Mathematics of Data Science

Martin Ehler

Tue 9:45 - 12:00 pm, SR 18

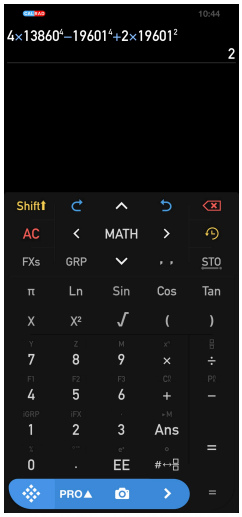
Copyright © 2023 Martin Ehler
All rights reserved

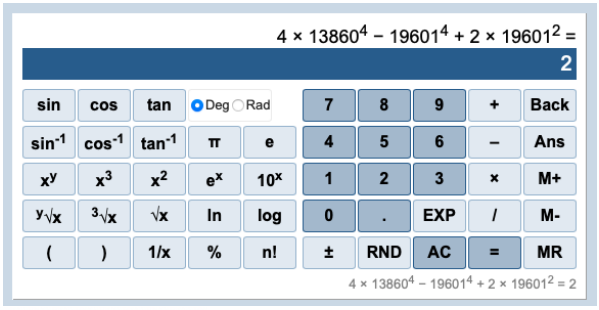
Contents

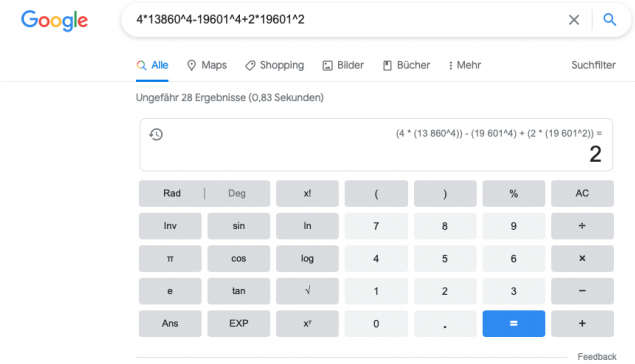
When the machine says, trust me. . .

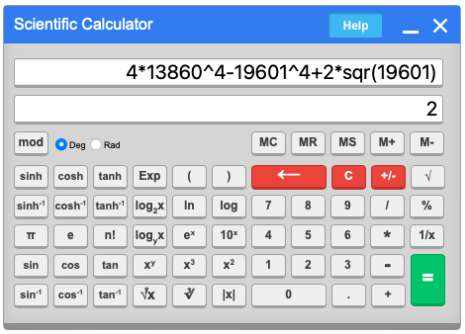
The success of machine learning and its use in modern daily life also raises few questions. For instance, how confident are we that the computations are correct?

The calculations below appear reliable. After all, we expect the result to be an integer.









The exact result, however, is 1. This is a simple example that shows how easy we may get fooled by machine responses that look trustworthy at first sight. To quantify accuracy and levels of trust, we need to understand the algorithms and learning schemes in use.

Chapter

1 Machine numbers

Numerical computations are usually not exact. To understand and quantify the accuracy, we discuss floating point numbers and the machine precision.

How are numbers typically represented on a digital computer?

If we have n bits available, then we may encode 2^n different objects.

Example 1.0.1

For $n = 3$ bits, we encode $2^3 = 8$ different objects by

000	001	010	100
011	101	110	111

A typical laptop or desktop computer (in the year 2022) runs a 64-bit system, i.e., it assigns 64 bits to encode an integer. Thus, we may have $2^{64} \approx 10^{19}$ different machine integers. The observation

$$2^{64} = \#\{0, \dots 2^{64} - 1\} = \#\{-2^{63}, \dots, 2^{63} - 1\}$$

leads to

- 64-bit integers:
$$\operatorname{Int} 64 = \left\{-2^{63}, \dots, 2^{63} - 1\right\}.$$
Integer computer arithmetics are exact if you stay within the range.
- 64-bit rationals:
$$\operatorname{Rational}\{\} = \operatorname{Rational}\{\operatorname{Int} 64\} = \left\{\frac{p}{q} : p, q \in \operatorname{Int} 64,\ q \neq 0\right\} \cup \{\pm\infty\}.$$
Rational computer arithmetics are exact.

α	11-bits
"special"	00000000000
-1022	00000000001
-1021	00000000010
$\vdots$	$\vdots$
-1	01111111110
0	01111111111
1	10000000000
$\vdots$	$\vdots$
1023	11111111110
"special"	11111111111

Table 1.1: Bit representations of the exponent α.

1.1 Floating point numbers

The *floating point numbers* are an attempt to cover a "large" part of the real numbers. We follow the IEEE 754 standard.

For a binary number such as 110 or $1,1001$, we put

$$(110)_2 := 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6$$
$$(1,1001)_2 := 1 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = 1 + \frac{9}{16} = \frac{25}{16}.$$

Consider

$$\pm \left(1, a_1 a_2 \dots a_{52}\right)_2 \cdot 2^\alpha = (-1)^s \left(1 + \sum_{k=1}^{52} a_k 2^{-k}\right) 2^\alpha, \qquad s, a_k \in \{0,1\}. \tag{1.1}$$

In 64 bit-encoding, bits are used as follows:

- 1 bit for the sign s
- 52 bits for *mantisse/significant digits* $a_1, \dots a_{52}$
- 11 bits for the *exponent* $\alpha \in \mathbb{Z}$.

We observe

$$2^{11} = 2048 = \#\{0, \dots, 2047\} = \#\{\underbrace{-1023}_{\text{special}}, -1022, \dots, 1023, \underbrace{1024}_{\text{special}}\}.$$

Since the bit stream of -1023 and 1024 are used for encoding special "numbers", see Table 1.1, we have $\alpha \in \{-1022, \dots, 1023\}$.

special value	sign s	mantisse $a_1 \dots a_{52}$	11 bits for α
$+0$	0	$0 \dots\dots 0$	$0 \dots 0$
-0	1	$0 \dots\dots 0$	$0 \dots 0$
$+\infty$	0	$0 \dots\dots 0$	$1 \dots 1$
$-\infty$	1	$0 \dots\dots 0$	$1 \dots 1$
NaN	$0, 1$	$10 \dots\dots$	$1 \dots 1$

Table 1.2: Bit representations of s, $a_1 \dots a_{52}$ and α for some special values. NaN stands for Not a Number such as $0/0$.

Definition 1.1.1: (floating point numbers)

The floating point numbers (for the 64-bit IEEE standard) are

$$\mathbb{F}_{64} := \left\{ \pm \left(1, a_1 a_2 \dots a_{52}\right)_2 \cdot 2^{\alpha} \ : \ a_k \in \{0,1\},\ \alpha \in \{-1022, \dots, 1023\} \right\},$$

and we define

$$R_{64} := \max(\mathbb{F}_{64}), \qquad r_{64} := \min(\{x \in \mathbb{F}_{64} : x > 0\}).$$

If $x \in \mathbb{F}_{64}$, then we have $|x| \in [r_{64}, R_{64}]$.

Lemma 1.1.2

The largest and smallest positive floating point numbers are

$$R_{64} = 2^{1024}\left(1 - 2^{-53}\right) > 10^{308}, \qquad r_{64} = 2^{-1022} < 3 \cdot 10^{-308},$$

and $\mathbb{F}_{64}$ is a union of equally spaced numbers with gaps $2^{\alpha-52}$, i.e.,

$$\mathbb{F}_{64} = \bigcup_{\alpha=-1022}^{1023} \{\pm 2^{\alpha}(1 + k2^{-52}) : k = 0, 1, \dots, 2^{52} - 1\}. \tag{1.2}$$

Proof. We directly compute

$$\begin{aligned} R_{64} &= (1, 1 \dots 1)_2 \cdot 2^{1023} \\ &= \sum_{k=0}^{52} 2^{-k} \cdot 2^{1023} \\ &= \frac{1 - 2^{-53}}{1/2} 2^{1023} = 2^{1024}\left(1 - 2^{-53}\right). \end{aligned}$$

The remaining claims also follow directly from the definitions. □

1.2 Machine precision

Let us quantify the error inherent to 64-bit arithmetic operations.

Definition 1.2.1: (machine precision)

The number $\text{eps}_{64} := 2^{-52}$ is called *machine precision*.

Example 1.2.2: (smallest machine number bigger than 5)

We have $5 = 2^2 + 2^0$ and

$$2^2\left(2^0 + 2^{-2} + 2^{-52}\right) = 5 + 4\,\text{eps}_{64}$$

is the smallest machine number bigger than 5.

Example 1.2.3: (smallest positive integer not in $\mathbb{F}_{64}$)

We have $2^{53} \in \mathbb{F}_{64}$ and Lemma 1.1.2 leads to $2^{53} + 1 \notin \mathbb{F}_{64}$.

Any number in $[-R_{64}, -r_{64}] \cup [r_{64}, R_{64}]$ can be written as

$$\pm\left(\sum_{k=0}^{\infty} a_k 2^{-k}\right) 2^{\alpha},$$

with $a_k \in \{0, 1\}$ and $\alpha \in \{-1022, \ldots, 1023\}$, while agreeing on $a_0 := 1$.

Definition 1.2.4: (rounding)

Rounding down $\text{fl}_{\downarrow} : [-R_{64}, -r_{64}] \cup [r_{64}, R_{64}] \to \mathbb{F}_{64}$ is defined by the truncation

$$\pm\left(\sum_{k=0}^{\infty} a_k 2^{-k}\right) 2^{\alpha} \mapsto \pm\left(\sum_{k=0}^{52} a_k 2^{-k}\right) 2^{\alpha}. \tag{1.3}$$

Rounding up $\text{fl}_{\uparrow}$ assigns the closest floating point number that is bigger or equal to its argument. The computer rounding is denoted by fl and assigns the closest floating point number.

Lemma 1.2.5: (rounding error)

For $x \in \mathbb{R}$ with $|x| \in [r_{64}, R_{64}]$, the relative rounding error satisfies

$$\left|\frac{\text{fl}(x) - x}{x}\right| \leq \text{eps}_{64}\,. \tag{1.4}$$

Proof. The rounding (1.3) yields

$$\left|\frac{\mathrm{fl}(x)-x}{x}\right| \leq \left|\frac{\left(\sum_{k=53}^{\infty} a_k 2^{-k}\right) 2^{\alpha}}{\left(\sum_{k=0}^{\infty} a_k 2^{-k}\right) 2^{\alpha}}\right| = \left|\frac{\sum_{k=53}^{\infty} a_k 2^{-k}}{\sum_{k=0}^{\infty} a_k 2^{-k}}\right|.$$

Since $a_0 = 1$, we observe $\sum_{k=0}^{\infty} a_k 2^{-k} \geq 1$, so that $\sum_{k=53}^{\infty} a_k 2^{-k} \leq 2^{-52}$ concludes the proof. □

We proceed with the computer addition $\oplus$ and multiplication $\odot$. If $\star$ is either $+$ or $\cdot$, then $\circledast$ is implemented such that, for all $x, y \in \mathbb{F}_{64}$ with $|x \star y| \in [r_{64}, R_{64}]$,

$$x \circledast y := \mathrm{fl}(x \star y). \tag{1.5}$$

The relative errors of computer addition and multiplication are bounded by the machine precision:

Theorem 1.2.6: (precision of computer arithmetics)

For all $x, y \in \mathbb{F}_{64}$ with $|x \star y| \in [r_{64}, R_{64}]$,

$$\left|\frac{x \circledast y - (x \star y)}{x \star y}\right| \leq \mathrm{eps}_{64}\,. \tag{1.6}$$

Proof. The assertion follows directly from (1.5) and Lemma 1.2.5. □

1.3 Subnormal floating point numbers

So far, $\mathbb{F}_{64}$ has a gap around 0, and we now fill the range $(-r_{64}, r_{64})$ by

$$\frac{2}{\mathrm{eps}_{64}} - 1 = 2^{53} - 1$$

many equally spaced numbers.

Definition 1.3.1: (subnormal floating point numbers)

The *subnormal* floating point numbers are

$$\begin{aligned}\mathbb{F}_{64,\mathrm{sub}} &= \{\pm k \cdot \mathrm{eps}_{64} \cdot r_{64} : k = 1, \ldots, \mathrm{eps}_{64}^{-1} - 1\} \\ &= \{\pm k 2^{-1074} : k = 1, \ldots, 2^{52} - 1\}.\end{aligned}$$

Example 1.3.2

The smallest positive subnormal floating point number is

$$r_{64,\mathrm{sub}} = r_{64} \cdot \mathrm{eps}_{64} = 2^{-1074}.$$

$\mathbb{F}_{64,\mathrm{sub}} \cup \{0\}$	52-bits mantisse
0	$0\cdots000$
2^{-1074}	$0\cdots001$
$2\cdot 2^{-1074}$	$0\cdots010$
$3\cdot 2^{-1074}$	$0\cdots011$
$\vdots$	$\vdots$
$(2^{52}-1)\cdot 2^{-1074}$	$1\cdots111$

Table 1.3: Bit representations of subnormal floating point numbers. The 11 bits of the exponent are all 0.

Computer rounding fl is extended to $\mathbb{F}\cup\mathbb{F}_{\mathrm{sub}}$.

Example 1.3.3

The number $(1+\frac{1}{2})2^{-1074}$ is rounded to 2^{-1073}. The estimates (1.4) and (1.6) do not extend to the range of $\mathbb{F}_{64,\mathrm{sub}}$ since

$$\left|\frac{2^{-1073}-(1+\frac{1}{2})2^{-1074}}{(1+\frac{1}{2})2^{-1074}}\right| = \frac{1}{3}.$$

Definition 1.3.4

The set of *machine numbers* is $\mathbb{F}_{64}\cup\mathbb{F}_{64,\mathrm{sub}}\cup\{0\}$.

Altogether, we define

$$\mathrm{Float}\,64 = \mathbb{F}_{64}\cup\mathbb{F}_{64,\mathrm{sub}}\cup\{\pm 0, \pm\,\mathrm{Inf}, \mathrm{NaN}\},$$

where $\pm\,\mathrm{Inf} = \pm\infty$ and NaN is Not a Number such as $0/0$.

Chapter 2

Data compression

When images are stored at your machine, they have usually undergone some compression steps. The file ending ".jpg" is quite common and refers to a particular compression scheme. This section is dedicated to the JPEG compression algorithm. It involves the cosine-transform and a distinct encoding strategy.

2.1 Huffman encoding

The alphabet $\{a, b, c, d, e, f\}$ can be encoded by 3 bits per element. The string

ffcebcffcafffaedbfebffdedfdecbfffcfeeecfdfcffcbfcffeadfffedddffddbcfbcfefffbdfefbeeffffcffffefdefaa

has length is 100. It can be encoded by the use of 3 bits per entry, which requires 300 bits. Can we do better?

We count the occurrences:

	a	b	c	d	e	f
occurrence	5	9	12	13	16	45

Example 2.1.1: Huffman encoding

We build the Huffman tree in Figure 2.1. We start with the two smallest frequencies and add a node with their sum in Figure 2.1(a). Then we take the next two smallest frequencies, etc. It leads to the encoding

	a	b	c	d	e	f
occurrence	5	9	12	13	16	45
bit-encoding	1100	1101	100	101	111	0

None of the bit-encodings is the beginning of another one. Therefore, we require only

$$45 + 3(12 + 13 + 16) + 4(5 + 9) = 224$$

bits and the encoded string starts with

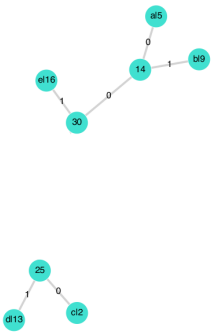
$$001001111101\ldots$$

that covers "ffceb…".

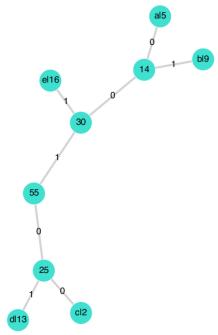
Works great if there are elements in the alphabet with high occurrences.

(a) $a|5$ and $b|9$ yields 14

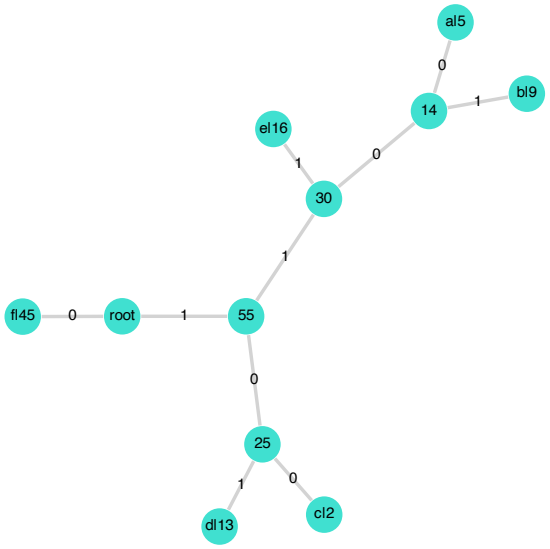
(b) $c|12$ and $d|13$ yields 25



(c) 14 and $e|16$ yields 30



(d) 25 and 30 yields 55



(e) $f|45$ and 55 yields the root

Figure 2.1: Huffman tree: Descending from the root, we put 0 to the smaller and 1 to the larger subnode.

2.2 Discrete cosine transform

Another ingredient of the JPEG algorithm is the discrete cosine transform.

Lemma 2.2.1

The vectors

$$\tilde{w}_k = \left(\cos(\pi k \tfrac{j+\frac{1}{2}}{n})\right)_{j=0,\dots,n-1}, \qquad k = 0, \dots, n-1,$$

are orthogonal in $\mathbb{R}^n$ with

$$\langle \tilde{w}_k, \tilde{w}_l \rangle = \begin{cases} 0, & k \neq l, \\ \frac{n}{2}, & k = l \neq 0, \\ n, & k = l = 0. \end{cases}$$

Proof. For $k \neq l$, we derive with some help of Wolfram Alpha

$$\begin{aligned} \langle \tilde{w}_k, \tilde{w}_l \rangle &= \sum_{j=0}^{n-1} \cos(\pi k \tfrac{j+\frac{1}{2}}{n}) \cos(\pi l \tfrac{j+\frac{1}{2}}{n}) \\ &= \dots \\ &= \frac{1}{4} \left(\frac{\sin(\pi(k-l))}{\sin(\frac{\pi(k-l)}{2n})} + \frac{\sin(\pi(k+l))}{\sin(\frac{\pi(k+l)}{2n})} \right) \\ &= \dots \end{aligned}$$

□

Let $\{w_0, \dots, w_{n-1}\}$ denote the normalized vectors that form an orthonormal basis for $\mathbb{R}^n$, see Figure 2.2 for a visualization with $n = 8$.

For $A = (a_{i,j}) \in \mathbb{R}^{n \times n}$, we define

$$\operatorname{trace}(A) = \sum_{i=1}^{n} a_{i,i}.$$

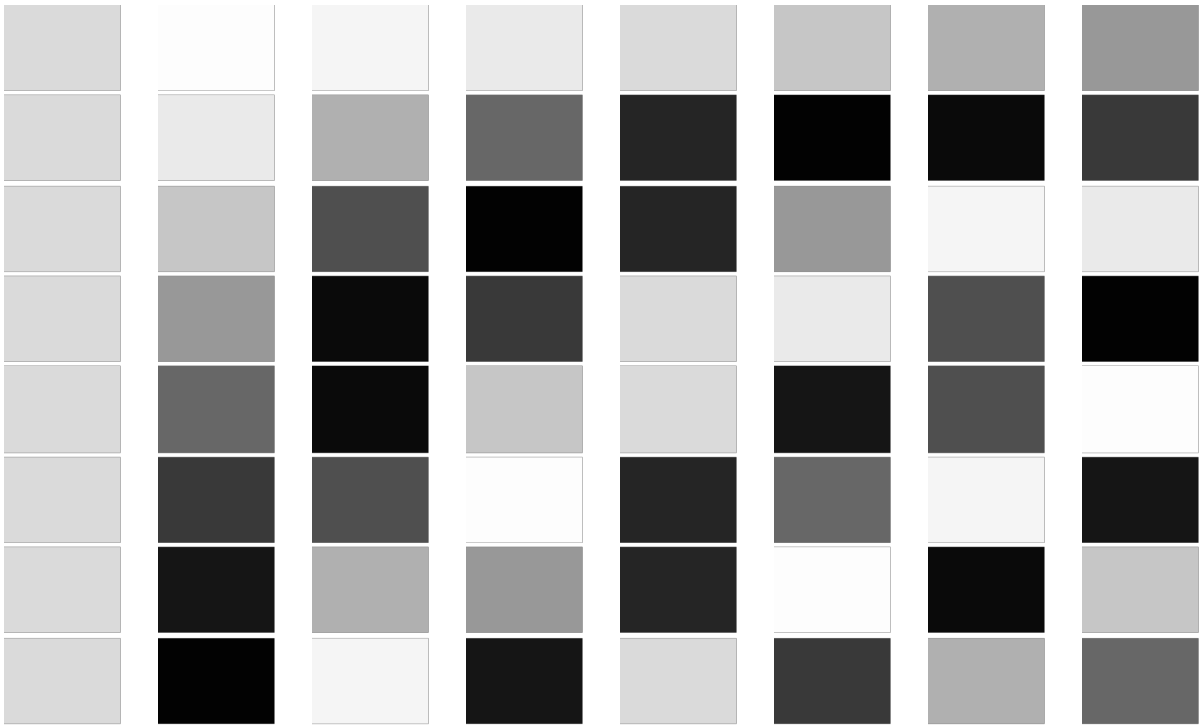
We may define an inner product on $\mathbb{R}^{n \times n}$ by

$$\langle A, B \rangle_{\mathbb{R}^{n \times n}} := \operatorname{trace}(AB^\top), \qquad A, B \in \mathbb{R}^{n \times n}.$$

Theorem 2.2.2

The n^2 many vectors

$$W_{k,l} := w_k w_l^\top \in \mathbb{R}^{n \times n}, \qquad k, l = 0, \dots, n-1$$



(a) w_0 (b) w_1 (c) w_2 (d) w_3 (e) w_4 (f) w_5 (g) w_6 (h) w_7

Figure 2.2: The 8 basis vectors

form an orthonormal basis for $\mathbb{R}^{n\times n}$.

See Figure 2.3 for $n = 8$.

Proof. We compute

$$\begin{aligned}\langle W_{i,j}, W_{k,l}\rangle_{\mathbb{R}^{n\times n}} &= \operatorname{trace}(W_{i,j}W_{k,l}^\top)\\ &= \operatorname{trace}(w_i w_j^\top (w_k w_l^\top)^\top)\\ &= \operatorname{trace}(w_i \underbrace{w_j^\top w_l}_{\delta_{j,l}} w_k^\top)\\ &= \langle w_i, w_k\rangle \delta_{j,l}\\ &= \delta_{i,k}\delta_{j,l}.\end{aligned}$$

Hence $\{W_{k,l} : k, l = 0, \dots, n-1\}$ is an orthonormal system with n^2 many elements. The dimension of $\mathbb{R}^{n\times n}$ is also n^2, so that we do have a basis. □

Thus, every matrix $X \in \mathbb{R}^{n\times n}$ is a linear combination

$$X = \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} \langle X, W_{k,l}\rangle_{\mathbb{R}^{n\times n}} W_{k,l}.$$

To store X, we may simply store the coefficients

$$c_{k,l} = \langle X, W_{k,l}\rangle_{\mathbb{R}^{n\times n}}.$$

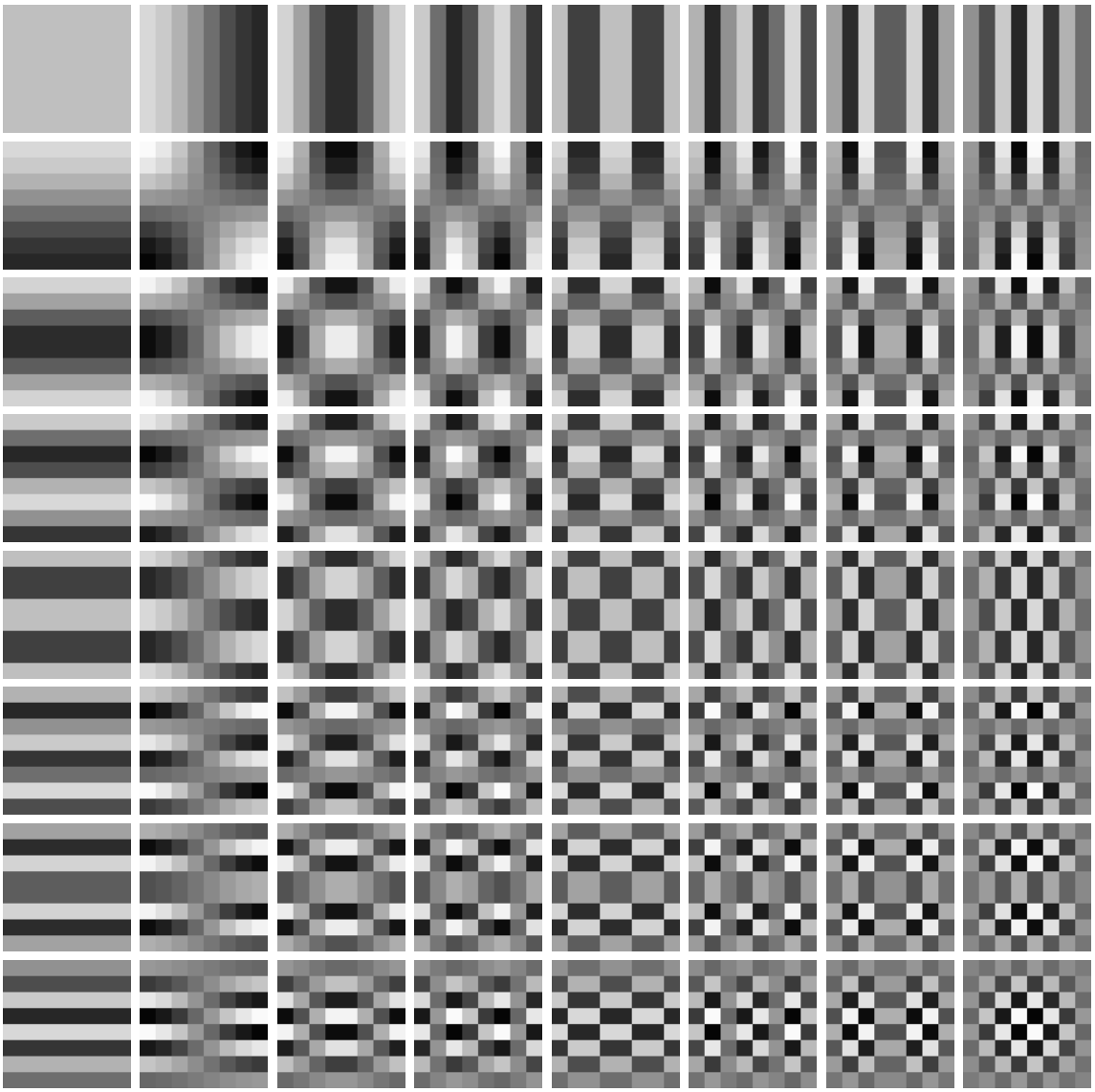


Figure 2.3: 64 vectors that form an orthonormal basis for $\mathbb{R}^{8\times 8}$

2.3 Jpeg (joint photographic experts group)

The JPEG-algorithm combines the cosine transform with some quantization step and the Huffman encoding.

At this point, suppose an image is a matrix $A \in \mathbb{R}^{M\times N}$ whose entries describe gray levels. We decompose A into blocks of size $n \times n$, for $n = 8$, see Figure 2.4.

Figure 2.4: The image is decomposed into patches of size 128×128 and 32×32. One actually uses patches of size 8×8.

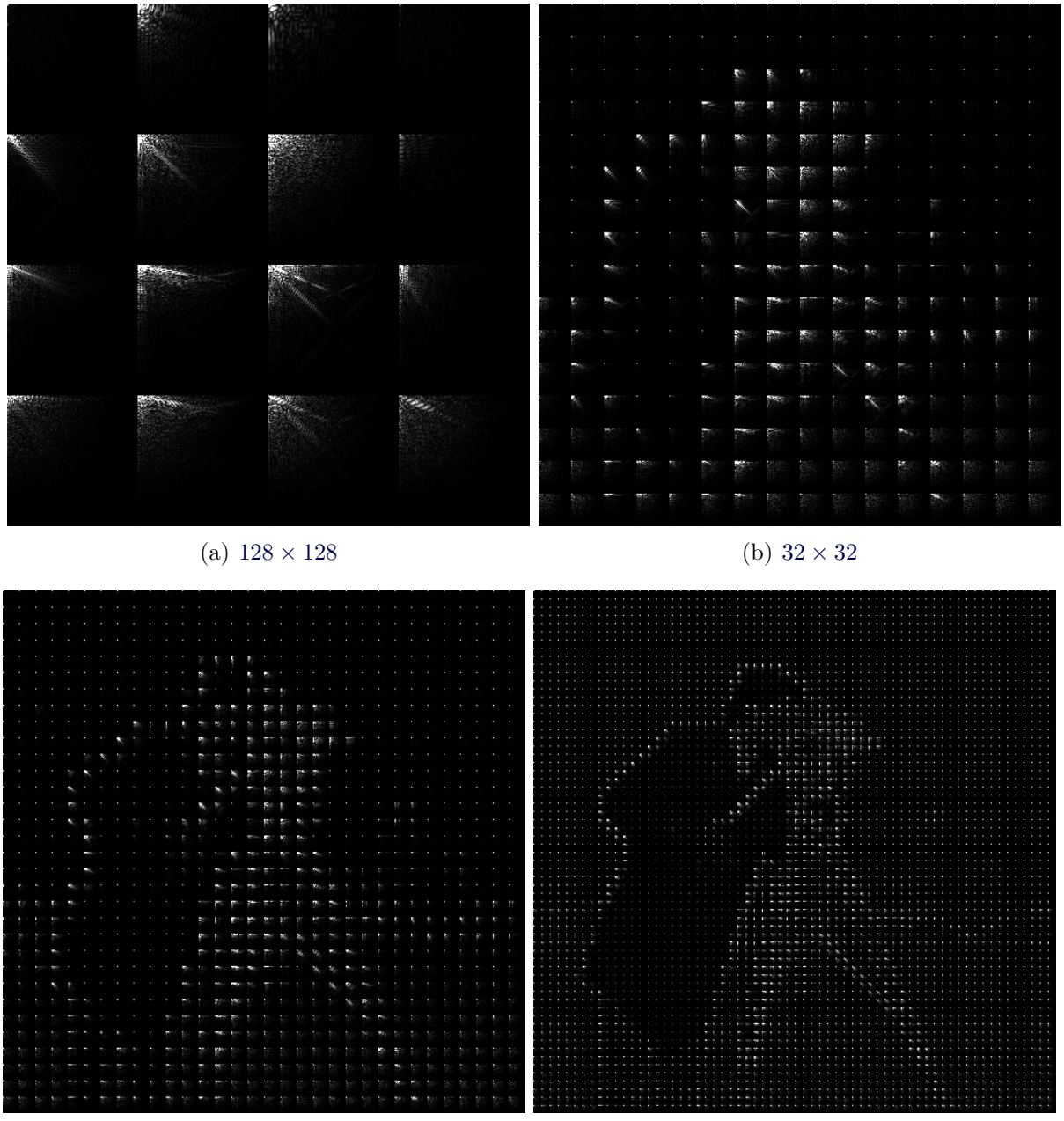
Thus, any matrix (submatrix) of size $n \times n$ can be written as a linear combination of $W_{k,l}$, for $k, l = 0 \ldots, n-1$,

$$\mathbb{R}^{n\times n} \ni B = \sum_{k,l=0}^{n-1} c_{k,l} W_{k,l}.$$

The coefficients

$$c_{k,l} = \langle B, W_{k,l}\rangle_{\mathbb{R}^{n\times n}} = \sum_{r,s=0}^{n} B(r,s) W_{k,l}(r,s)$$

are put into the matrix $C = (c_{k,l})_{k,l=0}^{n-1} \in \mathbb{R}^{n\times n}$. If we compute the coefficients $C^{(i,j)} \in \mathbb{R}^{n\times n}$ for each patch $B^{(i,j)}$, we derive the full coefficient matrix $C = \left(C^{(i,j)}\right)_{i,j}$ that has the same size as the original image, see Figure 2.5.



(a) 128×128 (b) 32×32

(c) 16×16 (d) 8×8

Figure 2.5: Coefficients are small except for the left upper corners in each subimage that correspond to low-pass. Right lower corner represent high-pass.

Since so many coefficients are close to zero, we put them to zero by using some threshold. We then reconstruct from the modified coefficient where now many are zero, see Figure 2.6.

(a) 60168 nonzero coefficients

(b) 16982 nonzero coefficients

(c) 7548 nonzero coefficients

(d) 3714 nonzero coefficients

Figure 2.6: There are 262144 many pixels, so that there are usually as many nonzero coefficients. We reconstruct from coefficient matrices that are thresholded, i.e., small values are put to zero

The jpeg-encoding uses the coefficient matrix and applies Huffman encoding to store the coefficients with minimal memory requirements. To do so, we first require to apply quantization, i.e., the coefficients are mapped onto a finite alphabet.

jpeg

For an 8-bit image $A \in \{0, \dots, 255\}^{N \times N}$, we do not threshold but apply the matrix

$$Q = (q_{k,l})_{k,l=0,\dots,n-1} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

entrywise to each coefficient patch and round towards the closest integer,

$$\tilde{c}_{k,l}^{(i,j)} = round\left(\frac{c_{k,l}^{(i,j)}}{q_{k,l}}\right).$$

The matrix $\tilde{C}^{(i,j)} = \left(\tilde{c}_{k,l}^{(i,j)}\right)_{k,l}$ contains the modified coefficients for patch (i, j). Huffman encoding is applied to the full matrix $\tilde{C} = \left(\tilde{C}^{(i,j)}\right)_{i,j}$. Since coefficients in the lower right corners are small, the division with Q and rounding puts them to zero.

Chapter 3

Image enhancement and feature extraction

Classical tasks for image enhancement are raising contrast, sharpening, and denoising. We also discuss in this section edge detection as a distinct feature extraction.

3.1 Contrast enhancement

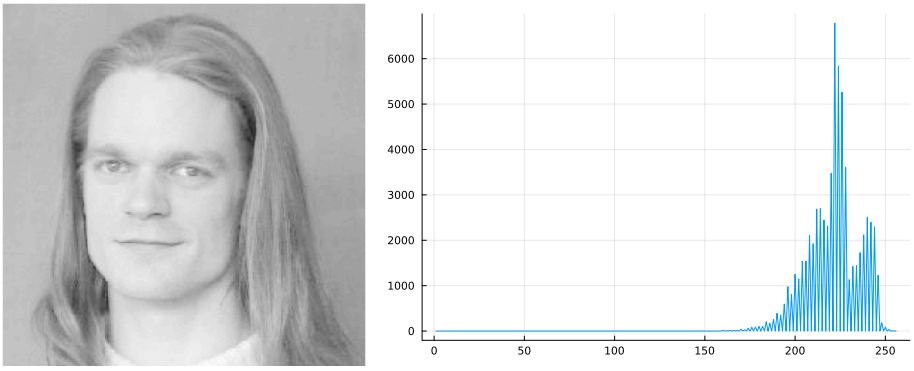
The image in Figure 3.1 has low contrast, which is reflected in its nonuniform histogram.

We enhance contrast in the image by histogram equalization.

Theorem 3.1.1

If $X : [0, a] \to \mathbb{R}_+$ is a random variable with a continuous, positive density g, then

$$Y := \int_0^X g(t)\mathrm{d}t$$



(a) low contrast image (b) histogram

Figure 3.1: Low contrast image with its histogram

is a random variable with density 1.

Proof. The function $S : [0, a] \to [0, 1]$ given by $S(x) = \int_0^x g(t)\mathrm{d}t$ is strictly increasing. Therefore it is invertible and its definition yields $0 = S(0) = S^{-1}(0)$. For $y \in [0, a]$, we observe

$$\begin{aligned}\mathbb{P}(0 \leq Y \leq y) &= \mathbb{P}(0 \leq S(X) \leq y) \\ &= \mathbb{P}(0 \leq X \leq S^{-1}(y)) \\ &= S(S^{-1}(y)) \\ &= y \\ &= \int_0^y 1\mathrm{d}t.\end{aligned}$$

□

Consider some probability space Ω and $\{r_1, \ldots, r_q\} \subset \mathbb{R}$ with $r_1 < \ldots < r_q$. Let $X : \Omega \to \{r_1, \ldots, r_q\}$ be a random variable, so that $X = r_k$ with probability $p_k > 0$, for $k = 1, \ldots, q$, and $\sum_{k=1}^q p_k = 1$.

We decompose Ω into $\Omega_k := X^{-1}(r_k)$, so that

$$\Omega = \bigcup_{k=1}^{q} \Omega_k.$$

In analogy to Theorem 3.1.1, we define

$$Y(\omega) = \sum_{l=1}^{k} p_l, \qquad \omega \in \Omega_k, \tag{3.1}$$

so that $Y(\omega) = p_1 + \ldots + p_k$, for $X(\omega) = r_k$.

Let us now consider an 8-bit image $f \in \ell_c(\mathbb{Z}^2)$, $f : \mathbb{Z}^2 \to \{0, r_1, \ldots, r_q\} \subset \{0, \ldots, 255\}$ with $0 < r_1 < \ldots < r_q$. Its histogram is derived from

$$N_k := \#\{(i,j) \in \mathbb{Z}^2 : f(i,j) = r_k, \qquad k = 1, \ldots, q.$$

For $N = \sum_{k=1}^{q} N_k$, the normalized histogram yields the probabilities

$$p_k := \frac{N_k}{N}$$

that provide the likelyhood that a pixel has the gray value r_k. For

$$\Omega_k = \{(i,j) \in \mathbb{Z}^2 : f(i,j) = r_k\},$$

according to (3.1), we define the contrast enhanced image F by

$$F(i,j) := \sum_{l=1}^{k} p_l, \qquad (i,j) \in \Omega_k,$$

see Figures 3.2 and 3.3.

An image is $f : \mathbb{Z}^d \to \mathbb{R}$ and a filter is $h : \mathbb{Z}^d \to \mathbb{R}$ such that h is only nonzero at finitely many points. We write $f \in \ell(\mathbb{Z}^d)$ and $h \in \ell_c(\mathbb{Z}^d)$. The convolution

$$f * h(n) = \sum_{k \in \mathbb{Z}^d} f(k) h(n-k)$$

is well-defined and $f * h \in \ell(\mathbb{Z}^d)$. Often, we may have $f \in \ell_c(\mathbb{Z}^d)$ as well.

3.2 Edge detection

We discuss two similar methods for detecting edges in an image, Prewitt and Sobel.

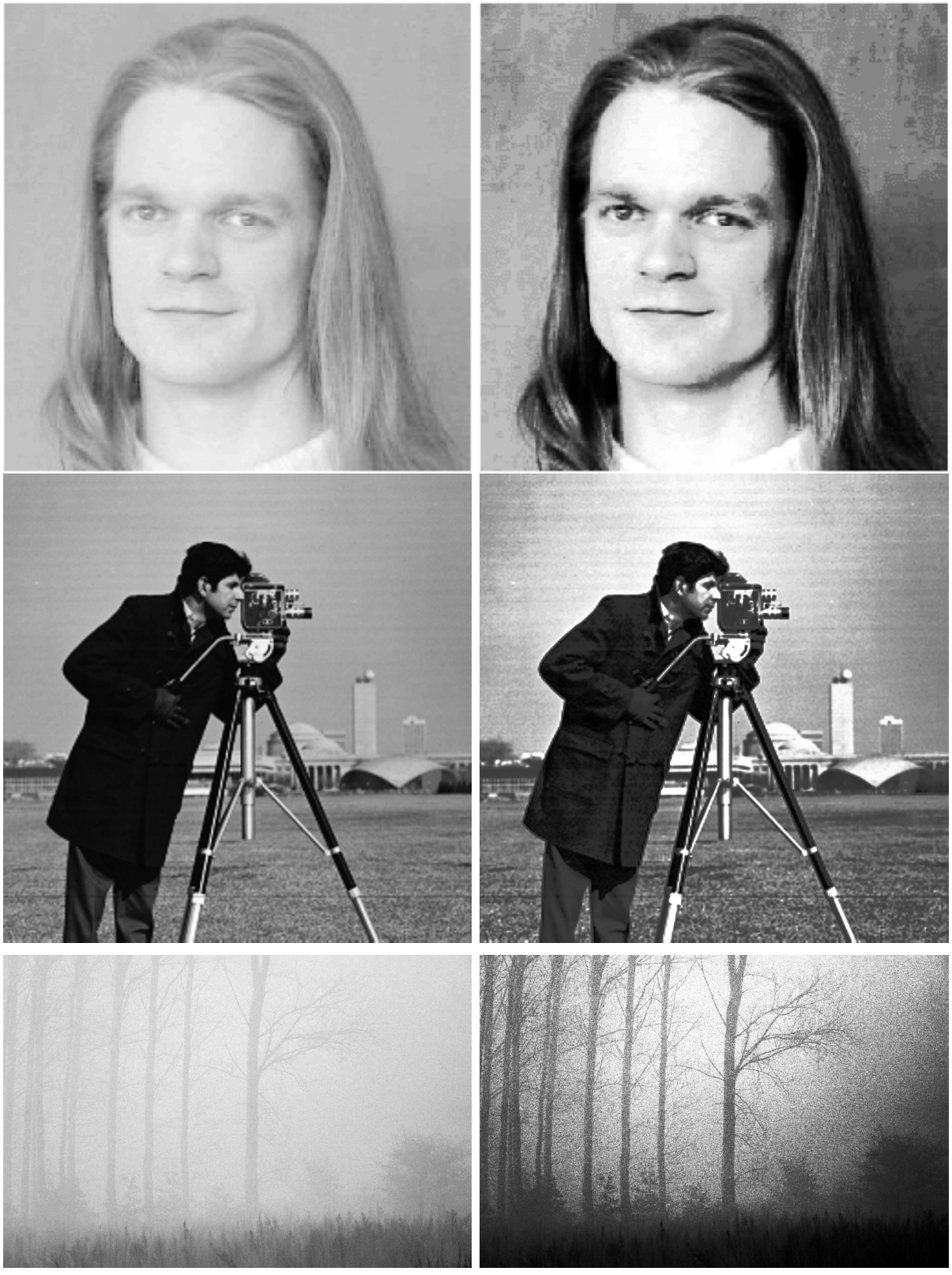
For a filter $h \in \ell_c(\mathbb{Z})$, we use the notion

$$h = (1, 2, -1)_{\mathbb{Z}} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}_{\mathbb{Z}} \quad :\Leftrightarrow \quad h(n) = \begin{cases} 1, & n = -1, \\ 2, & n = 0, \\ -1, & n = 1, \\ 0, & \text{otherwise}, \end{cases}$$

so that the 2-dimensional filter

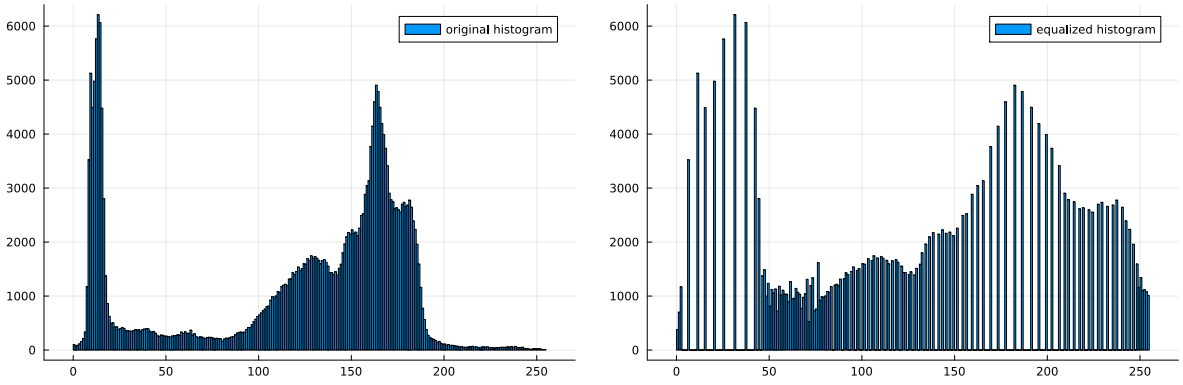
$$h = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}_{\mathbb{Z}^2}$$

is defined analogously.



(a) low contrast (b) histogram equalization

Figure 3.2: Contrast in the image is enhanced by histogram equalization



(a) original histogram (b) equalized histogram

Figure 3.3: Histogram equalization for the camera man. Large counts are moved away from each other and smaller counts are getting denser.

We first describe the Prewitt method. Consider the filters

$$h_x = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}_{\mathbb{Z}^2}, \qquad h_y = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}_{\mathbb{Z}^2}.$$

For $f \in \ell(\mathbb{Z}^2)$, we compute

$$f_x = f * h_x, \qquad f_y = f * h_y.$$

One picks up horizontal edges and the other vertical edges. Their combination

$$E_0 = \sqrt{f_x^2 + f_y^2} \in \ell(\mathbb{Z}^2)$$

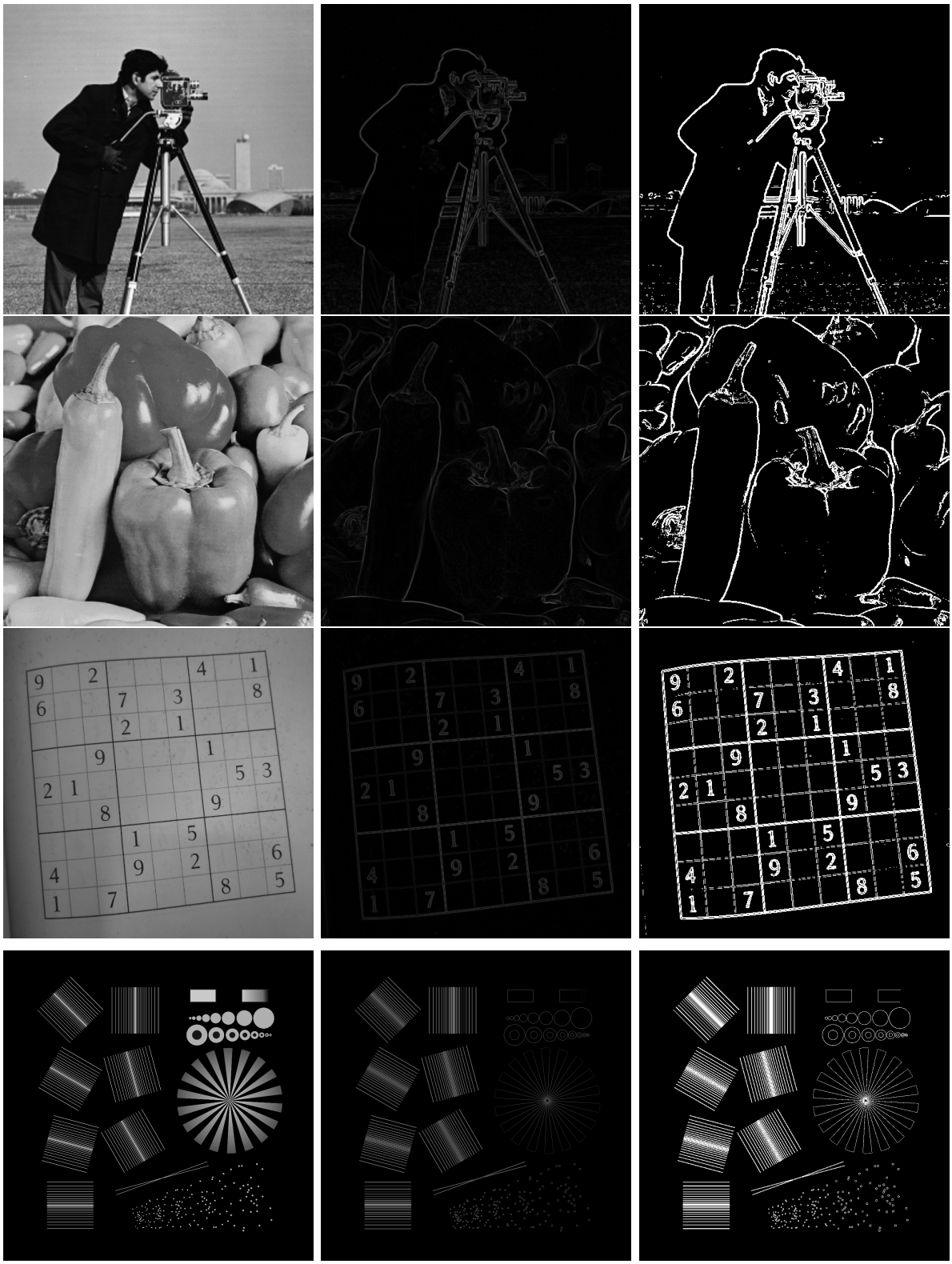
shows edges and its truncation with a parameters $s > 0$ yields the binary image

$$E(i,j) = \begin{cases} 1, & E(i,j) \geq s, \\ 0, & \text{otherwise.} \end{cases}$$

The Sobel method is identical to the Prewitt method but with the two filters

$$h_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}_{\mathbb{Z}^2}, \qquad h_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}_{\mathbb{Z}^2}.$$

See Figure 3.4 for the edge detection in images by the Sobel method.



(a) original f (b) $\sqrt{(f * h_x)^2 + (f * h_y)^2}$ (c) binary

Figure 3.4: Edge detection by the Sobel method. Images are derived from the Julia TestImages package.

3.3 Sharpening

If an image is somewhat blurry, then we may want to sharpen it. This section provides a simple tool for image sharpening.

First and second derivatives of a function f are often approximated by

$$f'(x) \approx \frac{f(x+\epsilon) - f(x)}{\epsilon}, \qquad f''(x) \approx \frac{f'(x) - f'(x-\epsilon)}{\epsilon}.$$

For the second derivative, this leads to

$$\begin{aligned} f''(x) &\approx \frac{\frac{f(x+\epsilon)-f(x)}{\epsilon} - \frac{f(x)-f(x-\epsilon)}{\epsilon}}{\epsilon} \\ &= \frac{f(x+\epsilon) - 2f(x) + f(x-\epsilon)}{\epsilon^2}. \end{aligned}$$

For $\epsilon = 1$, we obtain

$$f''(x) \approx f(x+1) - 2f(x) + f(x-1).$$

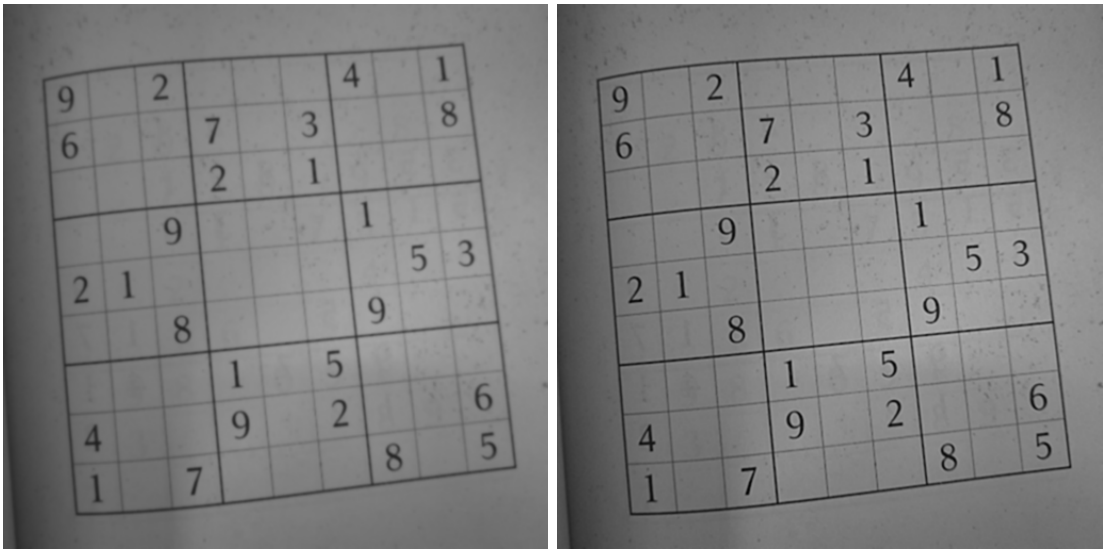
To sharpen an image f, we subtract the Laplacian, i.e., for the discrete Laplacian

$$L = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}_{\mathbb{Z}^2},$$

we derive a sharper image by

$$f - f * L = f * \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}_{\mathbb{Z}^2},$$

see Figure 3.5.



(a) unsharp (b) sharp

Figure 3.5: Image sharpening via $f - L * f$

3.4 Denoising

Images are often distorted and there are many different sorts of image distortions. Suppose that a few (or many) isolated pixels are lost. We assign them either black or white in a gray scale image, see Figure 3.6(a). This is also called Salt&Pepper noise.

To remove the noise, we apply a Median filter. The terminology is misleading, it is not a proper filter in the sense that it is not a convolution. Instead, we move a window of size s, i.e., a square of odd side length s over the image. The center of the pixel is now replaced by the median of its surrounding pixels (including itself).

(a) Image corrupted by Salt&Pepper noise

(b) Denoised by Median filter of length $s = 3$

(c) Denoised by Median filter of length $s = 5$

Figure 3.6: Image denoising by Median filter of length s

Chapter

4 Dimension reduction and linear separation

High-dimensional data is common, but visualization, clustering, and analysis in general are often difficult. We first try to reduce the dimension of data by identifying a distinct projection onto a lower dimensional subspace. In this regard, we discuss principal component analysis (PCA) and variations of it.

4.1 Singular value decomposition

If matrices have eigenvalues that are almost zero, then their inverse behaves bad.

Example 1
You only observe a local weighted average y of a 1-dimensional signal $x \in \mathbb{R}^n$ for large n, say

$$y = Ax, \qquad A = \frac{1}{4}\begin{pmatrix} 2 & 1 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & & \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 & 2 \end{pmatrix}. \tag{4.1}$$

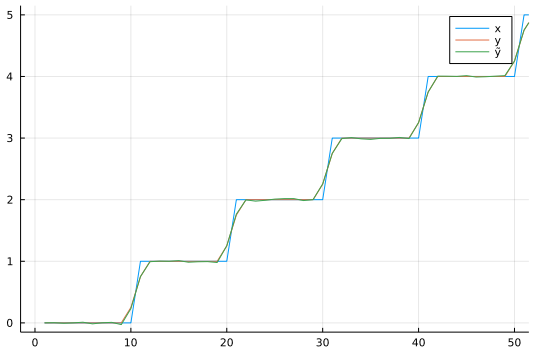
Often y is distorted by additive noise, i.e.,

$$\tilde{y} = y + \varepsilon.$$

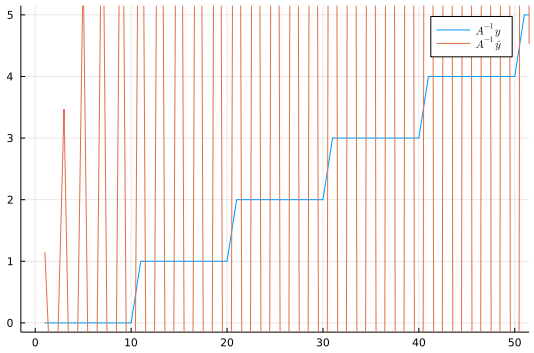
Even if $\varepsilon \in \mathbb{R}^n$ is so small that $\tilde{y}$ and y are difficult to distinguish, see part (a) of Figure 4.1, the consequences for the reconstruction can be fatal, i.e., $A^{-1}\tilde{y}$ is a horrible approximation to x, see part (b) of Figure 4.1.

Example 2
Smoothening of "image" $X \in \mathbb{R}^{n\times n}$ by the moving average

$$M_3 = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$



(a) The smoothened signal y and its distorted version $\tilde{y}$ are barely distinguishable.



(b) The reconstruction $A^{-1}\tilde{y}$ is useless.

Figure 4.1: Only the first 50 entries of $x \in \mathbb{R}^n$ with $n = 1000$ are shown.

(a) X (b) Y (c) $\tilde{Y}$

Figure 4.2: Image X, smoothed image Y, and perturbation $\tilde{Y}$

yields the smoothened image $Y = AXA$, where A is as in (4.1), see Figure 4.2. The perturbation by independent Gaussian noise $\varepsilon \in \mathbb{R}^{n\times n}$ with standard variation 10^{-6} is

$$\tilde{Y} = Y + \varepsilon.$$

We select ε so small, that Y and $\tilde{Y}$ are visually indistinguishable, see Figure 4.2.

The reconstructions are $X = A^{-1}YA^{-1}$ and $\tilde{X} = A^{-1}\tilde{Y}A^{-1}$, cf. Figure 4.3. Obviously, $\tilde{X}$ is useless.

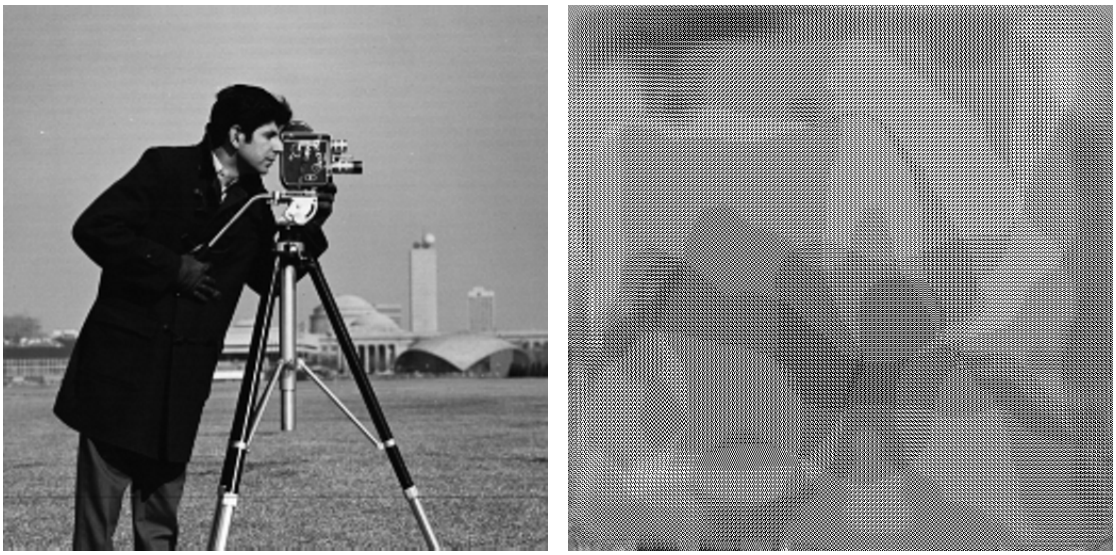
Turns out that it is beneficial to replace the actual inverse by some pseudo-inverse of a low-rank approximation. One step in this direction is the singular value decomposition.

Theorem 4.1.1: (reduced svd)

Let $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = s$.

- The matrix $A^\top A \in \mathbb{R}^{n\times n}$ has eigenvalues $\lambda_1 \geq \ldots \geq \lambda_s > 0$ with pairwise orthonormal eigenvectors $v_1, \ldots, v_s \in \mathbb{R}^n$. Define

$$V := (v_1, \ldots, v_s) \in \mathbb{R}^{n\times s}.$$



(a) $X = A^{-1}YA^{-1}$ (b) $\tilde{X} = A^{-1}\tilde{Y}A^{-1}$

Figure 4.3: Reconstruction from Y and $\tilde{Y}$

- The *singular values* σ_k, for $k = 1, \ldots, s$, are

$$\sigma_k := \sqrt{\lambda_k}, \qquad \Sigma := \operatorname{diag}(\sigma_1, \ldots, \sigma_s) \in \mathbb{R}^{s \times p}.$$

- Define $u_k := \frac{1}{\sigma_k} A v_k$, for $k = 1, \ldots, s$, and

$$U := (u_1, \ldots, u_s) \in \mathbb{R}^{m \times s}.$$

Then the svd

$$A = U \Sigma V^\top$$

holds.

Proof. For $k = 1, \ldots, s$, we derive

$$U\Sigma V^\top v_k = U\Sigma e_k = U\sigma_k e_k = \sigma_k u_k = A v_k. \tag{4.2}$$

Extend $v_1, \ldots, v_s$ to an orthonormal basis $v_1, \ldots, v_n$ of $\mathbb{R}^n$. Obviously, we have

$$U\Sigma V^\top v_{s+1}, \ldots, U\Sigma V^\top v_n = 0, \qquad A^\top A v_{s+1}, \ldots A^\top A v_n = 0.$$

Since $A^\top A x = 0$ implies

$$0 = \langle x, A^\top A x \rangle = \langle Ax, Ax \rangle = \|Ax\|_2^2,$$

we deduce $A v_{s+1}, \ldots, A v_n = 0$, so that (4.2) also holds for $k = s+1, \ldots, n$. □

Example 4.1.2: (svd)

Consider the matrix $A = \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 1 \end{pmatrix}$, so that $A^\top A = \begin{pmatrix} 5 & 2 \\ 2 & 2 \end{pmatrix}$. Its charateristic polynomial is

$$f(\lambda) = (5-\lambda)(2-\lambda) - 4 = \lambda^2 - 7\lambda + 6 = (\lambda - 6)(\lambda - 1),$$

so that $p = 2$ with $\lambda_1 = 6$ and $\lambda_2 = 1$. The corresponding normalized eigenvectors v_1 and v_2 are determined by

$$0 = (A^\top A - \lambda_1 I_2)v_1 = \begin{pmatrix} -1 & 2 \\ 2 & -4 \end{pmatrix} v_1, \qquad v_1 = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$
$$0 = (A^\top A - \lambda_2 I_2)v_2 = \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} v_2, \qquad v_2 = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 \\ -2 \end{pmatrix},$$

so that we obtain

$$V = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \sqrt{6} & 0 \\ 0 & 1 \end{pmatrix}, \qquad \sigma_1 = \sqrt{6}, \quad \sigma_2 = 1.$$

We also obtain

$$u_1 = \frac{1}{\sigma_1} A v_1 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 1 \end{pmatrix} \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{30}} \begin{pmatrix} 2 \\ 5 \\ 1 \end{pmatrix},$$
$$u_2 = \frac{1}{\sigma_2} A v_2 = \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 1 \end{pmatrix} \frac{1}{\sqrt{5}} \begin{pmatrix} 1 \\ -2 \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix},$$

so that we have

$$U = \begin{pmatrix} \sqrt{\frac{2}{15}} & \frac{1}{\sqrt{5}} \\ \sqrt{\frac{5}{6}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{-2}{\sqrt{5}} \end{pmatrix}, \qquad A = U\Sigma V^\top.$$

Definition 4.1.3

Let $A \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(A) = s$. The matrix

$$A^\# := V\Sigma^{-1}U^\top \in \mathbb{R}^{n \times m}$$

is called *pseudo-inverse* of A.

Lemma 4.1.4

The svd yields

a) $u_1, \dots, u_s$ are orthonormal, hence, $U^\top U = I_s$.

b) $AA^\#$ is an orthogonal projector of rank s.

c) if $\operatorname{rank}(A) = n$, then $A^\# = (A^\top A)^{-1} A^\top$ with $A^\# A = I_n$.

Proof. a) We observe

$$\langle u_k, u_l \rangle = \frac{1}{\sigma_k \sigma_l} \langle A v_k, A v_l \rangle = \frac{1}{\sigma_k \sigma_l} \langle A^\top A v_k, v_l \rangle = \frac{\lambda_k}{\sigma_k \sigma_l} \langle v_k, v_l \rangle = \delta_{k,l}.$$

b) We also obtain

$$AA^\# = U \Sigma V^\top V \Sigma^{-1} U^\top = U U^\top,$$

so that $AA^\#$ is symmetric with rank s. It also satisfies

$$AA^\# AA^\# = U U^\top U U^\top = U U^\top = AA^\#,$$

so that it is an orthogonal projector.
c) If $\operatorname{rank}(A) = n$, then $s = n$ and V is orthogonal, so that we have

$$(A^\top A)^{-1} A^\top = (V \Sigma \underbrace{U^\top U}_{I_n} \Sigma V^\top)^{-1} V \Sigma U^\top = V \Sigma^{-2} V^\top V \Sigma U^\top = V \Sigma^{-1} U^\top.$$

Moreover, we have

$$A^\# A = V \Sigma^{-1} U^\top U \Sigma V^\top = V V^\top = I_n.$$

□

Example 4.1.5

If $A \in \mathbb{R}^{n \times n}$ is invertible, then $s = n$ and $A^\# \in \mathbb{R}^{n \times n}$. Part c) and d) of Lemma 4.1.4 lead to $AA^\# = A^\# A = I_n$, so that $A^\# = A^{-1}$ must hold.

Example 4.1.2 has shown

$$V = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \sqrt{6} & 0 \\ 0 & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} \sqrt{\frac{2}{15}} & \frac{1}{\sqrt{5}} \\ \sqrt{\frac{5}{6}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{-2}{\sqrt{5}} \end{pmatrix},$$

so that the pseudo inverse is

$$A^\# = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{6}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \sqrt{\frac{2}{15}} & \sqrt{\frac{5}{6}} & \frac{1}{\sqrt{30}} \\ \frac{1}{\sqrt{5}} & 0 & \frac{-2}{\sqrt{5}} \end{pmatrix} = \dots = \frac{1}{6} \begin{pmatrix} 2 & 2 & -2 \\ -2 & 1 & 5 \end{pmatrix}.$$

Example 4.1.6: (pseudo inverse of low-rank approximation)

If $A \in \mathbb{R}^{n\times n}$ is invertible but has few tiny eigenvalues, then we may approximate A by its low-rank approximation A_p via svd, i.e.,

$$A = U \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \sigma_n \end{pmatrix} V^\top, \qquad A_p = U \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_p & & \\ & & & 0 & \\ & & & & \ddots \end{pmatrix} V^\top,$$

where $1 \le p \le n$. The matrix A_p is better conditioned than A. The pseudo-inverse $A_p^\#$ replaces A^{-1}. For $y = Ax$ and $\tilde{y} = y + \varepsilon$, we obtain

$$\begin{aligned} \|x - A_p^\# \tilde{y}\| &= \|A^{-1}y - A_p^\# y - A_p^\# \varepsilon\| \\ &\le \underbrace{\|A^{-1}y - A_p^\# y\|}_{\text{small for large } p} + \underbrace{\|A_p^\# \varepsilon\|}_{\text{small for small } p} . \end{aligned}$$

We usually find some p, such that $A_p^\#$ works much better than A^{-1}, see Figures 4.4 and 4.5.

4.2 Principal component analysis

Assume the data $x_1, \dots, x_n \in \mathbb{R}^d$ have zero mean $\bar{x} = \frac{1}{n}\sum_{j=1}^n x_j = 0$. Otherwise, switch to $x_i - \bar{x}$, for $i = 1, \dots, n$.

According to the zero mean, the variance of X is

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{j=1}^n \|x_j - \bar{x}\|^2 = \frac{1}{n}\sum_{j=1}^n \|x_j\|^2.$$

For fixed $s < d$, consider a matrix $V \in \mathbb{R}^{d\times s}$ with orthonormal columns. The dimension reduced vectors

$$V^\top x_1, \dots, V^\top x_n \in \mathbb{R}^s$$

still have zero mean. Therefore, their variance is

$$\mathrm{Var}(V^\top X) = \frac{1}{n}\sum_{i=1}^n \|V^\top x_i\|^2. \tag{4.3}$$

Principal component analysis identifies V that maximizes (4.3) among all matrices in $\mathbb{R}^{d\times s}$ with orthonormal columns. To understand the procedure, we consider the covariance matrix

$$C = \frac{1}{n}\sum_{j=1}^n x_j x_j^\top = \frac{1}{n} X X^\top \in \mathbb{R}^{d\times d}.$$

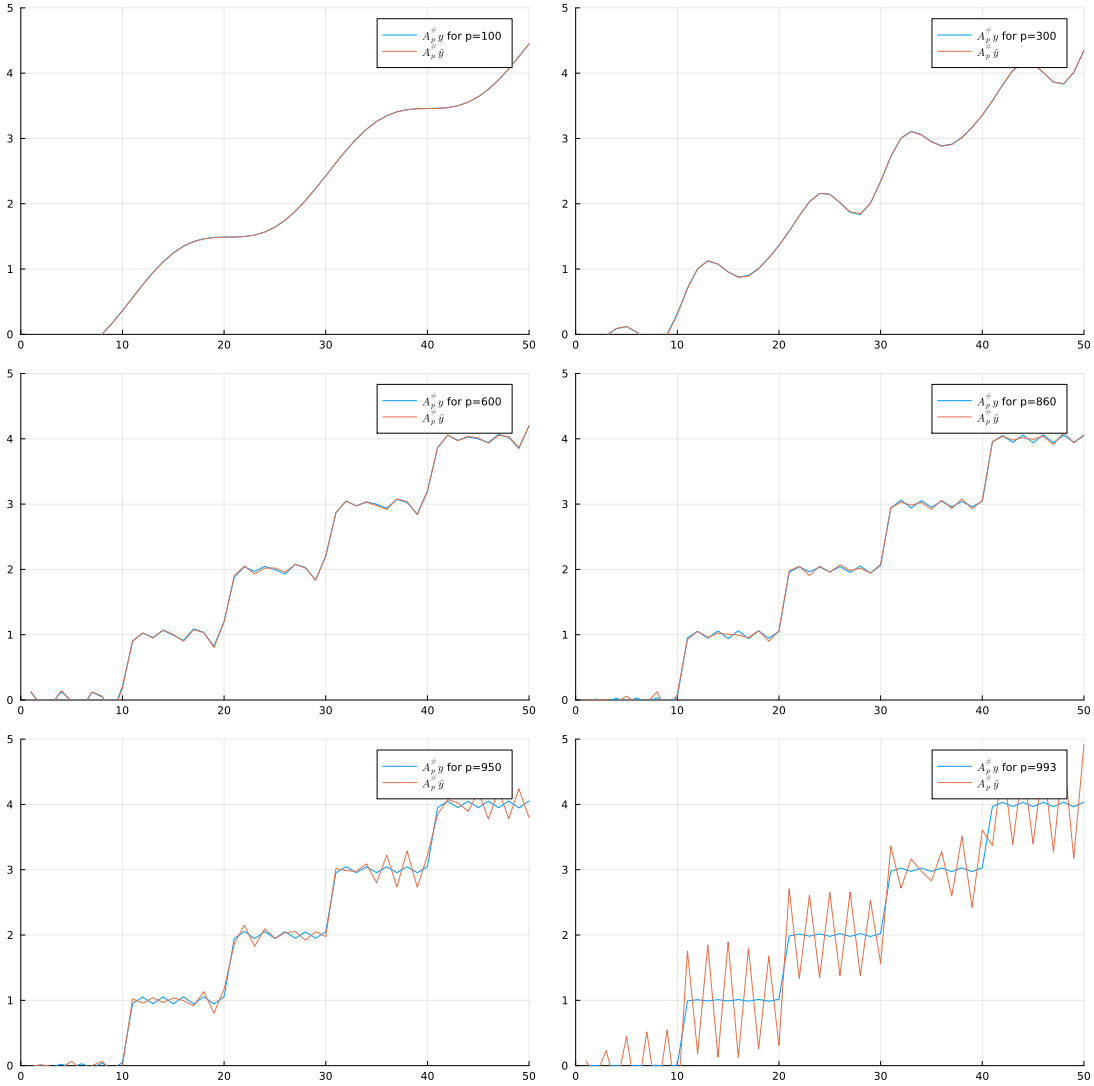


Figure 4.4: For $y = Ax$ and $\tilde{y} = y + \varepsilon$, the pseudo inverse $A_p^{\#}$ of the low-rank approximation A_p of A replaces A^{-1}. The distortion does not play any role for $1 < p < 860$. For larger p, the approximation A_p gets closer to A, but the distortions eventually take over.

(a) p=31 (b) p=100

(c) p=490 (d) p=504

(e) p=506 (f) p=508

Figure 4.5: For $y = AxA$ and $\tilde{y} = y + \varepsilon$, the pseudo inverse $A_p^{\#}$ of the low-rank approximation A_p of A replaces A^{-1}. The distortion does not play any role for $1 < p < 490$. For larger p, the approximation A_p gets closer to A, but the distortions eventually take over.

Direct computations yield

$$\begin{aligned}\mathrm{Var}(V^\top X) &= \frac{1}{n}\sum_{i=1}^{n}\langle V^\top x_i, V^\top x_i\rangle \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathrm{trace}(x_i^\top V V^\top x_i) \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathrm{trace}(x_i x_i^\top V V^\top) \\ &= \mathrm{trace}(CVV^\top).\end{aligned}$$

The requirement of orthonormal columns is equivalent to $V^\top V = I_s$. The matrix $VV^\top \in \mathbb{R}^{d\times d}$ is an orthogonal projector onto an s-dimensional subspace of $\mathbb{R}^d$.

Lemma 4.2.1: (Lemma of Fan)

Let $C \in \mathbb{R}^{d\times d}$ be symmetric and positive semi-definite with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq 0$ and associated orthonormal eigenvectors $v_1, \ldots, v_d$. Then

$$\max_{\substack{V\in\mathbb{R}^{d\times s} \text{ with} \\ \text{orthonormal columns}}} \mathrm{trace}(CVV^\top) = \lambda_1 + \ldots + \lambda_s$$

and the maximum is taken for $V = (v_1, \ldots, v_s)$.

Proof. Since C is symmetric, it can be diagonalized into $C = VDV^\top$ and D is diagonal with decreasing eigenvalues on the diagonal. So far, the matrix $V = (v_1, \ldots, v_d)$ contains the eigenvectors of C. For $Q = VV^\top$, we compute

$$\mathrm{trace}(CQ) = \mathrm{trace}(VDV^\top Q) = \mathrm{trace}(DV^\top QV) = \mathrm{trace}(D) = \lambda_1 + \ldots + \lambda_d.$$

□

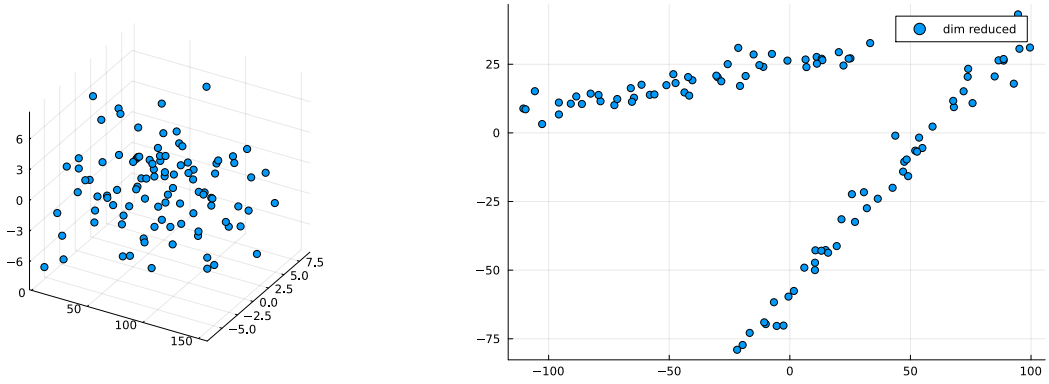


Figure 4.6: (left) Data that consists of 2 distinct groups that are not visible. (right) PCA with $s = 2$ enables identification of two groups that are linearly separable.

Principal Component Analysis (PCA)

1. Input: data $X = (x_1, \ldots, x_n) \in \mathbb{R}^{d \times n}$ with $\bar{x} = 0$.
2. covariance matrix
$$C = \frac{1}{n} \sum_{j=1}^{n} x_j x_j^\top = \frac{1}{n} X X^\top \in \mathbb{R}^{d \times d}$$
3. find eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots$ and eigenvectors $v_1, v_2, \ldots$
$$\lambda_k v_k = C v_k.$$
4. k-th principal component:
$$\left(v_k^\top x_1, \ldots, v_k^\top x_n\right) = v_k^\top X.$$
5. dimension reduced data: $V = (v_1, \ldots, v_s)$ and
$$Z = V^\top X \in \mathbb{R}^{s \times n}.$$

The matrix V is either computed from SVD of $X^\top$ or from SVD applied to C.

Theorem 4.2.2: (approximation vs. variance

Assume that $\bar{x} = 0$. Any maximizer V of the variance (4.3)

$$\max_{\substack{V \in \mathbb{R}^{d\times s} \text{ with} \\ \text{orthonormal columns}}} \operatorname{Var}(V^\top X)$$

is a minimizer of the approximation error

$$\min_{\substack{V \in \mathbb{R}^{d\times s} \text{ with} \\ \text{orthonormal columns}}} \sum_{i=1}^{n} \|x_i - VV^\top x_i\|^2.$$

and vice-versa.

Proof. We compute

$$\sum_{i=1}^{n} \|x_i - VV^\top x_i\|^2 = \sum_{i=1}^{n} \|x_i\|^2 - \frac{2}{n}\sum_{i=1}^{n}\langle x_i, VV^\top x_i\rangle + \frac{1}{n}\sum_{i=1}^{n}\|VV^\top x_i\|^2.$$

The term $\sum_{i=1}^{n} \|x_i\|^2$ does not depend on V. According to

$$\|VV^\top x_i\|^2 = \langle VV^\top x_i, VV^\top x_i\rangle = \langle x_i, VV^\top VV^\top x_i\rangle = \langle x_i, VV^\top x_i\rangle,$$

we further derive

$$\begin{aligned} -\frac{1}{n}\sum_{i=1}^{n}\langle x_i, VV^\top x_i\rangle &= -\frac{1}{n}\sum_{i=1}^{n}\operatorname{trace}(x_i^\top VV^\top x_i) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\operatorname{trace}(x_i x_i^\top VV^\top) \\ &= -\operatorname{trace}(CVV^\top). \end{aligned}$$

□

We have verified

$$\sum_{i=1}^{n} \|x_i - VV^\top x_i\|^2 = \sum_{i=1}^{n} \|x_i\|^2 - \operatorname{trace}(CVV^\top) = \sum_{i=1}^{n} \|x_i\|^2 - \operatorname{Var}(V^\top X).$$

It may happen that the principal components do not allow for linear separation, see Figure 4.7.

4.3 Kernel PCA

To circumvent the intrinsic limitations of a linear method, we consider kernel-PCA. This will allow the preservation of nonlinear features within a linear method combined with kernels.

Choose $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ to embed the data $x_1, \ldots, x_n \in \mathbb{R}^d$ into

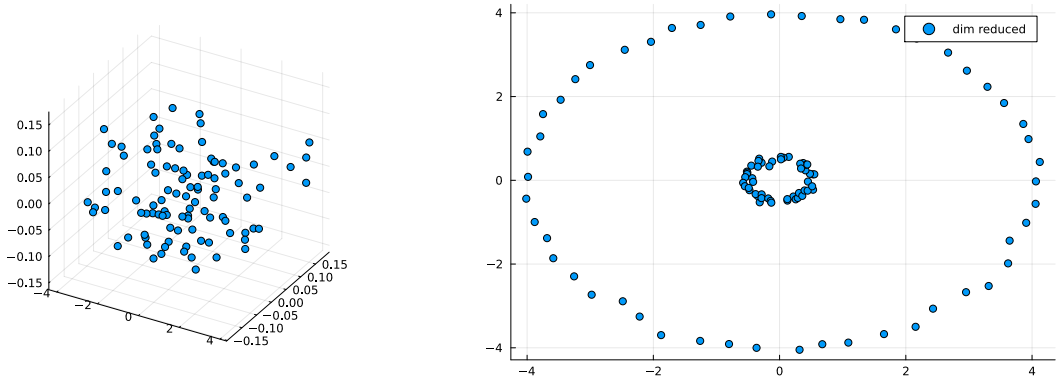


Figure 4.7: (left) Data that consists of 2 distinct groups that are not visible. (right) PCA with $s = 2$ still shows two groups but they are not linearly separable.

$$\phi_i = \Phi(x_i) \in \mathbb{R}^D, \qquad i = 1, \dots, n.$$

Apply PCA to $\phi_1, \dots, \phi_n \in \mathbb{R}^D$.

Define the kernel $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$ and the matrix

$$K = (\langle \phi_i, \phi_j \rangle)_{i,j=1,\dots,n} = (k(x_i, x_j))_{i,j=1,\dots,n} \in \mathbb{R}^{n \times n}$$

Theorem 4.3.1: (eigenvectors)

For $C = \frac{1}{n} \sum_{i=1}^n \phi_i \phi_i^\top$, we observe

$$n\lambda\alpha = K\alpha, \quad v = \sum_{j=1}^n \alpha_j \phi_j \qquad \Rightarrow \qquad \lambda v = Cv.$$

Proof. Since $v \in span\{\phi_1, \dots, \phi_n\}$, we have

$$\lambda v = Cv \qquad \Leftrightarrow \qquad \phi_k^\top \lambda v = \phi_k^\top C v, \quad k = 1, \dots, n.$$

Further computations lead to

$$\phi_k^\top \lambda v = \lambda \sum_{j=1}^n \alpha_j \phi_k^\top \phi_j = \lambda \sum_{j=1}^n \alpha_j K_{k,j} = \lambda (K\alpha)_k.$$

For the term $\phi_k^\top Cv$, we have

$$\begin{aligned}\phi_k^\top Cv &= \phi_k^\top \frac{1}{n}\sum_{i=1}^n \phi_i\phi_i^\top \sum_{j=1}^n \alpha_j\phi_j \\ &= \sum_{j=1}^n \alpha_j \frac{1}{n}\sum_{i=1}^n \phi_k^\top\phi_i\phi_i^\top\phi_j \\ &= \sum_{j=1}^n \alpha_j \frac{1}{n}\sum_{i=1}^n K_{k,i}K_{i,j} \\ &= \frac{1}{n}\sum_{j=1}^n \alpha_j (K^2)_{k,j} \\ &= \frac{1}{n}(K^2\alpha)_k.\end{aligned}$$

We have verified that $\phi_k^\top \lambda v = \phi_k^\top Cv$ is equivalent to $n\lambda K\alpha = K^2\alpha$. If α solves

$$n\lambda\alpha = K\alpha,$$

then it also satisfies $n\lambda K\alpha = K^2\alpha$. □

Theorem 4.3.2: (dimension reduction)

Choose α such that $\|\alpha\|^2 = \frac{1}{n\lambda}$. Then one principal component is

$$\left(v^\top\phi_1, \ldots, v^\top\phi_n\right) = \alpha^\top K.$$

Proof. The requirement $\|v\|^2 = 1$ yields

$$1 = \langle v, v\rangle = \sum_{i,j}\alpha_i\alpha_j\langle\phi_i,\phi_j\rangle = \sum_{i,j}\alpha_i\alpha_j K_{i,j} = \alpha^\top K\alpha = n\lambda\|\alpha\|^2.$$

Compute

$$v^\top(\phi_1,\ldots,\phi_n) = \sum_{j=1}^n \alpha_j\phi_j^\top(\phi_1,\ldots,\phi_n) = \sum_{j=1}^n \alpha_j\left(K_{j,1},\ldots,K_{j,n}\right).$$

□

In order to apply PCA, we still need to subtract the mean $\bar\phi = \frac{1}{n}\sum_{j=1}^n \phi_j$, so that we need to deal with the data

$$\underbrace{\phi_1 - \bar\phi}_{\tilde\phi_1}, \ldots\ldots, \underbrace{\phi_n - \bar\phi}_{\tilde\phi_n}.$$

Theorem 4.3.3: (kernel for mean zero)

The kernel matrix $\tilde{K}$ with $\tilde{K}_{i,j} = \langle \tilde{\phi}_i, \tilde{\phi}_j \rangle$ satisfies

$$\tilde{K} = K - 1_n K - K 1_n + 1_n K 1_n$$

and 1_n is $\frac{1}{n}$ times an $n \times n$-matrix filled with 1s.

Proof. We compute

$$\tilde{K}_{i,j} = \langle \phi_i - \bar{\phi}, \phi_j - \bar{\phi} \rangle = \langle \phi_i, \phi_j \rangle - \langle \phi_i, \bar{\phi} \rangle - \langle \bar{\phi}, \phi_j \rangle + \langle \bar{\phi}, \bar{\phi} \rangle$$

This leads to

$$\begin{aligned} \tilde{K}_{i,j} &= K_{i,j} - \frac{1}{n} \sum_l K_{i,l} - \frac{1}{n} \sum_l K_{l,j} + \frac{1}{n^2} \sum_{l,m} K_{l,m} \\ &= K_{i,j} - \frac{1}{n} (K(1, \ldots, 1)^\top)_i - \frac{1}{n} ((1, \ldots, 1)K)_j + \frac{1}{n} (1, \ldots, 1) K \frac{1}{n} (1, \ldots, 1)^\top \end{aligned}$$

□

Kernel-PCA

1. Input: data $X = (x_1, \dots, x_n) \in \mathbb{R}^{d \times n}$ with $\bar{x} = 0$ and kernel k.
2. kernel matrix $K = (k(x_i, x_j))_{i,j=1,\dots,n}$
3. zero mean kernel
$$\tilde{K} = K - 1_n K - K 1_n + 1_n K 1_n$$
4. find $\lambda_1 \geq \lambda_2 \geq \dots$ and eigenvectors $\alpha_1, \alpha_2, \dots$
$$n\lambda_k \alpha_k = K\alpha_k.$$
5. renormalize, so that $\|\alpha_k\|^2 = \frac{1}{n\lambda_k}$
6. k-th principal component:
$$\alpha_k^\top K.$$
7. dimension reduced data: $A = (\alpha_1, \dots, \alpha_s)$ and
$$Z = A^\top K \in \mathbb{R}^{s \times n}.$$

It turns out that we may forget about Φ altogether and simply start with a kernel $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$.

Definition 4.3.4

The function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{K}$ is called a positive definite kernel if $k(x, y) = \overline{k(y, x)}$, for all $x, y \in \mathbb{X}$, and, for any $n \in \mathbb{N}$, any $(x_j)_{j=1}^n \subseteq \mathbb{X}$, and any $(c_j)_{j=1}^n \subseteq \mathbb{K}$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i \bar{c}_j k(x_i, x_j) \geq 0. \tag{4.4}$$

The condition (4.4) means that the matrix $K = (k(x_i, x_j))_{i,j=1,\dots,n} \in \mathbb{K}^{n \times n}$ is positive semi-definite.

The following result clarifies the existence of Φ when we start with k.

Proposition 4.3.5: Feature map

For $k : \mathbb{X} \times \mathbb{X} \to \mathbb{K}$, the following are equivalent:

(i) k is a positive definite kernel,

(ii) $\exists$ pre-Hilbert space H and $\Phi : \mathbb{X} \to H$ such that

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle_H, \qquad x, y \in \mathbb{X}.$$

Proof. (ii)⇒(i): Simple computations yield:

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle_H = \overline{\langle \Phi(y), \Phi(x) \rangle_H} = \overline{K(y, x)}.$$

For $(c_j)_{j=1}^n \subset \mathbb{K}$ and $(x_j)_{j=1}^n \subset X$, we have

$$\begin{aligned} \sum_{i,j=1}^n c_i \overline{c_j} k(x_i, x_j) &= \sum_{i,j=1}^n c_i \overline{c_j} \langle \Phi(x_i), \Phi(x_j) \rangle_H \\ &= \left\langle \sum_{i=1}^n c_i \Phi(x_i), \sum_{j=1}^n c_j \Phi(x_j) \right\rangle_H \\ &= \left\| \sum_{i=1}^n c_i \Phi(x_i) \right\|_H^2 \geq 0. \end{aligned}$$

(i)⇒(ii): Define $k_x : \mathbb{X} \to \mathbb{K}$ such that $k_x(y) = k(x, y)$ and $H := \operatorname{span}\{k_x : x \in \mathbb{X}\}$ and

$$\left\langle \sum_{i=1}^n a_i k_{x_i}, \sum_{j=1}^m b_j k_{y_j} \right\rangle_H := \sum_{i=1}^n \sum_{j=1}^m a_i \overline{b_j} k(x_i, y_j). \tag{4.5}$$

Well-defined?!?: For $\sum_{i=1}^n a_i k_{x_i} = \sum_{k=1}^{\tilde{n}} \tilde{a}_k k_{\tilde{x}_k}$ and $\sum_{j=1}^m b_j k_{y_j} = \sum_{l=1}^{\tilde{m}} \tilde{b}_l k_{\tilde{y}_l}$, we have

$$\begin{aligned} \sum_{i=1}^n \sum_{j=1}^m a_i \overline{b_j} k(x_i, y_j) &= \sum_{i=1}^n a_i \overline{\sum_{j=1}^m b_j k(y_j, x_i)} = \sum_{i=1}^n a_i \overline{\sum_{j=1}^m b_j K_{y_j}(x_i)} \\ &= \sum_{i=1}^n a_i \overline{\sum_{l=1}^{\tilde{m}} \tilde{b}_l k_{\tilde{y}_l}(x_i)} = \sum_{l=1}^{\tilde{m}} \overline{\tilde{b}_l} \sum_{i=1}^n a_i k_{x_i}(\tilde{y}_l) \\ &= \sum_{l=1}^{\tilde{m}} \overline{\tilde{b}_l} \sum_{k=1}^{\tilde{n}} \tilde{a}_k k_{\tilde{x}_k}(\tilde{y}_l) = \sum_{k=1}^{\tilde{n}} \sum_{l=1}^{\tilde{m}} \tilde{a}_k \overline{\tilde{b}_l} k(\tilde{x}_k, \tilde{y}_l). \end{aligned}$$

Hence, (4.5) is well-defined. Since K is a positive definite kernel, (4.5) is a positive semi-definite sesquilinear form. To check for definiteness, for $f = \sum_{i=1}^n a_i k_{x_i}$, assume

$$0 = \langle f, f \rangle_H$$

and aim to verify that $f(x) = 0 \ \forall x \in \mathbb{X}$. For arbitrary $x \in \mathbb{X}$, assume without loss of generality that $x_1 = x$ (we may always "add" $0 \cdot k_x$ to f). Write

$$(k(x_i, x_j))_{i,j} = U^* D U,$$

where U is unitary and D diagonal. We have with $a := (a_1, \ldots, a_n)^\top$ and $\beta := Ua$,

$$0 = \langle f, f \rangle_H = \sum_{i=1}^n \sum_{j=1}^n a_i \overline{a_j} k(x_i, x_j) = a^* U^* D U a = \beta^* D \beta.$$

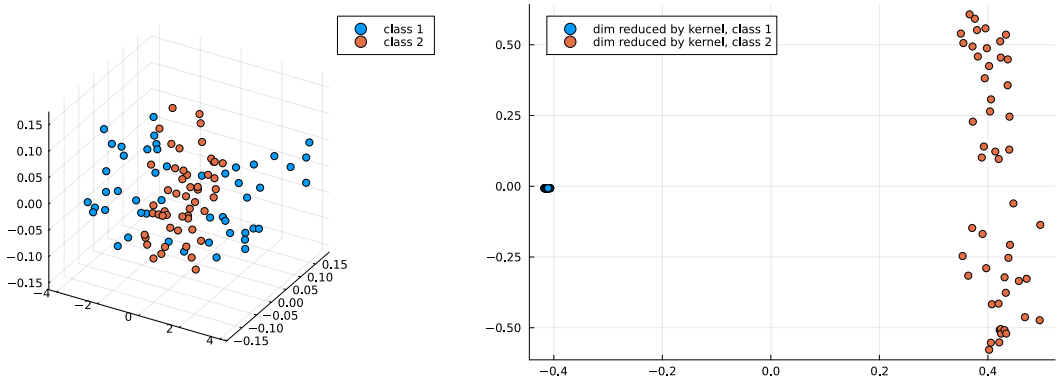


Figure 4.8: (left) Data that consists of 2 distinct groups. (right) kernel-PCA with $k(x,y) = e^{-\|x-y\|^2}$ and $s = 2$ enables linear separation.

Since D is diagonal, $D\beta = 0$. Now, $x_1 = x$ yields

$$f(x) = \sum_{i=1}^{n} a_i k(x_i, x_1) = e_1^* U^* DUa = e_1^* U^* D\beta = 0.$$

Thus, H with (4.5) is a pre-Hilbert space. Define $\Phi : \mathbb{X} \to H$ by $\Phi(x) := k_x$. Hence, $k(x,y) = \langle k_x, k_y \rangle_H = \langle \Phi(x), \Phi(y) \rangle_H$. □

Example 4.3.6

For $\mathbb{X} \subseteq \mathbb{R}^d$, the following kernels are positive definite,

- $k(x,y) = \langle x, y \rangle$,
- $k(x,y) = \langle x, y \rangle^2$,
- $k(x,y) = e^{-\|x-y\|^2}$.

Lemma 4.3.7

If k_1 and k_2 are positive definite kernels on $\mathbb{X}$ and $c > 0$, then the following kernels k are also positive definite on $\mathbb{X}$:

- $k(x,y) = ck_1(x,y)$,
- $k(x,y) = k_1(x,y) + k_2(x,y)$,
- $k(x,y) = k_1(x,y)k_2(x,y)$,
- $k(x,y) = x^\top Ay$ provided that $x, y \in \mathbb{X} = \mathbb{R}^d$ and $A \in \mathbb{R}^{d \times d}$ is positive semi-definite.

The issues of PCA in Figure 4.7 are resolved by kernel-PCA in Figure 4.8.

4.4 Support vector machines

Given data $x_1, \dots, x_n \in \mathbb{R}^d$ with corresponding labels $y_1, \dots, y_n \in \{\pm 1\}$, we aim to find a separating hyperplane $H_{w,b} := \{x \in \mathbb{R}^d : \langle x, w\rangle = b\}$, where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Here, we say that the hyperplane $H_{w,b}$ separates if and only if

$$y_i(\langle x, w\rangle - b) > 0, \qquad i = 1, \dots, n. \tag{4.6}$$

Once this hyperplane is identified, we may define the decision function

$$f(x) = \operatorname{sgn}(\langle x, w\rangle - b)$$

that classifies all of $\mathbb{R}^d$.

For $M \subseteq \mathbb{R}^d$, let $\operatorname{conv}(M)$ denote the smallest convex set that contains M.

Lemma 4.4.1

If $\operatorname{conv}(\{x_i : y_i = 1\}) \cap \operatorname{conv}(\{x_i : y_i = -1\}) = \emptyset$, then there exists a separating hyperplane $H_{w,b}$.

Lemma 4.4.2

If $H_{w,b}$ is a separating hyperplane, then

$$\operatorname{dist}(x_i, H_{w,b}) = \frac{y_i}{\|w\|}(\langle x_i, w\rangle - b), \quad i = 1, \dots, n.$$

Proof. Since w is the normal vector of $H_{w,b}$, there is $r_i \in \mathbb{R}$ such that $\operatorname{dist}(x_i, H_{w,b}) = \|r_i w\|$ and $x_i - r_i w \in H_{w,b}$. Therefore, we have

$$\langle x_i - r_i w, w\rangle - b = 0.$$

We deduce $\langle x_i, w\rangle - b = r_i \|w\|^2$.
If $r_i > 0$, then $y_i = 1$ and

$$y_i(\langle x_i, w\rangle - b) = \|r_i w\|\|w\|.$$

If $r_i < 0$, then

$$y_i(\langle x_i, w\rangle - b) = -r_i\|w\|\|w\| = \|r_i w\|\|w\|.$$

This concludes the proof. □

Definition 4.4.3

If $H_{w,b}$ is a separating hyperplane, then

$$M(w, b) := \frac{1}{\|w\|} \min_{i=1,\dots,n} |\langle x_i, w\rangle - b|$$

is called hard margin.

Theorem 4.4.4

If $x_1, \ldots, x_n \in \mathbb{R}^d$ are separable by a hyperplane, then the two optimization problems

$$\max_{w\in\mathbb{R}^d,\, b\in\mathbb{R}} M(w,b), \quad \text{subject to} \quad y_i(\langle x_i, w\rangle - b) \geq 0, \quad i = 1, \ldots, n, \tag{4.7}$$

$$\min_{w\in\mathbb{R}^d,\, b\in\mathbb{R}} \|w\|^2, \quad \text{subject to} \quad y_i(\langle x_i, w\rangle - b) \geq 1, \quad i = 1, \ldots, n, \tag{4.8}$$

are equivalent.

Proof. We may replace (4.8) with

$$\max_{w\in\mathbb{R}^d,\, b\in\mathbb{R}} \frac{1}{\|w\|}, \quad \text{subject to} \quad y_i(\langle x_i, w\rangle - b) \geq 1, \quad i = 1, \ldots, n. \tag{4.9}$$

Let A and B denote the maximums of (4.7) and (4.9), respectively. Suppose that $\hat{w}$ and $\hat{b}$ maximize (4.9). Then we derive

$$B = \frac{1}{\|\hat{w}\|} \leq \frac{1}{\|\hat{w}\|} \min_{i=1,\ldots,n} |\langle x_i, \hat{w}\rangle - \hat{b}| = M(\hat{w}, \hat{b}) \leq A.$$

Suppose now that $\hat{w}$ and $\hat{b}$ maximize (4.7). Since there exists a separating hyperplane by assumption, we must have $M(\hat{w}, \hat{b}) > 0$. Therefore, $\min_{i=1,\ldots,n} |\langle x_i, \hat{w}\rangle - \hat{b}| > 0$, so that there is $c > 0$ such that

$$\min_{i=1,\ldots,n} |\langle x_i, c\hat{w}\rangle - c\hat{b}| = 1.$$

We obtain

$$A = M(\hat{w}, \hat{b}) = M(c\hat{w}, c\hat{b}) = \frac{1}{\|c\hat{w}\|} \leq B.$$

□

The optimization of (4.8) and subsequent hyperplane separation is called support vector machine (SVM) with hard margin. Those x_i that satisfy $y_i(\langle x_i, w\rangle - b) = 1$ are called support vectors.

If w and b optimize (4.8), then the decision functions f according to (4.6) is given by

$$f(x) = \operatorname{sgn}(\langle x, w\rangle - b).$$

The PCA dimension reduced data in Figure 4.6 are separated by a hyperplane with SVM in Figure 4.9.

Kernel-SVM

As with PCA, there is also a kernel version of SVM. Choose $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ and apply SVM to the data set $\phi_i = \Phi(x_i)$, for $i = 1, \ldots, n$. For any positive definite kernel k, we put

$$K := (k(x_i, x_j))_{i,j=1,\ldots,n} \in \mathbb{R}^{n\times n}.$$

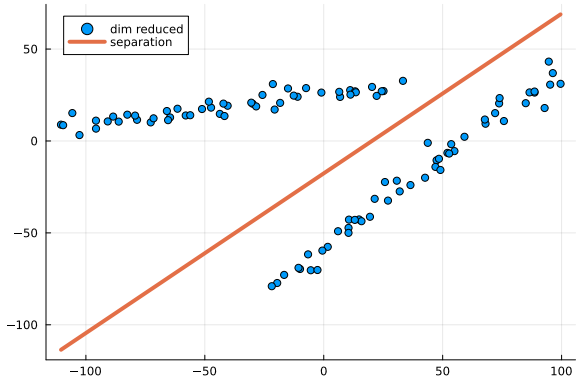


Figure 4.9: SVM identifies a separating hyperplane.

We may assume that the normal vector w lies in the span of $\phi_1, \ldots, \phi_n$ (why?). Hence, there are coefficients $(\alpha_k)_{k=1}^n \subseteq \mathbb{R}$ such that $w = \sum_{j=1}^n \alpha_j \phi_j$. The objective function $\|w\|^2$ is expressed in terms of K by

$$\|w\|^2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle \phi_i, \phi_j \rangle = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K_{i,j}.$$

The constraints $y_i(\langle x_i, w \rangle - b) \geq 1$ are reformulated by

$$\begin{aligned} y_i(\langle x_i, w \rangle - b) &= y_i(\sum_{j=1}^n \alpha_j \langle \phi_i, \phi_j \rangle - b) \\ &= y_i(\sum_{j=1}^n \alpha_j K_{i,j} - b). \end{aligned}$$

The decision function f of kernel-SVM is

$$f(x) = \operatorname{sgn}(\langle \Phi(x), w \rangle - b) = \operatorname{sgn}(\sum_{j=1}^n \alpha_j k(x, x_j) - b).$$

See Figure 4.10 for the decision function derived from kernel-SVM applied to the dimension reduced data in Figure 4.7.

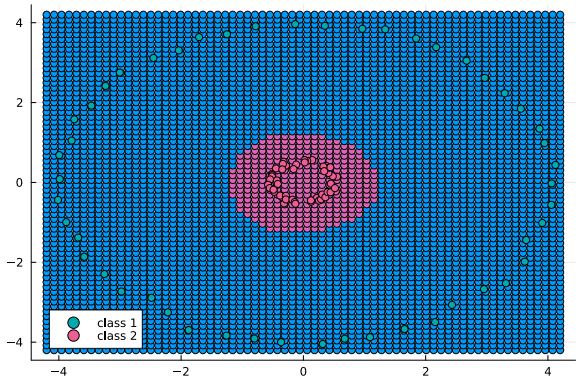


Figure 4.10: Decision function f of kernel-SVM with $k(x, y) = e^{-\|x-y\|^2}$ separates the data.

Chapter 5

Spectral clustering

5.1 Graph-cuts

An undirected graph $\mathcal{G}$ consists of a set of vertices $V = \{v_1, \dots, v_n\}$ and a symmetric weight matrix $W \in \mathbb{R}^{n \times n}$.

Aim

Split V into two separate clusters A and B, while W measures similarity.

For $A \cup B = V$ and $A \cap B = \emptyset$, ideally

- within A: $W(i, j)$ large
- within A: $W(i, j)$ large
- from A to B: $W(i, j)$ small

Definition 5.1.1

For $A, B \subseteq V$ with $A \cup B = V$ and $A \cap B = \emptyset$, define

$$\begin{aligned} \operatorname{cut}(A, B) &:= \sum_{\substack{a \in A \\ b \in B}} W(a, b), \\ \operatorname{vol}(A) &:= \sum_{\substack{a \in A \\ v \in V}} W(a, v), \\ \operatorname{Ncut}(a, B) &:= \operatorname{cut}(A, B) \left(\frac{1}{\operatorname{vol}(A)} + \frac{1}{\operatorname{vol}(B)} \right). \end{aligned}$$

Problem 1 (hard to compute, combinatorial optimization)

$$\arg \min_{A, B \subseteq V : \substack{A \cup B = V \\ A \cap B = \emptyset}} \operatorname{Ncut}(A, B)$$

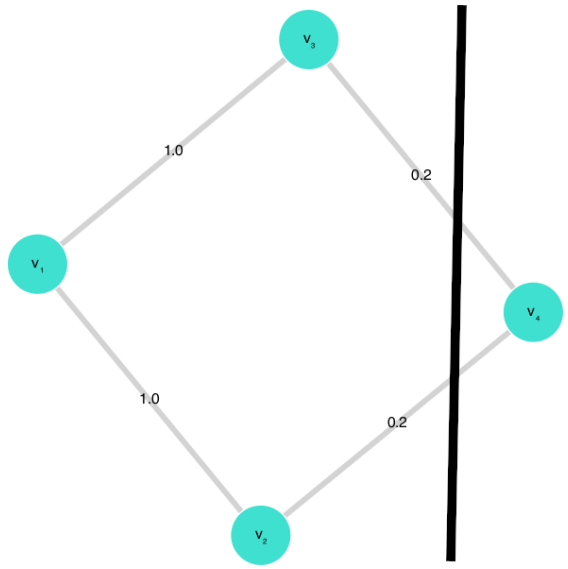


Figure 5.1: Graph $\mathcal{G}$ with cut.

Example 5.1.2

Consider the graph $\mathcal{G}$ with $V = \{v_1, \dots, v_4\}$ and

$$W = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & \frac{1}{5} \\ 1 & 0 & 0 & \frac{1}{5} \\ 0 & \frac{1}{5} & \frac{1}{5} & 0 \end{pmatrix},$$

see Figure 5.1. For $A_1 = \{v_1, v_2, v_3\}$ and $B_1 = \{v_4\}$, we obtain

- $\operatorname{cut}(A_1, B_1) = \frac{2}{5}$,
- $\operatorname{vol}(A_1) = 2 + \frac{6}{5} + \frac{6}{5} = \frac{22}{5}$,
- $\operatorname{vol}(B_1) = \frac{2}{5}$,
- $\operatorname{Ncut}(A_1, B_1) = \frac{2}{5}\left(\frac{5}{22} + \frac{5}{2}\right) = \frac{2}{5} \cdot \frac{60}{22} = \frac{12}{11}$.

For $A_2 = \{v_1, v_3\}$ and $B_2 = \{v_2, v_4\}$, we obtain

- $\operatorname{cut}(A_2, B_2) = \frac{6}{5}$,
- $\operatorname{vol}(A_2) = \frac{16}{5}$,
- $\operatorname{vol}(B_2) = \frac{8}{5}$,
- $\operatorname{Ncut}(A_2, B_2) = \frac{6}{5}\left(\frac{5}{16} + \frac{5}{8}\right) = \frac{6}{5} \cdot \frac{15}{16} = \frac{9}{8} > \frac{12}{11}$.

Definition 5.1.3

Given a graph $\mathcal{G}$ with symmetric weight matrix $W \in \mathbb{R}^n$. For $D = \operatorname{diag}(D_{1,1}, \dots, D_{n,n})$ with $D_{i,i} := \sum_{j=1}^n W(i,j)$, the matrix

$$L := D - W$$

is called the graph-Laplacian.

Lemma 5.1.4

For $x \in \mathbb{R}^n$, the graph-Laplacian satisfies

$$x^\top L x = \frac{1}{2} \sum_{i,j} (x_i - x_j)^2 W(i,j).$$

Proof. We directly compute

$$\begin{aligned}
x^\top L x &= x^\top D x - x^\top W x \\
&= \sum_{i=1}^n x_i^2 D_{i,i} - \sum_{i,j} x_i x_j W(i,j) \\
&= \sum_{i,j} x_i^2 W(i,j) - \sum_{i,j} x_i x_j W(i,j) \\
&= \frac{1}{2} \left(\sum_{i,j} x_i^2 W(i,j) - 2 \sum_{i,j} x_i x_j W(i,j) + \sum_{i,j} x_i^2 W(i,j) \right) \\
&= \frac{1}{2} \left(\sum_{i,j} x_i^2 W(i,j) - 2 \sum_{i,j} x_i x_j W(i,j) + \sum_{i,j} x_j^2 W(i,j) \right) \\
&= \frac{1}{2} \sum_{i,j} \left(x_i^2 - 2 x_i x_j + x_j^2 \right) W(i,j) \\
&= \frac{1}{2} \sum_{i,j} (x_i - x_j)^2 W(i,j).
\end{aligned}$$

□

Theorem 5.1.5

For $A \cup B = V = \{v_1, \dots, v_n\}$ and $A \cap B = \emptyset$, define $x \in \mathbb{R}^n$ by

$$x_i := \begin{cases} \frac{1}{\operatorname{vol}(A)}, & v_i \in A, \\ -\frac{1}{\operatorname{vol}(B)}, & v_i \in B, \end{cases} \qquad i = 1, \dots n.$$

Then the normalized cut satisfies

$$\operatorname{Ncut}(A,B) = \frac{x^\top L x}{x^\top D x}.$$

Proof. a) We first calculate $x^\top Lx$ and obtain

$$\begin{aligned} x^\top Lx &= \frac{1}{2}\sum_{i,j}(x_i - x_j)^2 W(i,j) \\ &= \frac{1}{2}\left(\sum_{\substack{v_i\in A\\ v_j\in B}}\left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)^2 W(i,j) + \sum_{\substack{v_i\in B\\ v_j\in A}}\left(-\frac{1}{\mathrm{vol}(B)}-\frac{1}{\mathrm{vol}(A)}\right)^2 W(i,j)\right) \\ &= \frac{1}{2}\left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)^2 \mathrm{cut}(A,B) + \frac{1}{2}\left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)^2 \mathrm{cut}(B,A) \\ &= \left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)^2 \mathrm{cut}(A,B). \end{aligned}$$

b) Let us now calculate $x^\top Dx$,

$$\begin{aligned} x^\top Dx &= \sum_{i=1}^n x_i^2 D_{i,i} \\ &= \sum_{v_i\in A}\left(\frac{1}{\mathrm{vol}(A)}\right)^2 \sum_{j=1}^n W(i,j) + \sum_{v_i\in B}\left(\frac{1}{\mathrm{vol}(B)}\right)^2 \sum_{j=1}^n W(i,j) \\ &= \left(\frac{1}{\mathrm{vol}(A)}\right)^2 \mathrm{vol}(A) + \left(\frac{1}{\mathrm{vol}(B)}\right)^2 \mathrm{vol}(B) \\ &= \left(\frac{1}{\mathrm{vol}(B)}\right) + \left(\frac{1}{\mathrm{vol}(B)}\right). \end{aligned}$$

c) By combining a) and b), we derive

$$\frac{x^\top Lx}{x^\top Dx} = \frac{\left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)^2 \mathrm{cut}(A,B)}{\left(\frac{1}{\mathrm{vol}(B)}\right)+\left(\frac{1}{\mathrm{vol}(B)}\right)} = \mathrm{Ncut}(A,B).$$

□

The vector $x = (1,\dots,1)^\top$ satisfies

$$Lx = Dx - Wx = \begin{pmatrix} D_{1,1} \\ \vdots \\ D_{n,n} \end{pmatrix} - \begin{pmatrix} \sum_{j=1}^n W(1,j) \\ \vdots \\ \sum_{j=1}^n W(n,j) \end{pmatrix} = \begin{pmatrix} D_{1,1} \\ \vdots \\ D_{n,n} \end{pmatrix} - \begin{pmatrix} D_{1,1} \\ \vdots \\ D_{n,n} \end{pmatrix} = 0$$

so that it is an eigenvector of L with associated eigenvalue 0.

Lemma 5.1.6

If $W(i,j) \geq 0$ for all i,j, then L is positive-semi-definite.

Proof. This follows from $x^\top Lx = \sum_{i,j}(x_i - x_j)^2 W(i,j) \geq 0$. □

We restrict us to symmetric weight matrices W with $W(i,j) \geq 0$ and assume that D is invertible, i.e., the graph $\mathcal{G}$ does not have any isolated vertices.

Idea

Find $x \in \mathbb{R}^n$ such that $x^\top Lx$ is small and put

$$A := \{v_i : x_i \geq 0\}, \qquad B := \{v_i : x_i < 0\} \tag{5.1}$$

The normalized Laplacian is

$$\mathscr{L} := D^{-1/2} L D^{-1/2}.$$

Problem 2

a) Minimize $\frac{x^\top Lx}{x^\top Dx}$ subject to $x^\top D \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0$.

b) Minimize $\frac{z^\top \mathscr{L} z}{z^\top z}$ subject to $z^\top D^{1/2} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0$.

What to do? Facts:

- $(1, \ldots, 1)^\top$ is eigenvector of L with associated eigenvalue 0
- $D^{1/2}(1, \ldots, 1)^\top$ is eigenvector of $\mathscr{L}$ with associated eigenvalue 0

To minimize b) of Problem 2,

take eigenvector z of $\mathscr{L}$ associated with the second smallest eigenvalue and build A and B by (5.1).

This eigenvector must have positive and negative entries because it is orthogonal to $D^{1/2} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ and the latter has only nonnegative entries.

This provides a "reasonable" approximative solution to Problem 1.