Data Sets

Linear Geometry

Principal Components

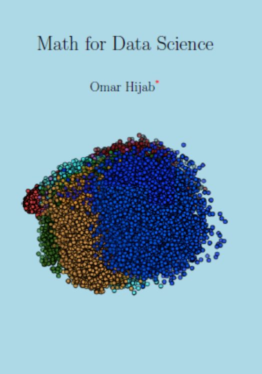
Mathematics for Data Science

Dr. S. M. Moosavi

smohsenmoosavi2009@gmail.com

November 10, 2024

The following slides are arranged (with some modifications) based on the book "*Math for Data Science*" by "**Omar Hijab**".



You can follow me on Linkedin. Also, for course materials such as slides and the related python codes, see this Github repository.

Outline

1. Data Sets
2. Linear Geometry
3. Principal Components

Outline

1. Data Sets
2. Linear Geometry
3. Principal Components

What is a dataset?

Definition 1.1

Geometrically, a dataset is a sample of N points $x_1, x_2, \cdots, x_N$ in d-dimensional space $\mathbb{R}^d$. Algebraically, a dataset is an N × d matrix.

Practically speaking, the following are all representations of datasets:

$$\text{matrix} = \text{CSV file} = \text{spreadsheet} = \text{SQL table} = \text{array} = \text{dataframe}$$

Definition 1.2

Each point $x = (t_1, t_2, \cdots, t_d)$ in the dataset is a sample or an example, and the components $t_1, t_2, \cdots, t_d$ of a sample point x are its features or attributes. As such, d-dimensional space $\mathbb{R}^d$ is feature space.

Definition 1.3

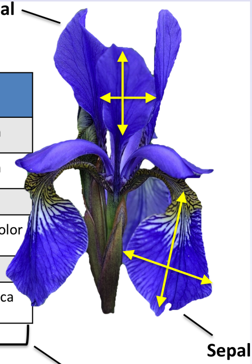
Sometimes one of the features is separated out as the label. In this case, the dataset is a labelled dataset.

Iris dataset

The *Iris dataset* contains 150 examples of four features of Iris flowers, and there are three classes of Irises, *Setosa*, *Versicolor* and *Virginica*, with 50 samples from each class.

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica



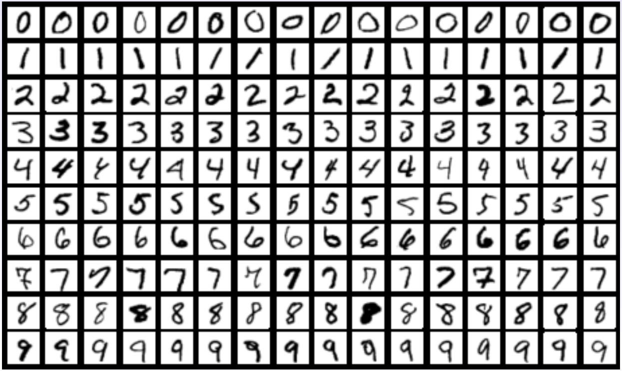
Class labels
(targets)

Features
(attributes, measurements, dimensions)

MNIST dataset

The *MNIST dataset* consists of $60,000$ images of hand-written digits. There are 10 classes of images, corresponding to each digit $0, 1, \cdots, 9$. We seek to compress the images while preserving as much as possible of the images' characteristics.

Each image is a grayscale 28×28 pixel image. Since $28^2 = 784$, each image is a point in $d = 784$ dimensions. Here there are $N = 60000$ samples and $d = 784$ features.



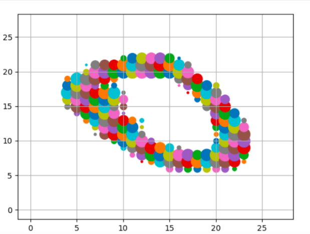
Exercises

Exercise 1.1

Use `sklearn` *to download Iris dataset.*

Exercise 1.2

- *From* `keras` *read the MNIST dataset.*
- *Let* `(train_X, train_y), (test_X, test_y) = mnist.load_data()`
- *Let* `pixels = train_X[1]`.
- *Do* `for` *loops over* `i` *and* `j` *in* `range(28)` *and use* `scatter` *to plot points at location* `(i,j)` *with size given by* `pixels[i,j]`, *then show the following image.*



Data Sets
Introduction
Averages and Vector Spaces
Two Dimensions
Complex Numbers
Mean and Covariance
Linear Geometry
Principal Components

Introduction

Suppose we have a population of things (people, tables, numbers, vectors, images, etc.) and we have a sample of size N from this population:

```
l = [x_1,x_2,...,x_N]
```

The total population is the *population* or the *sample space*.

Example 1.1

The sample space consists of all real numbers and we take $N = 5$ samples from

```
l = [3.95, 3.20, 3.10, 5.55, 6.93]
```

Example 1.2

The sample space consists of all integers and we take $N = 5$ samples from

```
l = [35, -32, -8, 45, -8]
```

Introduction

Example 1.3

The sample space consists of all Python strings and we take $N = 5$ samples from

```
l = ['a2e?','#%T','7y5,','kkk>><</','[[)*+']
```

Example 1.4

The sample space consists of all HTML colors and we take $N = 5$ samples from

```
from random import choice
import matplotlib.pyplot as plt

def hexcolor():
  return "#" + ''.join([choice('0123456789abcdef') for _ in range(6)])

for i in range(5): plt.scatter(i,0, c=hexcolor())
plt.show()
```

Mean

Let `l` be a list as above. The goal is to compute the sample *average* or *mean* of the list, which is

$$mean = average = \frac{x_1 + x_2 + \cdots + x_N}{N}.$$

In the Example (1.1), the average is

$$\frac{3.95 + 3.20 + 3.10 + 5.55 + 6.93}{5} = 4.546.$$

Example 1.5

```python
import numpy as np

dataset = np.array([3.95, 3.20, 3.10, 5.55, 6.93])
print(np.mean(dataset))

output: 4.546
```

In the Example (1.2), the average is $\frac{32}{5}$. In the Example (1.3), while we can add strings, we can't divide them by 5, so the average is undefined. Similarly for colors: the average is undefined.

Vector space

A sample space or population V is called a *vector space* if, roughly speaking, one can compute means or averages in V . In this case, we call the members of the population "*vectors*".

Definition 1.4 (Vector space)

Let V be a set. V is a vector space (over $\mathbb{R}$) if for every $u, v, w \in V$ and $r, s \in \mathbb{R}$:

1. *vectors can be added (and the sum $v + w$ is back in V);*
2. *vector addition is commutative $v + w = w + v$*
3. *vector addition is associative $u + (v + w) = (u + v) + w$;*
4. *there is a zero vector $\mathbf{0}$ ($\mathbf{0} + v = v$);*
5. *vectors v have negatives (or opposites) $-v$ ($v + (-v) = \mathbf{0}$);*
6. *vectors can be multiplied by real numbers (and the product rv is back in V);*
7. *multiplication is distributive over addition $(r + s)v = rv + sv$ and $r(u + v) = ru + rv$;*
8. *$1v = v$ and $0v = \mathbf{0}$;*
9. *$r(sv) = (rs)v$.*

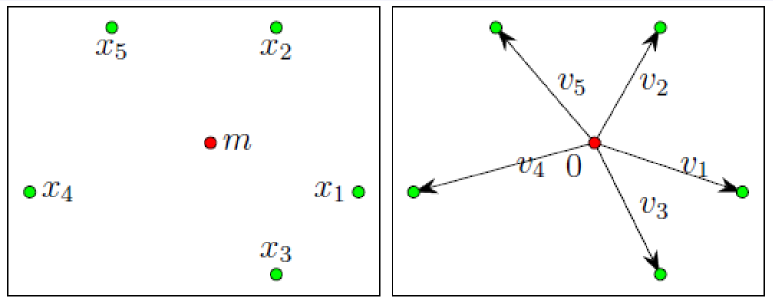
Centered dataset

Definition 1.5 (Centered Versus Non-Centered)

If $x_1, x_2, \cdots, x_N$ *is a dataset of points with mean* m *and*

$$v_1 = x_1 - m, v_2 = x_2 - m, \cdots, v_N = x_N - m,$$

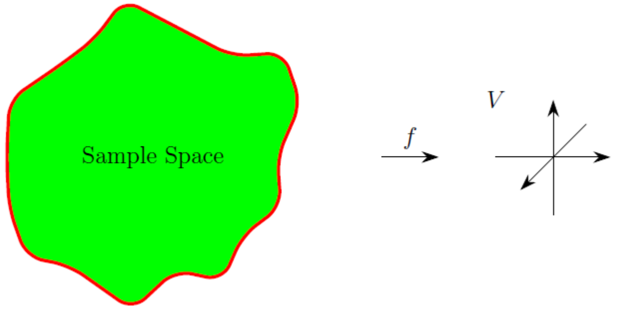
then $v_1, v_2, \cdots, v_N$ *is a centered dataset of vectors where its mean is zero.*



Some notes

- When we work with vector spaces, numbers are referred to as *scalars*.
- When we multiply a vector v by a scalar r to get the scaled vector rv, we call it *scalar multiplication*.
- The set of all real numbers $\mathbb{R}$ is a vector space.
- The set of all integers $\mathbb{Z}$ is not a vector space.
- The set of all rational numbers $\mathbb{Q}$ is a vector space over $\mathbb{Q}$ but not over $\mathbb{R}$.
- The set of all Python strings is not a vector space.
- Usually, we can't take sample means from a population, we instead take the sample mean of a *statistic* associated to the population. A statistic is an assignment of a number $f(\text{item})$ to each item in the population. For example, the human population on Earth is not a vector space (they can't be added), but their heights is a vector space (heights can be added). For the Python strings, a statistic might be the length of the strings. For the HTML colors, a statistic is the HTML code of the color.

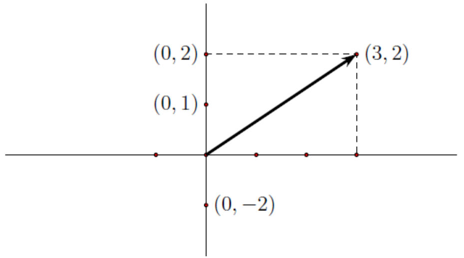
Statistic



In general, a statistic need not be a number. A statistic can be anything that "behaves like a number". For example, $f(\text{item})$ can be a vector or a matrix. More generally, a statistic's values may be anything that lives in a vector space V.

Cartesian plane

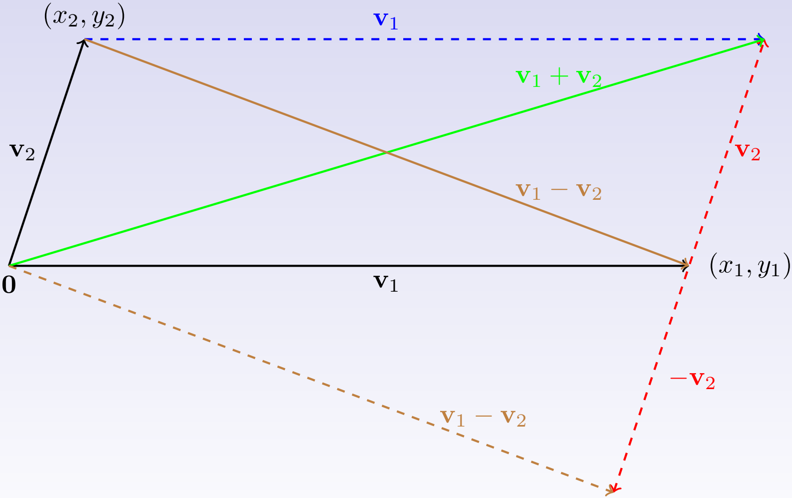
The *cartesian plane* $\mathbb{R}^2$, also called the 2-dimensional real space is a vector space.



For $\mathbf{v}_1 = (x_1, y_1), \mathbf{v}_2 = (x_2, y_2) \in \mathbb{R}^2$ and $t \in \mathbb{R}$ define

- $\mathbf{v}_1 + \mathbf{v}_2 = (x_1 + x_2, y_1 + y_2)$ (Addition).
- $\mathbf{0} = (0, 0)$ (Zero).
- $t\mathbf{v}_1 = (tx_1, ty_1)$ (Scaling).
- $-\mathbf{v}_1 = (-1)\mathbf{v}_1$ (Negative).
- $\mathbf{v}_1 - \mathbf{v}_2 = \mathbf{v}_1 + (-\mathbf{v}_2) = (x_1 - x_2, y_1 - y_2)$ (Subtraction).

Operations


(x_2, y_2)
$\mathbf{v}_1$
$\mathbf{v}_1 + \mathbf{v}_2$
$\mathbf{v}_2$
$\mathbf{v}_2$
$\mathbf{v}_1 - \mathbf{v}_2$
(x_1, y_1)
$\mathbf{0}$
$\mathbf{v}_1$
$-\mathbf{v}_2$
$\mathbf{v}_1 - \mathbf{v}_2$

Data Sets
Introduction
Averages and Vector Spaces
Two Dimensions
Complex Numbers
Mean and Covariance
Linear Geometry
Principal Components

2d example

Example 1.6

```
import numpy as np

v1 = (1,2)
v2 = (3,4)
print(v1 + v2 == (1+3,2+4)) # returns False

v1 = [1,2]
v2 = [3,4]
print(v1 + v2 == [1+3,2+4]) # returns False

v1 = np.array([1,2])
v2 = np.array([3,4])
print(v1 + v2 == np.array([1+3,2+4]))
# returns [ True  True]
print(3*v1 == np.array([3,6]))
# returns [ True  True]
print(-v1 == np.array([-1,-2]))
# returns [ True  True]
print(v1 - v2 == np.array([1-3,2-4]))
# returns [ True  True]
```

Data Sets
Introduction
Averages and Vector Spaces
Two Dimensions
Complex Numbers
Mean and Covariance
Linear Geometry
Principal Components

2d example

For the two-dimensional dataset

$$\mathbf{x}_1 = (1,2), \mathbf{x}_2 = (3,4), \mathbf{x}_3 = (-2,11), \mathbf{x}_4 = (0,66),$$

or, equivalently,

$$\mathbf{x} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ -2 & 11 \\ 0 & 66 \end{pmatrix},$$

the average is

$$\frac{(1,2)+(3,4)+(-2,11)+(0,66)}{4} = (0.5, 20.75).$$

Example 1.7

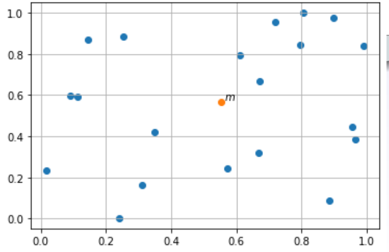
```python
import numpy as np

dataset = np.array([[1,2], [3,4], [-2,11], [0,66]])
print(np.mean(dataset, axis=0))
# returns [ 0.5 , 20.75]
```

2d example

Example 1.8

Generate a 2 dimensional dataset of random points and their mean

```
import numpy as np
from numpy.random import random as rd
import matplotlib.pyplot as plt
N = 20
dataset = np.array([[rd(), rd()] for _ in range(N)])
mean = np.mean(dataset,axis=0)
plt.grid()
X, Y = dataset[:,0], dataset[:,1]
plt.scatter(X,Y)
plt.scatter(*mean)
plt.annotate('$m$', xy=mean+0.01)
plt.show()
```



Magnitude

Definition 1.6 (Distance Formula)

If $\mathbf{v}_1 = (x_1, y_1)$ *and* $\mathbf{v}_2 = (x_2, y_2)$, *then the distance between* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *is*

$$|\mathbf{v}_1 - \mathbf{v}_2| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

The distance of $\mathbf{v} = (x, y)$ *to the origin* $\mathbf{0} = (0, 0)$ *is its magnitude or norm or length*

$$r = |\mathbf{v}| = |\mathbf{v} - \mathbf{0}| = \sqrt{x^2 + y^2}.$$

Example 1.9

For $\mathbf{v}_1 = (1, 2)$ and $\mathbf{v}_2 = (3, 4)$

$$|\mathbf{v}_1| = \sqrt{1^2 + 2^2} = \sqrt{5} \simeq 2.236,$$

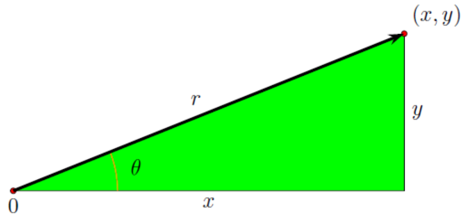
$$|\mathbf{v}_1 - \mathbf{v}_2| = \sqrt{(1-3)^2 + (2-4)^2} = \sqrt{4+4} = \sqrt{8} \simeq 2.828.$$

```
import numpy as np

v1 = np.array([1,2])
v2 = np.array([3,4])
print(np.linalg.norm(v1)) #returns 2.23606797749979
print(np.linalg.norm(v1-v2)) #returns 2.
    8284271247461903
```

Polar representation

In terms of r and θ, the *polar representation* of (x, y) is

$$x = r\cos\theta, \quad y = r\sin\theta.$$



The *unit circle* consists of the vectors which are distance 1 from the origin $\mathbf{0}$. When $\mathbf{v}$ is on the unit circle, the magnitude of $\mathbf{v}$ is 1, and we say $\mathbf{v}$ is a *unit vector*. In this case, the line formed by the scalings of $\mathbf{v}$ intersects the unit circle at $\pm\mathbf{v}$.

When $\mathbf{v}$ is a unit vector, then $r = 1$ and $\mathbf{v} = (x, y) = (\cos\theta, \sin\theta)$.

Polar representation

By the distance formula, a vector $\mathbf{v} = (x, y)$ is a unit vector when

$$x^2 + y^2 = 1.$$

More generally, any circle with *center* (a, b) and radius r consists of vectors $\mathbf{v} = (x, y)$ satisfying

$$(x - a)^2 + (y - b)^2 = r^2.$$

Let R be a point on the unit circle, and let $t > 0$. The scaled point tR is on the circle with center $(0, 0)$ and radius t. Moreover, if Q is any point, $Q + tR$ is on the circle with center Q and radius t. It is easy to check that $|t\mathbf{v}| = |t||\mathbf{v}|$ for any real number t and vector $\mathbf{v}$.

From this, if a vector $\mathbf{v}$ is unit and $r > 0$, then $r\mathbf{v}$ has magnitude r. If $\mathbf{v}$ is any vector not equal to the zero vector, then $r = |\mathbf{v}|$ is positive, and

$$\left|\frac{1}{r}\mathbf{v}\right| = \frac{1}{r}|\mathbf{v}| = \frac{1}{r}r = 1$$

so $\mathbf{v}/r$ is a unit vector.

Inner product

Definition 1.7

Let $\mathbf{v}_1 = (x_1, y_1), \mathbf{v}_2 = (x_2, y_2) \in \mathbb{R}^2$. *The inner product or the dot product of* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *is given algebraically as*

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = x_1 x_2 + y_1 y_2.$$

From the geometric view, we have:

Theorem 1.1 (Dot Product Identity)

$$x_1 x_2 + y_1 y_2 = \mathbf{v}_1 \cdot \mathbf{v}_2 = |\mathbf{v}_1||\mathbf{v}_2| \cos\theta,$$

where θ *is the angle between* $\mathbf{v}_1$ *and* $\mathbf{v}_1$.

Exercise 1.3

Prove the "Dot Product Identity", Theorem (1.1).
Hint: Use Pythagoras' theorem for general triangles.

The angle between two vectors

In Python, the dot product is given by `numpy.dot` and as a consequence of the dot product identity, we have the code for the angle between two vectors:

$$\theta_{\mathbf{v}_1,\mathbf{v}_2} = \arccos\left(\frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1||\mathbf{v}_2|}\right).$$

Example 1.10

Find the angle between the vectors $\mathbf{v}_1 = (1, 2)$ and $\mathbf{v}_2 = (3, 4)$.

```
import numpy as np

def angle(u,v):
  a = np.dot(u,v)
  b = np.dot(u,u)
  c = np.dot(v,v)
  theta = np.arccos(a / np.sqrt(b*c))
  return np.degrees(theta)

v1 = np.array([1,2])
v2 = np.array([3,4])
print(angle(v1,v2)) #returns 10.304846468766044 in degree
```

Cauchy-Schwarz Inequality

Recall that $-1 \leq \cos\theta \leq 1$. Using the dot product identity, we obtain the important inequality:

Theorem 1.2 (Cauchy-Schwarz Inequality)

If u and v are any two vectors, then

$$-|u||v| \leq u \cdot v \leq |u||v|.$$

Exercise 1.4

Prove the "Cauchy-Schwarz Inequality".

2d linear equations system

Consider the homogeneous system

$$\begin{cases} ax + by = 0 \\ cx + dy = 0 \end{cases} \tag{1.1}$$

and let A be the 2×2 matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}. \tag{1.2}$$

$(x, y) = (-b, a)$ is a solution of the first equation in (1.1). If we want this to be a solution of the second equation as well, we must have $cx + dy = ad - bc = 0$.

Definition 1.8 (Determinant)

The determinant of A is

$$\det(A) = \det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc.$$

2d linear equations system

Theorem 1.3 (Homogeneous System)

When $\det(A) = 0$, the homogeneous system (1.1) has a nonzero solution, and all solutions are scalar multiples of $(x, y) = (-b, a)$. When $\det(A) \neq 0$, the only solution is $(x, y) = (0, 0)$.

For the inhomogeneous case

$$\begin{cases} ax + by = e \\ cx + dy = f \end{cases} \tag{1.3}$$

we have

Theorem 1.4 (Inhomogeneous System)

When $\det(A) \neq 0$, the inhomogeneous system (1.3) has the unique solution

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{\det(A)} \begin{pmatrix} de - bf \\ af - ce \end{pmatrix}.$$

When $\det(A) = 0$, (1.3) has a solution iff $ce = af$ and $de = bf$.

2d linear equations system

When $a^2 + b^2 \neq 0$, a solution is

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{a^2 + b^2} \begin{pmatrix} ae \\ be \end{pmatrix}.$$

When $c^2 + d^2 \neq 0$, a solution is

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{c^2 + d^2} \begin{pmatrix} cf \\ df \end{pmatrix}.$$

Any other solution differs from these solutions by a scalar multiple of the homogeneous solution $(x, y) = (-b, a)$.

Exercise 1.5

Prove the Theorems (1.3) and (1.4).

Data Sets
Introduction
Averages and Vector Spaces
Two Dimensions
Complex Numbers
Mean and Covariance
Linear Geometry
Principal Components

Complex numbers

Roughly speaking, the set of all *complex numbers* is the set of all points in $\mathbb{R}^2$ with different multiplication rule.

Definition 1.9 (Complex numbers)

The complex numbers, $\mathbb{C}$*, is the set*

$$\mathbb{C} = \{(x, y) \in \mathbb{R}^2\}$$

with operations

- *Addition:* $(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$.
- *Scalar Multiplication:* $t(x, y) = (tx, ty)$
- *Multiplication:* $(x_1, y_1)(x_2, y_2) = (x_1x_2 - y_1y_2, x_1y_2 + x_2y_1)$.

Then, in $\mathbb{C}$, we have

- zero: $0 = (0, 0)$.
- opposite or additive inverse: $-(x, y) = (-x, -y)$.
- one: $1 = (1, 0)$.

Example

Example 1.11

- $(1,2)+(3,4)=(4,6)$.
- $(0,0)+(1,2)=(1,2)$.
- $3(1,2)=(3,6)$.
- $(1,0)(1,2)=(1-0,2+0)=(1,2)$.
- $(1,2)(3,4)=(3-8,4+6)=(-5,10)$.
- $(x,0)+(y,0)=(x+y,0)$.
- $(x,0)(y,0)=(xy,0)$.

Note. By the last two examples, we see that complex numbers with 0 as their second component act like real numbers in addition and multiplication. So, from now on, we set $x=(x,0)$.

Example 1.12

- $0=(0,0)$.
- $1=(1,0)$.
- $-1=(-1,0)$.

Imaginary number

Definition 1.10 (Imaginary number)

$i = (0, 1)$.

Note. Python uses the symbol j for imaginary number.

Theorem 1.5

For each $z = (x, y) \in \mathbb{C}$, we can write

$$z = x + iy.$$

We call x as the real part of z, and y the imaginary part of z.

$$x = Re(z), \quad y = Im(z).$$

Proof. $x + iy = (x, 0) + (0, 1)(y, 0) = (x, 0) + (0 - 0, 0 + y) = (x, y)$.

Theorem 1.6

$i^2 = -1$.

Proof. $i^2 = (0, 1)(0, 1) = (0 - 1, 0 + 0) = (-1, 0) = -1$.

Data Sets
Introduction
Averages and Vector Spaces
Two Dimensions
Complex Numbers
Mean and Covariance
Linear Geometry
Principal Components

Example

Example 1.13

In complex numbers:

- $\sqrt{-1} = i.$
- $\sqrt{-4} = 2i.$
- $$\begin{aligned}(1,2)(3,4) &= (1+2i)(3+4i)\\ &= 3+4i+6i+8i^2\\ &= 3+10i-8\\ &= -5+10i\\ &= (-5,10).\end{aligned}$$
- $$\begin{aligned}(1,2)^3 &= (1+2i)^3\\ &= (1)^3+3(1)^2(2i)+3(1)(2i)^2+(2i)^3\\ &= 1+6i+12i^2+8i^3\\ &= 1+6i-12-8i\\ &= -11-2i\\ &= -(11,2).\end{aligned}$$

Conjugate

Definition 1.11 (Conjugate)

For $z = (x, y) \in \mathbb{C}$, *the conjugate is*

$$\bar{z} = (x, -y) = x - iy \in \mathbb{C}.$$

Some properties.

- $z + \bar{z} = 2Re(z)$, $z - \bar{z} = 2iIm(z)$.
- $z\bar{z} = Re(z)^2 + Im(z)^2$,

$$\Rightarrow |z| = \sqrt{Re(z)^2 + Im(z)^2} = \sqrt{z\bar{z}}$$
$$\Rightarrow |z|^2 = z\bar{z}.$$

Example 1.14

For $z = (4, -3) \in \mathbb{C}$:

- $\bar{z} = (4, 3) = 4 + 3i$,
- $z + \bar{z} = 2 \times 4 = 8$, $z - \bar{z} = 2i \times (-3) = -6i$.
- $z\bar{z} = (4)^2 + (-3)^2 = 16 + 9 = 25 \Rightarrow |z| = \sqrt{25} = 5$.
- $z^2 = (4 - 3i)^2 = 7 - 24i$.
- $|z|^2 = 25$.

Inverse

Theorem 1.7

For a non-zero $z \in \mathbb{C}$, the inverse of z is

$$z^{-1} = \frac{1}{z} = \frac{\bar{z}}{z\bar{z}} = \frac{\bar{z}}{|z|^2}.$$

Proof. Firstly, if $z = (x, y)$ then $\frac{1}{z} \in \mathbb{C}$, because,

$$\frac{1}{z} = \frac{x - iy}{x^2 + y^2} = (\frac{x}{x^2 + y^2}, \frac{-y}{x^2 + y^2}) \in \mathbb{C}.$$

Secondly,

$$zz^{-1} = (x + iy)\left(\frac{x - iy}{x^2 + y^2}\right) = \frac{x^2 + y^2}{x^2 + y^2} = 1.$$

Corollary 1.1 (Division)

For $z_1 \in \mathbb{C}$ and $0 \neq z_2 \in \mathbb{C}$

$$\frac{z_1}{z_2} = z_1 z_2^{-1}.$$

Definitions

Definition 1.12 (Mean-squared distance)

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ *be a dataset, say* D*, in* $\mathbb{R}^d$*, and let* $\mathbf{x} \in \mathbb{R}^d$*. The mean-squared distance of* $\mathbf{x}$ *to* D *is*

$$MSD(\mathbf{x}) = \frac{1}{N} \sum_{k=1}^{N} |\mathbf{x}_k - \mathbf{x}|^2.$$

Definition 1.13 (Mean)

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ *be a dataset in* $\mathbb{R}^d$*. The mean or sample mean is*

$$\mathbf{m} = \bar{\mathbf{x}}_N = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_N}{N}.$$

Theorem 1.8 (Point of Best-fit)

The mean is the point of best-fit: The mean minimizes the mean-squared distance to the dataset.

Exercise 1.6

Prove the Theorem (1.8).

Point of Best-fit

Example 1.15

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(1)
N = 20
rnd = np.random.random
dataset = np.array([ [rnd(), rnd()] for _ in range(N) ])
# Mean
m = np.mean(dataset, axis=0)
#Random point
p = np.array([rnd(), rnd()])

plt.grid()
X, Y = dataset[:,0], dataset[:,1]
plt.scatter(X,Y)
for v in dataset:
    plt.plot([m[0],v[0]],[m[1],v[1]],c='green')
    plt.plot([p[0],v[0]],[p[1],v[1]],c='red')
plt.show()

# Comparison of MSD of the mean and a random point
MSD_m = np.sum(np.abs(dataset-m)**2)/N
MSD_p = np.sum(np.abs(dataset-p)**2)/N
print(MSD_m, MSD_p) # 0.160478187272121 0.5984208474157081
```

Point of Best-fit

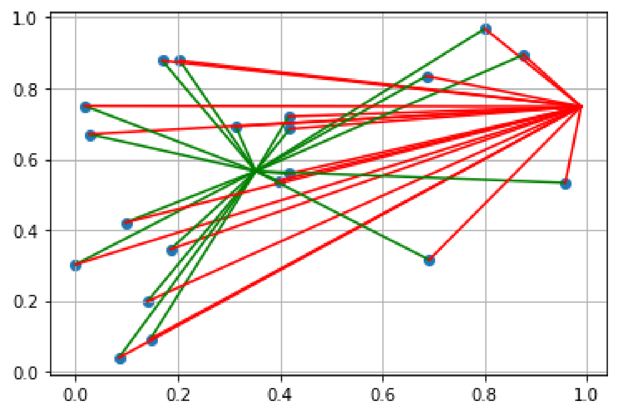


Figure 1.1: MSD for the mean (green) versus MSD for a random point (red).

Tensor product

For simplicity, let $\mathbf{u} = (a, b)$ and $\mathbf{v} = (c, d, e)$ be two vectors.

Definition 1.14 (Tensor product)

The tensor product of $\mathbf{u}$ *and* $\mathbf{text}$ *is the matrix*

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} ac & ad & ae \\ bc & bd & be \end{pmatrix} = \begin{pmatrix} c\mathbf{u} & d\mathbf{u} & e\mathbf{u} \end{pmatrix} = \begin{pmatrix} a\mathbf{v} \\ b\mathbf{v} \end{pmatrix}$$

Definition 1.15 (Trace of a matrix)

The trace of a squared matrix A *is the sum of the diagonal entries.*

Note. For any vectors $\mathbf{u}, \mathbf{v}$ and $\mathbf{w}$:

- $\mathbf{v} \otimes \mathbf{u} = (\mathbf{u} \otimes \mathbf{v})^t$.

  In square case:
- $\det(\mathbf{u} \otimes \mathbf{v}) = 0$.
- $trace(\mathbf{u} \otimes \mathbf{v}) = \mathbf{u} \cdot \mathbf{v}$.
- $trace(\mathbf{u} \otimes \mathbf{u}) = |\mathbf{u}|^2$.
- $(\mathbf{u} \otimes \mathbf{v})\mathbf{w} = (\mathbf{v} \cdot \mathbf{w})\mathbf{u}$.

Covariance

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ be a dataset in $\mathbb{R}^d$ with $\mathbf{m}$ as its mean.

Definition 1.16 (1d Covariance)

When $d = 1$, the covariance q is a scalar

$$q = \frac{1}{N}\sum_{k=1}^{N}(x_k - m)^2 = MSD(m).$$

In the scalar case, the covariance is called the variance of the scalar dataset.

In general, the covariance is a symmetric $d \times d$ matrix Q. We can center the dataset as

$$\mathbf{v}_1 = \mathbf{x}_1 - \mathbf{m}, \mathbf{v}_2 = \mathbf{x}_2 - \mathbf{m}, \ldots, \mathbf{v}_N = \mathbf{x}_N - \mathbf{m}.$$

Then the *covariance matrix* is the $d \times d$ matrix Q as

$$Q = \frac{\mathbf{v}_1 \otimes \mathbf{v}_1 + \mathbf{v}_2 \otimes \mathbf{v}_2 + \ldots + \mathbf{v}_N \otimes \mathbf{v}_N}{N}. \tag{1.4}$$

Example

Example 1.16

Suppose $N = 5$ and

$$\mathbf{x}_1 = (1,2), \quad \mathbf{x}_2 = (3,4), \quad \mathbf{x}_3 = (5,6), \quad \mathbf{x}_4 = (7,8), \quad \mathbf{x}_5 = (9,10).$$

Then $m = (5,6)$ and

$$\mathbf{v}_1 = \mathbf{x}_1 - \mathbf{m} = (-4,-4), \quad \mathbf{v}_2 = \mathbf{x}_2 - \mathbf{m} = (-2,-2),$$
$$\mathbf{v}_3 = \mathbf{x}_3 - \mathbf{m} = (0,0), \quad \mathbf{v}_4 = \mathbf{x}_4 - \mathbf{m} = (2,2), \quad \mathbf{v}_5 = \mathbf{x}_5 - \mathbf{m} = (4,4).$$

Since

$$(\pm 4, \pm 4) \otimes (\pm 4, \pm 4) = \begin{pmatrix} 16 & 16 \\ 16 & 16 \end{pmatrix},$$
$$(\pm 2, \pm 2) \otimes (\pm 2, \pm 2) = \begin{pmatrix} 4 & 4 \\ 4 & 4 \end{pmatrix},$$
$$(0,0) \otimes (0,0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},$$

then

$$Q = \begin{pmatrix} 8 & 8 \\ 8 & 8 \end{pmatrix}.$$

Example

Example 1.17

```
import numpy as np

def tensor(u,v):
  return np.array([ [ a*b for b in v] for a in u ])

np.random.seed(1)
N = 20
rnd = np.random.random
dataset = np.array([[rnd(), rnd()] for _ in range(N)])
# mean
m = np.mean(dataset,axis=0)
# center dataset
vectors = dataset - m
# covariance
Q = np.mean([ tensor(v,v) for v in vectors ],axis=0)
print(Q)
```

Standardized

Note. The covariance matrix as written in (1.4) is the *biased covariance matrix*. If the denominator is instead $N - 1$, the matrix is the *unbiased covariance matrix*.

For datasets with large N, it doesn't matter, since N and $N - 1$ are almost equal.

In `numpy`, the Python covariance constructor is

Example 1.18

```
import numpy as np

np.random.seed(1)
N = 20
rnd = np.random.random
dataset = np.array([[rnd(), rnd()] for _ in range(N)])
# covariance
Q = np.cov(dataset, bias=True, rowvar=False)
print(Q)
```

Total variance

Definition 1.17 (Total variance)

From $trace(\mathbf{u} \otimes \mathbf{u}) = |\mathbf{u}|^2$*, if* Q *is the covariance matrix then*

$$trace(Q) = \frac{1}{N} \sum_{k=1}^{N} |\mathbf{x}_k - \mathbf{m}|^2. \tag{1.5}$$

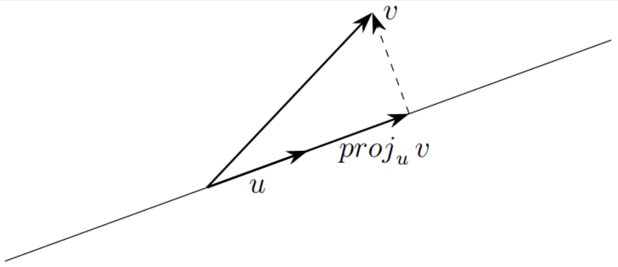
We call (1.5) the total variance of the dataset. Thus the total variance equals $MSD(\mathbf{m})$.

Example 1.19

```python
import numpy as np

np.random.seed(1)
N = 20
rnd = np.random.random
dataset = np.array([[rnd(), rnd()] for _ in range(N)])
# covariance
Q = np.cov(dataset.T,bias=True)
print(Q.trace()) # returns 0.16047818727212101
```

Projections

We would like to project a $2d$ dataset onto a line. Let $\mathbf{u}$ be a unit vector (a vector of length one, $|\mathbf{u}| = 1$), and let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ be a $2d$ dataset, assumed for simplicity to be centered. We wish to project this dataset onto the line through $\mathbf{u}$. This will result in a $1d$ dataset.



When a vector $\mathbf{v}$ is projected onto the line through $\mathbf{u}$, the length of the projected vector reads

$$|proj_{\mathbf{u}}\mathbf{v}| = |\mathbf{v}| \cos\theta,$$

where θ is the angle between the vectors $\mathbf{v}$ and $\mathbf{u}$. Since $|\mathbf{u}| = 1$, this length equals the dot product $\mathbf{v} \cdot \mathbf{u}$. Hence the projected vector is

$$proj_{\mathbf{u}}\mathbf{v} = (\mathbf{v} \cdot \mathbf{u})\mathbf{u}.$$

Projections

Hence,

Definition 1.18 (Reduced dataset)

The projected dataset of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ *onto the line through* $\mathbf{u}$ *is the dataset*

$$(\mathbf{v}_1 \cdot \mathbf{u})\mathbf{u}, (\mathbf{v}_2 \cdot \mathbf{u})\mathbf{u}, \ldots (\mathbf{v}_N \cdot \mathbf{u})\mathbf{u}.$$

The projected datasetc is in $\mathbb{R}^2$*. The reduced dataset is*

$$(\mathbf{v}_1 \cdot \mathbf{u}), (\mathbf{v}_2 \cdot \mathbf{u}), \ldots (\mathbf{v}_N \cdot \mathbf{u}),$$

which is in $\mathbb{R}$*.*

Exercise 1.7

Show that when a $2d$ *dataset is centered then the mean of the reduced dataset is 0.*

Exercise 1.8

Prove that if Q *is the covariance matrix of a* $2d$ *dataset, then the variance of the projected dataset onto the line through the vector* $\mathbf{u}$ *equals the quadratic function* $\mathbf{u} \cdot Q\mathbf{u}$*:*

$$q = \frac{1}{N} \sum_{k=1}^{N} \mathbf{u} \cdot (\mathbf{v}_k \otimes \mathbf{v}_k)\mathbf{u} = \mathbf{u} \cdot Q\mathbf{u}.$$

Covariance ellipse

Hence,

Definition 1.19 (Covariance ellipse)

The contour of all points $\mathbf{x}$ satisfying $\mathbf{x} \cdot Q\mathbf{x} = 1$ is the covariance ellipsoid. In two dimensions $d = 2$, this is the covariance ellipse. The contour of all points $\mathbf{x}$ satisfying $\mathbf{x} \cdot Q^{-1}\mathbf{x} = 1$ is the inverse covariance ellipsoid. In two dimensions $d = 2$, this is the inverse covariance ellipse.

In two dimensions $d = 2$, a covariance matrix has the form

$$Q = \begin{pmatrix} a & b \\ b & c \end{pmatrix}.$$

If we write $\mathbf{u} = (x, y)$ for a vector in the plane, the covariance ellipse is

$$\mathbf{u} \cdot Q\mathbf{u} = (x, y) \cdot \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = ax^2 + 2bxy + cy^2 = 1.$$

The covariance ellipse and inverse covariance ellipses described above are centered at the origin $(0, 0)$. When a dataset has mean $\mathbf{m}$ and covariance Q, the ellipses are drawn centered at $\mathbf{m}$.

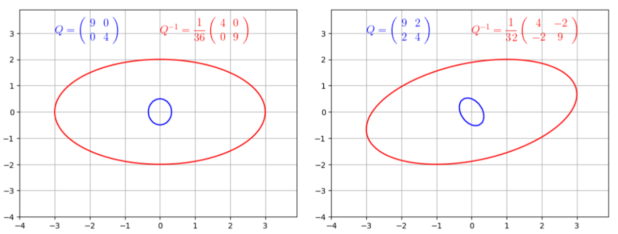
In particular, when $a = c$ and $b = 0$, then $Q = aI$ is a multiple of the identity, the inverse covariance ellipse is the circle of radius $\sqrt{a}$, and the covariance ellipse is the circle of radius $\frac{1}{\sqrt{a}}$.

Covariance ellipse I

Example 1.20

Plot the contour ellipses for

$$Q_1 = \begin{pmatrix} 9 & 0 \\ 0 & 4 \end{pmatrix}, \quad Q_2 = \begin{pmatrix} 9 & 2 \\ 2 & 4 \end{pmatrix}.$$



Covariance ellipse II

```
import matplotlib.pyplot as plt
import numpy as np

def ellipse(a, b, c, levels, color):
  L, delta = 4, .1
  x = np.arange(-L,L,delta)
  y = np.arange(-L,L,delta)
  X,Y = np.meshgrid(x, y)
  plt.contour(X, Y, a*X**2 + 2*b*X*Y + c*Y**2, levels,
                            colors=color)

# Q1 Covariance entities
a, b, c = 9, 0, 4

# Inverse Covariance entities
det = a*c - b**2
A, B, C = c/det, -b/det, a/det

plt.grid()
ellipse(a, b, c, [20], 'blue')
ellipse(A, B, C, [1], 'red')
plt.show()
```

Covariance ellipse III

```

# Q2 Covariance entities
a, b, c = 9, 2, 4

# Inverse Covariance entities
det = a*c - b**2
A, B, C = c/det, -b/det, a/det

plt.grid()
ellipse(a, b, c, [1], 'blue')
ellipse(A, B, C, [1], 'red')
plt.show()
```

Standardization

Here, we describe how to standardize datasets in $\mathbb{R}^2$. *Standardizing* the dataset means to center the dataset and to place the x and y features on the same scale.

Consider the dataset $\mathbf{x}_1 = (x_1, y_1), \mathbf{x}_2 = (x_2, y_2), \ldots, \mathbf{x}_N = (x_N, y_N)$ with mean $\mathbf{m} = (m_x, m_y)$. Then the covariance matrix is

$$Q = \begin{pmatrix} a & b \\ b & c \end{pmatrix},$$

where

$$a = \frac{1}{N}\sum_{k=1}^{N}(x_k - m_x)^2, \quad b = \frac{1}{N}\sum_{k=1}^{N}(x_k - m_x)(y_k - m_y),$$

$$c = \frac{1}{N}\sum_{k=1}^{N}(y_k - m_y)^2.$$

Standardization

If a and c differ, the different scales of x's and y's distorts the relation between them, and b may not accurately reflect the correlation. To correct for this, we center and re-scale

$$x_1, x_2, \ldots, x_N \to x_1' = \frac{x_1 - m_x}{\sqrt{a}}, x_2' = \frac{x_2 - m_x}{\sqrt{a}}, \ldots, x_N' = \frac{x_N - m_x}{\sqrt{a}}$$

and

$$y_1, y_2, \ldots, y_N \to y_1' = \frac{y_1 - m_y}{\sqrt{c}}, y_2' = \frac{y_2 - m_y}{\sqrt{c}}, \ldots, y_N' = \frac{y_N - m_y}{\sqrt{c}}$$

This results in a new dataset

$\mathbf{v}_1 = (x_1', y_1'), \mathbf{v}_2 = (x_2', y_2'), \ldots, \mathbf{v}_N = (x_N', y_N')$ that is centered:

$$\frac{\mathbf{v}_1 + \mathbf{v}_2 + \ldots + \mathbf{v}_N}{N} = 0,$$

with each feature standardized to have unit variance,

$$\frac{1}{N}\sum_{k=1}^{N} x_k' = 1, \quad \frac{1}{N}\sum_{k=1}^{N} y_k' = 1.$$

This is the *standardized dataset*.

Data Sets
- Introduction
- Averages and Vector Spaces
- Two Dimensions
- Complex Numbers
- Mean and Covariance

Linear Geometry

Principal Components

Standardization

The covariance matrix of the standardized dataset has the form

$$Q' = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix},$$

where

$$\rho = \frac{1}{N}\sum_{k=1}^{N} x'_k y'_k = \frac{b}{\sqrt{ac}} = \frac{\sum_{k=1}^{N}(x_k - m_x)(y_k - m_y)}{\sqrt{\left(\sum_{k=1}^{N}(x_k - m_x)^2\right)\left(\sum_{k=1}^{N}(y_k - m_y)^2\right)}}$$

is the *Pearson correlation coefficient* of the dataset. The matrix Q' is the *correlation matrix*, or the *standardized covariance matrix*.

Example 1.21

$$Q = \begin{pmatrix} 9 & 2 \\ 2 & 4 \end{pmatrix} \quad \Rightarrow \quad \rho = \frac{b}{\sqrt{ac}} = \frac{1}{3} \quad \Rightarrow \quad Q' = \begin{pmatrix} 1 & 1/3 \\ 1/3 & 1 \end{pmatrix}.$$

Standardization

From the Cauchy-Schwarz inequality, the correlation coefficient ρ is always between -1 and 1 . When $\rho = \pm 1$, the dataset samples are perfectly correlated and lie on a line passing through the mean. When $\rho = 1$, the line has slope 1, and when $\rho = -1$, the line has slope -1. When $\rho = 0$, the dataset samples are completely uncorrelated and are considered two independent one-dimensional datasets (In standardized case).

In Python `numpy`, the correlation matrix is returned by

```python
import numpy as np
np.corrcoef(dataset.T)
```

Here again, we input the transpose of the dataset if our default is vectors as rows.

Notice the $1/N$ cancels in the definition of ρ. Because of this, corrcoef is the same whether we deal with biased or unbiased covariance matrices.

Outline

1 Data Sets

2 Linear Geometry

3 Principal Components

Introduction

Definition 2.1

A matrix is a listing arranged in a rectangle of rows and columns. Specifically, an $N \times d$ matrix A has N rows and d columns,

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1d} \\ a_{21} & a_{22} & \dots & a_{2d} \\ \vdots & \vdots & \dots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{Nd} \end{pmatrix}$$

The transpose of A is

$$A^t = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{N1} \\ a_{12} & a_{22} & \dots & a_{N2} \\ \vdots & \vdots & \dots & \vdots \\ a_{1d} & a_{2d} & \dots & a_{Nd} \end{pmatrix}$$

Introduction

Example 2.1

$$A = \begin{pmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{pmatrix} \Rightarrow A^t = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}.$$

```python
import numpy as np

A = np.array([[1,6,11],[2,7,12],[3,8,13],[4,9,14],[5,10,15]])
print(A)
print(A.shape)
print(len(A))
print(A[1])
print(A[1,2])
print(A[1:3])

# transpose
A_t = np.transpose(A)
print(A_t)
print(A_t.shape)
print(len(A_t))
print(A_t[1])
print(A_t[1,2])
print(A_t[1:3])
```

Introduction

Definition 2.2

A d-dimensional vector $\mathbf{v}$ may be written as a $1 \times d$ matrix

$$\mathbf{v} = \begin{pmatrix} t_1 & t_2 & \cdots & t_d \end{pmatrix}.$$

In this case, we call $\mathbf{v}$ a row vector.

Definition 2.3

An N-dimensional vector $\mathbf{v}$ may be written as an $N \times 1$ matrix

$$\mathbf{v} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}.$$

In this case, we call $\mathbf{v}$ a column vector.

Introduction

Vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ with the same dimension may be stacked as columns (`np.column_stack` in `Python`) of a matrix,

$$A = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_d \end{pmatrix}.$$

Similarly, vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ with the same dimension may be stacked as rows (`np.row_stack` in `Python`) of a matrix,

$$A = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{pmatrix}.$$

Introduction

Example 2.2

The row stack of $\mathbf{v}_1 = (1, 6, 11)$, $\mathbf{v}_2 = (2, 7, 12)$, $\mathbf{v}_3 = (3, 8, 13)$, $\mathbf{v}_4 = (4, 9, 14)$ and $\mathbf{v}_5 = (5, 10, 15)$ reads:

$$A = \begin{pmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{pmatrix},$$

and the column stack of them is:

$$A^t = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}.$$

```python
import numpy as np

v1 = [1,6,11]
v2 = [2,7,12]
v3 = [3,8,13]
v4 = [4,9,14]
v5 = [5,10,15]
A = np.row_stack((v1,v2,v3,v4,v5))
print(A)
A_t = np.column_stack((v1,v2,v3,v4,v5))
print(A_t)
```

Introduction

Definition 2.4

A matrix is square if the number of rows equals the number of columns.

Definition 2.5

A matrix is diagonal if the off-diagonal entities are zero.

Example 2.3

The matrix

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{pmatrix},$$

is square and diagonal.

The following matrices are not square but they are diagonal:

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \end{pmatrix}, \quad \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \\ 0 & 0 & 0 \end{pmatrix}.$$

Datasets

Definition 2.6

A dataset is a collection of points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ *in* $\mathbb{R}^d$*. After centering the mean to the origin, the dataset becomes a collection of vectors* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$*. Usually the vectors are presented as the rows of an* $N \times d$ *matrix* A*.*

Corresponding to this, datasets are often provided as a CSV file. The matrix A is the dataset matrix. In excel, this is called a spreadsheet. In SQL, this is called a table. In `numpy`, it's an array. In `pandas`, it's a dataframe. So, effectively,

matrix = dataset = CSV file = spreadsheet = table = array = dataframe

Datasets

Example 2.4

For the Iris dataset:

```python
import numpy as np
import pandas as pd
from sklearn import datasets

iris = datasets.load_iris()

# The dataset
dataset = iris["data"]

# To center the dataset
m = np.mean(dataset,axis=0)
vectors = dataset - m

# To make a data frame
centered_df = pd.DataFrame(data=vectors)
```

Addition & scalar multiplication

Matrices consisting of numbers are added and multiplied by scalars as follows. With t as an scalar and the matrices

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1d} \\ a_{21} & a_{22} & \dots & a_{2d} \\ \vdots & \vdots & \dots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{Nd} \end{pmatrix} \quad \text{and} \quad A' = \begin{pmatrix} a'_{11} & a'_{12} & \dots & a'_{1d} \\ a'_{21} & a'_{22} & \dots & a'_{2d} \\ \vdots & \vdots & \dots & \vdots \\ a'_{N1} & a'_{N2} & \dots & a'_{Nd} \end{pmatrix}$$

we have

$$A + A' = \begin{pmatrix} a_{11} + a'_{11} & a_{12} + a'_{12} & \dots & a_{1d} + a'_{1d} \\ a_{21} + a'_{21} & a_{22} + a'_{22} & \dots & a_{2d} + a'_{2d} \\ \vdots & \vdots & \dots & \vdots \\ a_{N1} + a'_{N1} & a_{N2} + a'_{N2} & \dots & a_{Nd} + a'_{Nd} \end{pmatrix},$$

and

$$tA = \begin{pmatrix} ta_{11} & ta_{12} & \dots & ta_{1d} \\ ta_{21} & ta_{22} & \dots & ta_{2d} \\ \vdots & \vdots & \dots & \vdots \\ ta_{N1} & ta_{N2} & \dots & ta_{Nd} \end{pmatrix}.$$

Matrices may be added only if they have the same shape.

Addition & scalar multiplication

Example 2.5

```
import numpy as np

A = np.zeros((4,3))
print(A)
B = np.eye(3)
print(B)
C = np.eye(4,3)
print(C)
D = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
print(D)
E = np.diag([1,2,3,4])
print(E)

print(A+C)
print(C+D)
print(4*D)
print(-D)
print(-2*D)
```

Introduction

Let t be a scalar, $\mathbf{u}, \mathbf{v}, \mathbf{w}$ be vectors, and let A, B be matrices. We already know how to compute $t\mathbf{u}$, $t\mathbf{v}$, and tA, tB. In this section, we compute the *dot product* $\mathbf{u} \cdot \mathbf{v}$, the *matrix-vector product* $A\mathbf{v}$, and the *matrix-matrix product* AB.

Dot product

In the first chapter, we defined the dot product in two dimensions. We now generalize it to any dimension d. Suppose $\mathbf{u}, \mathbf{v}$ are vectors in $\mathbb{R}^d$. Then their dot product $\mathbf{u} \cdot \mathbf{v}$ is the scalar obtained by multiplying corresponding features and then summing the products. This only works if the dimensions of $\mathbf{u}$ and $\mathbf{v}$ agree.

In other words, if $\mathbf{u} = (u_1, u_2, \ldots, u_d)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_d)$, then

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \ldots + u_d v_d.$$

It's best to think of this as "row-times-column" multiplication,

$$\mathbf{u} \cdot \mathbf{v} = \begin{pmatrix} u_1 & u_2 & \cdots & u_d \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix} = u_1 v_1 + u_2 v_2 + \ldots + u_d v_d.$$

Dot product

Example 2.6

In Python, calculate the dot product of $\mathbf{u} = (1, 2, 3)$ and $\mathbf{v} = (4, 5, 6)$.

```
import numpy as np

u = np.array([1,2,3])
v = np.array([4, 5, 6])

u_dot_v = np.dot(u,v)
print(u_dot_v)

u_dot_v_ = u[0]*v[0] + u[1]*v[1] + u[2]*v[2]
print(u_dot_v_)

print(u_dot_v == u_dot_v_)
```

Dot product

As we mentioned in 2 dimensions, we have the following generalizations in d dimension:

Definition 2.7

The length or norm or magnitude of a vector $\mathbf{v}$ is the square root of the dot product $\mathbf{v}\cdot\mathbf{v}$,

$$|\mathbf{v}| = \sqrt{\mathbf{v}\cdot\mathbf{v}}$$

Theorem 2.1 (Dot Product)

The dot product $\mathbf{u}\cdot\mathbf{v}$ satisfies

$$\mathbf{u}\cdot\mathbf{v} = |\mathbf{u}||\mathbf{v}|\cos\theta,$$

where θ is the angle between $\mathbf{u}$ and $\mathbf{v}$.

Corollary 2.1

To calculate the angle θ between $\mathbf{u}$ and $\mathbf{v}$ we have:

$$\cos\theta = \frac{\mathbf{u}\cdot\mathbf{v}}{\sqrt{|\mathbf{u}||\mathbf{v}|}} = \frac{\mathbf{u}\cdot\mathbf{v}}{\sqrt{(\mathbf{u}\cdot\mathbf{u})(\mathbf{v}\cdot\mathbf{v})}}.$$

Data Sets

Linear Geometry
- Matrices
- Products
- Matrix Inverse
- Span and Linear Independence
- Zero Variance Directions
- Pseudo-Inverse
- Projections
- Basis
- Rank

Principal Components

Dot product

Corollary 2.2 (Cauchy-Schwarz Inequality)

The dot product of two vectors is absolutely less or equal to the product of their lengths,

$$|\mathbf{u}\cdot\mathbf{v}| \leq |\mathbf{u}||\mathbf{v}| \quad \text{or} \quad |\mathbf{u}\cdot\mathbf{v}| \leq (\mathbf{u}\cdot\mathbf{u})(\mathbf{v}\cdot\mathbf{v}).$$

Definition 2.8

Vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be perpendicular or orthogonal if $|\mathbf{u}\cdot\mathbf{v}| = 0$. A collection of vectors is orthogonal if any pair of vectors in the collection are orthogonal.
Vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ are said to be orthonormal if they are both unit vectors and orthogonal.

Exercise 2.1

The zero vector is orthogonal to every vector. The converse is true as well: if a vector is orthogonal to every vector then it is the zero vector.

Matrix-vector product

Definition 2.9

Suppose $\mathbf{v}$ is a vector and A is a matrix. If the rows of A have the same dimension as that of $\mathbf{v}$, we can take the dot product of each row of A with $\mathbf{v}$, obtaining the matrix-vector product $A\mathbf{v}$: $A\mathbf{v}$ is the vector whose features are the dot products of the rows of A with $\mathbf{v}$.

Note:

- In Python we use again `np.dot(A,v)` for matrix-vector product.
- If $\mathbf{u}$ and $\mathbf{v}$ are vectors, we can think of $\mathbf{u}$ as a row vector, or a matrix consisting of a single row. With this interpretation, the matrix-vector product $\mathbf{u}\mathbf{v}$ equals the dot product $\mathbf{u} \cdot \mathbf{v}$.
- If $\mathbf{u}$ and $\mathbf{v}$ are vectors, we can think of $\mathbf{u}$ as a column vector, or a matrix consisting of a single column. With this interpretation, $\mathbf{u}^t$ is a single row, and the matrix-vector product $\mathbf{u}^t\mathbf{v}$ equals the dot product $\mathbf{u} \cdot \mathbf{v}$.
- $(A\mathbf{v})^t = \mathbf{v}^t A^t$.
- $(A\mathbf{u}) \cdot \mathbf{v} = \mathbf{u} \cdot (A^t\mathbf{v})$.

Matrix-vector product

Example 2.7

Calculate $A\mathbf{v}$, when

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \quad \text{and} \quad \mathbf{v} = (1, 2, 3, 4).$$

Answer:

$$A\mathbf{v} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$$= \begin{pmatrix} (1 \times 1) + (2 \times 2) + (3 \times 3) + (4 \times 4) \\ (5 \times 1) + (6 \times 2) + (7 \times 3) + (8 \times 4) \\ (9 \times 1) + (10 \times 2) + (11 \times 3) + (12 \times 4) \end{pmatrix} = \begin{pmatrix} 30 \\ 70 \\ 110 \end{pmatrix}$$

```python
import numpy as np

A = np.arange(1,13).reshape(3,4)
v = np.array([1,2,3,4])

Av = np.dot(A, v)
print(Av)
```

Matrix-matrix product

Definition 2.10

Let A and B be two matrices. If the row dimension of A equals the column dimension of B, the matrix-matrix product AB is defined. When this condition holds, the entries in the matrix AB are the dot products of the rows of A with the columns of B.

Note:

- In Python we use again `np.dot(A,B)` for matrix-vector product.
- $(AB)^t = B^t A^t$.

Matrix-vector product

Example 2.8

Calculate AB, when

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \\ 19 & 20 \end{pmatrix}.$$

Answer:

$$AB = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \\ 19 & 20 \end{pmatrix}$$

$$= \begin{pmatrix} (1 \times 13) + (2 \times 15) + (3 \times 17) + (4 \times 19) & (1 \times 14) + (2 \times 16) + (3 \times 18) + (4 \times 20) \\ (5 \times 13) + (6 \times 15) + (7 \times 17) + (8 \times 19) & (5 \times 14) + (6 \times 16) + (7 \times 18) + (8 \times 20) \\ (9 \times 13) + (10 \times 15) + (11 \times 17) + (12 \times 19) & (9 \times 14) + (10 \times 16) + (11 \times 18) + (12 \times 20) \end{pmatrix}$$

$$= \begin{pmatrix} 170 & 180 \\ 426 & 452 \\ 682 & 724 \end{pmatrix}$$

```python
import numpy as np

A = np.arange(1,13).reshape(3,4)
B = np.arange(13,21).reshape(4,2)

AB = np.dot(A, B)
print(AB)
```

Orthonormal Rows and Columns

Assume the rows of a matrix A are $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$. Since matrix-matrix multiplication is $row \times column$, we have

$$AA^t = \begin{pmatrix} \mathbf{v}_1 \cdot \mathbf{v}_1 & \mathbf{v}_1 \cdot \mathbf{v}_2 & \cdots & \mathbf{v}_1 \cdot \mathbf{v}_N \\ \mathbf{v}_2 \cdot \mathbf{v}_1 & \mathbf{v}_2 \cdot \mathbf{v}_2 & \cdots & \mathbf{v}_2 \cdot \mathbf{v}_N \\ \vdots & \vdots & & \vdots \\ \mathbf{v}_N \cdot \mathbf{v}_1 & \mathbf{v}_N \cdot \mathbf{v}_2 & \cdots & \mathbf{v}_N \cdot \mathbf{v}_N \end{pmatrix}.$$

Corollary 2.3

Let U be a matrix.

- *U has orthonormal rows iff $UU^t = I$.*
- *U has orthonormal columns iff $U^tU = I$.*

Tensor product

Definition 2.11

If $\mathbf{u}$ *and* $\mathbf{v}$ *are vectors, the tensor product* $\mathbf{u} \otimes \mathbf{v}$ *is the matrix-matrix product* $\mathbf{u}^t\mathbf{v}$*, with* $\mathbf{u}$ *and* $\mathbf{v}$ *row vectors. If* $\mathbf{u}$ *is* N*-dimensional and* $\mathbf{v}$ *is* d*-dimensional, then* $\mathbf{u} \otimes \mathbf{v}$ *is an* $N \times d$ *matrix.*

Example 2.9

if $\mathbf{u} = (a, b, c)$ and $\mathbf{v} = (\alpha, \beta)$, then

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \begin{pmatrix} \alpha & \beta \end{pmatrix} = \begin{pmatrix} a\alpha & a\beta \\ b\alpha & b\beta \\ c\alpha & c\beta \end{pmatrix}.$$

Using the tensor product, we have

Theorem 2.2 (Tensor Identity)

Let A *be a matrix with rows* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$*. Then*

$$A^tA = \mathbf{v}_1 \otimes \mathbf{v}_1 + \mathbf{v}_2 \otimes \mathbf{v}_2 + \cdots + \mathbf{v}_N \otimes \mathbf{v}_N.$$

Exercise 2.2

Prove the tensor identity.

Some definitions

Definition 2.12

A matrix Q is symmetric if $Q = Q^t$.
For any matrix A, $Q = AA^t$ and $Q = A^tA$ are symmetric.
A symmetric matrix Q satisfying $\mathbf{v} \cdot Q\mathbf{v} \geq 0$ for every vector $\mathbf{v}$ is nonnegative.
A symmetric matrix Q satisfying $\mathbf{v} \cdot Q\mathbf{v} > 0$ for every nonzero vector $\mathbf{v}$ is positive.

Definition 2.13

The trace of a square matrix is the sum of its diagonal elements.

Even though in general $AB \neq BA$, it is always true that

Exercise 2.3

$trace(AB) = trace(BA)$.

Exercise 2.4

$\mathbf{u} \cdot Q\mathbf{v} = trace(Q(\mathbf{v} \otimes \mathbf{u}))$.

Data Sets

Linear Geometry
- Matrices
- Products
- Matrix Inverse
- Span and Linear Independence
- Zero Variance Directions
- Pseudo-Inverse
- Projections
- Basis
- Rank

Principal Components

Norm squared

Definition 2.14

If $A = (a_{ij})$ is any matrix, then the norm squared of A is

$$\parallel A \parallel^2 = \sum_{i,j} a_{ij}^2.$$

Theorem 2.3 (Norm Squared of Matrix)

Let A be a matrix with rows $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$. Then

$$\parallel A \parallel^2 = |\mathbf{v}_1|^2 + |\mathbf{v}_2|^2 + \ldots + |\mathbf{v}_N|^2,$$

and

$$\parallel A \parallel^2 = trace(A^t A).$$

Exercise 2.5

Prove Theorem (2.3).

Iris

If $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ is a dataset of points in $\mathbb{R}^d$ with mean $\mathbf{m}$, and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ is the corresponding centered dataset, then we saw that the covariance matrix Q is the average of tensor products

$$Q = \frac{\mathbf{v}_1 \otimes \mathbf{v}_1 + \mathbf{v}_2 \otimes \mathbf{v}_2 + \cdots + \mathbf{v}_N \otimes \mathbf{v}_N}{N}.$$

Let A be the matrix with rows $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$. By Theorem (2.2), the last equation is the same as

$$Q = \frac{1}{N} A^t A.$$

Iris

Example 2.10

Calculate the mean, covariance and total variance of the Iris dataset.

```python
import numpy as np
from sklearn import datasets

iris = datasets.load_iris()

# The dataset
dataset = iris["data"]

# Mean
m = np.mean(dataset, axis=0)

# Centered dataset
vectors = dataset - m

# Covariance
N = len(vectors)
#   Biased
Q = np.dot(vectors.T, vectors)/N
Q = np.cov(dataset, rowvar=False, ddof=0) # ddof = delta degrees of freedom
Q = np.cov(dataset.T, ddof=0)

#   Unbiased
Q = np.dot(vectors.T, vectors)/(N-1)
Q = np.cov(dataset, rowvar=False)
Q = np.cov(dataset.T)

# Total Variance
TV = np.trace(Q)
```

Standardized dataset

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ is a dataset of points in $\mathbb{R}^d$. Each sample point $\mathbf{x}$ has d features $(t_1, t_2, \ldots, t_d)$. We compute the variance of each feature separately.

Let $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d$ be the standard basis in $\mathbb{R}^d$, and, for each $j = 1, 2, \ldots, d$, project the dataset onto $\mathbf{e}_j$, obtaining the scalar dataset $\mathbf{x}_1 \cdot \mathbf{e}_j, \mathbf{x}_2 \cdot \mathbf{e}_j, \ldots, \mathbf{x}_N \cdot \mathbf{e}_j$, consisting of the j-th feature of the samples. If q_{jj} is the variance of this scalar dataset, then $q_{11}, q_{22}, \ldots, q_{dd}$ are the diagonal entries of the covariance matrix.

To standardize the dataset, we center it, and rescale the features to have variance one, as follows. Let $\mathbf{m} = (m_1, m_2, \ldots, m_d)$ be the dataset mean. For each sample point $\mathbf{x} = (t_1, t_2, \ldots, t_d)$, the standardized vector is

$$\mathbf{v} = \left(\frac{t_1 - m_1}{\sqrt{q_{11}}}, \frac{t_2 - m_2}{\sqrt{q_{22}}}, \ldots, \frac{t_d - m_d}{\sqrt{q_{dd}}} \right).$$

Then the standardized dataset is $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$.

Data Sets

Linear Geometry

Matrices

Products

Matrix Inverse

Span and Linear Independence

Zero Variance Directions

Pseudo-Inverse

Projections

Basis

Rank

Principal Components

Standardized dataset

Definition 2.15

If $Q = (q_{ij})$ is the covariance matrix, then the correlation matrix is the $d \times d$ matrix $Q' = (q'_{ij})$ with entries

$$q'_{ij} = \frac{q_{ij}}{\sqrt{q_{ii}q_{jj}}}, \quad i, j = 1, 2, \ldots, d.$$

Theorem 2.4 (Standardized Covariance Equals Correlation)

The covariance matrix of the standardized dataset equals the correlation matrix of the original dataset.

Exercise 2.6

Prove Theorem (2.4).

Iris

Example 2.11

For the Iris dataset check Theorem (2.4).

```python
import numpy as np
from sklearn import datasets
from sklearn.preprocessing import StandardScaler

iris = datasets.load_iris()

# The dataset
dataset = iris["data"]

# standardize dataset
vectors = StandardScaler().fit_transform(dataset)
Qcorr = np.corrcoef(dataset.T)
Qcov = np.cov(vectors.T,bias=True)
np.allclose(Qcov,Qcorr)
```

Matrix Invers

Definition 2.16

Given a square matrix A, the inverse matrix is the matrix B satisfying

$$AB = I = BA.$$

When A has an inverse, we say A is invertible. If a matrix is $d \times d$, then the inverse is also $d \times d$. We write $B = A^{-1}$ for the inverse matrix of A.

Here I is the identity matrix. Not every square matrix has an inverse. For example, the zero matrix does not have an inverse.

Example 2.12

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \Rightarrow A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

Since we can't divide by zero, a 2×2 matrix is invertible only if $ad - bc \neq 0$.

Notes

Exercise 2.7

Prove that $(AB)^{-1} = B^{-1}A^{-1}$.

Exercise 2.8

Prove that for a linear system $A\mathbf{x} = \mathbf{b}$, *if* A *is invertible then* $\mathbf{x} = A^{-1}\mathbf{b}$.

Example 2.13

Solve the following linear system

$$\begin{cases} x + 2y + 3z = 1 \\ -3x + 6y = 2 \\ 10x - 5y + 23z = 3 \end{cases}$$

```python
import numpy as np

A = np.array([[1,2,3],[-3,6,0], [10,-5,23]])
b = np.array([1,2,3])
# Determinant of A
np.linalg.det(A)
# Inverse of A
np.linalg.inv(A)
# Solution of Ax=b
x = np.dot(np.linalg.inv(A),b)
```

Linear combination

Definition 2.17 (Linear combination)

A linear combination of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ *is*

$$t_1\mathbf{v}_1 + t_2\mathbf{v}_2 + \ldots + t_d\mathbf{v}_d,$$

where $t_1, t_2, \ldots, t_d$ *are scalars.*

Example 2.14

Let $\mathbf{u}, \mathbf{v}, \mathbf{w}$ be three vectors. Then

$$3\mathbf{u} - \frac{1}{6}\mathbf{v} + 9\mathbf{w} \quad , 5\mathbf{u} + 0\mathbf{v} - \mathbf{w}, \quad 0\mathbf{u} + 0\mathbf{v} + 0\mathbf{w},$$

are linear combinations of $\mathbf{u}, \mathbf{v}, \mathbf{w}$.

Example 2.15

Let A be a matrix with columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$, and let $\mathbf{x} = (t_1, t_2, \ldots, t_d)$. Then $A\mathbf{x}$ is a linear combination of the columns of A as:

$$A\mathbf{x} = t_1\mathbf{v}_1 + t_2\mathbf{v}_2 + \ldots + t_d\mathbf{v}_d.$$

Span

Definition 2.18 (Span)

The span of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ *is the set* S *of all linear combinations of* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$, *and we write*

$$S = span(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d).$$

Exercise 2.9

Let A *be the matrix with columns* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$. *Then* $S = span(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d)$ *is the set* S *of all vectors of the form* $A\mathbf{x}$.

Exercise 2.10

If each vector $\mathbf{v}_k$ *of* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ *is a linear combination of vectors* $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N$, *then*

$$span(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d) \subseteq span(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N).$$

Column space

Definition 2.19

Let A be a matrix. The column space of A is the span of its columns.

Example 2.16

```python
import sympy as sp
import scipy as sc
import numpy as np

A = sp.Matrix([[1, 6, 11], [2, 7, 12], [3, 8, 13], [4, 9, 14], [5, 10, 15]])
A
# column vectors
u = sp.Matrix([1,2,3,4,5])
v = sp.Matrix([6,7,8,9,10])
w = sp.Matrix([11,12,13,14,15])
A = sp.Matrix.hstack(u,v,w)
A
# returns minimal spanning set for column space of A
A.columnspace()
# returns minimal spanning orthonormal set for column space of A
A = np.array([[1, 6, 11], [2, 7, 12], [3, 8, 13], [4, 9, 14], [5, 10, 15]])
sc.linalg.orth(A)
```

`A.columnspace()` returns a minimal set of vectors spanning the column space of A. The *column rank* of A is the number of vectors returned: for A in the above example, the column rank is 2. `sc.linalg.orth(A)` returns a minimal orthonormal set of vectors spanning the column space of A.

Notes

Exercise 2.11

As in example 2.16, show that if

$$\mathbf{v}_1 = (1, 2, 3, 4, 5), \quad \mathbf{v}_2 = (6, 7, 8, 9, 10), \quad \mathbf{v}_3 = (11, 12, 13, 14, 15)$$

then $span(\mathbf{v}_1, \mathbf{v}_2) = span(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$.

Exercise 2.12

Show that: the column space of a matrix A consists of all vectors of the form $A\mathbf{x}$. A vector $\mathbf{b}$ is in the column space of A when $A\mathbf{x} = \mathbf{b}$ has a solution.

The augmented matrix $\bar{A} = (A, \mathbf{b})$ is obtained by adding $\mathbf{b}$ as an extra column next to the columns of A.

Exercise 2.13

Let $\bar{A}$ be the matrix A augmented by a vector $\mathbf{b}$. Then $\mathbf{b}$ is in the column space of A iff

$$\textit{column rank}(A) = \textit{column rank}(\bar{A}).$$

Notes

Exercise 2.14

Show that the vectors

$$\begin{aligned}
\mathbf{e}_1 &= (1, 0, 0, \ldots, 0, 0) \\
\mathbf{e}_2 &= (0, 1, 0, \ldots, 0, 0) \\
\mathbf{e}_3 &= (0, 0, 1, \ldots, 0, 0) \\
&\vdots \\
\mathbf{e}_d &= (0, 0, 0, \ldots, 0, 1)
\end{aligned}$$

span $\mathbb{R}^d$.

The set $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d\}$ is the *standard basis* for $\mathbb{R}^d$.

Row space

Definition 2.20

The row space of a matrix is the span of its rows.

Example 2.17

```python
import sympy as sp
import scipy as sc
import numpy as np

A = sp.Matrix([[1, 6, 11], [2, 7, 12], [3, 8, 13], [4, 9, 14], [5, 10, 15]])
A

# returns minimal spanning set for row space of A
A.rowspace()

# returns minimal spanning orthonormal set for column space of A
A = np.array([[1, 6, 11], [2, 7, 12], [3, 8, 13], [4, 9, 14], [5, 10, 15]])
sc.linalg.orth(A.T)
```

The *row rank* of a matrix is the number of vectors returned by `A.rowspace()`. This is the minimal number of vectors spanning the row space of A which for the above example is 2. `sc.linalg.orth(A.T)` returns a minimal orthonormal set of vectors spanning the row space of A.

Linearly independence

Definition 2.21

A linear combination $t_1\mathbf{v}_1 + t_2\mathbf{v}_2 + \ldots + t_d\mathbf{v}_d$ *is trivial if all the coefficients are zero:* $t_1 = t_2 = \ldots = t_d = 0$. *Otherwise it is non-trivial: if at least one coefficient is not zero.*
A linear combination $t_1\mathbf{v}_1 + t_2\mathbf{v}_2 + \ldots + t_d\mathbf{v}_d$ *vanishes if it equals the zero vector:*

$$t_1\mathbf{v}_1 + t_2\mathbf{v}_2 + \ldots + t_d\mathbf{v}_d = \mathbf{0}.$$

We say $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ *are linearly dependent if there is a non-trivial vanishing linear combination of* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$. *Otherwise, we say* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ *are linearly independent.*

Example 2.18

The vectors $\mathbf{v}_1 = (1, 2, 3, 4, 5)$, $\mathbf{v}_2 = (6, 7, 8, 9, 10)$, $\mathbf{v}_3 = (11, 12, 13, 14, 15)$ are linearly dependent, because

$$\mathbf{v}_3 + \mathbf{v}_1 - 2\mathbf{v}_2 = \mathbf{0}.$$

We can see $\mathbf{v}_3 = 2\mathbf{v}_2 - \mathbf{v}_1$.

Note

Exercise 2.15

Show that if $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ *are linearly dependent then at least one of the vectors is a linear combination of the remaining vectors.*

Exercise 2.16 (Homogeneous Linear Systems)

Let A *be the matrix with columns* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$. *Then* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$

- *are linearly dependent when* $A\mathbf{x} = \mathbf{0}$ *has a nonzero solution* $\mathbf{x}$, *and*
- *are linearly independent when* $A\mathbf{x} = \mathbf{0}$ *has only the zero solution* $\mathbf{x} = 0$.

Exercise 2.17

Show that if $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ *are orthonormal then they are linearly independent.*

Null space

Definition 2.22

The set of vectors $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{0}$, or the set of solutions $\mathbf{x}$ of $A\mathbf{x} = \mathbf{0}$, is the null space of the matrix A.
The cardinality of a minimal set of vectors spanning the null space of A is called the nullity of A.

Example 2.19

Show that the nullity of the following matrix is 1.

$$A = \begin{pmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{pmatrix}.$$

```python
import sympy as sp
import scipy as sc
import numpy as np

# using sympy
A = sp.Matrix([[1, 6, 11], [2, 7, 12], [3, 8, 13], [4, 9, 14], [5, 10, 15]])
A.nullspace()

# using numpy and scipy
A = np.array([[1, 6, 11], [2, 7, 12], [3, 8, 13], [4, 9, 14], [5, 10, 15]])
sc.linalg.null_space(A)
```

Data Sets
Linear Geometry
Matrices
Products
Matrix Inverse
Span and Linear Independence
Zero Variance Directions
Pseudo-Inverse
Projections
Basis
Rank
Principal Components

Note

Exercise 2.18

Let A be any matrix. Show that the null space, row space and column space of A equals the null space, row space and column space of A^tA, respectively.

Definition 2.23 (Orthogonal complements)

Let S and T be spans. We say S and T are orthogonal complements if every vector in S is orthogonal to every vector in T. In symbols, we write $S = T^\perp$ and $T = S^\perp$ (pronounced "T-perp" and "S-perp").

Exercise 2.19

Show that, if $S = span(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N)$, and A is the matrix with rows $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$, then $S^\perp$ equals the null space of A.

Exercise 2.20

For a matrix A, show that $(nullspace^\perp = rowspace)$ and $(rowspace^\perp = nullspace)$

Subspace

Definition 2.24 (Subspace)

A subspace is a set of vectors closed under addition and scalar multiplication. precisely: A subset $S \subseteq V$ is a subspace of the vector space V whenever for every $\mathbf{x}_1, \mathbf{x}_2 \in S$ and every scalar t we have

- $\mathbf{x}_1 + \mathbf{x}_2 \in S$ *and*
- $t\mathbf{x}_1 \in S$.

or equivalently: $t\mathbf{x}_1 + \mathbf{x}_2 \in S$.

Exercise 2.21

If V is a vector space then $\emptyset$ and V are the trivial subspaces of V.

Exercise 2.22

Show that

- *the null space: all $\mathbf{x}$'s satisfying $A\mathbf{x} = \mathbf{0}$,*
- *the row space: the span of the rows of A, and*
- *the column space: the span of the columns of A*

are subspaces, but

- *the solution space: the solutions $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$*

is not a subspace.

Projected dataset

Let $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N$ be the centered dataset of the dataset $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ in $\mathbb{R}^d$ with mean $\mathbf{m}$. Then the covariance is

$$Q = \frac{\mathbf{v}_1 \otimes \mathbf{v}_1 + \mathbf{v}_2 \otimes \mathbf{v}_2 + \ldots + \mathbf{v}_N \otimes \mathbf{v}_N}{N} = \frac{1}{N} A^t A,$$

where A is the matrix with rows $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N$.

If $\mathbf{b}$ is a vector, the projection of the centered dataset onto the line through $\mathbf{b}$ results in the reduced dataset

$$\mathbf{v}_1 \cdot \mathbf{b}, \mathbf{v}_2 \cdot \mathbf{b}, \dots, \mathbf{v}_N \cdot \mathbf{b}.$$

The mean of this projected dataset is zero, and its variance is

$$\frac{(\mathbf{v}_1 \cdot \mathbf{b})^2 + (\mathbf{v}_2 \cdot \mathbf{b})^2 + \ldots + (\mathbf{v}_N \cdot \mathbf{b})^2}{N} = \frac{1}{N} \mathbf{b}^t A^t A \mathbf{b} = \mathbf{b} \cdot Q \mathbf{b}.$$

Zero variance direction

Definition 2.25

Let $\mathbf{m}$ be a point in $\mathbb{R}^d$ and $\mathbf{b}$ a vector in $\mathbb{R}^d$. The hyperplane passing through $\mathbf{m}$ and orthogonal to $\mathbf{b}$ is the set of points $\mathbf{x}$ satisfying the equation

$$\mathbf{b} \cdot (\mathbf{x} - \mathbf{m}) = 0.$$

Example 2.20

In $\mathbb{R}^3$, a hyperplane is a plane, and in $\mathbb{R}^2$, a hyperplane is a line. In general, in $\mathbb{R}^d$, a hyperplane is $(d-1)$-dimensional, always one less than the ambient dimension.

Data Sets

Linear Geometry
- Matrices
- Products
- Matrix Inverse
- Span and Linear Independence
- Zero Variance Directions
- Pseudo-Inverse
- Projections
- Basis
- Rank

Principal Components

Hyperplane

Definition 2.26

A vector $\mathbf{b}$ is a zero variance direction of Q if the projected variance is zero:

$$\mathbf{b} \cdot Q\mathbf{b} = 0.$$

Theorem 2.5

Let $\mathbf{m}$ and Q be the mean and covariance of a dataset in $\mathbb{R}^d$. Then $\mathbf{b} \cdot Q\mathbf{b} = 0$ is the same as saying every point in the dataset lies in the hyperplane passing through $\mathbf{m}$ and orthogonal to $\mathbf{b}$: $\mathbf{b} \cdot (\mathbf{x} - \mathbf{m}) = 0$.

Theorem 2.6

Let Q be a covariance matrix. Then the null space of Q equals the zero variance directions of Q.

Corollary 2.4

Let Q be a covariance matrix of a centered dataset A. Then the null space of A equals the zero variance directions of Q.

Example

Example 2.21

Suppose the dataset

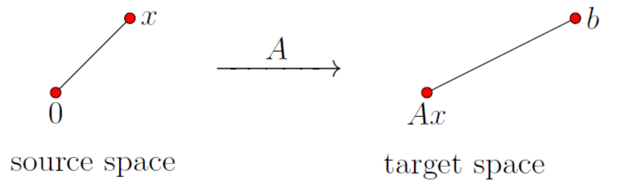
	t_1	t_2	t_3	t_4	t_5
$\mathbf{x}_1$	1	2	3	4	5
$\mathbf{x}_2$	6	7	8	9	10
$\mathbf{x}_3$	11	12	13	14	15
$\mathbf{x}_4$	16	17	18	19	20

Here we have 5 features. By the following code the null space of the covariance matrix, say Q, has 4 vectors which means it is 4-dimensional (or the nullity of Q is 4). Hence the dataset is a 1-dimensional dataset $(5-4=1)$. It means that there is a hyperplane (here a line) in $\mathbb{R}^5$ which we can project the dataset on it without loosing any information.

```python
import numpy as np
import scipy as sc

dataset = np.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20]])
Q = np.cov(dataset.T)
N = sc.linalg.null_space(Q)
nullity = N.shape[1]
```

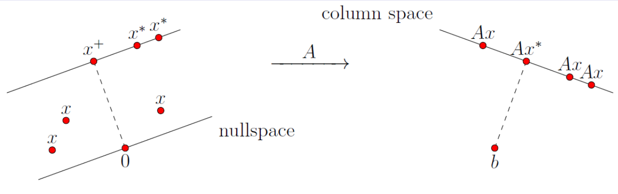
Concept

Think of $\mathbf{b}$ and $A\mathbf{x}$ as points, and measure the distance between them, and think of $\mathbf{x}$ and the origin $\mathbf{0}$ as points, and measure the distance between them.



If $A\mathbf{x} = \mathbf{b}$ is solvable, then, among all solutions $\mathbf{x}^*$, select the solution $\mathbf{x}^+$ closest to $\mathbf{0}$. More generally, if $A\mathbf{x} = \mathbf{b}$ is not solvable, select the points $\mathbf{x}^*$ so that $A\mathbf{x}^*$ is closest to $\mathbf{b}$, then, among all such $\mathbf{x}^*$, select the point $\mathbf{x}^+$ closest to the origin.

Concept



Even though the point $\mathbf{x}^+$ may not solve $A\mathbf{x} = \mathbf{b}$, this procedure results in a uniquely determined $\mathbf{x}^+$: While there may be several points $\mathbf{x}^*$, there is only one $\mathbf{x}^+$.

Concept

The results in this section are as follows. Let A be any matrix. There is a unique matrix A^+ — the *pseudo-inverse* of A — with the following properties:

- the linear system $A\mathbf{x} = \mathbf{b}$ is solvable, when $\mathbf{b} = AA^+\mathbf{b}$.
- $\mathbf{x}^+ = A^+\mathbf{b}$ is a solution of
  1. the linear system $A\mathbf{x} = \mathbf{b}$, if $A\mathbf{x} = \mathbf{b}$ is solvable.
  2. the *regression equation* $A^tA\mathbf{x} = A^t\mathbf{b}$, always.
- In either case,
  1. there is exactly one solution with minimum norm.
  2. Among all solutions, $\mathbf{x}^+$ has minimum norm.
  3. Every other solution is $\mathbf{x} = \mathbf{x}^+ + \mathbf{v}$ for $\mathbf{v}$ in the null space of A.

Key concepts in this section are the *residual*

$$|A\mathbf{x} - \mathbf{b}|^2$$

and the *regression equation*

$$A^tA\mathbf{x} = A^t\mathbf{b}.$$

Exercise 2.23

$\mathbf{x}$ *is a solution of* $A\mathbf{x} = \mathbf{b}$ *iff the residual is zero.*

Example

Example 2.22

For A and $\mathbf{b}$ as below

$$A = \begin{pmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -9 \\ -3 \\ 3 \\ 9 \\ 10 \end{pmatrix},$$

the linear system $A\mathbf{x} = \mathbf{b}$ and the regression equation $A^tA\mathbf{x} = A^t\mathbf{b}$ are

$$\begin{cases} x + 6y + 11z = -9 \\ 2x + 7y + 12z = -3 \\ 3x + 8y + 13z = 3 \\ 4x + 9y + 14z = 9 \\ 5x + 10y + 15z = 10 \end{cases}, \quad \begin{cases} 11x + 26y + 41z = 16 \\ 13x + 33y + 53z = 13 \\ 41x + 106y + 171z = 36 \end{cases},$$

respectively.

Data Sets
Linear Geometry
Matrices
Products
Matrix Inverse
Span and Linear Independence
Zero Variance Directions
Pseudo-Inverse
Projections
Basis
Rank
Principal Components

Residual minimizer

Let $\mathbf{b}$ be any vector, not necessarily in the column space of A. To see how close we can get to solving $A\mathbf{x} = \mathbf{b}$, we minimize the residual $|A\mathbf{x} - \mathbf{b}|^2$.

Definition 2.27 (Residual minimizer)

We say $\mathbf{x}^*$ is a residual minimizer if

$$|A\mathbf{x}^* - \mathbf{b}|^2 = \min_x |A\mathbf{x} - \mathbf{b}|^2 .$$

Theorem 2.7 (Existence of Residual Minimizer)

There is a residual minimizer $\mathbf{x}^*$ in the row space of A.

Exercise 2.24

Prove Theorem 2.7.

Data Sets

Linear Geometry
- Matrices
- Products
- Matrix Inverse
- Span and Linear Independence
- Zero Variance Directions
- Pseudo-Inverse
- Projections
- Basis
- Rank

Principal Components

Residual minimizer

Theorem 2.8

$\mathbf{x}^*$ *is a residual minimizer iff* $\mathbf{x}^*$ *solves the regression equation.*

Proof: let $\mathbf{v}$ be any vector, and t a scalar. Insert $\mathbf{x} = \mathbf{x}^* + t\mathbf{v}$ into the residual:

$$\begin{aligned} |A\mathbf{x} - \mathbf{b}|^2 &= |A(\mathbf{x}^* + t\mathbf{v}) - \mathbf{b}|^2 \\ &= |(A\mathbf{x}^* - \mathbf{b}) + At\mathbf{v}|^2 \\ &= |A\mathbf{x}^* - \mathbf{b}|^2 + 2t(A\mathbf{x}^* - \mathbf{b}) \cdot A\mathbf{v} + t^2|A\mathbf{v}|^2 \\ &:= f(t). \end{aligned}$$

If $\mathbf{x}^*$ is a residual minimizer, then $f(t)$ is minimized when $t = 0$. But a parabola $f(t) = a + 2bt + ct^2$ is minimized at $t = 0$ only when $b = 0$. Thus the linear coefficient vanishes, $(A\mathbf{x}^* - \mathbf{b}) \cdot A\mathbf{v} = 0$. This implies

$$A^t(A\mathbf{x}^* - \mathbf{b}) \cdot \mathbf{v} = (A\mathbf{x}^* - \mathbf{b}) \cdot A\mathbf{v} = 0.$$

Since $\mathbf{v}$ is any vector, this implies

$$A^t(A\mathbf{x}^* - \mathbf{b}) = \mathbf{0},$$

which is the regression equation. Conversely, if the regression equation holds, then the linear coefficient in the parabola $f(t)$ vanishes, so $t = 0$ is a minimum, establishing that $\mathbf{x}^*$ is a residual minimizer. $\square$

Note

Exercise 2.25

Any two residual minimizers differ by a vector in the nullspace of A.

Definition 2.28

We say $\mathbf{x}^+$ is a minimum norm residual minimizer if $\mathbf{x}^+$ is a residual minimizer and

$$|\mathbf{x}^+|^2 \leq |\mathbf{x}^*|^2$$

for any residual minimizer $\mathbf{x}^*$.

Theorem 2.9

Let $\mathbf{x}^*$ be a residual minimizer. Then $\mathbf{x}^*$ is a minimum norm residual minimizer iff $\mathbf{x}^*$ is in the row space of A.

Exercise 2.26

Prove Theorem 2.9.

Residual minimizer

Theorem 2.10 (Uniqueness of Residual Minimizer)

There is exactly one minimum norm residual minimizer $\mathbf{x}^+$*.*

Proof: If $\mathbf{x}_1^+$ and $\mathbf{x}_2^+$ are minimum norm residual minimizers, then $\mathbf{v} = \mathbf{x}_1^+ - \mathbf{x}_2^+$ is both in the row space and in the null space of A, $\mathbf{x}_1^+ - \mathbf{x}_2^+ = \mathbf{0}$. Hence $\mathbf{x}_1^+ = \mathbf{x}_2^+$. □

Putting the above all together, each vector $\mathbf{b}$ leads to a unique $\mathbf{x}^+$. Defining A^+ by setting

$$\mathbf{x}^+ = A^+\mathbf{b},$$

we obtain A^+, the pseudo-inverse of A.

Notice if A is, for example, 5×4, then $A\mathbf{x} = \mathbf{b}$ implies $\mathbf{x}$ is a 4-vector and $\mathbf{b}$ is a 5-vector. Then from $\mathbf{x}^+ = A^+\mathbf{b}$, it follows A^+ is 4×5. Thus the shape of A^+ equals the shape of A^t.

Theorem 2.11 (Regression Equation is Always Solvable)

The regression equation $A^tA\mathbf{x} = A^t\mathbf{b}$ *is always solvable. The solution of minimum norm is* $\mathbf{x}^+ = A^+\mathbf{b}$*. Any other solution differs by a vector in the null space of* A*.*

Example

Example 2.23

For A and $\mathbf{b}$ as below

$$A = \begin{pmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -9 \\ -3 \\ 3 \\ 9 \\ 10 \end{pmatrix},$$

the minimum norm solution of the regression equation $A^t A\mathbf{x} = A^t\mathbf{b}$ is

$$\mathbf{x}^+ = A^+\mathbf{b} = \frac{1}{150}\begin{pmatrix} -37 & -20 & -3 & 14 & 31 \\ -10 & -5 & 0 & 5 & 10 \\ 17 & 10 & 3 & -4 & -11 \end{pmatrix}\begin{pmatrix} -9 \\ -3 \\ 3 \\ 9 \\ 10 \end{pmatrix} = \frac{1}{15}\begin{pmatrix} 82 \\ 25 \\ -32 \end{pmatrix}.$$

```python
import sympy as sm

u = sm.Matrix([1,2,3,4,5])
v = sm.Matrix([6,7,8,9,10])
w = sm.Matrix([11,12,13,14,15])
A = sm.Matrix.hstack(u,v,w)

A.pinv()

b = sm.Matrix([-9,-3,3,9,10])
A.pinv()*b
```

Linear systems

Returning to the linear system, we have

Theorem 2.12

If the linear system is solvable, then every solution of the regression equation is a solution of the linear system, and vice-versa.

Corollary 2.5

The linear system $A\mathbf{x} = \mathbf{b}$ *is solvable iff* $\mathbf{b} = AA^{+}\mathbf{b}$. *When this happens,* $\mathbf{x}^{+} = A^{+}\mathbf{b}$ *is the solution of minimum norm.*

Example 2.24

For A and $\mathbf{b}$ as in Example 2.23, since

$$AA^{+}\mathbf{b} = \begin{pmatrix} -8 \\ -3 \\ 2 \\ 7 \\ 12 \end{pmatrix}$$

is not equal to $\mathbf{b}$, the linear system $A\mathbf{x} = \mathbf{b}$ is not solvable.

Note

Corollary 2.6

If A is invertible, then $A^+ = A^{-1}$.

Theorem 2.13 (Properties of Pseudo-Inverse)

1. $AA^+A = A$.
2. $A^+AA^+ = A^+$.
3. AA^+ *and* A^+A *are symmetric.*
4. *If A has orthonormal columns or orthonormal rows, then* $A^+ = A^t$.

Exercise 2.27

Prove Theorem 2.12, Corollary 2.5, Corollary 2.6 and Theorem 2.13.

Math for Data

Dr. S. M. Moosavi

Data Sets

Linear Geometry

Matrices

Products

Matrix Inverse

Span and Linear Independence

Zero Variance Directions

Pseudo-Inverse

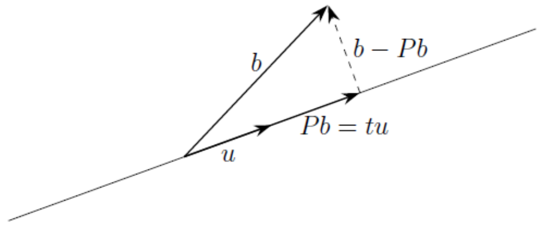
Projections

Basis

Rank

Principal Components

Projection onto a line



Let $\mathbf{u}$ be a unit vector, and let $\mathbf{b}$ be any vector. Let $span(\mathbf{u})$ be the line through $\mathbf{u}$. The *projection* of $\mathbf{b}$ onto $span(\mathbf{u})$ is the vector $\mathbf{v}$ in $span(\mathbf{u})$ that is closest to $\mathbf{b}$ (Exercise). It turns out this closest vector $\mathbf{v}$ equals $P\mathbf{b}$ for some matrix P, the *projection matrix*. Since $span(\mathbf{u})$ is a line, the projected vector $P\mathbf{b}$ is a multiple $t\mathbf{u}$ of $\mathbf{u}$. We have $\mathbf{b} - P\mathbf{b}$ is orthogonal to $\mathbf{u}$, so

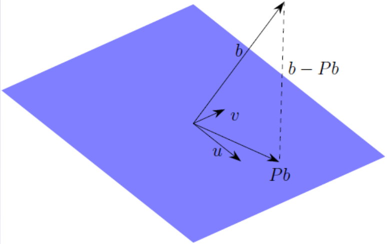
$$0 = (\mathbf{b} - P\mathbf{b}) \cdot \mathbf{u} = \mathbf{b} \cdot \mathbf{u} - P\mathbf{b} \cdot \mathbf{u} = \mathbf{b} \cdot \mathbf{u} - t\mathbf{u} \cdot \mathbf{u} = \mathbf{b} \cdot \mathbf{u} - t.$$

Solving for t, this implies $t = \mathbf{b} \cdot \mathbf{u}$. If U is the matrix with column $\mathbf{u}$

$$P\mathbf{b} = (\mathbf{b} \cdot \mathbf{u})\mathbf{u} = (\mathbf{u} \otimes \mathbf{u})\mathbf{b} = UU^t\mathbf{b}.$$

We call $\mathbf{b} \cdot \mathbf{u} = U^t\mathbf{b}$ the *reduced vector*.

Projecting onto a plane



Let $\mathbf{u}, \mathbf{v}$ be an orthonormal pair of vectors, so $\mathbf{u} \cdot \mathbf{v} = 0$, $\mathbf{u} \cdot \mathbf{u} = 1 = \mathbf{v} \cdot \mathbf{v}$. We project a vector $\mathbf{b}$ onto $span(\mathbf{u}, \mathbf{v})$. As before, there is a matrix P, the *projection matrix*, such that the projection of $\mathbf{b}$ onto the plane equals $P\mathbf{b}$. Then $\mathbf{b} - P\mathbf{b}$ is orthogonal to the plane:

$$(\mathbf{b} - P\mathbf{b}) \cdot \mathbf{u} = 0 \quad \text{and} \quad (\mathbf{b} - P\mathbf{b}) \cdot \mathbf{v} = 0.$$

Since $P\mathbf{b}$ lies in the plane, $P\mathbf{b} = r\mathbf{u} + s\mathbf{v}$ is a linear combination of $\mathbf{u}$ and $\mathbf{v}$. So:

$$r = \mathbf{b} \cdot \mathbf{u}, \quad s = \mathbf{b} \cdot \mathbf{v}.$$

If U is the matrix with columns $\mathbf{u}, \mathbf{v}$, then

$$P\mathbf{b} = (\mathbf{b} \cdot \mathbf{u})\mathbf{u} + (\mathbf{b} \cdot \mathbf{v})\mathbf{v} = (\mathbf{u} \otimes \mathbf{u} + \mathbf{v} \otimes \mathbf{v})\mathbf{b} = UU^t\mathbf{b}.$$

We call $(\mathbf{b} \cdot \mathbf{u}, \mathbf{b} \cdot \mathbf{v}) = U^t\mathbf{b}$ the *reduced vector*.

Projection matrices in general

Definition 2.29

Let S be a span. A matrix P is a projection matrix onto S if

1. $P\mathbf{b}$ *is in* S *for any vector* $\mathbf{b}$,
2. $P\mathbf{b} = \mathbf{b}$ *if* $\mathbf{b}$ *is in* S,
3. $\mathbf{b} - P\mathbf{b}$ *is orthogonal to* S *for any vector* $\mathbf{b}$.

Exercise 2.28

Show that, the projection of a vector onto a span equals the vector itself when the vector is already in the span.

Theorem 2.14 (Projection Onto a Column Space)

Let A be a matrix and $\mathbf{v}$ a vector. Then the projected vector onto the column space of A is $P\mathbf{v} = AA^+\mathbf{v}$ and the reduced vector is $\mathbf{x} = A^+\mathbf{v}$.

Theorem 2.15 (Projection Onto a Row Space)

Let A be a matrix and $\mathbf{v}$ a vector. Then the projected vector onto the row space of A is $P\mathbf{v} = A^+A\mathbf{v}$.

Exercise 2.29

Prove Theorems 2.14 and 2.15.

Example

Example 2.25

```python
import numpy as np

# projection of column vector b onto column space of A
def project_col(A,b):
  Aplus = np.linalg.pinv(A)
  x = np.dot(Aplus,b) # reduced
  return np.dot(A,x) # projected

# projection of column vector b onto row space of A
def project_row(A,b):
  Aplus = np.linalg.pinv(A)
  AplusA = np.dot(Aplus,A)
  return np.dot(AplusA,b) # projected

A = np.array([[1,6,11],[2,7,12],[3,8,13],[4,9,14],[5,10,15]])

b = np.array([-9,-3,3,9,10])
project_col(A, b.T)

b = np.array([-9,-3,3])
project_row(A, b.T)
```

Projecting onto Orthonormal Vectors

Theorem 2.16 (Projection Onto Orthonormal Vectors)

If the columns of U are orthonormal and $\mathbf{v}$ is a vector. Then the projected vector onto the column space of U is $P\mathbf{v} = UU^t\mathbf{v}$ and the reduced vector is $\mathbf{x} = U^t\mathbf{v}$.

Example 2.26

```python
import numpy as np

# projection of column vector b onto column space of U
# with orthonormal columns
def project_col_ortho(U,b):
  x = np.dot(U.T,b) # reduced
  return np.dot(U,x) # projected

# Matrices with orthnormal columns
U1 = np.array([[1,0,0],[0,1,0],[0,0,1]])
U2 = np.array([[1,1,1]/np.sqrt(3),[1,-1,0]/np.sqrt(2),[1,1,-2]/np.sqrt(6)])
U3 = np.array([[1,0,0],[0,1,0],[0,0,1],[0,0,0],[0,0,0]])

b = np.array([1,2,3]).reshape(3,1)

project_col_ortho(U1, b)
project_col_ortho(U2, b)

b = np.array([1,2,3,4,5]).reshape(5,1)

project_col_ortho(U3, b)
```

Data Sets

Linear Geometry

Matrices

Products

Matrix Inverse

Span and Linear Independence

Zero Variance Directions

Pseudo-Inverse

Projections

Basis

Rank

Principal Components

Direct sum

Let S and T be spans. Let $S + T$ consist of all sums of vectors $\mathbf{u} + \mathbf{v}$ with $\mathbf{u}$ in S and $\mathbf{u}$ in T. Then a moment's thought shows $S + T$ is itself a span. When the intersection of S and T is the zero vector, we write $S \oplus T$, and we say $S \oplus T$ is the *direct sum* of S and T.

Theorem 2.17

If S is a span in $\mathbb{R}^d$, then

$$\mathbb{R}^d = S \oplus S^{\perp}.$$

Theorem 2.18

If A is an $N \times d$ matrix,

$$nullspace \oplus rowspace = \mathbb{R}^d,$$

and the null space and row space are orthogonal to each other.

Note

Corollary 2.7

From Theorem 2.18, the projection matrix onto the null space of A is $P = I - A^+A$.

Theorem 2.19 (Projection is the Nearest Point in the Span)

Let $P\mathbf{b} = AA^+\mathbf{b}$ be the projection of $\mathbf{b}$ onto the column space of A, and let $\mathbf{x}^+ = A^+\mathbf{b}$ be the reduced vector. Then

$$|A\mathbf{x}^+ - \mathbf{b}|^2 = \min_{\mathbf{x}} |A\mathbf{x} - \mathbf{b}|^2.$$

Exercise 2.30

Prove Theorems 2.17, 2.18, 2.19 and Corollary 2.7.

Data Sets
Linear Geometry
Matrices
Products
Matrix Inverse
Span and Linear Independence
Zero Variance Directions
Pseudo-Inverse
Projections
Basis
Rank
Principal Components

Definition

Let S be the span of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$. Then there are many other choices of spanning vectors for S. For example,
$\mathbf{v}_1 + \mathbf{v}_2, \mathbf{v}_2, \ldots, \mathbf{v}_N$ also spans S.
If S cannot be spanned by fewer than N vectors, then we say $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ is a basis for S, and we call N is the dimension of S.

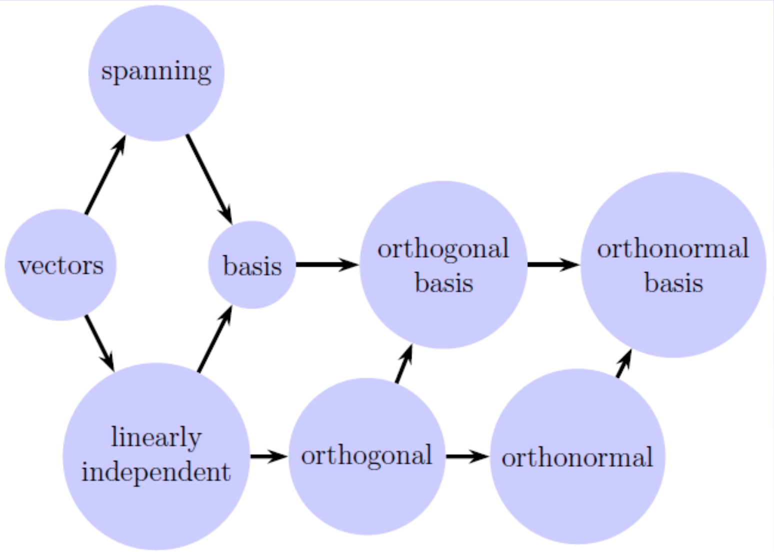
Definition 2.30 (Basis and Dimension)

A basis for a span S is a minimal spanning set of vectors. The dimension of S is the number of vectors in any basis for S.

Definition 2.31

When a basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ consists of orthogonal vectors, we say $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ is an orthogonal basis.
When $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ are also unit vectors, we say $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ is an orthonormal basis.

Vector classes



Note

Theorem 2.20

If $S = span(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N)$, *then* $\dim S \leq N$.

Theorem 2.21

If a span $S_1 \subseteq S_2$, *then* $\dim S_1 \leq \dim S_2$.

- `rowspace()` returns a basis of the row space,
- `columnspace()` returns a basis of the column space,
- `nullspace()` returns a basis for the null space,
- row rank equals the dimension of the row space,
- column rank equals the dimension of the column space,
- nullity equals the dimension of the null space.

Exercise 2.31

Prove all the above statements.

Note

Theorem 2.22 (Spanning Plus Linearly Independent Equals Basis)

Let S be the span of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$. Then the vectors are a basis for S if and only if they are linearly independent.

Note: To check for linear independence of given vectors:

- assemble the vectors as columns of a matrix A, and check whether `A.nullspace()` equals zero. If that is the case, the vectors are a basis for their span. If not, the vectors are not a basis for their span.
- assemble the vectors as columns of a matrix A, if `np.linalg.matrix_rank(A)` equals the number of vectors then the vectors are independent.

Exercise 2.32

Prove Theorem 2.22.

MNIST example

The MNIST dataset consists of vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N$ in $\mathbb{R}^d$, where $N = 60000$ and $d = 28 \times 28 = 784$. For the MNIST dataset, the dimension is 712, as returned by the code

Example 2.27

```python
from keras.datasets import mnist
import numpy as np

(train_X,train_y), (test_X,test_y) = mnist.load_data()

vectors = train_X.reshape(60000,784) # each image in one row

vectors = np.array(vectors)
rank = np.linalg.matrix_rank(vectors) # returns 712
```

In particular, since $712 < 784$, approximately 10% of pixels are never touched by any image. For example, a likely pixel to remain untouched is at the top left corner $(0,0)$. For this dataset, there are $72 = 784 - 712$ zero variance directions.

Concept

If A is an $N \times d$ matrix, then $\mathbf{x} \mapsto A\mathbf{x}$ is a linear transformation that sends a vector $\mathbf{x}$ in $\mathbb{R}^d$ (*the source space*) to the vector $A\mathbf{x}$ in $\mathbb{R}^N$ (*the target space*). The transpose A^t goes in the reverse direction: The linear transformation $\mathbf{b} \mapsto A^t\mathbf{b}$ sends a vector $\mathbf{b}$ in $\mathbb{R}^N$ (the target space) to the vector $A^t\mathbf{b}$ in $\mathbb{R}^d$ (the source space). It follows that for an $N \times d$ matrix, the dimension of the source space is d, and the dimension of the target space is N,

$$\dim(\text{source space}) = d, \quad \dim(\text{target space}) = N.$$

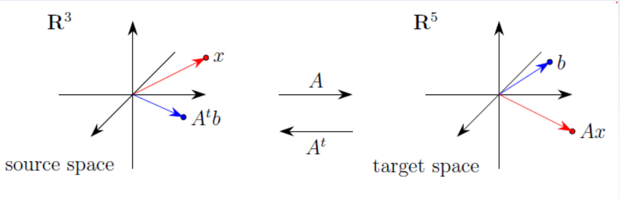


Figure 2.1: A 5×3 matrix A is a linear transformation from $\mathbb{R}^3$ to $\mathbb{R}^5$.

Rank Theorem

We know that, the column space is in the target space, and the row space is in the source space. Thus we always have

$$0 \leq \text{row rank} \leq d, \quad \text{and} \quad 0 \leq \text{column rank} \leq N.$$

Example 2.28

For the matrix a as below, the column rank is 2, the row rank is 2, and the nullity is 1. Thus the column space is a 2-d plane in $\mathbb{R}^5$, the row space is a 2-d plane in $\mathbb{R}^3$, and the null space is a 1-d line in $\mathbb{R}^3$.

$$A = \begin{pmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{pmatrix}.$$

The main result in this section is

Theorem 2.23 (Rank Theorem)

Let A be any matrix. Then row rank$(A) =$ *column rank*(A).

Note

Exercise 2.33

Prove Theorem 2.23.

Because the row rank and the column rank are equal, we just say rank of a matrix, and we write rank(A). In Python, the following code returns the rank of a matrix.

```python
import sympy import sm
A = sm.Matrix(...)
rank = A.rank()

import numpy as np
A = np.array(...)
rank = np.linalg.matrix_rank(A)
```

Note

Theorem 2.24 (Upper bound for Rank)

For any $N \times d$ matrix, the rank is never greater than $\min(N, d)$.

Definition 2.32

An $N \times d$ matrix A is full-rank if its rank is the highest it can be: rank$(A) = \min(N, d)$.

Note. For any $N \times d$ matrix A:

- When $N \geq d$, full-rank is the same as rank$(A) = d$, which is the same as saying the columns are linearly independent and the rows span $\mathbb{R}^d$.
- When $N \leq d$, full-rank is the same as rank$(A) = N$, which is the same as saying the rows are linearly independent and the columns span $\mathbb{R}^N$.
- When $N = d$, full-rank is the same as saying the rows are a basis of $\mathbb{R}^d$, and the columns are a basis of $\mathbb{R}^N$.

When A is a square matrix, we can say more:

Theorem 2.25

Let A be a square matrix. Then A is full-rank iff A is invertible.

Exercise 2.34

Prove all the above statements.

Orthogonal matrix

Theorem 2.26

Let U be a matrix.

- *U has orthonormal rows iff $UU^t = I$.*
- *U has orthonormal columns iff $U^tU = I$.*

If U is square and either holds, then they both hold.

Definition 2.33 (Orthogonal Matrix)

A square matrix U satisfying

$$UU^t = I = U^tU$$

is an orthogonal matrix.

Equivalently, we can say

Exercise 2.35

A matrix U is orthogonal iff its rows are an orthonormal basis iff its columns are an orthonormal basis.

Orthogonal matrix

For orthogonal matrices, say U, since

$$U\mathbf{u} \cdot U\mathbf{v} = \mathbf{u} \cdot U^t U\mathbf{v} = \mathbf{u} \cdot \mathbf{v},$$

U preserves dot products. Since lengths are dot products, U also preserves lengths. Since angles are computed from dot products, U also preserves angles. Summarizing,

Exercise 2.36

Orthogonal Matrices Preserve Angles, Lengths, and Dot Products.

As a consequence,

Exercise 2.37

Let U be an orthogonal matrix. If $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ are orthonormal, then $U\mathbf{v}_1, U\mathbf{v}_2, \ldots, U\mathbf{v}_N$ are orthonormal.

Orthogonal matrix

Exercise 2.38

In two dimensions, $d = 2$, an orthogonal matrix must have two orthonormal columns, so must be of the form

$$U = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad \textit{or} \quad U = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix}.$$

In the first case, U is a rotation, while in the second, U is a rotation followed by a reflection.

Exercise 2.39 (Orthonormal Basis Expansion)

If $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ is an orthonormal basis, and $\mathbf{v}$ is any vector, then

$$\mathbf{v} = (\mathbf{v}\cdot\mathbf{v}_1)\mathbf{v}_1 + (\mathbf{v}\cdot\mathbf{v}_2)\mathbf{v}_2 + \ldots + (\mathbf{v}\cdot\mathbf{v}_d)\mathbf{v}_d = \sum_{i=1}^{d}(\mathbf{v}\cdot\mathbf{v}_i)\mathbf{v}_i$$

and

$$|\mathbf{v}|^2 = |\mathbf{v}\cdot\mathbf{v}_1|^2 + |\mathbf{v}\cdot\mathbf{v}_2|^2 + \ldots + |\mathbf{v}\cdot\mathbf{v}_d|^2 = \sum_{i=1}^{d}|\mathbf{v}\cdot\mathbf{v}_i|^2.$$

Dataset

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ be a dataset, and let A be the dataset matrix with rows $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

The dataset is full-rank if A is full-rank. This is the same as saying the span of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ is the whole feature space.

The *dimension* of the dataset is the rank of A. Hence the dimension of the dataset equals the rank of $A^t A$.

When the dataset is centered, the covariance is the matrix $Q = A^t A/N$.

Since scaling a matrix has no effect on the rank, we conclude:

Exercise 2.40

The dimension of a dataset equals the rank of its covariance.

Outline

1. Data Sets
2. Linear Geometry
3. Principal Components

Linear transformation

Matrix multiplication by an $N \times d$ matrix A sends a point $\mathbf{x}$ in the source space $\mathbb{R}^d$ to a point $\mathbf{b} = A\mathbf{x}$ in the target space $\mathbb{R}^N$ (Figure 2.1).

Equivalently, since points in $\mathbb{R}^d$ are essentially the same as vectors in $\mathbb{R}^d$, an $N \times d$ matrix A sends a vector $\mathbf{v}$ in $\mathbb{R}^d$ to a vector $A\mathbf{v}$ in $\mathbb{R}^N$.

So, a matrix A induces a *linear transformation*: Matrix multiplication by A satisfies

$$A(\mathbf{v}_1 + \mathbf{v}_2) = A\mathbf{v}_1 + A\mathbf{v}_2, \quad A(t\mathbf{v}) = tA\mathbf{v}.$$

If we let

$$\mathbf{u} = \frac{\mathbf{v}_1 - \mathbf{v}_2}{|\mathbf{v}_1 - \mathbf{v}_2|},$$

then $\mathbf{u}$ is a unit vector, $|\mathbf{u}| = 1$, and by linearity

$$|A\mathbf{u}| = \left|\frac{A(\mathbf{v}_1 - \mathbf{v}_2)}{|\mathbf{v}_1 - \mathbf{v}_2|}\right| = \frac{|A\mathbf{v}_1 - A\mathbf{v}_2|}{|\mathbf{v}_1 - \mathbf{v}_2|}.$$

This ratio is a scaling factor of the linear transformation which depends on the given vectors $\mathbf{v}_1, \mathbf{v}_2$.

Scaling distortions

Definition 3.1

Let

$$\sigma_1 = \max_{\mathbf{u}} |A\mathbf{u}| \quad \textit{and} \quad \sigma_2 = \min_{\mathbf{u}} |A\mathbf{u}|.$$

Here the maximum and minimum are taken over all unit vectors $\mathbf{u}$*. Then* σ_1 *is the distance of the furthest image from the origin, and* σ_2 *is the distance of the nearest image to the origin.*

It turns out σ_1 and σ_2 are the top and bottom singular values of A.

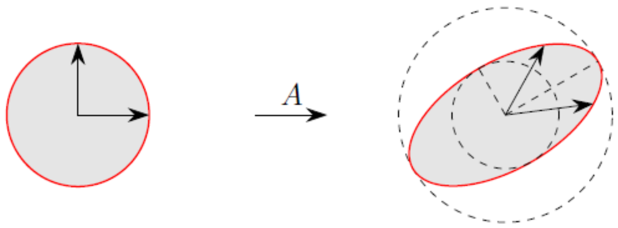


Figure 3.1: Image of the unit circle (in $\mathbb{R}^2$) with $\sigma_1 = 1.5$ and $\sigma_2 = 0.75$.

Scaling distortions

For simplicity, assume both the source space and the target space are $\mathbb{R}^2$; then A is 2×2.

Definition 3.2

The unit circle (in red in Figure 3.1) is the set of vectors $\mathbf{u}$ *satisfying*

$$\{\mathbf{u} : |\mathbf{u}| = 1\}.$$

The image of the unit circle (also in red in Figure 3.1) is the set of vectors of the form

$$\{A\mathbf{u} : |\mathbf{u}| = 1\}.$$

The annulus is the set (the region between the dashed circles in Figure 3.1) of vectors $\mathbf{b}$ *satisfying*

$$\{\mathbf{b} : \sigma_2 < |\mathbf{b}| < \sigma_1\}.$$

It turns out the image is an ellipse, and this ellipse lies in the annulus.

Thus the numbers σ_1 and σ_2 constrain how far the image of the unit circle is from the origin, and how near the image is to the origin.

Scaling distortions

To relate σ_1 and σ_2 to what we've seen before, let $Q = A^t A$. Then,

$$\sigma_1^2 = \max |A\mathbf{u}|^2 = \max \{(A\mathbf{u}) \cdot (A\mathbf{u})\} = \max \left\{\mathbf{u} \cdot A^t A\mathbf{u}\right\}$$
$$= \max \{\mathbf{u} \cdot Q\mathbf{u}\}.$$

Thus σ_1^2 is the maximum projected variance corresponding to the covariance Q. Similarly, σ_2^2 is the minimum projected variance corresponding to the covariance Q.

Now let $Q = AA^t$, and let $\mathbf{b}$ be in the image. Then $\mathbf{b} = A\mathbf{u}$ for some unit vector $\mathbf{u}$, and

$$\mathbf{b} \cdot Q^{-1}\mathbf{b} = (A\mathbf{u}) \cdot Q^{-1}A\mathbf{u} = \mathbf{u} \cdot A^t(AA^t)^{-1}A\mathbf{u} = \mathbf{u} \cdot I\mathbf{u} = |\mathbf{u}|^2 = 1.$$

This shows the image of the unit circle is the inverse covariance ellipse corresponding to the covariance Q, with major axis length $2\sigma_1$ and minor axis length $2\sigma_2$.

Eigenvalues and Eigenvectors

Definition 3.3

If A is a square matrix. An eigenvector of A is a nonzero vector $\mathbf{v}$ such that $A\mathbf{v} = \lambda\mathbf{v}$ for some scalar λ, called the corresponding eigenvalue.

Theorem 3.1

If $\mathbf{v}$ is an eigenvector corresponding to eigenvalue λ, any scalar multiple $\mathbf{u} = t\mathbf{v}$ is also an eigenvector corresponding to the same eigenvalue λ.

Exercise 3.1

Prove Theorem 3.1.

Note. To find the eigenvalues of a matrix A we have to solve the system $\mathbf{0} = A\mathbf{v} - \lambda\mathbf{v} = A\mathbf{v} - \lambda I\mathbf{v} = (A - \lambda I)\mathbf{v}$. This represents a *homogeneous system of linear equations* and it has a non-trivial solution only when the determinant of the coefficient matrix is 0. So, we have to solve $\det(A - \lambda I) = 0$. This equation is called the *characteristic equation* (where $\det(A - \lambda I) = 0$ is called the *characteristic polynomial*) and by solving this for λ, we get the eigenvalues.
To find the eigenvectors we have to solve the systems $(A - \lambda I)\mathbf{v} = \mathbf{0}$, for each eigenvalue, separately.

Example

Example 3.1

Let

$$Q = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

The characteristic polynomial is

$$\begin{aligned} \det(Q - \lambda I) &= \left| \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| = \left| \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \right| \\ &= \left| \begin{pmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{pmatrix} \right| = (2-\lambda)(2-\lambda) - (1)(1) \\ &= (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3. \end{aligned}$$

The characteristic equation is $\lambda^2 - 4\lambda + 3 = 0$. Then Q has eigenvalues $\lambda_1 = 3$ and $\lambda_2 = 1$. Now by solving the systems

$$\mathbf{0} = (A - \lambda_1 I)\mathbf{v}_1 = \begin{pmatrix} 2-3 & 1 \\ 1 & 2-3 \end{pmatrix} \begin{pmatrix} v_{11} \\ v_{12} \end{pmatrix} \Rightarrow \begin{cases} -v_{11} + v_{12} = 0 \\ v_{11} - v_{12} = 0 \end{cases}$$

and

$$\mathbf{0} = (A - \lambda_2 I)\mathbf{v}_2 = \begin{pmatrix} 2-1 & 1 \\ 1 & 2-1 \end{pmatrix} \begin{pmatrix} v_{21} \\ v_{22} \end{pmatrix} \Rightarrow \begin{cases} v_{21} + v_{22} = 0 \\ v_{21} + v_{22} = 0 \end{cases}$$

we find the corresponding eigenvectors $\mathbf{v}_1 = (v_{11}, v_{12}) = (1, 1)$ and $\mathbf{v}_2 = (v_{21}, v_{22}) = (-1, 1)$. These are not unit vectors, but the corresponding unit eigenvectors are $\mathbf{u}_1 = (1/\sqrt{2}, 1/\sqrt{2})$ and $\mathbf{u}_2 = (-1/\sqrt{2}, 1/\sqrt{2})$.

Example

For Example 3.1, we have the following code:

Example 3.2

```
import numpy as np

A = np.array([[2, 1], [1, 2]])
eigenvalues, eigenvectors = np.linalg.eig(A)
print(f'{eigenvalues = }')
print(f'{eigenvectors = }')
```

Example

Example 3.3

Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & -5 & -6 \\ 1 & 4 & -9 \end{pmatrix}.$$

The characteristic polynomial is

$$\begin{aligned} \det(A - \lambda I) &= \left| \begin{pmatrix} 1-\lambda & 2 & 3 \\ 3 & -5-\lambda & -6 \\ 1 & 4 & -9-\lambda \end{pmatrix} \right| \\ &= (1-\lambda)\big((-5-\lambda)(-9-\lambda) - (-6)(4)\big) \\ &\quad - 2\big((3)(-9-\lambda) - (-6)(1)\big) \\ &\quad + 3\big((3)(4) - (-5-\lambda)(1)\big) \end{aligned}$$

The characteristic equation is $\lambda^3 + 13\lambda^2 + 46\lambda - 162 = 0$. Here, we have complex eigenvalues:

```python
import numpy as np

A = np.array([[1,2,3], [3,-5,-6], [1,4,-9]])
eigenvalues, eigenvectors = np.linalg.eig(A)
print(f'{eigenvalues = }')
print(f'{eigenvectors = }')
```

Note

Theorem 3.2

The eigenvalues of A and the eigenvalues of A^t are the same.

Theorem 3.3

If $\mathbf{v}$ is a unit eigenvector of a symmetric matrix Q, then $\mathbf{v} \cdot Q\mathbf{v}$ equals the corresponding eigenvalue. In particular, the eigenvalues of a covariance matrix are nonnegative.

Theorem 3.4

For a symmetric matrix Q, eigenvectors corresponding to distinct eigenvalues are orthogonal.

Exercise 3.2

Prove Theorems 3.2, 3.3 and 3.4.

Note

Definition 3.4 (Spectrum)

The spectrum of a matrix is the list of eigenvalues, repeated according to multiplicity. An important quantity associated with the spectrum is the maximum absolute value of any eigenvalue. This is known as the spectral radius of the matrix.

Theorem 3.5

Let A be an arbitrary $d \times d$ matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_d$.

1. $trace(A) = \sum_{i=1}^{d} a_{ii} = \sum_{i=1}^{d} \lambda_i = \lambda_1 + \lambda_2 + \cdots + \lambda_d.$
2. $\det(A) = \prod_{i=1}^{d} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_d.$
3. *The eigenvalues of the kth power of A; i.e., the eigenvalues of A^k, for any positive integer k, are $\lambda_1^k, \lambda_2^k, \ldots, \lambda_d^k$.*
4. *The matrix A is invertible iff every eigenvalue is nonzero.*
5. *If A is invertible, then the eigenvalues of A^{-1} are $1/\lambda_1, 1/\lambda_2, \cdots, 1/\lambda_d$ and each eigenvalue's geometric multiplicity coincides.*

Exercise 3.3

Prove Theorem 3.5.

Diagonalization

Definition 3.5 (Diagonalization)

A square matrix A is called diagonalizable or non-defective if it is similar to a diagonal matrix. That is, if there exists an invertible matrix P and a diagonal matrix D such that $P^{-1}AP = D$. This is equivalent to $A = PDP^{-1}$. (Such P, D are not unique.)
A square matrix that is not diagonalizable is called defective.

Theorem 3.6 (Eigenvalue Decomposition (EVD))

Let Q be a symmetric $d \times d$ matrix. There is an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ in $\mathbb{R}^d$ of eigenvectors of Q, with corresponding eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d.$$

Exercise 3.4

Prove Theorem 3.6.

Text

As a corollary of Theorem 3.6:

Theorem 3.7 (Diagonalization)

Let Q be a symmetric matrix. There is an orthogonal matrix V and a diagonal matrix D such that

$$Q = VDV^t.$$

When this happens, the columns of V are the eigenvectors of Q, and the diagonal entries of D are the eigenvalues of Q.

Exercise 3.5

Prove Theorem 3.7.

Definition 3.6

The orthonormal basis eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ (as in Theorem 3.6) are the principal components of the symmetric matrix Q. The eigenvalues and eigenvectors of Q, taken together, are the eigendata of Q.

Example

Example 3.4

For the symmetric matrix

$$Q = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

we have the following diagonalization:

$$1) \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix},$$

$$2) \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$$

```python
import numpy as np

Q = np.array([[2, 1], [1, 2]])
eigenvalues, eigenvectors = np.linalg.eig(Q)
V = eigenvectors
D = np.diag(eigenvalues)
VDV_t = np.dot(V, np.dot(D,V.T))
print(f'{V = }')
print(f'{D = }')
print(np.allclose(Q, VDV_t))

import sympy as sm

Q = sm.Matrix([[2, 1], [1, 2]])
V, D = Q.diagonalize()
V
D
```

Example

Example 3.5

For a symmetric matrix

$$Q = \begin{pmatrix} a & b \\ b & c \end{pmatrix},$$

with $b \neq 0$, we have a diagonalization, $Q = VDV^t$, where

$$D = \frac{1}{2}\begin{pmatrix} a+c-\sqrt{(a-c)^2+4b^2} & 0 \\ 0 & a+c+\sqrt{(a-c)^2+4b^2} \end{pmatrix}$$

$$V = \frac{1}{2b}\begin{pmatrix} a+c-\sqrt{(a-c)^2+4b^2} & a+c+\sqrt{(a-c)^2+4b^2} \\ 2b & 2b \end{pmatrix}$$

```python
import sympy as sm

a, b, c = sm.symbols("a b c")
Q = sm.Matrix([[a, b], [b, c]])
V, D = Q.diagonalize()
```

Note

Theorem 3.8

The rank of a square symmetric matrix equals the number of its nonzero eigenvalues.

Theorem 3.9

Let $Q = VDV^t$ be a diagonalization of a symmetric matrix Q. Then Q is invertible iff all its eigenvalues are nonzero. When this happens, we have

$$Q^{-1} = VD^{-1}V^t.$$

Theorem 3.10

If $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r$ are the nonzero eigenvalues of Q, then $1/\lambda_1 \leq 1/\lambda_2 \leq \cdots \leq 1/\lambda_r$ are the nonzero eigenvalues of Q^+. Moreover, if $Q = VDV^t$ is a diagonalization, then $Q^+ = VD^+V^t$.

Note

Theorem 3.11

Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ be the orthonormal basis of eigenvectors of Q corresponding to eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_d$. Then the linear system $Q\mathbf{x} = \mathbf{b}$ has a solution $\mathbf{x}$ for every vector $\mathbf{b}$ iff all eigenvalues are nonzero, in which case

$$\mathbf{x} = \frac{1}{\lambda_1}(\mathbf{b}\cdot\mathbf{v}_1)\mathbf{v}_1 + \frac{1}{\lambda_2}(\mathbf{b}\cdot\mathbf{v}_2)\mathbf{v}_2 + \cdots + \frac{1}{\lambda_d}(\mathbf{b}\cdot\mathbf{v}_d)\mathbf{v}_d.$$

Theorem 3.12

Let Q be a symmetric $d \times d$ matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_d$ and orthonormal eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$. Then

$$Q = \lambda_1 \mathbf{v}_1 \otimes \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 \otimes \mathbf{v}_2 + \cdots + \lambda_d \mathbf{v}_d \otimes \mathbf{v}_d.$$

In particular, when Q is nonnegative, the dataset consisting of the $2d$ points

$$\pm\sqrt{\lambda_1}\mathbf{v}_1, \pm\sqrt{\lambda_2}\mathbf{v}_2, \ldots, \pm\sqrt{\lambda_d}\mathbf{v}_d$$

is centered and has covariance Q/d.

Exercise 3.6

Prove Theorems 3.8-3.12.

Covariance matrix

Let Q be a covariance matrix. We know that the eigenvalues of a covariance matrix are nonnegative.

Definition 3.7

An eigenvalue λ_1 of Q is the top eigenvalue if $\lambda_1 \geq \lambda$ for any other eigenvalue. An eigenvalue λ_d of Q is the bottom eigenvalue if $\lambda_d \leq \lambda$ for any other eigenvalue.

Definition 3.8

We say that a unit vector $\mathbf{b}$ is best-fit for Q or best-aligned with Q if the maximum

$$\lambda_1 = \max_{|\mathbf{v}|=1} \mathbf{v} \cdot Q\mathbf{v}$$

is achieved at $\mathbf{v} = \mathbf{b}$: i.e. $\lambda_1 = \mathbf{b} \cdot Q\mathbf{b}$.

When Q is a covariance matrix, this means the unit vector $\mathbf{b}$ is chosen so that the variance $\mathbf{b} \cdot Q\mathbf{b}$ of the dataset projected onto $\mathbf{b}$ is maximized.

Covariance matrix

Theorem 3.13 (Maximum Projected Variance is an Eigenvalue)

Let Q be a symmetric matrix. Then $\lambda_1 = \max_{|\mathbf{v}|=1} \mathbf{v} \cdot Q\mathbf{v}$ is the top eigenvalue of Q.

Theorem 3.14 (Best-aligned vector is an eigenvector)

Let Q be a symmetric matrix. Then a best-aligned vector $\mathbf{b}$ is an eigenvector of Q corresponding to the top eigenvalue λ_1.

Exercise 3.7

Prove the above theorems.

Just as the maximum variance, the minimum variance

$$\lambda_d = \min_{|\mathbf{v}|=1} \mathbf{v} \cdot Q\mathbf{v}$$

is the bottom eigenvalue, and the corresponding eigenvector $\mathbf{v}_d$ is the worst-aligned vector.

By the eigenvalue decomposition, the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ of a symmetric matrix Q may be arranged in decreasing order, and may be positive, zero, or negative scalars. When Q is a covariance, the eigenvalues are nonnegative, and the bottom eigenvalue is at least zero. When the bottom eigenvalue is zero, the corresponding eigenvectors are zero variance directions.

Example

Example 3.6

For the Iris dataset, the eigenvalues are

$$4.22 > 0.24 > 0.08 > 0.02.$$

The total variance of the Iris dataset is

$$\text{Total Variance} = trace(Q) = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \simeq 4.57.$$

1. The top eigenvalue accounts for 92.5% of the total variance.
2. The top two eigenvalues account for 97.8% of the total variance.
3. The top three eigenvalues account for 99.5% of the total variance.
4. The top four eigenvalues account for 100% of the total variance.

```python
import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()
dataset = iris.data
# Covariance matrix
Q = np.cov(dataset.T)
# Eigen data
eigenvalues, eigenvectors = np.linalg.eig(Q)
# Compute total variance
total_variance = np.trace(Q)
# Percentage
percent = 0
for i in range(len(eigenvalues)):
  percent += (eigenvalues[i]/total_variance)*100
  print(percent)
```

Singular Value

Definition 3.9 (Singular Value)

A positive number $\sigma > 0$ *is a singular value of a matrix* A*, if there are nonzero vectors* $\mathbf{v}$ *and* $\mathbf{u}$ *satisfying*

$$A\mathbf{v} = \sigma\mathbf{u} \quad \text{and} \quad A^t\mathbf{u} = \sigma\mathbf{v}.$$

When this happens, $\mathbf{v}$ *is a right singular vector and* $\mathbf{u}$ *is a left singular vector associated to* σ*.*

Exercise 3.8

The singular values of A *are the same as the singular values of* A^t*.*

Theorem 3.15

Let A *be any matrix and* Q *be the symmetric matrix* $Q = A^tA$*. Then*

1. *the rank of* A *equals the rank of* Q*,*
2. σ *is a singular value of* A *iff* $\lambda = \sigma^2$ *is a positive eigenvalue of* Q*.*

Exercise 3.9

Prove Theorem 3.15.

Example

Example 3.7

Find the singular values of the following matrix.

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Answer. Let

$$Q = A^t A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

The characteristic equation for Q reads:

$$0 = \det(Q - \lambda I) = \lambda^2 - 3\lambda + 1 \Rightarrow \lambda_{1,2} = \frac{3 \pm \sqrt{5}}{2}.$$

Hence, $\lambda_1 = 2.62$ and $\lambda_2 = 0.38$. So, the singular values of A are $\sigma_1 = \sqrt{2.62} = 1.62$ and $\sigma_2 = \sqrt{0.38} = 0.62$.
Now, for example, if $\mathbf{v}_1$ is the eigenvector corresponding to λ_1 and we set $\mathbf{u}_1 = A\mathbf{v}_1/\sigma_1$ then

- $A\mathbf{v}_1 = \sigma_1 \mathbf{u}_1$ and
- $A^t \mathbf{u}_1 = A^t A \mathbf{v}_1/\sigma_1 = Q\mathbf{v}_1/\sigma_1 = \lambda_1 \mathbf{v}_1/\sigma_1 = \sigma_1 \mathbf{v}_1.$

Thus $\mathbf{v}_1$, $\mathbf{u}_1$ are right and left singular vectors corresponding to the singular value σ_1 of A.

SVD

Theorem 3.16 (Singular Value Decomposition)

Let A be any $N \times d$ matrix and let r be the rank of A. Then there is an orthonormal basis $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ of the target space and an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ of the source space and positive scalars $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, such that

$$A\mathbf{v}_k = \sigma_k \mathbf{u}_k, \quad A^t \mathbf{u}_k = \sigma_k \mathbf{v}_k, \quad k = 1, 2, \ldots, r,$$

and

$$A\mathbf{v}_k = 0, \quad A^t \mathbf{u}_k = 0, \quad k > r.$$

Theorem 3.17 (Diagonalization (SVD))

If A is any matrix, there is a diagonal matrix S with nonnegative diagonal entries, with the same shape as A, and orthogonal matrices U and V, satisfying $A = USV$.
The rows of V are an orthonormal basis of right singular vectors, and the columns of U are an orthonormal basis of left singular vectors.

Exercise 3.10

Prove Theorems 3.16 and 3.17.

Example

Example 3.8

Find the SVD diagonalization of the Iris dataset.

```python
import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()
dataset = iris.data

# SVD
U, sigma, V = np.linalg.svd(dataset)
p = np.min(dataset.shape)
S = np.zeros(dataset.shape)
S[:p,:p] = np.diag(sigma)

np.allclose(dataset, np.dot(U, np.dot(S, V)))
```

Note

Exercise 3.11

Let A be any matrix and let $Q = A^t A$. $\mathbf{v}$ is an eigenvector of Q iff $\mathbf{v}$ is a right singular vector of A.

Example 3.9

Check the above exercise for the Iris dataset.

```
import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()
dataset = iris.data

# center dataset
m = np.mean(dataset, axis=0)
A = dataset - m

# rows of V are right singular vectors of A
V = np.linalg.svd(A)[2]

# any of these will work
Q = np.dot(A.T,A)
Q = np.cov(dataset.T, bias=False)
Q = np.cov(dataset.T, bias=True)

# columns of U are eigenvectors of Q
U = np.linalg.eigh(Q)[1]

# compare columns of U and rows of V
import sympy as sm
U = sm.Matrix(U)
V = sm.Matrix(V)
```

PCA

Let Q be the covariance matrix of a dataset in $\mathbb{R}^d$. Then Q is a $d \times d$ symmetric matrix, and the eigenvalue decomposition guarantees an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ in $\mathbb{R}^d$ consisting of eigenvectors of Q,

$$Q\mathbf{v}_k = \lambda_k \mathbf{v}_k, \quad k = 1, \ldots, d.$$

These eigenvectors are the principal components of the dataset. Principal Component Analysis (PCA) consists of projecting the dataset onto lower dimensional subspaces spanned by some of the eigenvectors.

Let A be the dataset matrix of a given dataset with N samples, and d features. If the samples are the rows of A, then A is $N \times d$. If we assume the dataset is centered, then, the covariance is $Q = A^t A/N$. Since multiplying Q by a scalar does not change the eigenvectors, the eigenvectors of the covariance Q equal the eigenvectors of $A^t A$. From Exercise 3.11, these are the same as the right singular vectors of A.

PCA

Corollary 3.1

The principal components of a dataset are the right singular vectors of the centered dataset matrix.

This shows there are two approaches to the principal components of a dataset:

1. either through EVD and eigenvectors of the covariance matrix,
2. or through SVD and right singular vectors of the centered dataset matrix.

Data Sets
Linear Geometry
Principal Components
Geometry of Matrices
Eigenvalue Decomposition
Singular Value Decomposition
Principal Component Analysis

MNIST Example I

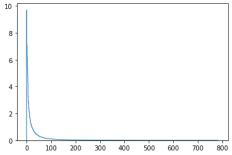
Example 3.10

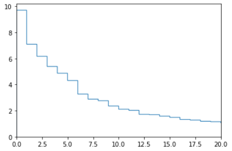
```python
from keras.datasets import mnist
import numpy as np

(train_X, train_y), (test_X, test_y) = mnist.load_data()

dataset = train_X.reshape(60000,784)
labels = train_y

# Covariance and total variance
Q = np.cov(dataset.T)
totvar = Q.trace()

# Eigendata
eigenvalues, eigenvectors = np.linalg.eig(Q)

# Percentage of eigenvalues in total varince
percent = eigenvalues*100/totvar

# cumulative sums
sums = np.cumsum(percent)

data = np.array([percent,sums])
data20 = data.T[:20].real.round(decimals=3)
for index in range(len(data20)):
  print(f'{index+1} ==> {data20[index][0]} ==> {data20[index][1]}')
```

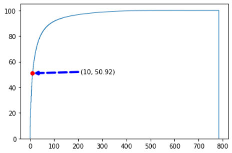
MNIST Example II

Example 3.10

```
# Plots
import matplotlib.pyplot as plt

plt.stairs(percent, range(len(eigenvalues)+1))

plt.stairs(percent, range(len(eigenvalues)+1))
plt.xlim(0,20)

plt.stairs(sums, range(len(eigenvalues)+1))
indices_above_50 = np.where(sums ¿ 50)[0][0]
plt.scatter(indices_above_50, sums[indices_above_50], color='red', label='Above 50',
                                        zorder=5)
text = f'({indices_above_50}, {sums[indices_above_50].real:.2f})'
plt.annotate(text, xy=(indices_above_50, sums[indices_above_50]), xytext=(
                                        indices_above_50+200, sums[indices_above_50
                                        ]),arrowprops=dict(arrowstyle='—>', color=
                                        'blue', lw=3.5, ls='——'))

# projection matrix onto top 11
# eigenvectors of covariance
# of dataset
order = eigenvalues.argsort()[::-1]
V = eigenvectors[:,order[:11]]
P = np.dot(V,V.T)
```







MNIST Example

Example 3.11

```
from keras.datasets import mnist
import numpy as np

(train_X, train_y), (test_X, test_y) = mnist.load_data()

dataset = train_X.reshape(60000,784)
dataset = dataset[:2000,:]
labels = train_y

# center dataset
m = np.mean(dataset,axis=0)
vectors = dataset - m

# rows of V are right singular vectors
V = np.linalg.svd(vectors)[2]

# no need to sort, already decreasing order
U = V[:11].T # top n rows as columns
P = np.dot(U,U.T)
```

MNIST Example I

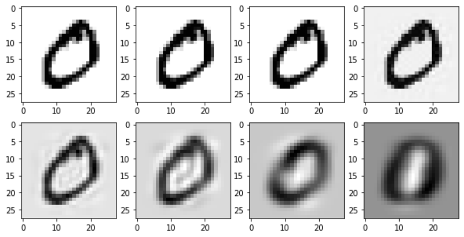
Example 3.12

```
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt

def pca_evd(dataset,n):
  Q = np.cov(dataset.T)
  eigenvalues, eigenvectors = np.linalg.eig(Q)
  order = eigenvalues.argsort()[::-1]
  V = eigenvectors[:,order[:n]]
  P = np.dot(V,V.T)
  return P

def pca_svd(dataset,n):
  dataset = dataset[:2000,:]
  m = np.mean(dataset,axis=0)
  vectors = dataset - m
  V = np.linalg.svd(vectors)[2]
  U = V[:n].T
  P = np.dot(U,U.T)
  return P
```

MNIST Example II

Example 3.12

```python
def plot_mnist(dataset, func_name):
    plt.figure(figsize=(10,5))
    # eight subplots
    rows, cols = 2, 4
    v = dataset[1] # second image
    plt.subplot(rows, cols, 1)
    plt.imshow(np.reshape(v,(28,28)),cmap="gray_r")

    for i,n in enumerate([784,600,350,150,50,10,1],start=2):
      P = func_name(dataset,n)
      projv = np.dot(P.real,v)
      A = np.reshape(projv,(28,28))
      plt.subplot(rows, cols, i)
      plt.imshow(A,cmap="gray_r")

(train_X, train_y), (test_X, test_y) = mnist.load_data()
dataset = train_X.reshape(60000,784)
plot_mnist(dataset, pca_evd)
plot_mnist(dataset, pca_svd)
```



MNIST Example

Example 3.13

```
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

(train_X, train_y), (test_X, test_y) = mnist.load_data()
dataset = train_X.reshape(60000,784)

N = len(dataset)
n = 10
engine = PCA(n_components = n)

reduced = engine.fit_transform(dataset)
reduced.shape

projected = engine.inverse_transform(reduced)
projected.shape

plt.figure(figsize=(10,5))
# eight subplots
rows, cols = 2, 4
v = dataset[1] # second image
plt.subplot(rows, cols, 1)
plt.imshow(np.reshape(v,(28,28)),cmap="gray_r")

for i,n in enumerate([784,600,350,150,50,10,1],start=2):
  engine = PCA(n_components = n)
  reduced = engine.fit_transform(dataset)
  projected = engine.inverse_transform(reduced)
  projv = projected[1]
  A = np.reshape(projv,(28,28))
  plt.subplot(rows, cols, i)
  plt.imshow(A,cmap="gray_r")
```