJ(p, q, \theta)\,.$$

This process converges if $\texttt{off}(X)$ is smaller than a predefined threshold ϵ. Prove that one iteration will not increase $\texttt{off}(X)$.

(f) Given the specific choice in the classical Jacobi method, prove that it always converges.