Geometric Foundations of Data Analysis I

Joshua Maglione

October 8, 2025

Contents

1 Introduction

Data analysis and more broadly Data Science is a vast and important field within Computer Science and Mathematics. At the core, the goal is to make sense of data, which can be measurements, survey results, behavior patterns, etc. Often this data comes to us in a very "high dimension". That is, there are so many variables that it is impossible to visualize, and even in low dimensions, it may not be clear what the best conclusion is based on the analysis.

A few references seem to agree that the *total data* on all computers is something like $\mathbf{10^{23}}$ **bytes** or about **100 zettabytes**. While all of this data is not concentrated in one organization, we still require highly sophisticated tools to make sense of a huge amount of data. My goal with this course is that you will have a solid foundation with some standard tools. From this bedrock one could explore more sophisticated methods of data analysis more easily.

We will consider four key topics:

1. Least Squares Fitting,
2. Principal Component Analysis,
3. k-Means and Hierarchical Clustering,
4. Nearest Neighbors and the Johnson–Lindenstrauss Theorem

We will be working with the assumption that **the data we care about is preserved by orthogonal and linear transformations**. This is not true with all data—for example, one should not take (proper) linear combinations of people. However, for data like grams of different kinds of food, this is completely plausible. This assumption will not always be necessary, but we will just keep this in mind.

Half of our time will be spent bringing these ideas to life and getting our hands dirty. We will be working with Jupyter Notebooks to build familiarity with the concepts we will discuss. This will be done using Python and standard data analysis packages like Pandas.

2 Least squares fitting

This method of analysis is both simple and powerful. Like with most things in mathematics, without some good guiding examples, we can get lost in the formulas.

2.1 Build up

Suppose we have the following data as seen in Figure 2.1. (Maybe a company produces parts once per month in lots that vary in size depending on demand. We write i for the production run, x_i for the lot size, and y_i for the hours of labor.)

i	x_i	y_i
1	30	73
2	20	50
3	60	128
4	80	170
5	40	87
6	50	108
7	60	135
8	30	69
9	70	148
10	60	132

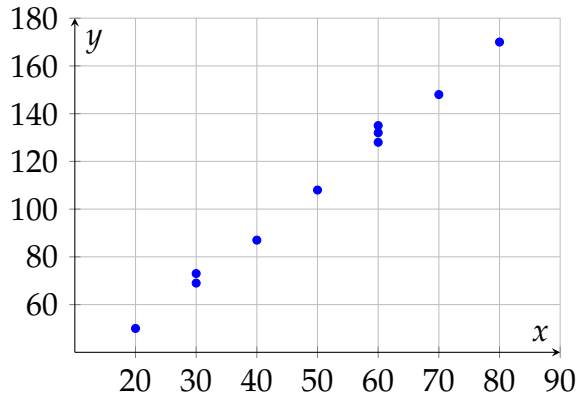


Figure 2.1: Data points exhibiting a linear relationship.

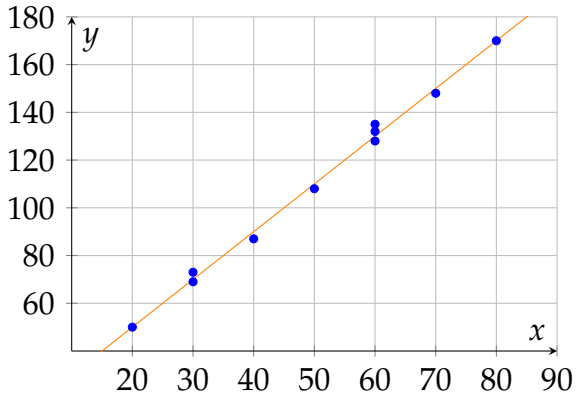


Figure 2.2: Line of best fit with data points.

It seems clear from the plot that the data fits a geometric pattern—there is a linear phenomenon. If we try to find the line that is somehow closest to all the data points, we might draw something like in Figure 2.2.

Although the line is not a perfect fit, it seems to tell us something about the relationship between the lots size and the amount of hours.

Questions:

- What makes this line "better" than alternatives?
- How are we quantifying "better"?
- Why are we using a line?

We will answer the first question later, probably. Let us consider the question of quantifying "better". Least squares fitting is all about minimizing the squares of differences between the line and the actual data points. We will make this precise very soon.

2.2 Line of best fit

We know that the equation of a non-vertical line has the form

$$y = b_0 + b_1 x.$$

If we had n data points of the form (x_i, y_i), we could choose b_0 and b_1 to minimize the following sum

$$S(b_0, b_1) = \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2 .$$

We can even solve for these values. Since S is a function in terms of b_0 and b_1, all possible minima occur when the partial derivatives of S are 0. In other words, the minima arise as values (b_0, b_1) such that

$$\begin{aligned} \frac{\partial S}{\partial b_0} &= -2 \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i)) = 0, \\ \frac{\partial S}{\partial b_1} &= -2 \sum_{i=1}^{n} (x_i y_i - x_i (b_0 + b_1 x_i)) = 0. \end{aligned}$$

These equations are linear equations, so we can solve for these with techniques from linear algebra. This mean, we need to solve two equations in the unknown b_0 and b_1:

$$\begin{aligned} n b_0 + b_1 \sum x_i &= \sum y_i, \\ b_0 \sum x_i + b_1 \sum x_i^2 &= \sum x_i y_i. \end{aligned} \tag{2.1}$$

Using the data from our example in Section 2.1, we have

$$\sum x_i = 500, \qquad \sum y_i = 1100, \qquad \sum x_i^2 = 28400, \qquad \sum x_i y_i = 61800.$$

Thus, the equations we need to solve are

$$\begin{aligned} 10 b_0 + 500 b_1 &= 1100, \\ 500 b_0 + 28400 b_1 &= 61800, \end{aligned}$$

which yield $b_0 = 10$ and $b_1 = 2$. Going back to the context of the initial problem: this solution tells us that by increasing the lot size by one, we expect to increase the labor hours by two.

2.2.1 Written as matrices

Let us write the equations in (2.1) with matrices. This might seem like overkill at this stage, but it will set us up nicely to generalize. Let

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}, \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \qquad B = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}. \tag{2.2}$$

Therefore, the equations in (2.1) are equivalent to the single matrix equation:

$$X^tXB = X^tY. \tag{2.3}$$

If X^tX is invertible, then $B = (X^tX)^{-1}X^tY$.

2.3 In class exercises pt. I

1. (a) With X, Y, and B as defined in Equation (2.2), show that

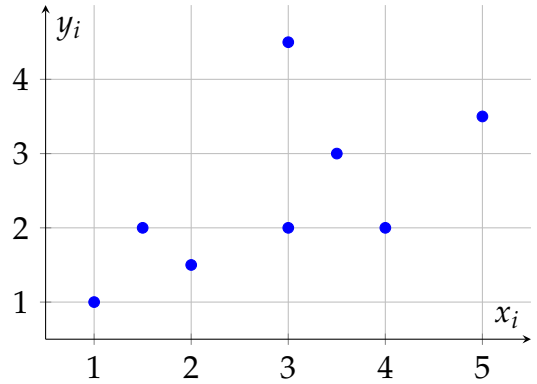
$$X^tX = \begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}, \qquad X^tY = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix}.$$

   (b) Show that $X^tXB = X^tY$ is equivalent to Equation (2.1):

$$\begin{aligned} nb_0 + b_1 \sum x_i &= \sum y_i, \\ b_0 \sum x_i + b_1 \sum x_i^2 &= \sum x_i y_i. \end{aligned}$$

2. Find a least squares fitting line to the following data and draw in the line:

i	x_i	y_i
1	1.0	1.0
2	2.0	1.5
3	3.0	2.0
4	1.5	2.0
5	3.5	3.0
6	3.0	4.5
7	4.0	2.0
8	5.0	3.5



(Round b_0 and b_1 to the nearest half integer.)

Week 1

2.4 Plane of best fit

We consider two independent variables and one dependent variable now. Consider the following data points as given in Figure 2.3.

We can put some meaning to these data. For example, suppose a company is selling a product, and we have 16 populations of people labeled 0 through 15. The values x_{i1} are the population sizes in 100s of people; the values x_{i2} are the average yearly income in €1000 per capita; and the values y_i are the number of sales of the product. (There might be dependencies between population size and average income, but our model treats them as independent.)

i	x_{i1}	x_{i2}	y_i
0	278	36	287
1	252	31	256
2	344	35	300
3	134	33	182
4	215	35	248
5	261	40	271
6	131	39	149
7	463	43	411
8	167	46	214
9	298	42	291
10	230	60	314
11	293	67	352
12	290	37	298
13	271	31	252
14	385	63	439
15	354	36	328

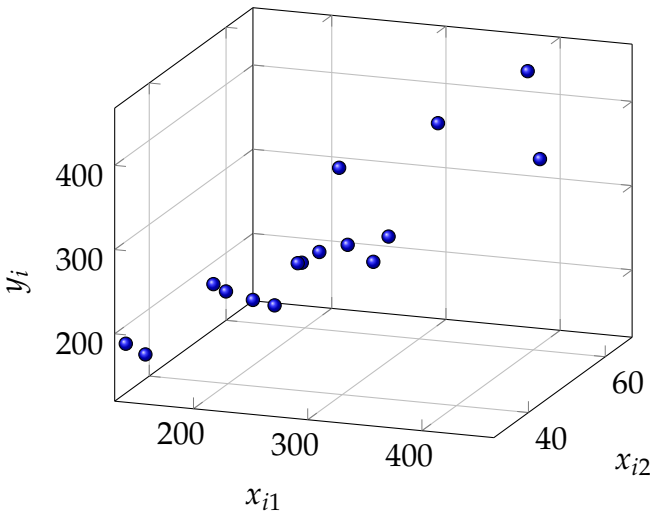


Figure 2.3: Data points in $\mathbb{R}^3$.

It looks like though there is a plane of best fit for the data—thanks to the suggestive viewing angle. Our goal is to find a plane, given by

$$y = b_0 + b_1 x_1 + b_2 x_2.$$

We can just do what we did last time. That is, for

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix}, \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \qquad B = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix},$$

we need to solve for B in the equation

$$X^{\mathrm{t}} X B = X^{\mathrm{t}} Y.$$

Thus, if $X^{\mathrm{t}}X$ is invertible, there is a unique B, which is equal to $(X^{\mathrm{t}}X)^{-1}X^{\mathrm{t}}Y$. For our example, we have

$$X^{\mathrm{t}}X = \begin{pmatrix} 16 & 4366 & 674 \\ 4366 & 1309480 & 187024 \\ 674 & 187024 & 30330 \end{pmatrix}, \qquad X^{\mathrm{t}}Y = \begin{pmatrix} 4592 \\ 1343400 \\ 200571 \end{pmatrix}.$$

Therefore, the plane of best fit is approximately

$$y = -11.3 + 0.7x_1 + 2.6x_2.$$

Putting all the data together we have a plane of best fit as seen in Figure 2.4.

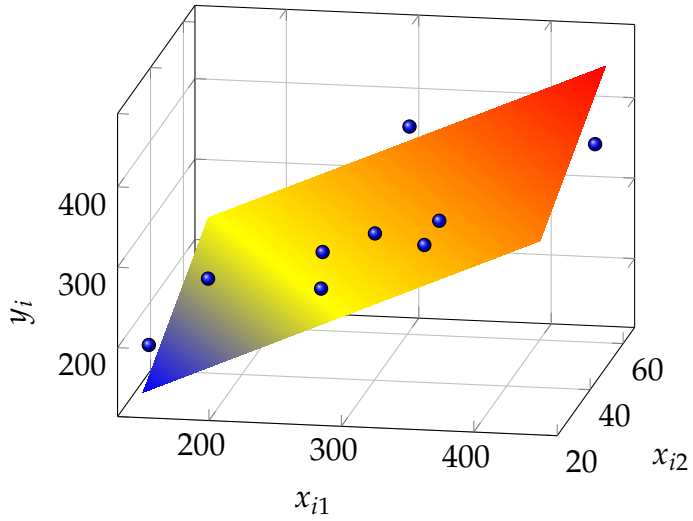


Figure 2.4: Data points together with the plane of best fit.

2.5 Hyperplane of best fit

Now we go to the general case. Suppose we have $p-1$ independent variables and 1 dependent variable, where $p \geqslant 2$. We assume we have n data points of the form

$$(x_{i1}, x_{i2}, \ldots, x_{i,p-1}, y_1) \in \mathbb{R}^p.$$

The least squares fitting for these data is a hyperplane of the form

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_{p-1} x_{p-1}.$$

To solve for the values b_i, we do as we did before. We define matrices

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1,p-1} \\ 1 & x_{21} & x_{22} & \cdots & x_{2,p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{n,p-1} \end{pmatrix}, \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \qquad B = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{pmatrix}.$$

As before, the values we want are given by the equation

$$X^{\mathrm{t}} X B = X^{\mathrm{t}} Y. \tag{2.4}$$

2.6 Why Equation (2.4) works

The heart of least squares is (Euclidean) distance. The distance between two points $x = (x_1, \ldots, x_p)$ and $y = (y_1, \ldots, y_p)$ in $\mathbb{R}^p$ is

$$d(x, y) = \|x - y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_p - y_p)^2}.$$

For a vector $v = (v_1, \ldots, v_p) \in \mathbb{R}^p$, the **length** of v is

$$\|v\| = d(0, v) = \sqrt{v_1^2 + v_2^2 + \cdots + v_p^2}.$$

Recall that the dot product of two (column) vectors u and v is

$$u \cdot v = u^t v = u_1 v_1 + u_2 v_2 + \cdots + u_p v_p.$$

Thus, the length of v is $\|v\| = \sqrt{v \cdot v}$; in other words $\|v\|^2 = v \cdot v$. In addition, if $u \cdot v = 0$, we say that u and v are **orthogonal** (or perpendicular).

The goal of least squares is to *minimize distance*; more specifically to minimize $\|Y - XB\|$. Note that the column vector Y has entries that are the *actual* y_i *values*, and the column vector

$$XB = \begin{pmatrix} B \cdot (1, x_{11}, x_{12}, \ldots, x_{1,p-1}) \\ B \cdot (1, x_{21}, x_{22}, \ldots, x_{2,p-1}) \\ \vdots \\ B \cdot (1, x_{n1}, x_{n2}, \ldots, x_{n,p-1}) \end{pmatrix} = \begin{pmatrix} b_0 + b_1 x_{11} + b_2 x_{12} + \cdots + b_{p-1} x_{1,p-1} \\ b_0 + b_1 x_{21} + b_2 x_{22} + \cdots + b_{p-1} x_{2,p-1} \\ \vdots \\ b_0 + b_1 x_{n1} + b_2 x_{n2} + \cdots + b_{p-1} x_{n,p-1} \end{pmatrix}$$

Therefore, $\|Y - XB\|$ is the square root of a sum of squares of the form

$$y_i - b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_{p-1} x_{i,p-1}.$$

Hence minimizing $\|Y - XB\|$ is the same as minimizing $\|Y - XB\|^2$, which is a sum of *squares*.

Proposition 2.1. *The minimal distance $\|Y - XB\|$ is achieved by solving for B in*

$$X^t XB = X^t Y.$$

Proof. Consider the subspace $U = \{Xu \mid u \in \mathbb{R}^p\}$ of $\mathbb{R}^n$, and observe that our desired solution XB is contained in U. Since $\|Y - XB\|$ is minimal, we must have that the vector $Y - XB$ is orthogonal to all vectors contained in U.[1] That is, $(Xu) \cdot (Y - XB) = 0$ for all $u \in \mathbb{R}^p$. In other words, we have for all $u \in \mathbb{R}^p$,

$$\begin{aligned} 0 = (Xu)^t (Y - XB) &= u^t X^t (Y - XB) \\ &= u^t \left(X^t Y - X^t XB \right). \end{aligned}$$

Because $u^t \left(X^t Y - X^t XB \right) = 0$ for all $u \in \mathbb{R}^p$, it follows that $X^t Y - X^t XB = 0$. □

2.7 In class exercises pt. II

1. Determine $X^t X$ and $X^t Y$ with

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1,p-1} \\ 1 & x_{21} & x_{22} & \cdots & x_{2,p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{n,p-1} \end{pmatrix}, \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

[1] To see why this is true, see Section 6.3.1 of [5], which is all about orthogonal decompositions.

2. Using (1) and by taking partial derivatives of

$$S(b_0, \ldots, b_{p-1}) = \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_{p-1} x_{i,p-1}))^2, \quad (2.5)$$

show that the hyperplane of best fit is obtained by solving $X^t X B = X^t Y$. (You could try this for $p = 3$ first.)

2.8 Nonlinear fittings

Although all of our examples so far have been linear fittings, we will demonstrate that least squares fittings works in the nonlinear case. What is important is that we have a candidate equation to fit. In the linear cases, we tried to fit

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_{p-1} x_{p-1}.$$

Suppose we have the following data as given in Figure 2.5. Instead of trying to fit the line $y = b_0 + b_1 x$, we could try to fit the parabola:

$$y = b_0 + b_1 x + b_2 x^2.$$

We can treat this the same way as before. Of course the quantities x and x^2 are *not* independent, but we can ignore this. Set

$$x_{i1} = x_i, \qquad x_{i2} = x_i^2.$$

Therefore, the hyperplane of best fit for the data (x_{i1}, x_{i2}, y_i) will give us the parabola of best fit. *Try this on your own!*

So one can fit any hypersurface $y = f(x_1, \ldots, x_{p-1})$ to the given data. The function f in this case is called the **regression function**. This general method of analysis is known as **regression analysis**. A few questions arise:

- Which surface is "best"?
- How can we quantify "best"?
- Even in the line case $(p = 2)$, how can we quantify how well data fits our line?

Week 3

x_i	y_i
2.27	2.50
5.06	-16.13
1.45	4.23
5.89	-22.46
0.48	1.37
-0.22	0.86
1.44	11.85
-1.77	-14.71
2.45	9.42
-1.54	-14.07
7.55	-55.62
1.76	4.45
5.16	-19.56
3.26	-2.79
3.23	5.20
0.85	8.09

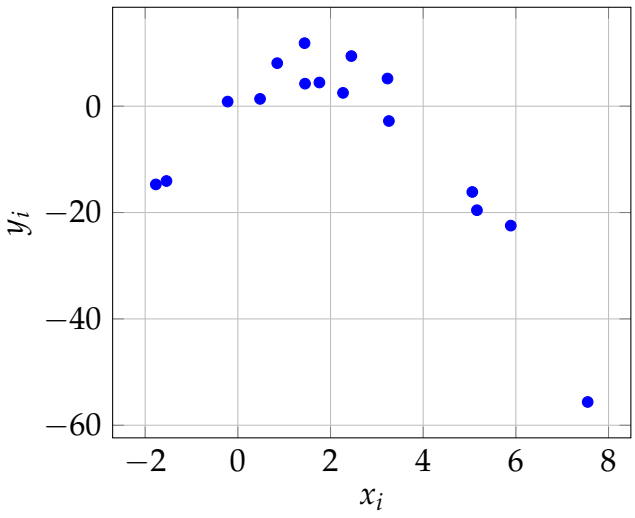


Figure 2.5: Data points demonstrating a nonlinear relationship.

2.9 Coefficient of determination (R^2 values)

We are going to make precise how well our hyperplane fits our data. Recall that hyperplanes can be replaced by hypersurfaces; see Section 2.8. First we establish some notation. Suppose we have n data points $(x_{i1}, x_{i2}, \ldots, x_{i,p-1}, y_i) \in \mathbb{R}^p$. Then we define

$$\begin{aligned} \text{(Fitted value)} \qquad & \widehat{y}_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_{p-1} x_{i,p-1}, \\ \text{(Residual)} \qquad & e_i = y_i - \widehat{y}_i, \\ \text{(Sample mean)} \qquad & \overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i. \end{aligned}$$

These yield vectors in $\mathbb{R}^n$ as follows

$$\widehat{Y} = \begin{pmatrix} \widehat{y}_1 \\ \vdots \\ \widehat{y}_n \end{pmatrix} = XB, \qquad E = \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} = Y - \widehat{Y}, \qquad \overline{Y} = \begin{pmatrix} \overline{y} \\ \vdots \\ \overline{y} \end{pmatrix} = \overline{y} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

From our n data points, we have three points in $\mathbb{R}^n$ given by Y, $\widehat{Y}$, and $\overline{Y}$. Three points always lie on a plane, so the three point determine a triangle on such a plane. What does this triangle look like? If it is a triangle (and not a line or a single point), then the next lemma proves it must be a right triangle.

Lemma 2.2. *The vectors $E = Y - \widehat{Y}$ and $\widehat{Y} - \overline{Y}$ are orthogonal.*

Proof. Suppose $X^t X B = X^t Y$. We need to prove two equations. For the first,

$$0 = X^t(Y - XB) = X^t(Y - \widehat{Y}) = X^t E.$$

Hence, $X^{\mathrm{t}}E = 0$. For the second,

$$\begin{aligned}\overline{Y}^{\mathrm{t}}E &= \overline{y}\sum_{i=1}^{n}(y_i - \widehat{y}_i)\\ &= \overline{y}\sum_{i=1}^{n}(y_i - (b_0 + b_1x_{i1} + b_2x_{i2} + \cdots + b_{p-1}x_{i,p-1}))\\ &= -\frac{\overline{y}}{2}\cdot\frac{\partial S}{\partial b_0} = 0,\end{aligned}$$

where S is defined in Equation (2.5), so $\overline{Y}^{\mathrm{t}}E = 0$. Thus, we have

$$(\widehat{Y} - \overline{Y})\cdot E = (XB)^{\mathrm{t}}E - \overline{Y}^{\mathrm{t}}E = 0. \qquad \square$$

Remark 2.3. One can simplify the proof for Theorem 2.2 by applying an isometry to the data, so that $\overline{y} = 0$. That is, one only needs to prove that E and $\widehat{Y}$ are orthogonal.

The lengths of the differences of the vectors are important and have names:

$$\begin{aligned}\text{(Sums of Squares Total – SST)} &: \|Y - \overline{Y}\|^2,\\ \text{(Sums of Squares Error – SSE)} &: \|Y - \widehat{Y}\|^2,\\ \text{(Sums of Squares Regression – SSR)} &: \|\widehat{Y} - \overline{Y}\|^2.\end{aligned}$$

The value SST measures the *total variability* of the data set. For example, $\sqrt{SST} = \|Y - \overline{Y}\|$ is the distance from the actual data Y to the sample mean Y. Using the same ideas, we can see that SSE measures the error of our regression and that SSR measures the distance from our regression to the sample mean.

Proposition 2.4.

$$SST = SSE + SSR.$$

Proof. Apply Theorem 2.2 and the Pythagorean Theorem:

$$\|Y - \overline{Y}\|^2 = \|Y - \widehat{Y}\|^2 + \|\widehat{Y} - \overline{Y}\|^2. \qquad \square$$

Now we can describe a quantity that measures how good our regression fits the given data.

Definition 2.5. The **coefficient of determination** (also known as the R^2**-value**) is

$$R^2 = \frac{SSR}{SST} = \frac{\|\widehat{Y} - \overline{Y}\|^2}{\|Y - \overline{Y}\|^2}.$$

Proposition 2.6. $0 \leqslant R^2 \leqslant 1$.

Proof. Since each SST and SSR are squares, they are nonnegative. By Theorem 2.4, we have $0 \leqslant SSR \leqslant SST$. $\square$

2.9.1 What do the extremes means?

The one case where R^2 is meaningless is when $SST = 0$. This implies both $SSR = SSE = 0$. Moreover, $Y = \overline{Y} = \overline{y}\mathbb{1}$, where $\mathbb{1}$ is the all ones column vector. Hence, every data point y_i is the same and, therefore, equal to the mean. Let's never return to this case.

We can have $SSR = 0$, which is equivalent to $R^2 = 0$. This implies that $\|\widehat{Y} - \overline{Y}\|^2 = 0$, so that $\widehat{Y} = \overline{Y}$. In other words, our prediction $\widehat{y}_i$ is just simply the mean. This means we have not found any relationship between the independent variables and the dependent variables.

In the other extreme we have $SSR = SST$, which is equivalent to $R^2 = 1$. This implies that $Y = \widehat{Y}$, so the given data lies (exactly) on the surface given by $y = f(x_1, \ldots, x_{p-1})$. That is, the regression function exactly predicts the data.

To summarize, when $R^2 = 0$, we cannot deduce any relationship between the independent and dependent variables, and when $R^2 = 1$, we understand completely the relationship between the independent and dependent variables. Very roughly speaking, the R^2 can be thought of as the ratio of how well the regression fits the data.

2.10 In class exercises pt. III

1. Prove the following.
   (a) $\|\overline{Y}\|^2 = n\overline{y}$.
   (b) $Y \cdot \overline{Y} = \widehat{Y} \cdot \overline{Y} = \|\overline{Y}\|^2$.
   (c) $Y \cdot \widehat{Y} = \|\widehat{Y}\|^2$.
2. Use (1) to show that
   (a) $SST = \|Y\|^2 - \|\overline{Y}\|^2$,
   (b) $SSE = \|Y\|^2 - \|\widehat{Y}\|^2$,
   (c) $SSR = \|\widehat{Y}\|^2 - \|\overline{Y}\|^2$.
3. What are the R^2 values for the examples above?

3 Principal component analysis

Principal component analysis (PCA) is a power method of analysis that comes standard in all data science tool kits. With little effort, one can reduce a complex data set to data that we can more easily see structure. More specifically, the goal of PCA is to find the "best" basis to express the data. In other words, our initial reference frame may not be the one that best expresses the structure of our data—PCA is a method to find the "best" reference frame.

3.1 Introducing PCA

We will start with a toy example, where the analysis is quite simple. Suppose we have many many data points in $\mathbb{R}^2$ as seen in Figure 3.1.

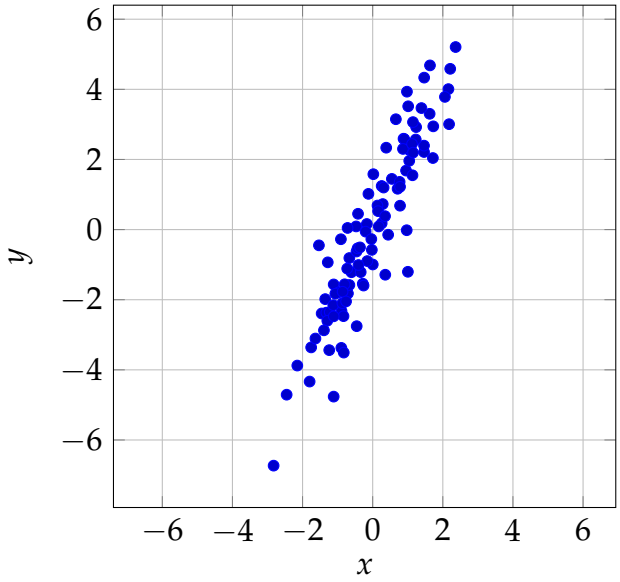


Figure 3.1: Some data in $\mathbb{R}^2$.

Let's assume that our data points live in $\mathbb{R}^m$, so that $m = 2$ in Figure 3.1. Suppose we write those data points in an $m \times n$ matrix X. Our current (and default) basis is $\{(1,0,\ldots,0),(0,1,0,\ldots,0),\ldots,(0,\ldots,0,1)\}$, and we want a basis $\{p_1, p_2, \ldots, p_m\}$ that better reflects the structure of our data. That is, we want an $m \times m$ matrix P, whose rows are the p_i, that provides us a better reference frame. Therefore, we want to transform our data X into a new data set Y such that

$$PX = Y.$$

The columns of X are the "old" data, and the columns of Y are the "new" data. If the p_i are row vectors and the x_i column vectors, then we want

$$PX = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \cdots x_n \end{pmatrix} = \begin{pmatrix} p_1 \cdot x_1 & p_1 \cdot x_2 & \cdots & p_1 \cdot x_n \\ p_2 \cdot x_1 & p_2 \cdot x_2 & \cdots & p_2 \cdot x_n \\ \vdots & \vdots & \ddots & \vdots \\ p_m \cdot x_1 & p_m \cdot x_2 & \cdots & p_m \cdot x_n \end{pmatrix}.$$

Thus, if the columns of Y are written y_i, we have

$$y_i = \begin{pmatrix} p_1 \cdot x_i \\ p_2 \cdot x_i \\ \vdots \\ p_m \cdot x_i \end{pmatrix}. \tag{3.1}$$

Back to our example from Figure 3.1. We want data with a high *signal-to-noise* (SNR) ratio as to minimize noise. Assuming our data in Figure 3.1 was

collected reasonably well, the direction of largest variance is the direction of most interesting dynamics. Therefore, the variance of signal, σ_s^2, would correspond to the length of the orange vector in fig. 3.2 pointing to the top right, and the variance of the noise, σ_n^2, would correspond to the length of the orange vector pointing to the top left.

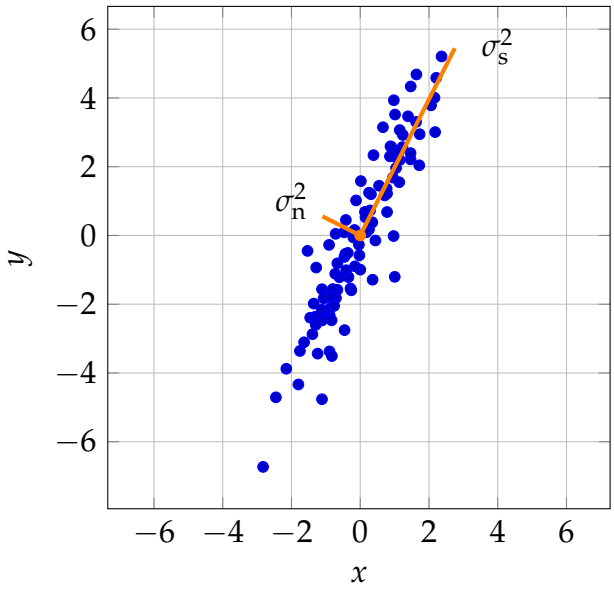


Figure 3.2: Signal and noise variances represented graphically.

Note also that in Figure 3.2 knowing the x value gives one a good approximation for the y value and vice versa. In this case, we might say that the data has a moderate amount of redundancy, whereas if the data points had a much higher R^2 value to its line of best fit, we would say the data have high redundancy. (And if the data had a much lower R^2 value, we would say the data have low redundancy.) One of the aims of PCA is to lower redundancy. For the 2-dimensional case, this is simple—take the line of best fit, but for arbitrarily higher dimensions, this is not obvious.

Week 4

Suppose we have n measurements of the same kind (like length)

$$U = \{u_1, u_2, \ldots, u_n\} \qquad \text{and} \qquad V = \{v_1, v_2, \ldots, v_n\}$$

with mean equal to 0. The *variances* are equal to

$$\sigma_U^2 = \frac{1}{n}\sum_{i=1}^{n} u_i^2, \qquad\qquad \sigma_V^2 = \frac{1}{n}\sum_{i=1}^{n} v_i^2.$$

The *covariance* between the data sets U and V is

$$\sigma_{UV}^2 = \frac{1}{n}\sum_{i=1}^{n} u_i v_i.$$

The covariance measures the degree of the linear relationship between the two variables. Thus, a large positive value would imply that the data are positively correlated, and a large negative value would imply negatively correlated. And $\sigma_{UV}^2 = 0$ if and only if the data U and V are uncorrelated. Moreover the absolute magnitude of the covariance measures the degree of redundancy.

If instead we wrote $u = (u_1, u_2, \ldots, u_n)$ and $v = (v_1, v_2, \ldots, v_n)$ as row vectors, then

$$\sigma_{uv}^2 = \frac{1}{n} u v^{\mathrm{t}}. \tag{3.2}$$

We generalize from two vectors to m vectors. Write

$$X = \begin{pmatrix} x_1' \\ x_2' \\ \vdots \\ x_m' \end{pmatrix}$$

where x_i' is a row vector. Note that the rows of X correspond to measurements of a particular type, and the columns of X correspond to all measurements of a particular trial. In other words, our data points are the columns of X, and each entry of a data point is a measurement of a particular type.

Using Equation (3.2), we define the **covariance matrix** of X to be

$$C_X = \frac{1}{n} X X^{\mathrm{t}}.$$

Lemma 3.1.

1. *The matrix C_X is symmetric. That is, $C_X = C_X^{\mathrm{t}}$.*
2. *The diagonal entries of C_X are variances.*
3. *The off-diagonal entries of C_X are covariances.*

Recall our goal is to find a new (and better) basis $\{p_1, p_2, \ldots, p_m\}$. Namely, we want an invertible matrix P to turn our data X into a data set Y where we can better understand the structure. If we could do this on the level of covariance matrices, then we could pick an ideal covariance matrix, one where the diagonal entries are large in absolute magnitude and where the off-diagonal entries are small in absolute magnitude. (Variances being high in magnitude suggest interesting dynamics, and covariances low in magnitude suggest low redundancy.) In other words, we would like to find a matrix P such that

$$C_Y = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_m \end{pmatrix}, \tag{3.3}$$

where $Y = PX$. Typically we have $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$.

The main work of PCA is finding such a matrix P. We will describe how to construct such a P later, but let us return to our running example.

The first principal component is in the direction of the largest variance, and the second principal component is in the orthogonal direction. So in $\mathbb{R}^2$, this is quite simple. Although we have not given all the data point explicitly, the covariance matrix is

$$C_X = \begin{pmatrix} 1.27 & 2.52 \\ 2.52 & 5.95 \end{pmatrix}.$$

The slope of the line in the direction of the highest variance is 1.98, and it passes through the point $(0, 0)$. By rotating and permuting, we get

$$P = \begin{pmatrix} 0.40 & 0.92 \\ 0.92 & -0.40 \end{pmatrix},$$

and the graph of the data $Y = PX$ is seen in Figure 3.3.

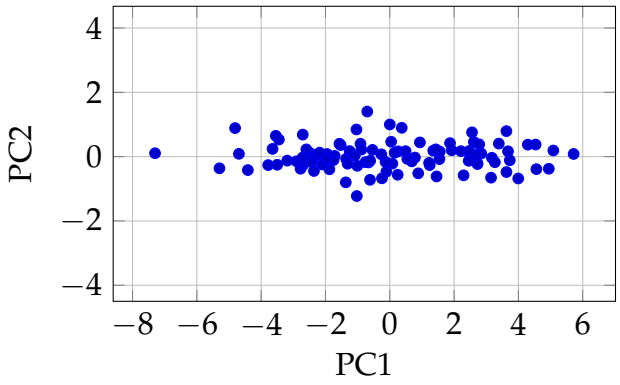


Figure 3.3: A new basis for our data.

Remark 3.2. There is a whole art of scaling data in a pre-processing stage that we will not explore in this course. Basically, if one variable ranges between ± 10 and another $\pm 10^3$, the second variable will bias the process simply by its scale. There are many different methods to rescale the data as not to lose (too much) information. *Throughout we will assume our data has roughly the same scale and not worry about rescaling*, but in practice this is an important issue.

3.1.1 Assumptions of PCA

Before we close this introduction to PCA, let us come back to some of the assumptions we have made along the way. There are three key assumptions we have made in introducing PCA. We will not focus much on these, but users of PCA should know that assumptions have been made. These statements may not necessarily hold for a particular data set.

1. *Linearity.*

   This allows us to reframe the problem as a change of basis problem.

2. *Large variance is important (and SNR > 1).*

   This can be a very strong assumption and really needs to take into account how the data was collected.

3. *Principal components are orthogonal.*

   This is not always the case, but orthogonality allows us to use linear algebra.

3.2 In class exercises pt. IV

1. (a) Show that C_X is symmetric.
   (b) Prove that the diagonal entries of C_X are variances and the off-diagonal entries are covariances.

2. Suppose X is an $m \times n$ matrix with sample mean $\overline{x} \in \mathbb{R}^m$. Let X' be the shifted data of X, so that its sample mean is 0. What is $C_{X'}$ in terms of X and $\overline{x}$?

3.3 Performing PCA

At the heart of performing PCA in practice is the following question. Recall that C_Y is a diagonal matrix; see the discussion around Equation (3.3).

Question 3.3. What is the relationship between C_X and C_Y if $Y = PX$?

Proof. From above, we have

$$C_Y = \frac{1}{n} YY^{\mathsf{t}} = \frac{1}{n}(PX)(PX)^{\mathsf{t}} = PC_X P^{\mathsf{t}}. \qquad \square$$

In order to perform a PCA, we need to find a matrix P such that $PC_X P^{\mathsf{t}}$ is diagonal. Importantly, we do not want to change the variances, so we want P to be *distance preserving*; that is, we want P to satisfy

$$\|Pv\| = \|v\| \tag{3.4}$$

for all vectors $v \in \mathbb{R}^m$. We say such a matrix P is an **isometry**. We can take Equation (3.4) and massage it, so that P must satisfy

$$v^{\mathsf{t}} P^{\mathsf{t}} Pv = (Pv) \cdot (Pv) = v \cdot v = v^{\mathsf{t}} v$$

for all $v \in \mathbb{R}^m$. Since this needs to hold for all vectors, it must hold for all pairs of basis vectors (e_i, e_j) for all $i, j \in \{1, \ldots, m\}$. Thus, distance preserving is equivalent to $P^{\mathsf{t}} P = I_m$, but these matrices are called **orthogonal**. (Note: pairs of distinct columns of such a matrix P are pairwise orthogonal. Can you prove this?!)

Week 5

Let us bring this back to the equation we established. We want P to be an orthogonal matrix such that

$$C_Y = PC_XP^{\mathrm{t}}. \tag{3.5}$$

Moreover, we want C_Y to be a diagonal matrix. Since $P^{\mathrm{t}}P = I_m$, it follows that $P^{-1} = P^{\mathrm{t}}$, so using this identity we have

$$C_Y = PC_XP^{-1}.$$

Since C_Y is diagonal, this is accomplished through *eigendecomposition*. Therefore, the rows of P are eigenvectors, and the diagonal entries of C_Y are eigenvalues.

All the entries of C_Y are *real*, and we know that some matrices have complex eigenvalues. For example, the matrix

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

has eigenvalues i and $-i$, for $i = \sqrt{-1}$.

Question 3.4. Is it always possible to find a real matrix P such that Equation (3.5) holds?

The answer to Theorem 3.4 is "Yes," and we will prove it later. For now, let us assume it is always possible.

We give a recipe for cooking up the principal components.

Given: n data points $x_i \in \mathbb{R}^m$ (of roughly the same scale),

Return: m principal components.

1. Compute the mean of each coordinate: $\mu_j = \sum_i x_{ij}$,
2. Organize the normalised data into a matrix $X = (x_{ij} - \mu_j)$,
3. Compute the covariance matrix C_X of X,
4. Compute the eigenvectors of C_X, and sort them based on their eigenvalues: largest is first and smallest is last.
5. Return the (ordered) orthonormal basis of eigenvectors.

PCA is a phrase used for this algorithm together with its analysis, and one of the main ways PCA is performed is by taking only the first k principal components (rather than all m). Here, k is usually determined by the *eigenvalues*.

Both C_X and C_Y have the same eigenvalues: $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. Since the trace of matrix is the sum of its eigenvalues, both C_X and C_Y have the same trace. In other words,

$$\operatorname{tr}(C_X) = \sum_{i=1}^{m} \lambda_i,$$

and by Theorem 3.1, the sum of the variances is the sum of the eigenvalues. Because we view variability as an important measurement to keep track of, we can choose a k that both maximizes the amount of variability "seen" and while minimizing the value of k. This is a bit more of an art than a science, but one general rule could be to choose the smallest k such that

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geqslant 0.95. \tag{3.6}$$

For such a k, one might say that the first k principal components capture 95% of the total variability.

Question 3.5. Are the principal components $k+1$ through m useless?

3.4 Projections

The answer to Theorem 3.5 is essentially "Yes", and it can be helpful to consider an idealised example.

Recall that X is our $m \times n$ matrix whose columns are our n data points in $\mathbb{R}^m$. The matrix P is obtained from the eigenvectors of C_X, which are the rows of P, and $Y = PX$. Moreover, P is an orthogonal matrix. Suppose $k < m$ and

$$\lambda_{k+1} = \lambda_{k+2} = \cdots = \lambda_m = 0.$$

Therefore, we have

$$C_Y = \begin{pmatrix} \lambda_1 & & & & & \\ & \ddots & & & & \\ & & \lambda_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}.$$

Remark 3.6. In this situation, $y_{rj} = 0$ for all $k+1 \leqslant r \leqslant m$ and all $1 \leqslant j \leqslant n$. (Can you show this?!) In other words, the "new" data point y_j has a tail of zeroes.

Lemma 3.7. *Let Q be the first k rows of P, so that Q is $k \times m$. Then for all columns $x_i, x_j \in \mathbb{R}^m$ of X,*

$$d_{\mathbb{R}^m}(x_i, x_j) = d_{\mathbb{R}^k}(Qx_i, Qx_j).$$

Proof. For each column x_i of X, we have $Px_i = y_i$, where y_i is the ith column of Y. Since P is orthogonal, $d_{\mathbb{R}^m}(Px_i, Px_j) = d_{\mathbb{R}^m}(x_i, x_j)$. Thus, by Theorem 3.6

$$\begin{aligned} d_{\mathbb{R}^m}(Px_i, Px_j)^2 &= d_{\mathbb{R}^m}(y_i, y_j)^2 \\ &= \sum_{r=1}^{m} (y_{ri} - y_{rj})^2 \\ &= \sum_{r=1}^{k} (y_{ri} - y_{rj})^2 \\ &= d_{\mathbb{R}^k}(Qx_i, Qx_j)^2. \end{aligned}$$

□

The key application of Theorem 3.7 is that ignoring rows $k+1$ through m in the matrix P does not change the geometry! That is, we can project the data to a smaller dimension and distances between the data points remain unchanged!

In practice the λ_i are strictly greater than 0, so this idealised situation does not occur. Using our rule in Equation (3.6), then the projected data would approximate the original geometry quite well. The larger the ratio, the better the approximation, so there is indeed a trade off. Thus, after constructing all of the principal components, one can take the first k, and project the original data (i.e. the matrix X) into a smaller dimension by constructing the matrix Q from the first k rows of P.

3.5 PCA is always possible—the Spectral Theorem

The real power of PCA is that we can *always* perform it. One does not need to input parameters; just the data. Note there is a pre-processing stage of normalising and rescaling, but this can be applied to all data. We address Theorem 3.4 which, at the time, we just assumed was true. In order to do this, we prove the Spectral Theorem.

Theorem 3.8 (Spectral Theorem)**.** *Let M be a real symmetric $n \times n$ matrix. Then $\mathbb{R}^n$ has an orthonormal basis coming from eigenvectors of M.*

Corollary 3.9. *Every eigenvalue and eigenvector of a real symmetric matrix are real.*

In particular, the Spectral Theorem makes principal component analysis possible since the covariance matrix is always a real symmetric matrix.

Remark 3.10. Covariance matrices are examples of *positive semi-definite matrices*, which are real matrices whose eigenvalues are all real and nonnegative. We will not need this much, nor will we prove this, but it might be useful to know that covariance matrices are particularly nice.

Let's unpack what the Spectral Theorem says exactly. Recall that an orthonormal basis $\{b_1, \ldots, b_n\}$ for a vector space V satisfies

$$b_i \cdot b_j = \begin{cases} 1 & i = j, \\ 0 & i \neq j. \end{cases}$$

There are two components to being an orthonormal basis: the basis vectors are pairwise orthogonal and each basis vector has unit length. The latter condition is not so particularly important (though useful); really, the magic is in the pairwise orthogonal condition.

We already saw in Section 3.3 that the Spectral Theorem does not hold if we drop 'symmetric'. Now we will build our way to prove the Spectral Theorem.

Lemma 3.11. *Let M be a real symmetric matrix. Then eigenvectors corresponding to distinct eigenvalues are orthogonal.*

Proof. Let u and v be eigenvectors of M corresponding to λ and μ with $\lambda \neq \mu$. Then

$$\mu u^t v = u^t(Mv) = (u^t M)v = (Mu)^t v = \lambda u^t v.$$

Suppose via contradiction that $u^t v \neq 0$, but then $\mu = \lambda$, which is a contradiction, so we must have $u^t v = 0$. Hence, u and v are orthogonal. □

From Theorem 3.11, we can already prove that all eigenvalues (and therefore) all eigenvectors are real. Before we dive into that proof, recall the operation of *complex conjugation*. If $a, b \in \mathbb{R}$, then $a + bi$ is a complex number, and the complex conjugate is

$$\overline{a + bi} = a - bi.$$

If $z \in \mathbb{C}$, then $\overline{z} = z$ if and only if $z \in \mathbb{R}$.

Proposition 3.12. *Let M be a real symmetric matrix. Then all eigenvalues of M are real.*

Proof. Suppose $Mv = \lambda v$, and assume, via contradiction, that $\lambda \in \mathbb{C} \setminus \mathbb{R}$. If we apply complex conjugation to $Mv = \lambda v$, we have $\overline{M}\overline{v} = \overline{\lambda}\overline{v}$. Since M is a real matrix, $\overline{M} = M$. Thus, we have a new eigenvector and eigenvalue of M since

$$M\overline{v} = \overline{\lambda}\overline{v}.$$

The dot product of v and $\overline{v}$ is a sum of squares: suppose M is written with respect to some basis $\{b_1, \ldots, b_m\}$; then

$$v \cdot \overline{v} = \sum_{i=1}^{m} v_i \overline{v}_i > 0.$$

Thus, v and $\overline{v}$ are not orthogonal, but $\lambda \neq \overline{\lambda}$, so we get a contradiction thanks to Theorem 3.11. Therefore, $\lambda \in \mathbb{R}$, and hence all eigenvalues of M are real. □

It follows from Theorem 3.12 that all eigenvectors of a real symmetric matrix are real, but we still want more—namely orthogonality.

Week 6

Definition 3.13. Let M be an $n \times n$ matrix with entries in a field K. A subspace U of K^n is **M-invariant** if $Mu \in U$ for all $u \in U$.

Definition 3.14. Let U be a subspace of a vector space V. Define $U^\perp$, sometimes called the **perp-space** (or orthogonal space) of U, to be the set of all $v \in V$ such that $v \cdot u = 0$ for all $u \in U$.

One thing to try on your own: Show that $U^\perp$ is a subspace of V if U is a subspace of V.

Lemma 3.15. *Let M be an $n \times n$ real symmetric matrix. If U is an M-invariant subspace of $\mathbb{R}^n$, then $U^\perp$ is M-invariant.*

Proof. Exercise. □

Lemma 3.16. *Let M be an $n \times n$ real symmetric matrix. If U is a nonzero M-invariant subspace of $\mathbb{R}^n$, then U contains an eigenvector of M.*

Proof. Let Q be a matrix whose columns define an orthonormal basis for U, and assume U is m-dimensional. Thus, $Q^tQ = I_m$. Since U is M-invariant, each basis vector is mapped to some linear combination of all of the basis vectors. Therefore, there exists an $m \times m$ real matrix N such that

$$MQ = QN.$$

Since $QQ^t = I_m$, we have $Q^tMQ = N$. Because M is symmetric, so is N. By Theorem 3.12, all the eigenvalues of N are real, and since $m \geqslant 1$, let $0 \neq v \in \mathbb{R}^m$ such that $Nv = \lambda v$ for some $\lambda \in \mathbb{R}$. Hence,

$$MQv = QNv = \lambda Qv.$$

Since the columns of Q are linearly independent, $Qv \neq 0$ and is, therefore, an eigenvector of M. But $Qv \in U$, so U contains an eigenvector of M. □

Proof of the Spectral Theorem. We will prove this by induction on n. For $n = 1$, there is nothing to prove, so we assume that the statement of the theorem holds for n. We will show that it holds for $n + 1$.

Let v be an eigenvector of M, which is real by Theorem 3.12. Let U be the subspace of $\mathbb{R}^{n+1}$ that is orthogonal to v, so $\dim(U) = n$. Since the span of v is M-invariant, so is U by Theorem 3.15. By Theorem 3.16, U contains a real eigenvector of M. By iterating this argument, U contains all other eigenvectors of M. Hence, by induction, the statement holds. □

3.6 Random Projections and Johnson–Lindenstrauss

We now cover a topic somewhat related to **Principal Component Analysis (PCA)** but also very different. The main question we are concerned with is whether we can accomplish dimension reduction without PCA?

Suppose we have n data points in $\mathbb{R}^m$ and we want to project it to $\mathbb{R}^k$ with k much smaller than m while preserving the geometry as much as we can. In other words, we want a map $\varphi : \mathbb{R}^m \to \mathbb{R}^k$ such that all distances are preserved. In symbols, we want, for a small $\varepsilon > 0$ and a subset $S \subseteq \mathbb{R}^m$, for all $u, v \in S$:

$$(1 - \varepsilon)\|u - v\|^2 \leqslant \|\varphi(u) - \varphi(v)\|^2 \leqslant (1 + \varepsilon)\|u - v\|^2 \tag{3.7}$$

So that ε gives us an **approximation threshold**.

This is different from what happens with PCA. In that case, φ is given by matrix multiplication: using only the first k principal components. There, the **average error** is small, so that some distances can have large errors, and most would be small. In our context right now, φ is much more controlled.

The idea to create $\varphi : \mathbb{R}^m \to \mathbb{R}^k$ so that (3.7) is satisfied is simple: project onto a random k-dimensional subspace. See Algorithm 1.

The following theorem states the probability of this approach yielding the desired outcome.

Algorithm 1 Random Projection

Input: $X \in \mathrm{Mat}_{m\times n}(\mathbb{R})$ and $k \in \mathbb{N}$.
Output: $Y \in \mathrm{Mat}_{k\times n}(\mathbb{R})$.
  for $i \in \{1, \ldots, k\}$ **do**
    Choose random vector v_i from a **Gaussian distribution**.
    Rescale v_i: $v_i = \sqrt{\frac{m}{\|v_i\|^2}} v_i$ # Useful for proof.
  end for
  for each column of X, x_i **do**
    $y_i = (x_i \cdot v_1, x_i \cdot v_2, \ldots, x_i \cdot v_k)^T$
  end for
  $Y = [y_1, y_2, \ldots, y_n]$
  return Y.

Theorem 3.17 (Johnson–Lindenstrauss (1984))**.** *There exists $\varphi : \mathbb{R}^m \to \mathbb{R}^k$ satisfying the inequalities in (3.7) with probability at least $1 - \frac{1}{n}$ as long as*

$$k \geqslant \frac{8 \ln(n)}{\varepsilon^2}$$

where φ is of the form

$$\varphi(x) = \frac{1}{\sqrt{k}} Ax$$

*where A is a matrix with independent and identically distributed **Gaussian entries** with zero mean and unit variance.*

We will not prove Theorem 3.17.

Remark 3.18. Although this is called the JL Lemma (or Theorem), this formulation benefits from recent research on these problems. One can get better bounds for k; see Frankl–Maehara [2]. The conditions on the independent and identically distributed Gaussian entries is an improvement due to Har-Peled–Indyk–Motwani; see [4].

Now that we have two ways to perform dimension reduction, when should we use one over the other?

- Generally, PCA is the safest. It preserves largest distances, but may not preserve the smaller ones. This is essentially down to PCA caring about **high variance** and disregarding **low variance**. However, PCA might be computationally expensive.
- If you want more control over the distances (i.e. want them all essentially the same) or if you need to optimize time or memory, then a random projection would be better, provided the hypotheses of Theorem 3.17 hold.

It is possible to use both with good success; see [7].

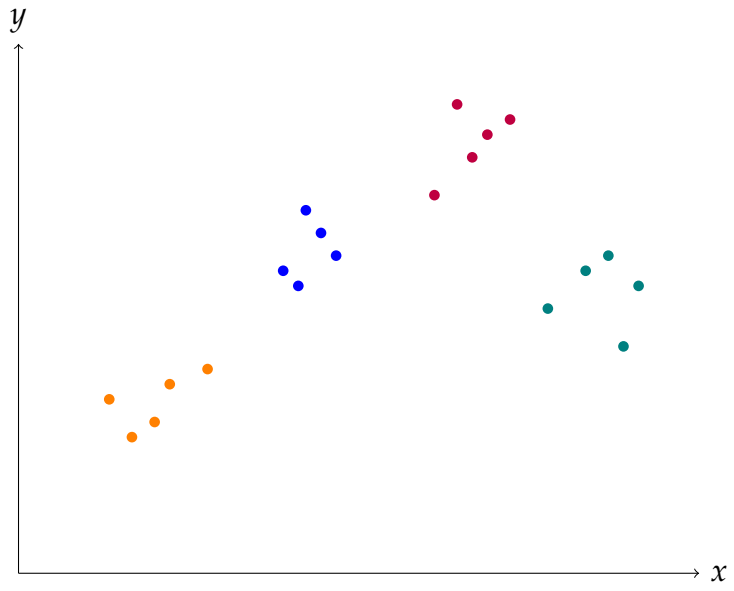


Figure 4.1: Data points in $\mathbb{R}^2$ that form four clusters.

4 Clustering

Similar to PCA, clustering is a way to understand structure of data. Clustering or cluster analysis is a generic term for algorithms designed to partition a given data into clusters. Clusters is a word that does not really have a precise meaning, and its meaning might even depend on situation, but generally we intuitively understand that clusters are groups of data points as seen in Figure 4.1.

Clustering is a fantastic tool for answering certain kinds of questions about the data like the following.

- What sub-populations exist in the data?
- Are there outliers?
- Are the sub-populations cohesive or can they be partitioned further?

"Populations" presupposes a certain perspective that need not be true. For example, clustering algorithms are used on gene sequences in medicine and biological sciences. See for example the hierarchical clustering done on sequences of genes for breast cancer in Figure 4.2.

Clustering is also one of the first steps in machine learning—an example of unsupervised learning. This just means that the machine "learns" of patterns and structures from the unlabeled data. Sometimes data are put into clusters, and then nearly all of the data, except for the cluster ID is discarded, which can save significant space in the analysis stage. To quote Google [3]:

> At Google, clustering is used for generalization, data compression, and privacy preservation in products such as YouTube videos, Play apps, and Music tracks.

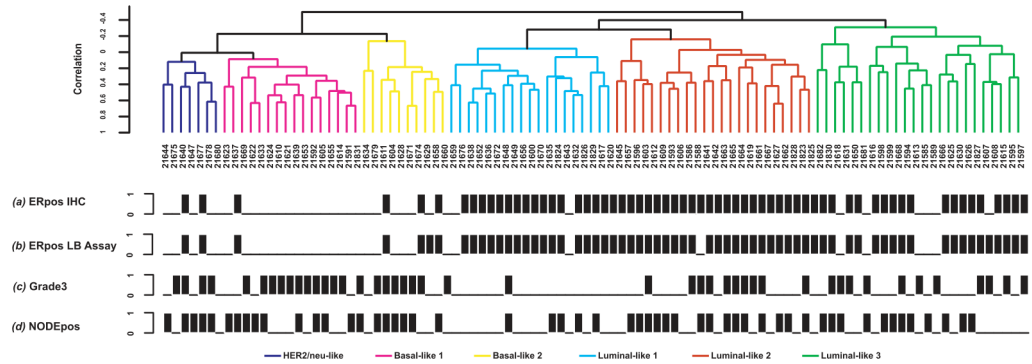


Figure 4.2: A hierarchical figure taken from [1]. The clusters are seen with the different colours.

There is no one clustering algorithm that is optimal in every situation; in fact, because data can cluster in many different ways, one needs a whole clustering toolkit. We will talk about two common methods for clustering: *k-means* and *single linkage* clustering. At the core of both of these clustering algorithms is a different assumption on the definition of a cluster.

4.1 Introduction to k-means clustering

The idea around k-means clustering is incredibly simple: group the data into k clusters based on the shortest distance to the center of the cluster. From this, we see how "cluster" is interpreted in the k-means algorithm—a cluster forms a "cell" in Euclidean space and data points outside of this cell are outside of the cluster. The shape of these cells depend on the layout of the centers of the clusters.

Before we go through the algorithm for k-means clustering, let's explore some of the many applications of this algorithm.

4.2 Image compression with k-means

Suppose we have a `jpg` file that is $m \times n$ pixels. We can turn this image file into a 3-dimensional array, namely an $m \times n \times 3$ array. Alternatively, we can interpret this as three $m \times n$ matrices corresponding to the red, green, and blue values of the image.

We can use k-means clustering by interpreting each pixel as a data point in $\mathbb{R}^3$ and determining clusters. Once we find our clusters, we can replace all the pixels with the center it is closest to. The impact is that we replace all the different colours with exactly k colours—essentially compressing the image.

Let's see how this might look on one of my photos. Meet my dog, Sherlock; he is depicted in Figure 4.3. There is not very many colours

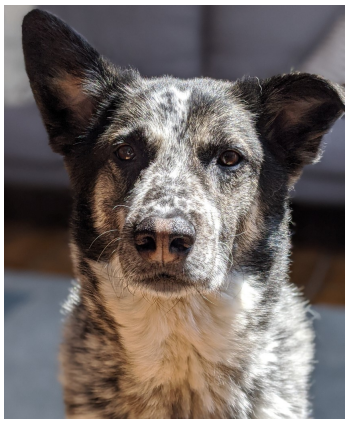


Figure 4.3: Sherlock.

in the image, so we should expect that low values of k will produce an image that looks fairly similar. Running a k-means clustering for each $k \in \{2, \ldots, 10\}$ yields nine different images seen in Figure 4.4.

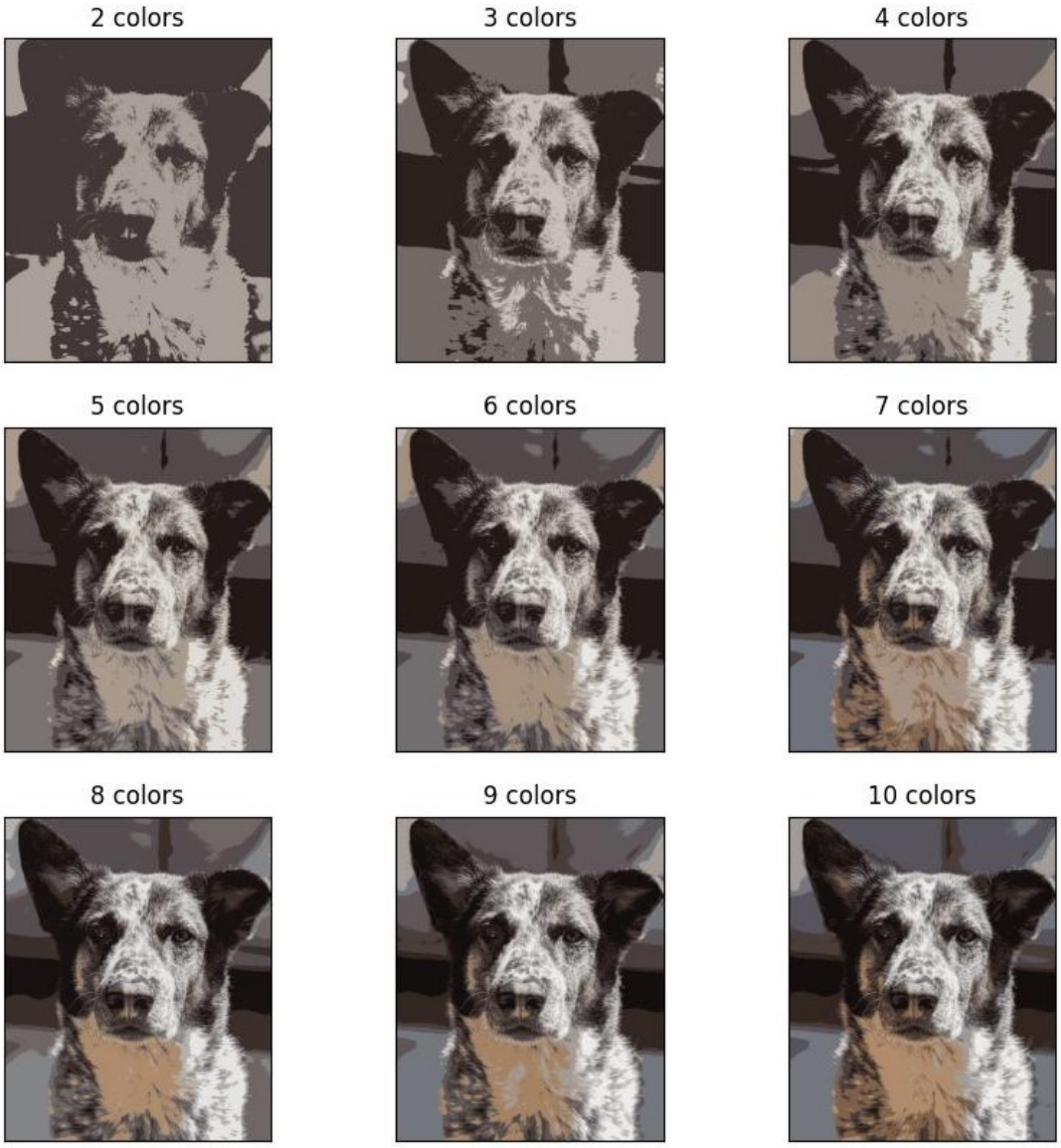


Figure 4.4: k-means clustering on the image in Figure 4.3.

4.3 The k-means algorithm

There are many versions of the basic algorithm we will describe. We use pseudocode to give the details of the algorithm; this is a way of giving the idea of the structure of the code without diving into the specifics of any particular language.

We provide pseudo-code for a basic k-means algorithm in Algorithm 2. The algorithm returns a function $\Lambda : \{1, \ldots, n\} \to \{1, \ldots, k\}$, which can be interpreted as a label for the n data points. The algorithm accomplishes this by randomly choosing the k centroids, and then clustering the data points accordingly. Once all the data points have been labeled, we recompute the centroids by taking the mean point of all the data points in a given cluster to be the new centroid.

Algorithm 2 (Basic) k-means

Input: a positive integer k and n data points $x_i \in \mathbb{R}^m$,
Output: a map $\Lambda : \{1, \ldots, n\} \to \{1, \ldots, k\}$.

1: Randomly assign the k centroids c_i to k data points.
2: Let Λ_0 and Λ be constant functions mapping to -1 and 0, respectively.
3: **while** $\Lambda_0 \neq \Lambda$ **do**
4: Set $\Lambda_0 = \Lambda$.
5: **for** $i \in \{1, \ldots, n\}$ **do**
6: Set $\Lambda(i) = j$ if $\|x_i - c_j\| = \min\{\|x_i - c_r\| : 1 \leqslant r \leqslant k\}$.
7: **end for**
8: **for** $i \in \{1, \ldots, k\}$ **do**
9: Set $c_i = \left(\sum_{j \in \Lambda^{-1}(i)} x_j\right) \Big/ \#\left(\Lambda^{-1}(i)\right)$.
10: **end for**
11: **end while**
12: **return** Λ.

In the first step of k-means (see line 1 in Algorithm 2), we randomly assign the k centroids. For different initial assignments, one can get different outputs. This is like trying to find global extrema on a "bumpy" graph by starting at a random point on the graph and then looking for the nearest local extrema. There are a number of ways to improve the k-means algorithm, we will not cover them, but many consider different variations for their specific data at hand.

4.4 A small example with k-means

We will consider 40 points in $\mathbb{R}^2$, which can be seen in Figure 4.5, and we will look for 4 clusters. You might try to see how you would cluster the data.

We will randomly select four data points for our initial centroids, and then we will group the data points into the four clusters. We will show this by colouring the Voronoi cells different colours and indicating the centroids by white crosses. All of the iterations of one k-means algorithm can be seen in Figure 4.6. Just to illustrate how sensitive k-means is to the initial choice of centroids, we run the algorithm again in Figure 4.7.

4.5 Introduction to hierarchical clustering

Hierarchical clustering is, in some sense, the opposite approach to clustering than k-means. We will primarily be working with Single-linkage clustering. Single-linkage or, more generally, *neighbour joining* clustering is a hierarchical clustering

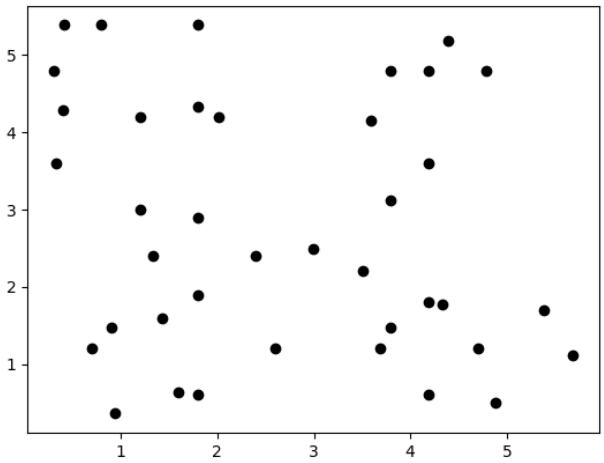


Figure 4.5: The plotted data we will use to run a 4-means clustering.

algorithm. With k-means, we need to give the number of clusters k in order to run the algorithm. With neighbour joining, we look at all the possible clusterings under prescribed conditions by means of a *dendrogram*. We have already seen an example of the output of a hierarchical clustering algorithm; see Figure 4.2.

The basic idea of neighbour joining clustering is that every data point is its own cluster and under certain conditions, two clusters are merged at each step until all points are in one cluster. The output is not a specific clustering of the data, but rather all the possible clusterings. This is communicated through a dendrogram.

Before we look at the algorithm, let's consider a specific application.

4.6 An application of hierarchical clustering

We will look at the study in [1] a little closely; see also Figure 4.2. People diagnosed with breast cancer can have very different clinical outcomes. We do not really know why this is the case, but one suggestion might be that our knowledge of the specific breast cancer tumors is lacking. Sotiriou et al. [1] propose clustering the tumors by gene expression profiles, via unsupervised hierarchical clustering.

Prior to their work, the tumors could be partitioned into two major subgroups based on their estrogen receptors. Using clustering methods, Sotiriou et al. were able to obtain this partition at the top of the dendrogram—if one cuts it so there are exactly two clusters, these align with the two subgroups based on estrogen receptors. They were able to further partition the two subgroups based on their basal and luminal characteristics. This led to the six clusters (represented by different colours) in Figure 4.2. This is also in line with previous work done by different research teams, and offers more evidence that their hypothesis is correct—that breast cancer tumors ought to be clustered in the established way.

4.7 The neighbour joining algorithm

We will describe a class of *agglomerative* algorithms that can be altered to produce different outcomes based on certain given conditions. In order to do this, one of the inputs is a distance function, which can be obtained from a *metric*. We want the metric to be defined on finite subsets of $\mathbb{R}^m$, which will be interpreted as clusters in the actual algorithm.

Definition 4.1. A function $d : \mathscr{M} \times \mathscr{M} \to \mathbb{R}$ on a set $\mathscr{M}$ is a *metric* if

1. for all $X, Y \in \mathscr{M}$, we have $d(X, Y) \geqslant 0$, where equality holds if and only if $X = Y$,
2. for all $X, Y \in \mathscr{M}$, we have $d(X, Y) = d(Y, X)$, and
3. for all $X, Y, Z \in \mathscr{M}$, the triangle inequality is satisfied:

$$d(X, Y) + d(Y, Z) \geqslant d(X, Z).$$

Functions satisfying (1) and (2) are called *semimetrics*.

Lemma 4.2. *The Euclidean distance $d(x, y) = \|x - y\|$ is a metric on $\mathbb{R}^m$. The square of the Euclidean distance, $d_\square(x, y) = \|x - y\|^2$, is not a metric but a semimetric.*

Proof. To show that $d_\square$ does not satisfy the triangle inequality, let $a \in \mathbb{R}$ be positive. Then $|-a|^2 + |a|^2 < |2a|^2$, so

$$d_\square(a, 0) + d_\square(-a, 0) < d_\square(a, -a). \qquad \square$$

In the context of clustering, having a metric is not necessary. For example, because $x \mapsto x^2$ is monotonically increasing on the $\mathbb{R}_{\geqslant 0}$, it follows that $d(x, y) < d(u, v)$ if and only if $d_\square(x, y) < d_\square(u, v)$; here we are using the notation from Theorem 4.2. Thus, it is irrelevant if we cluster two points based on d or $d_\square$, so we allow our distance functions to come from semimetrics.

Let's write $\mathscr{F}_m$ for the set of finite subsets of $\mathbb{R}^m$, so we will describe a few distance functions on $\mathscr{F}_m$.

Single-linkage: For $C_1, C_2 \in \mathscr{F}_m$,

$$d(C_1, C_2) = \min \{\|x - y\| : x \in C_1, y \in C_2\}.$$

Complete-linkage: For $C_1, C_2 \in \mathscr{F}_m$,

$$d(C_1, C_2) = \max \{\|x - y\| : x \in C_1, y \in C_2\}.$$

Average-linkage: For $C_1, C_2 \in \mathscr{F}_m$,

$$d(C_1, C_2) = \frac{1}{\#C_1 \cdot \#C_2} \sum_{x \in C_1} \sum_{y \in C_2} \|x - y\|.$$

Technically the single-linkage is not a semimetric: if $C_1 \cap C_2 \neq \varnothing$ yet $C_1 \neq C_2$, then $d(C_1, C_2) = 0$. However we only consider sets C_1 and C_2 with trivial intersection anyways, so this problem is avoided.

Lemma 4.3. *The complete-linkage function defines a metric on $\mathscr{F}_m$.*

There are of course other distances one could use. Single-linkage clustering, or nearest neighbour, corresponds to neighbour joining with the single-linkage metric. Likewise, complete-linkage clustering, or farthest neighbour, corresponds to neighbour joining with the complete-linkage metric. One could design a metric that might be most relevant for the particular data set at hand.

The output for a hierarchical clustering algorithm is a *dendrogram* or data equivalent to one. Figure 4.2 contains one example. Another very similar example is a phylogenetic tree seen in Figure 4.8.

To read a dendrogram, one first finds the many leaves or ends—often these are labeled—these are the initial clusters. As one goes through the dendrogram (e.g. bottom to top or left to right), clusters get linked, which is depicted by joining the ends of the initial clusters. This happens until there is exactly one cluster left. Two ends are closer (or more similar) if they become joined early on compared to two ends that only become joined near the end of the dendrogram.

Let's explain the algorithm using pseudo-code. Instead of outputting a dendrogram, we will output a list of pairs of the form $(t, [C_1, \ldots, C_r])$, where $t \geqslant 0$ and each C_i is a finite set of data points. This list will start with $(0, [C_1, \ldots, C_n])$, where each C_i is a singleton, and it will end with $(t_\infty, [C_1])$, where C_1 contains all of the data points. Note that this data is also equivalent to a dendrogram.

Algorithm 3 Neighbour joining

Input: a semimetric $d : \mathscr{F}_m \times \mathscr{F}_m \to \mathbb{R}$ and n data points $x_i \in \mathbb{R}^m$,
Output: a list D equivalent to a dendrogram.

Set $t = 0$.
Set $C = \{\{x_1\}, \ldots, \{x_n\}\}$.
Set $D = [(t, C)]$.
while $\#C > 1$ **do**
    Construct distance matrix $M = (M_{ij})$, with $M_{ij} = d(C_i, C_j)$.
    Determine the two closest distinct clusters C_r and C_s from M.
    Set $t = t + M_{rs}$.
    Set $C = (C \setminus \{C_r, C_s\}) \cup \{C_r \cup C_s\}$.
    Append to D the pair (t, C).
end while
return D.

We will soon go through Algorithm 3 on a small example. Unlike k-means, neighbour joining is *deterministic*; that is, it runs the same every time on the same input. Notice in Algorithm 3 that the actual points x_i do not matter; instead, it's the distances between the points (and clusters) that matter. Thus, we won't specify points but a *graph*.

Definition 4.4. A (finite, simple) **graph** G is a pair of vertices $V = \{v_1, \ldots, v_n\}$ and edges $E = \{e_1, \ldots, e_k\}$, where each $e_i \subseteq V$ of size exactly two. Vertices v_i and v_j are **adjacent**, written $v_i \sim v_j$, if $\{v_i, v_j\} \in E$.

Graphs are ubiquitous in mathematics and computer science, and they can be easily visualised (for a relatively small amount of vertices). Three graphs can be seen in Figure 4.9.

4.8 A small example of hierarchical clustering

Instead of writing down or plotting data points, we will write down a distance matrix. The distance matrix is a symmetric matrix with zeros along the diagonal. (Why?) The (i, j) entry is equal to the distance between vertices (or clusters) v_i and v_j. Let's consider the following distance matrix.

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	4	8	1	7	8
v_2	4	0	6	5	5	6
v_3	8	6	0	7	2	2
v_4	1	5	7	0	8	8
v_5	7	5	2	8	0	1
v_6	8	6	2	8	1	0

We will apply single- and complete-linkages and cluster the data accordingly. Because we know how to compute them both given the distances from all pairs of vertices, we will not recompute our distance matrix each time. Instead we will draw nine graphs labeled G_t for $t \in \{0, \ldots, 8\}$ since the maximal distance between any two points is 8. We will include an edge between v_i and v_j in G_t if and only if $d(v_i, v_j) \leqslant t$. The graph G_0 is the graph on 8 vertices with no edges, and the graph G_8 is the graph on 8 vertices where every vertex is adjacent to every other vertex (this is called a *complete graph*). All of the graphs are shown in Figure 4.10.

Now we will produce dendrograms, one for the single-linkage and one for the complete-linkage. There are two concepts in Graph Theory that we can use to determine these dendrograms.

Definition 4.5. Two vertices v_i and v_j of a graph G are **connected** if there exists a sequence of adjacent vertices of the form $v_i = v_0\, v_1\, \cdots\, v_\ell = v_j$.

Definition 4.6. A subset S of vertices of a graph G form a **clique** if for all distinct $v_i, v_j \in S$, the edge $\{v_i, v_j\}$ is in G.

Let's consider the graphs in Figure 4.10 again. Using single-linkage, we know that vertices v_i and v_j lie in the same cluster if and only if v_i and v_j are connected. Therefore, we have exactly one cluster starting at G_5 using single-linkage. Using complete-linkage, we know that vertices $\{v_{i_1}, \ldots, v_{i_\ell}\}$ form a cluster if and only if they form a clique. Hence, we have exactly one cluster starting at G_8 using complete-linkage. The dendrograms can be seen in Figures 4.11 and 4.12.

References

[1] Christos Sotiriou, Soek-Ying Neo, Lisa M. McShane, Edward L. Korn, Philip M. Long, Amir Jazaeri, Philippe Martiat, Steve B. Fox, Adrian L. Harris, and Edison T. Liu, *Breast cancer classification and prognosis based on gene expression profiles from a population-based study*, Proceedings of the National Academy of Sciences **100** (2003), no. 18, 10393–10398.

[2] Peter Frankl and Hiroshi Maehara, *Some geometric applications of the beta distribution*, Ann. Inst. Statist. Math. **42** (1990), no. 3, 463–474.

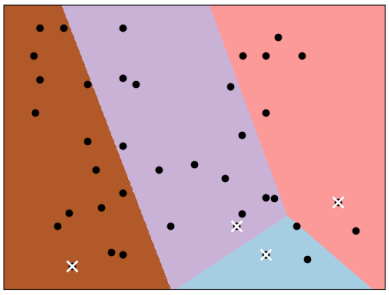
[3] Google, *What is Clustering?* (2023), date accessed: 16 Oct. 2023. https://developers.google.com/machine-learning/clustering/overview.

[4] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani, *Approximate nearest neighbor: towards removing the curse of dimensionality*, Theory Comput. **8** (2012), 321–350, DOI 10.4086/toc.2012.v008a014. MR2948494

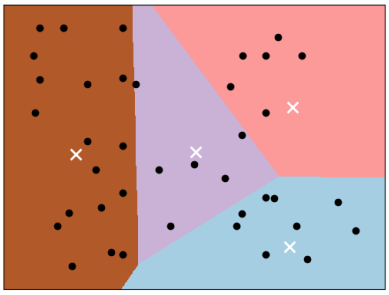
[5] Dan Margalit and Joseph Rabinoff, *Interactive Linear Algebra*, 2019, https://textbooks.math.gatech.edu/ila/.

[6] Jonathon Shlens, *A tutorial on principal component analysis*, 2014, arXiv:1404.1100.

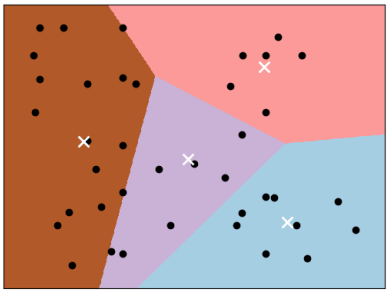
[7] Fan Yang and Sifan Liu and Edgar Dobriban and David P. Woodruff, *How to reduce dimension with PCA and random projections?*, 2021, arXiv:2005.00511.



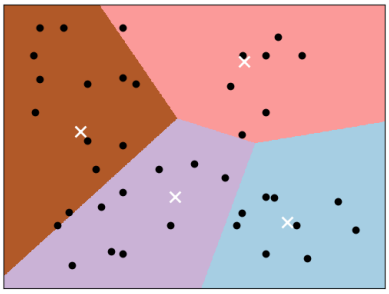
(a) Iteration 0



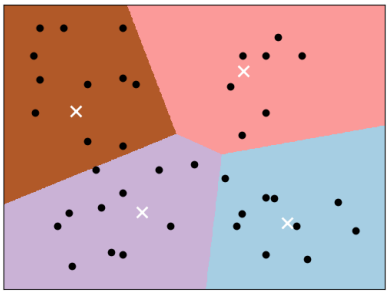
(b) Iteration 1



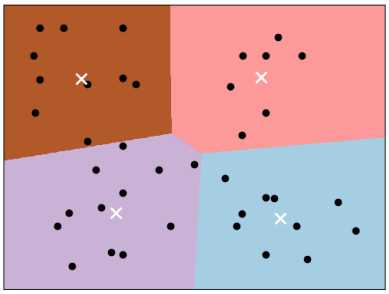
(c) Iteration 2



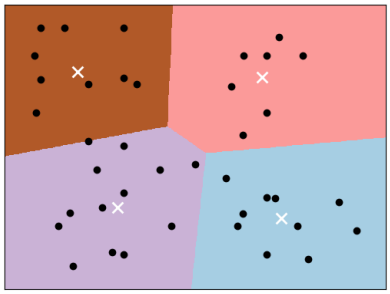
(d) Iteration 3



(e) Iteration 4

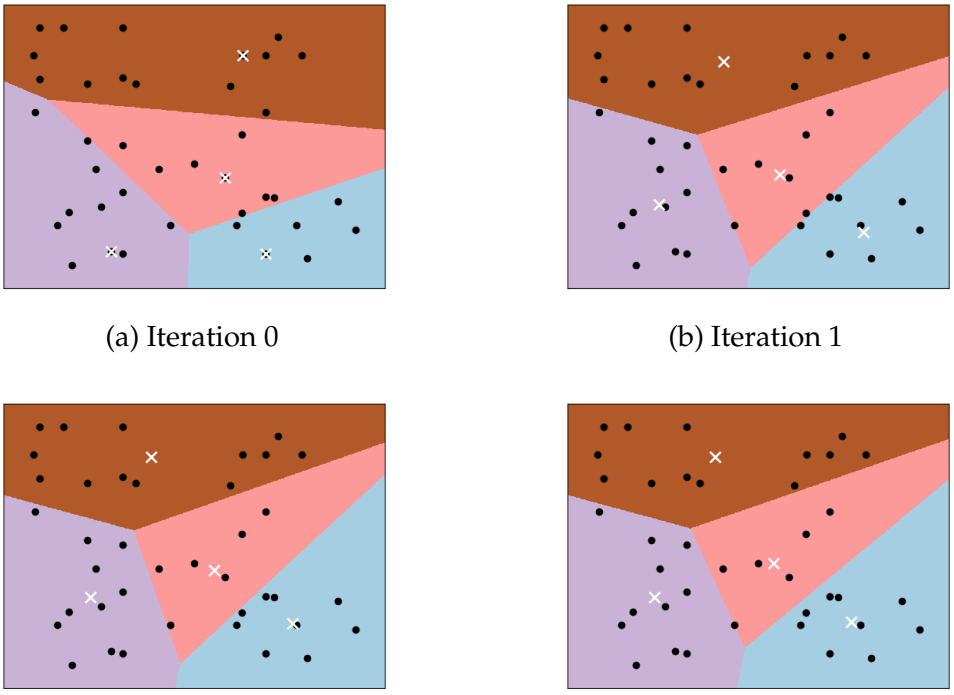


(f) Iteration 5



(g) Iteration 6

Figure 4.6: Seven iterations of one k-means algorithm.



(a) Iteration 0

(b) Iteration 1

(c) Iteration 2

(d) Iteration 3

Figure 4.7: Four iterations of another k-means algorithm.

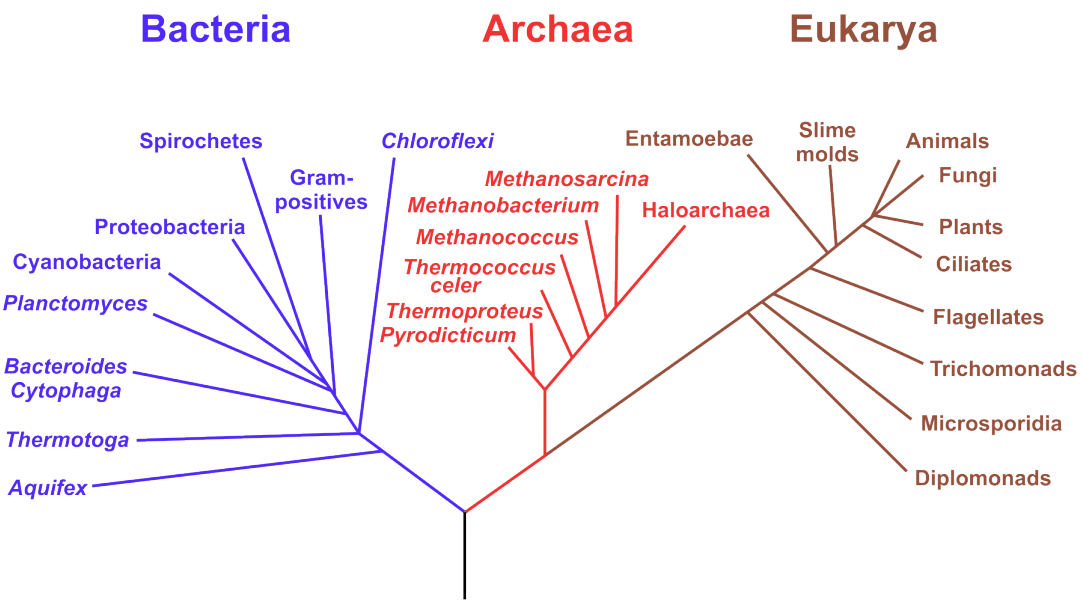
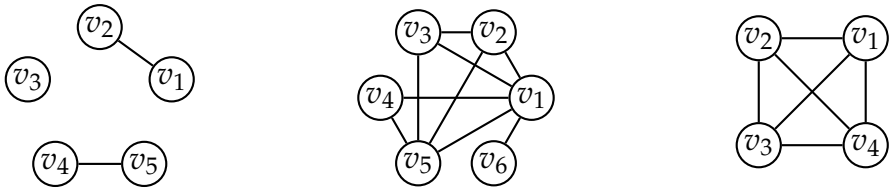


Figure 4.8: A phylogenetic tree is equivalent to a dendrogram.



G_1 G_2 G_3

Figure 4.9: Three different graphs.

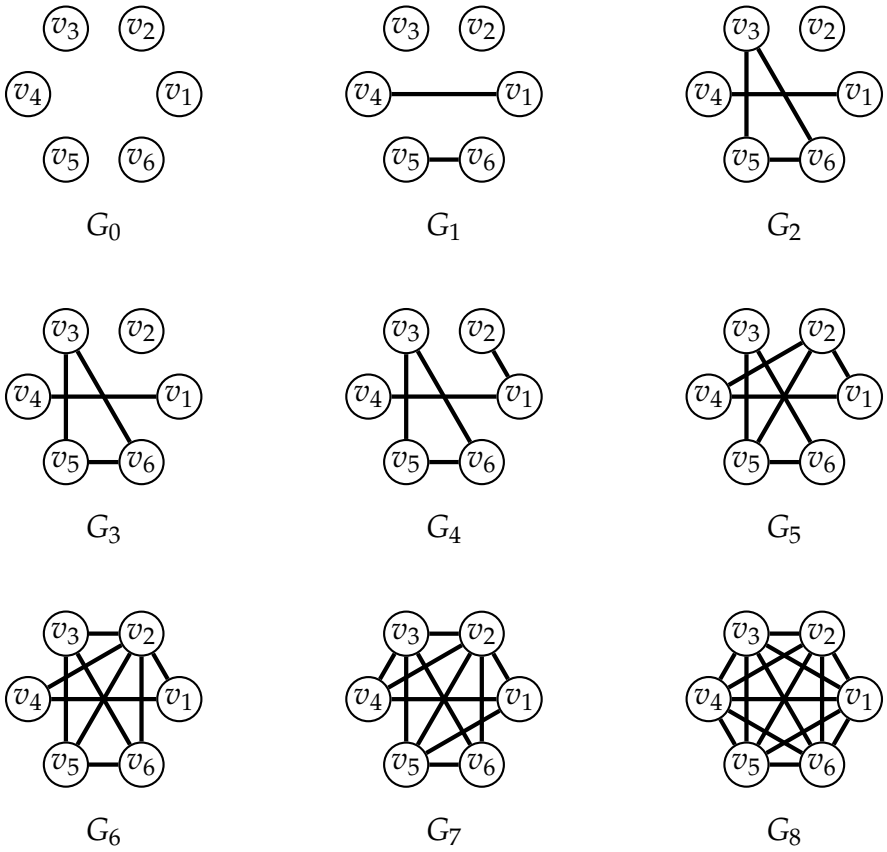


Figure 4.10: The 9 graphs visualising the possible clustering of the 8 vertices.

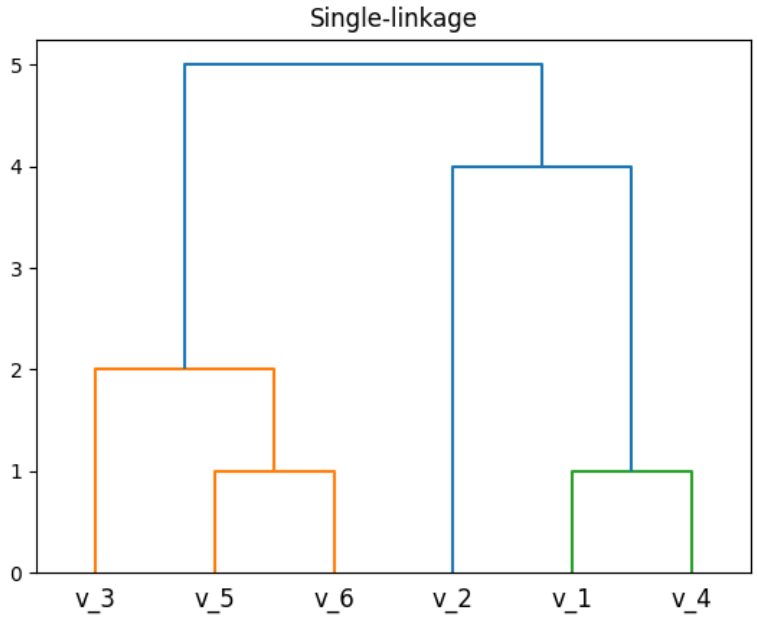


Figure 4.11: The dendrogram corresponding to the graphs in Figure 4.10 using single-linkage.

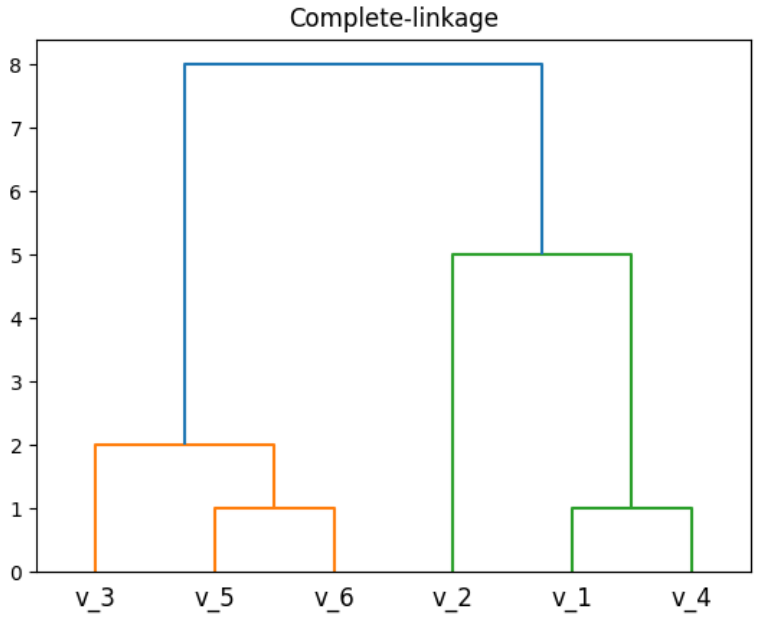


Figure 4.12: The dendrogram corresponding to the graphs in Figure 4.10 using complete-linkage.