Solution for Exercises in

— Introduction to Applied Linear Algebra

Author : utoppia

Last Update : May 31, 2020
Version : 1.0

Forewords

This document is a collection of solution for the exercises in textbook A Introduction to Applied Linear Algebra written by Stephen Boyd.

Due to the COVID-19, I was self-isolated during the whole May 2020 in Ankara, Turkey, and I was boring to learn the Turkish during the periods, I wanted to make the time more valueable, so I choose this book and try to learn and solve the exercises. It is not easy for me since I had forgot almost all mathematics I learned, and the English. When I want to find the solution of some exercise I can't solve on the internet, I found that I can't find anything but a website in which I should pay before assesing the answer. I am a poor student and I don't want to waste my money, so I decided to make this document.

During formatting this document, I learned not only the ideas about Linear Algebra, but also the LaTeX, and English, I am so happy to finish this document, and now it belongs to you, my friends, who are learning this book, and I hope this document may help you when you encouter some hard job solving the exercises.

If you find any error in this document, please contact me by the e-mail *utoppia@163.com*.

Thank you very much.

Contents

1 Vectors

1.1 Vector equations. Determine whether each of the equations below is true, false, or contains bad notation (and therefore does not make sense).

(a) $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = (1,2,1)$. ✓ **YES**.

(b) $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = [1,2,1]$. × **NO**.

(c) $(1,(2,1)) = ((1,2),1)$. ✓ **YES**.

1.2 Vector notation. Which of the following expressions use correct notation? When the expression does make sense, give its length. In the following, a and b are 10-vector, and c is a 20-vector.

(a) $a + b - c_{3:12}$. ✓ **YES**, length 10.

(b) $(a, b, c_{3:13})$. ✓ **YES**, legnth 31.

(c) $2a + c$. × **NO**.

(d) $(a,1) + (c_1, b)$. ✓ **YES**, length 11.

(e) $((a,b),a)$. ✓ **YES**, length 30.

(f) $[ab] + 4c$. × **NO**.

(g) $\begin{bmatrix} a \\ b \end{bmatrix} + 4c$. ✓ **YES**, length 20.

1.3 Overloading. Which of the following expressions use correct notation? If the notation is correct, is it also unambiguous? Assume that a is a 10-vector and b is a 20-vector.

(a) $b = (0,a)$. ✓ **YES**, and it is unambiguous.

(b) $a = (0,b)$. × **NO**.

(c) $b = (0,a,0)$. ✓ **YES**, but it is ambiguous.

(d) $a = 0 = b$. ✓ **YES**, and it is unambiguous.

1.4 Periodic energe usage. The 168-vector w gives the hourly electricity consumption of a manufacturing plant, starting on Sunday midnight to 1AM, over one week, in MWh (megawatt-hours). The consmption pattern is the

same each day, *i.e.*, it is 24-periodic, which means that $w_{t+24} = w_t$ for $t = 1, \ldots, 114$. Let d be the 24-vector that gives the energe consumption over one day, starting at midnight.

(a) Use vector notation to express w in terms of d.

(b) Use vector notation to express d in terms of w.

Solution:

(a) $w = (d_{1:24}, d, d, d, d, d, d, d_1)$

(b) $d = w_{24:47}$

1.5 Interpreting sparsity. Suppose the n-vector x is sparse, *i.e.*, has only a few nonzero entries. Give a short sentence or two explaining what this means in each of the following contexts.

(a) x represents the daily cash flow of some business over n days.

(b) x represents the annual dollar value purchases by a customer of n products or services.

(c) x represents a portfilio, say, the dollar value holding of n stocks.

(d) x represents a bill of materials for a project, *i.e.*, the amounts of n materials needed.

(e) x represents a monochrome image, *i.e.*, the brightness values of n pixels.

(f) x is the daily rainfall in a location over one year.

Solution:

(a) The cash flow of this business change a little over n days.

(b) The customer buy only some of the n products.

(c) This portfilio only invested in some of stocks.

(d) This project made from few of the materials.

(e) This iamge is dark.

(f) This location has more dry days than rainfall days.

1.6 Vector of differences. Suppose x is an n-vector. The associated vector of differences is the $(n-1)$-vector d given by $d = (x_2 - x_1, x_3 - x_2, \ldots, x_n - x_{n-1})$.

Express d in terms of x using vector operations (*e.g.*, slicing notation, sum, difference, linear combinations, inner product). The difference vector has a simple interpretation when x represents a time series. For example, if x gives the daily value of some quantity, d gives the day-to-day changes in quantity.

Solution:

$d = x_{2:n} - x_{1:n-1}$.

1.7 Transforming between two encodings for Boolean vectors. A Boolean n-vector is one for which all entries either 0 or 1. Such vectors are used to encode whether each of n conditions holds, with $a_i = 1$ meaning that condition i holds. Another common encoding of the same information uses the two values -1 and $+1$ for the entries. For example the Boolean vector $(0,1,1,0)$ would be written using this alternative encoding as $(-1,+1,+1,-1)$. Suppose that x is a Boolean vector with entries that are 0 and 1, and y is a vector encoding the same information using the values -1 and $+1$. Express y in terms of x using vector notation. Also, express x in terms of y using vector notation.

Solution:

$y = 2x - 1$.
$x = \frac{1}{2}(y + 1)$.

1.8 Profit and sales vectors. A company sells n different products or items. The n-vector p gives the profit, in dollars per unit, for each of the n items. (The entrie of p are typically positive, but a few items might have negative entries. These items are called *loss leaders*, and are used to increase customer engagement in the hope that the customer will make other, profitable purchases.) The n-vector s gives the total sales of each of the items, over some period (such as a month), *i.e.*, s_i is the total number of units of item i sold. (These are also typically nonnegative, but negative entries can be used to reflect items that were purchased in a previous time period and returned in this one.) Express the total profit in terms of p and s using vector notation.

Solution:

$p^T s$.

1.9 Symptoms vector. A 20-vector s records whether each of 20 different symptoms is present in a medical patient, with $s_i = 1$ meaning the patient has the symptom and $s_i = 0$ meaning she does not. Express the following using vector notation.

(a) The total number of symptoms the patient has.

(b) The patient exhibits five out of the first ten symptoms.

Solution:

(a) $\mathbf{1}^T s$

(b) $\mathbf{1}^T s_{1:10} \geq 5$

1.10 Total score from course record. The record for each student in a class is given as a 10-vector r, whether $r_1, \ldots, r_8$ are the grades for the 8 homeworks assignments, each on a $0-10$ scale, r_9 is the midterm example grade on a $0-120$ scale, and r_{10} is final exam score on a $0-160$ scale. The student's total course score s, on a $0-100$ scale, is based 25% on the homework, 35% on the midterm exam, and 40% on the final exam. Express s in the form $s = w^T r$. (That is, determine the 10-vector w.) You can give the coefficients of w to 4 digits after the decimal point.

Solution:

Let $w = k(0.25, 0.25, \ldots, 0.25, 0.35, 0.4)$ where k is the coefficient which promise that the final score is between 0 and 100, suppose a student get full score in all homeworks, midterm and final exam, the final score she get is

$$k \times (0.25 \times 10 \times 8 + 120 \times 0.35 + 160 \times 0.4) = 126k,$$

and we need $126k = 100$, implies

$$k \approx 0.7937.$$

Thus we have

$$w = (0.1984, \ldots, 0.1984, 0.2778, 0.3175).$$

1.11 Word count and word count histogram vectors. Suppose the n-vector w is the word count vector associated with a document and a dictionary of n words. For simplicity we will assume that all words in the document appear in the dictionary.

(a) What is $\mathbf{1}^T w$?

(b) What does $w_{282} = 0$ mean?

(c) Let h be the n-vector that gives the histogram of the word counts, *i.e.*, h_i is the fraction of the words in the document that are word i. Use vector notation to express h in terms of w. (You can assume that the document contains at least one word.)

Solution:

(a) $\mathbf{1}^T w$ means the total word counts in the document.

(b) The word 282 doesn't appear in this document.

(c) $h = \dfrac{w}{\mathbf{1}^T w}$.

1.12 Total cash value. An international company holds cash in five currencies: USD (US dollar), RMB (Chinese yuan), EUR (euro), GBP (British pound), and JPY (Japanese yen), in amounts given by the 5-vector c. For example, c_2 gives the number of RMB held. Negative entries in c represent liabilities or amounts owed. Express the total (net) value of the cash in USD, using vector notation. Be sure to give the size and define the entries of any vectors that you introduce in your solution. Your solution can refer to currency exchange rates.

Solution:

We define a currency exchange rate 5-vector r, for example, r_2 refer to the exchange rate from RMB to USD, and we have $r_1 = 1$, therefore the total value of the cash in USD is

$$r^T c.$$

1.13 Average age in a population. Suppose the 100-vector x represents the distribution of ages in some population of people, with x_i being the number of $i-1$ year olds, for $i = 1, \ldots, 100$. (You can assume that $x \neq 0$, and that

there is no one in the population over age 99.) Find expressions, using vector notation, for the following quantities.

(a) The total number of people in the population.

(b) The total number of people in the population age 65 and over.

(c) The average age of the population. (You can use ordinary division of number in your expression.)

Solution:

(a) $\mathbf{1}^T x$.

(b) $\mathbf{1}^T x_{66:100}$.

(c) Define 100-vector denote the age from 0 to 99, *i.e.*, $a_i = i - 1$ for $i = 1, \ldots, 100$, the average age of the population is

$$\frac{a^T x}{\mathbf{1}^T w}.$$

1.14 Industry or sector exposure. Consider a set of n assets or stocks that we invest in. Let f be an n-vector that encodes whether each asset is in some specific Industry or sector, *e.g.*, pharmaceuticals or consumer electronics. Specifically, we take $f_i = 1$ if asset i is in the sector, and $f_i = 0$ if it is not. Let the n-vector h denote a portfilio, with h_i the dollar value held on asset i (with negative meaning a short position). The inner product $f^T h$ is called the (dollar value) *exposure* of our portfilio to the sector. It gives the net dollar value of the portfilio that is invested in assets from the sector. A portfilio h is called *neutral* (to a sector or industry) if $f^T h = 0$.

A portfilio h is called *long only* if each entry is nonnegative, *i.e.*, $h_i \geq 0$ for each i. This means the portfilio does not include any short positions.

What does it mean if a long-only portfilio is neutral to a sector, say, pharmaceuticals? Your answer should be in simple English, but you should back up your conclusion with an argument.

Solution:

The portfilio doesn't hold any assets associated with this industry, *i.e.*, the portfilio doesn't invest in this industry.

1.15 Cheapest supplier. You must buy n raw materials in quantities given by the n-vector q, where q_i is the amount of raw material i that you must buy. A set of K potential suppliers offer the raw materials at prices given by the n-vectors $p_1, \ldots, p_K$. (Note that p_k is an n-vector; $(p_k)_i$ is the price that supplier k charges per unit of raw materials i.) We will assume that all quantities and prices are positive.

If you must choose just one supplier, how would you do it? Your answer should use vector notation.

A (highly paid) consultant tells you that you might do better (*i.e.*, get a better total cost) by splitting your order into two, by choosing two suppliers and ordering $(1/2)q$ (*i.e.*, half the quantities) from each of the two. He argues that having a diversity of suppliers is better. Is he right? If so, explain how to find the two suppliers you would use to fill half the order.

Solution:

The total cost if we buy these materials from supplier k is $p_k^T q$, therefore we should choose the supplier which has minimum $p_k^T q$, which can be expressed as

$$\operatorname*{argmin}_{k=1,\ldots,K} p_k^T q.$$

He is wrong, suppose there are two suppliers which imply fewest total cost, let the suppliers be p_{k_1} and p_{k_2}, and we can assume that $p_{k_1}^T q < p_{k_2}^T q$, then we can acheive less total cost if we buy all meterials from supplier k_1, because

$$p_{k_1}^T q = \frac{1}{2} p_{k_1}^T q + \frac{1}{2} p_{k_1}^T q < \frac{1}{2} p_{k_1}^T q + \frac{1}{2} p_{k_2}^T q.$$

1.16 Inner product of nonnegative vectors. A vector is called *nonnegative* if all its entries are nonnegative.

(a) Explain why the inner product of two nonnegative vectors is nonnegative.

(b) Suppose the inner product of two nonnegative is zero. What can you say about them? Your answer shoud be in terms of their respective sparsity patters, *i.e.*, which entries are zero and nonzero.

Solution:

Denote the two vectors as a and b, we have

$$a^T b = \sum_{i=1}^{n} a_i b_i.$$

a As $a_i \geq 0$ and $b_i \geq 0$ for $i = 1, \ldots, n$, we have $a_i b_i \geq 0$ for each i, thus $a^T b \geq 0$.

b Since $a^T b = 0$, means the entries of a must be zero if the corresponded entries of b are nonzero, and the entries of b must be zero if the corresponded entries of a are nonzero, *i.e.*

$$\begin{cases} a_i = 0, & \text{if } b_i = 0 \\ b_i = 0, & \text{if } a_i = 0 \end{cases}$$

1.17 Linear combinations of cash flows. We consider cash flow vectors over T time periods, with a positive entry meaning a payment received, and negative meaning a payment made. A (unit) *single period loan*, at time period t, is the T-vector l_t that corresponds to a payment received of \$1 in period t and a payment made of $\$(1+r)$ in period $t+1$, with all other payment zero. Here $r > 0$ is the interst rate (over one period).

Let c be a \$1 $T-1$ period loan, starting at period 1. This means that \$1 is received in period 1, $\$(1+r)^{T-1}$ is paid in period T, and all other payments (*i.e.*, $c_2, \ldots, c_{T-1}$) are zero. Express c as a linear combination of single period loans.

Solution:

$c = l_1 + (1+r)l_2 + \cdots + (1+r)^{T-2} l_T$

1.18 Linear combinations of linear combinations. Suppose that each of the vectors $b_1, \ldots, b_k$ is a linear combination of the vectora $a_1, \ldots, a_m$, and c is a linear combination of $b_1, \ldots, b_k$. The c is a linear combination of $a_1, \ldots, a_m$. Show this for the case with $m = k = 2$. (Showing it in general is not much more difficult, but the notation gets more complicated.)

Solution:

Suppose

$$b_1 = k_1 a_1 + k_2 a_2$$
$$b_2 = q_1 a_1 + q_2 a_2,$$

then

$$\begin{aligned} c &= p_1 b_1 + p_2 b_2 \\ &= p_1(k_1 a_1 + k_2 a_2) + p_2(q_1 a_1 + q_2 a_2) \\ &= (p_1 k_1 + p_2 q_1) a_1 + (p_1 k_2 + p_2 q_2) a_2, \end{aligned}$$

i.e., c is a linear combination of a_1, a_2.

1.19 Auto-regressive model. Suppose that $z_1, z_2, \ldots$ is a time series, with the number z_t giving the value in period or time t. For example z_t could be the gross sales at a particular store on day t. An *auot-regressive* (AR) model is used to predict z_{t+1} from the previous M values, $z_t, z_{t-1}, \ldots, z_{t-M+1}$:

$$\hat{z}_{t+1} = (z_t, z_{t-1}, \ldots, z_{t-M+1})^T \beta, t = M, M+1, \ldots.$$

Here $\hat{z}_{t+1}$ denotes the AR model's prediction of z_{t+1}, M is the memory length of the AR model, and the M-vector β is the AR model coefficient vector. For this problem we will assume that the time period is daily, and $M = 10$. Thus, the AR model predicts tomorrow's value, given the value over the last 10 days.

For each of the following cases, give a short interpretation or description of the AR model in English, without referring to mathematical concepts like vectors, inner product, and so on. You can use words like 'yesterday' or 'today'.

(a) $\beta \approx e_1$.

(b) $\beta \approx 2e_1 - e_2$.

(c) $\beta \approx e_6$.

(d) $\beta \approx 0.5e_1 + 0.5e_2$.

Solution:

(a) The prediction for tomorrow is equal to the value today.

(b) The prediction for tomorrow is equal to the double of value of today minus the value of yesterday.

(c) The prediction for tomorrow is equal to the value 5 days ago.

(d) The prediction for tomorrow is equal to the average of the values of today and yesterday.

1.20 How many bytes does it take to store 100 vectors of length 10^5? How many flops does it take to form a linear combination of them (with 100 nonzero coefficients)? About how long would this take on a computer capable of carrying out 1 Gflops/s?

Solution:

It take 10^7 bytes to store these vectors, and about 2×10^7 flops to form a linear combination, it will use about $20ms$ to take out in this computer.

2 Linear Functions

2.1 Linear or not? Determine whether each of the following scalar-valued functions of n-vectors is linear. If it is a linear function, give its inner product representation, *i.e.*, an n-vector a for which $f(x) = a^T x$ for all x. If it is not linear, give specific x, y, α and β for which superposition fails, *i.e.*,

$$f(\alpha x = \beta x) \neq \alpha f(x) + \beta f(y)$$

.

(a) The spread of values of the vector, defined as $f(x) = max_k x_k - min_k x_k$.

(b) The difference of the last element and the first, $f(x) = x_n - x_1$.

(c) The median if an n-vector, where we will assume $n = 2k+1$ is odd. The median of the vector x is defined as the $(k+1)$th largest number among the entries of x. For example, the median of $(-7.1, 3.2, -1.5)$ is -1.5.

(d) The average of the entries with odd indices, minus the average of the entries with even indices. You can assume that $n = 2k$ is even.

(e) Vector extrapolation. defined as $x_n + (x_n - x_{n-1})$, for $n \geq 2$. (This is simple prediction of what x_{n+1} would be, based on a straight line drawn through x_n, and x_{n-1}.)

Solution:

(a) × **NO.** Suppose $x = (1, 0, \ldots, 0)$ and $y = (0, 1, \ldots, 1)$, $\alpha = 1$ and $\beta = 1$, then $f(\alpha x + \beta y) = f(1, 1, \ldots, 1) = 1 - 1 = 0$, but $\alpha f(x) + \beta f(y) = (1 - 0) + (1 - 0) = 2$, so fot this specific x, y, α and β, it doesn't satisfied the property of a *linear function*.

(b) ✓ **YES.** We can define $a = (1, 0, \ldots, 0, 1)$ which will satisfy $f(x) = a^T x$.

(c) × **NO.** Suppose $x = (1, -1, 2), y = (2, 1, -1), \alpha = 1, \beta = 1$, then $f(\alpha x + \beta y) = f(3, 0, 1) = 1$, but $\alpha f(x) + \beta f(y) = 1 + 1 = 2$.

(d) ✓ **YES.** We can create a vector a which $a_{2k+1} = \frac{1}{2}, a_{2k} = -\frac{1}{2}$, that will satisfy $f(x) = a^T x$.

(e) ✓ **YES.** $a = (0, 0, \ldots, -1, 2)$ satisfy $f(x) = a^T x$.

2.2 Processor powers and temperature. The temperature T of an electronic device containing three processor is an affine function of the power dissipated by the three processor, $P = (P_1, P_2, P_3)$. When all three processors are idling, we have $P = (10, 10, 10)$, which results in a temperature $T = 30$. When the first processor operates at full power and the other two are idling, we have $P = (100, 10, 10)$, and the temperature rise to $T = 60$. When the second processor operates at full power and other two are idling, we have $P = (10, 100, 10)$ and $T = 70$. When the third processor operates at full power and the other two are idling, we have $P = (10, 10, 100)$ and $T = 65$. Now suppose that all three processor are operate at the same power P^{same}. How large can P^{same} be, if we require the $T \leq 85$? *Hint.* From the given data, find the 3-vector a and number b for which $T = a^T x + b$.

Solution:

According to the given data, we can easyly get a equation as :

$$\begin{cases} 10a_1 + 10a_2 + 10a_3 + b &= 30 \\ 100a_1 + 10a_2 + 10a_3 + b &= 60 \\ 10a_1 + 100a_2 + 10a_3 + b &= 70 \\ 10a_1 + 10a_2 + 100a_3 + b &= 65 \end{cases} \rightarrow \begin{cases} a_1 &= \frac{1}{3} \\ a_2 &= \frac{4}{9} \\ a_3 &= \frac{7}{18} \\ b &= \frac{35}{3} \end{cases}$$

There for $T = a^T + b = \frac{1}{3}P_1 + \frac{4}{9}P_2 + \frac{7}{18}P_3 + \frac{35}{3}$. For the same P^{same}, we have $T(P^{same}) = \frac{7}{6}P^{same} + \frac{35}{3}$, so the $\frac{7}{6}P^{same} + \frac{35}{3} \leq 85$ makes the $P^{same} \leq \frac{440}{7} \approx 52.8$.

2.3 Motive of a mass in response to applied force. A unit mass moves on a straight line(in one dimension). The position of the mass as time t(in seconds) is denoted by $s(t)$, and its derivatives (the velocity and acceleration) by $s'(t)$ and $s''(t)$. The position as a function of time can be determined from Newton's second law

$$s''(t) = F(t).$$

Where $F(t)$ is the force applied at time t, and the the initial conditions $s(0), s'(0)$. We assume $F(t)$ is piecewise-constant, and is kept constant in intervals of one second. The sequence of forces $F(t)$, for $0 \leq t \leq 10$, can then be reoresented by a 10-vector f, with

$$F(t) = f_k, \qquad k - 1 \leq t < k.$$

Derive expressions for the final velocity $s'(10)$ and final position $s(10)$. Show that $s(10)$ and $s'(10)$ are affine functions of x, and give 10-vectors a, c and constants b, d for which

$$s'(10) = a^T f + b, \qquad s(10) = c^T f + d.$$

This means that the mapping from the applied force sequence to the final position and velocity is affine. *Hint*. You can use

$$s'(t) = s'(0) + \int_0^t F(\tau)d\tau, \qquad s(t) = s(0) + \int_0^t s'(\tau)d\tau.$$

You will find out that the mass velocity $s'(t)$ is piecewise-linear.

Solution:

Suppose $K-1 < t \leq K$, where K is a nonegative integer, therefore

$$\begin{aligned} s'(t) &= s'(0) + \int_0^t F(\tau)d\tau \\ &= s'(0) + \sum_{i=1}^{K-1} \int_{i-1}^{i} f_i d\tau + \int_{K-1}^{t} f_K d\tau \\ &= s'(0) + f_1 + \cdots + f_{K-1} + (t-K+1)f_K \end{aligned}$$

therefore, for an integer K, we have

$$s'(K) = s'(0) + f_1 + \cdots + f_K.$$

or we can write $s'(t)$ as

$$s'(t) = s'(K-1) + (t-K+1)f_K.$$

And with $s(t) = s(0) + \int_0^t s'(\tau)d\tau$, we have

$$\begin{aligned} s(t) &= s(0) + \sum_{i=1}^{K-1} \left(\int_{i-1}^{i} s'(\tau)d\tau \right) + \int_{K-1}^{t} s'(\tau)d\tau \\ &= s(0) + \sum_{i=1}^{K-1} \left(\int_{i-1}^{i} s'(i-1) + (\tau - i + 1)f_i d\tau \right) + \int_{K-1}^{t} s'(K-1) + (\tau - K + 1)f_K d \\ &= s(0) + \sum_{i=1}^{K-1} (s'(i-1) + (1/2)f_i) + (t-K+1)s'(K-1) + (t-K+1)^2/2)f_K \\ &= s(0) + Ks'(0) + \frac{2K-1}{2}f_1 + \frac{2K-3}{2}f_2 + \cdots + \frac{3}{2}f_{K-1} + (t-K)s'(K-1) + (t-K+1)^2/2)f_K \end{aligned}$$

For an integer K, we have

$$s(K) = s(0) + Ks'(0) + \frac{2K-1}{2}f_1 + \frac{2K-3}{2}f_2 + \cdots + \frac{3}{2}f_{K-1} + + \frac{3}{2}f_K.$$

Thus,

$$s'(10) = s'(0) + f_1 + \cdots + f_{10},$$

let $b = s'(10)$ and $a = (1, 1, \ldots, 1)$, we have

$$s'(10) = a^T f + b.$$

Also we have

$$s(10) = s(0) + 10s'(0) + \frac{19}{2}f_1 + \cdots + \frac{1}{2}f_{10},$$

let $d = s(0) + 10s'(0)$ and $c = (\frac{19}{2}, \frac{17}{2}, \ldots, \frac{1}{2})$, we have

$$s(10) = c^T f + d.$$

2.4 Linear function? The function $\phi : \mathrm{R}^3 \to \mathrm{R}$ satisfies

$$\phi(1,1,0) = -1, \qquad \phi(-1,1,1) = 1, \qquad \phi(1,-1,-1) = 1.$$

Choose one of the following, and justify your choice: ϕ must be linear; ϕ could be linear; ϕ cannot be linear.

Solution:

ϕ cannot be linear. Suppose ϕ is a linear function, then it must satisfies the *Additivity property, which means*

$$\begin{aligned} -1 + 1 + 1 &= \phi(1,1,0) + \phi(-1,1,1) + \phi(1,-1,-1) \\ &= \phi(1-1+1, 1+1-1, 0+1-1) \\ &= \phi(1,1,0) \\ &= -1 \end{aligned}$$

then we derive to $1 = -1$, which is wrong, so the function ϕ cannot be linear.

2.5 Affine function. Suppose $\psi : \mathrm{R}^2 \to \mathrm{R}$ is an affine function, with $\psi(1,0) = 1, \psi(1,-2) = 2$.

(a) What can you say about $\psi(1,-1)$? Either give the value of $\psi(1,-1)$, or state that it cannot be determined.

(b) What can you say about $\psi(2,-2)$? Either give the value of $\psi(2,-2)$, or state that it cannot be determined.

Justify your answers.

Solution:

(a) $\psi(1,-1) = \frac{3}{2}$. $\psi(1,-1) = \psi(\frac{1}{2}(1,0) + \frac{1}{2}(1,-2)) = \frac{1}{2}\psi(1,0) + \frac{1}{2}\phi(0,-2) = \frac{1}{2} + 1 = \frac{3}{2}$.

(b) $\psi(2,-2)$ cannot be determined.

2.6 Questionnaire scoring. A questionnaire in a magezine has 30 questions, broken into two sets of 15 questions. Someone taking the questionnaire answers each question with 'Rarely', 'Sometimes', or 'Often'. The answers are recorded as a 30-vector a, with $a_i = 1,2,3$ if question i is answered Rarely, Sometimes, or Often, respectively. The total score on a completed questionnaire is found by adding up 1 point for every question answered Sometimes and 2 points for every question answered Often on questions 1-15, and by adding 2 and 4 points for those responses on questions 16-30. (Nothing is added to the score for Rarely response.) Express the total score s in the form of an affine function $s = w^T a + v$, where w is a 30-vector and v is a scalar (number).

Solution:

The w and v satisfy

$$\begin{cases} w_i = 1, & 1 \le i \le 15 \\ w_i = 2, & 16 \le i \le 30 \end{cases} \qquad \begin{cases} v_i = -1, & 1 \le i \le 15 \\ v_i = -2, & 16 \le i \le 16 \end{cases}$$

but will this satisfy our question? Let's justify it: for questions 1-15, the quesiton answered Sometime will get $s_i = w_i \times 2 + v_i = 2 - 1 = 1$ score, and the question answered Often will get $s_i = w_i \times 3 + v_i = 3 - 1 = 2$ score. Now for questions 16-30, the question answered Sometime will get $s_i = w_i \times 2 + v_i = 2 \times 2 - 2 = 2$ score, and the question answered Often will get $s_i = w_i \times 3 + v_i = 2 - 2 = 4$ score. It means that the w and v we create satisfy the function $s = w^T a + v$.

2.7 General formula for affine functions. Verify that the formula

$$f(x) = f(0) + x_1(f(e_1) - f(0)) + \cdots + x_n(f(e_n) - f(0)),$$

holds for any affine function $f : (R)^n \to (R)$. You can use the fact that $f(x) = a^T x + b$ for some n-vector a and b.

Solution:

Because $f : (R)^n \to (R)$ is an affine function, so there existed n-vector a and constant b, which satisfy $f(x) = a^T x + b$, we have $f(e_i) = a^T e_i + b, f(0) = b$, therefore $x_i(f(e_i) - f(0)) = x_i(a^T e_i + b - b) = x_i a^T e_i = a^T x_i e_i$, thus $f(0) + x_1(f(e_1) - f(0)) + \cdots + x_n(f(e_n) - f(0)) = b + a^T(x_1 e_1 + \cdots + x_n e_n) = a^T x + b = f(x)$.

2.8 Integral and derivative of polynomial. Suppose the n-vector c gives the coeffients of a polynomial $p(x) = c_1 + c_2 x + \ldots c_n x^{n-1}$.

(a) Let α and β be numbers with $\alpha < \beta$. Find an n-vector a for which

$$a^T c = \int_\alpha^\beta p(x)dx$$

always holds. This means that the integral of a polynomial over an interval is a linear function of its coefficients.

(b) Let α be a number. Find an n-vector b for which

$$b^T c = p'(\alpha).$$

This means that the derivation of the polynomial at a given point is a linear function of its coefficients.

Solution:

(a) As

$$\int p(x)dx = c_0 + c_1 x + \frac{c_2 x^2}{2} + \cdots + \frac{c_n x^n}{n},$$

therefore

$$\int_\alpha^\beta p(x)dx = (\beta - \alpha)c_1 + \frac{\beta^2 - \alpha^2}{2}c_2 + \cdots + \frac{\beta^n - \alpha^n}{n}c_n.$$

Let

$$a_i = \frac{\beta^i - \alpha^i}{i},$$

previous equation can be rewrited as

$$\int_\alpha^\beta p(x)dx = a_1 c_1 + a_2 c_2 + \cdots + a_n c_n = a^T c$$

.

(b) Because $p'(\alpha) = c_2 + 2c_3\alpha + \cdots + (n-1)c_n\alpha^{n-1}$. Let $b_i = (i-1)a^{i-1}$, previois equation can be rewrited as $p'(\alpha) = b_1 c_1 + b_2 c_2 + \cdots + b_n c_n = b^T c$.

2.9 Taylor approximation. Consider the function $f : \mathrm{R}^2 \to \mathrm{R}$ given by $f(x_1, x_2) = x_1 x_2$. Find the Taylor approximation $\hat{f}$ at point $z = (1,1)$. Compare $f(x)$ and $\hat{f}(x)$ for the following values of x:

$$x = (1,1), \qquad x = (1.05, 0.95), \qquad x = (0.85, 1.25), \qquad x = (-1,2).$$

Make a brief comment about the accuracy if the Taylor approximation in each case.

Solution:

The gradient at point z is

$$\nabla f(z) = (1,1),$$

therefore the Taylor approximation at point z is

$$\begin{aligned}\hat{f}(x) &= f(z) + \nabla f(z)^T (x - z) \\ &= 1 + (1,1)^T (x_1 - 1, x_2 - 1) \\ &= 1 + x_1 + x_2 - 2 \\ &= x_1 + x_2 - 1.\end{aligned}$$

The values are listed below

Point x	Real value $f(x)$	Approximate value $\hat{f}(x)$
$(1,1)$	1	1
$(1.05, 0.95)$	0.997	1.0
$(0.85, 1.25)$	1.0625	1.0
$(-1,2)$	-2	0

2.10 Regression model. Consider the regression mode $\hat{y} = x^T \beta + v$, where $\hat{y}$ is the predicted response, x is an 8-vector of features, β is an 8-vector of coefficients, and v is the offset term. Determine whether each of the following statements is true or false.

(a) if $\beta_3 > 0$ and $x_3 > 0$, then $\hat{y} \geq 0$.

(b) if $\beta_2 = 0$ the the prediction $\hat{y}$ does not depend on the second feature x_2.

(c) if $\beta_6 = -0.8$, then increasing x_6 (keeping all other x_is the same) will decrease $\hat{y}$.

Solution:

(a) × **NO.** Obviously.

(b) ✓ **YES.**

(c) ✓ **YES.**

2.11 Sparse regression weight vector. Suppose that x is an n-vector that gives n features for some object, and the scalar y is some outcome associated with the object. What does it maen if a regression model $\hat{y} = x^T\beta + v$ uses a sparse weight vector β? Give your answer in English, referring to $\hat{y}$ as our prediction of the outcome.

Solution:

It means that outcome $\hat{y}$ is depended on a few features.

2.12 Price change to maximize profit. A business sells n products, and it considering changing the price of *one* of the products to increase its total profits. A business analyst develops a regression model that (reasonably accurately) predicts the total profit when the product prices are changed, given by $\hat{P} = \beta^T x + P$, where the n-vector x denotes the fractional change in the product prices, $x_i = (p_i^{new} - p_i)/p_i$. Here P is the profit with the current price, $\hat{P}$ is the predicted profit with the changed prices, p_i is the current (positive) price of product i, and p_i^{new} is the new price of product i.

(a) What does it mean if $\beta_3 < 0$? (And yes, this can occur.)

(b) Suppose that you are given permission to change the price of *one* product, by up tp 1%, to increase total profit. Which product would you choose, and would you increase or decrease the price? By how much?

(c) Repeat part(b) assuming you are allowed to change the price of two products, each by up to 1%.

Solution:

(a) It means if you increase the price of 3th product, the total profit will decrease.

(b) I will choose the index for witch has the maximum absolute value of β, which means for the choosed index i, it satisfy $\beta_i = max(\beta)$, and if $\beta_i > 0$, increase the price, otherwise, decrease the price.

(c) Like (b), I just need to choose the first two index witch have the maximum absolute value of β.

3 Norm and distance

3.1 Distance between Boolean vectors. Suppose that x and y are Boolean n-vectors, which means that each of their entries are either 0 or 1. What is their distance $||x - y||$?

Solution:

For $i \in [1,\ldots,n]$, if x_i and y_i have same values, means that $x_i = y_i$, then $x_i - y_i = 0$; and if x_i and y_i have different values, means that $x_i \neq y_i$, then $|x_i - y_i| = 1$. Suppose there are k entries which satisfy $x_i \neq y_i$, according to the difinition of *norm* and *distance* of two vector: $||x - y|| = (x_1 - y_1)^2 + \cdots + (x_n + y_n)^2 = k$.

3.2 RMS value and average of block vectors. Let x be a block vector with two vector elements, $x = (a, b)$, where a and b are vectors of size n and m, respectively.

(a) Express $\mathbf{rms}(x)$ in terms of $\mathbf{rms}(a)$, $\mathbf{rms}(b)$, m, and n.

(b) Express $\mathbf{avg}(x)$ in terms of $\mathbf{avg}(a)$, $\mathbf{avg}(b)$, m, and n.

Solution:

(a) According to the difinition of RMS, for any k-vector v, $\mathbf{rms}(v) = \frac{||v||}{\sqrt{k}}$, therefore, $||v||^2 = k\mathbf{rms}^2(v)$, and as for the block vector x, we

have $||x||^2 = ||a||^2 + ||b||^2$, thus,

$$\begin{aligned}\mathbf{rms}(x) &= \frac{||x||}{\sqrt{n+m}} \\ &= \frac{\sqrt{||x||^2}}{\sqrt{n+m}} \\ &= \frac{\sqrt{||a||^2 + ||b||^2}}{\sqrt{n+m}} \\ &= \sqrt{\frac{n\mathbf{rms}^2(a) + m\mathbf{rms}^2(b)}{n+m}}\end{aligned}$$

(b) For any k-vector v, we have $\sum_1^k v_i = k\mathbf{avg}(v)$, so

$$(n+m)\mathbf{avg}(x) = \sum_1^{n+m} x_i = \sum_1^n a_i + \sum_1^m b_i = n\mathbf{avg}(a) + m\mathbf{avg}(b),$$

thus,

$$\mathbf{avg}(x) = \frac{n\mathbf{avg}(a) + m\mathbf{avg}(b)}{n+m}.$$

3.3 Reverse triangle inequality. Suppose a and b are vectors of same size. The triangle inequality state that $||a+b|| \le ||a|| + ||b||$. Show that we also have $||a+b|| \ge ||a|| - ||b||$. *Hints.* Draw a picture to get the idea. To show the inequality, apply the triangle inequality to $(a+b)+(-b)$.

Solution:

As the *Hints* tells us, we can derive that

$$||a|| = ||(a+b)+(-b)|| \le ||a+b|| + ||b||$$

where we use the fact that $||-b|| = ||b||$, so we can get $||a+b|| \ge ||a|| - ||b||$.

3.4 Norm identities. Verify that the following identities hold for any two vectors a and b of the same size.

(a) $(a+b)^T(a-b) = ||a||^2 - ||b||^2$.

(b) $||a+b||^2 + ||a-b||^2 = 2(||a||^2 + ||b||^2)$. This is called *parallelogram law*.

Solution:

(a) $(a+b)^T(a-b) = (a^T+b^T)(a-b) = a^Ta - a^Tb + b^Ta - b^Tb = a^Ta + b^Tb = ||a||^2 - ||b||^2$. We used the facts $a^Tb = b^Ta$ and $a^Ta = ||a||^2$.

(b) $||a+b||^2 + ||a-b||^2 = ||a||^2 + 2a^Tb + ||b||^2 + ||a||^2 - 2a^Tb + ||b||^2 = 2(||a||^2 + ||b||^2)$. We used the facts $\|x+y\| = \sqrt{\|x\|^2 + 2x^Ty + \|y\|^2}$ in Page-31 formula (3.1).

3.5 General norms. Any real-valued function f that satisfies the four properties given on Page 46 (nonnegative homogeneity, triangle inequality, nonnegativity, and definiteness) is called a *vector norm*, and is usually written as $f(x) = ||x||_{mn}$, where the subscript is some kind of identifier or mnemonic to identify it. The most commonly used norm is the one we use in this book, the Euclidean norm, which is sometimes written with the subscript 2, as $||x||_2$. Two other common vector norms for n-vectors are the *1-norm* $||x||_1$ and the *∞-norm* $||x||_\infty$, defined as

$$||x||_1 = |x_1| + \cdots + |x_n|, \qquad ||x||_\infty = \mathbf{max}\{|x_1|, \ldots, |x_n|\}.$$

These norms are the sum and the maximum of the absolute values of the entries in the vector, respectively. The 1-norm and the ∞-norm arise in some recent and advanced applications, but we will not encounter them in this book. Verify that the 1-norm and the ∞-norm satisfy the four norm properties listed on page 46.

Solution:

The four properties of norm on page 46 are:

- *Nonnegative homogeneity.* $\|\beta x\| = |\beta|\,\|x\|$.
- *Triangle inequality.* $\|x+y\| \le \|x\| + \|y\|$.
- *Nonnegativity.* $\|x\| \ge 0$.
- *Definitenese.* $\|x\| = 0$ only if $x = 0$.

For 1-norm:

- $\|\beta x\|_1 = |\beta x_1| + \cdots + |\beta x_n| = |\beta|(|x_1| + \cdots + |x_n|) = |\beta|\,\|x\|_1$.
- For any $i \in [1, \ldots, n]$, $|x_i + y_i| \le |x_i| + |y_i|$, so

$$\|x+y\| = |x_1+y_1| + \cdots + |x_n+y_n| \le |x_1| + |y_1| + \cdots + |x_n| + |y_n| = \|x\| + \|y\|\,.$$

- Obviously, $\|x\| = |x_1| + \cdots + |x_n| \geq 0 + \cdots + 0 = 0$.
- Only when all $iin[1,\ldots,n]$, $x_i = 0$, means $x = 0$, then $\|x\| = 0$.

For ∞-norm:

- $\|\beta x\|_\infty = \mathbf{max}\{|\beta x_1|,\ldots,|\beta x_n|\} = |\beta|\mathbf{max}\{|x_1|,\ldots,|x_n|\} = |\beta|\,\|x\|_\infty$
- Suppose $k \in [1,\ldots,n]$ satisfies $|x_k + y_k| = \mathbf{max}\{|x_1 + y_1|,\ldots,|x_n + y_n|\}$, means $\|x+y\| = |x_k + y_k| \leq |x_k| + |y_k|$, with $|x_k| \leq \mathbf{max}\{|x_1|,\ldots,|x_n|\} = \|x\|$ and $|y_k| \leq \mathbf{max}\{|y_1|,\ldots,|y_n|\} = \|y\|$, we have $\|x+y\| \leq \|x\| + \|y\|$.
- $\|x\| = |x_k| \geq 0$.
- $\forall i \in [1,\ldots,n]$, $|x_i| \geq 0$, so $\mathbf{math}\{|x_1|,\ldots,|x_n|\} = 0$ only when $\forall i \in [1,\ldots,n]$, $|x_i| = 0$, means $x = 0$.

3.6 Taylor approximation of norm. Find a general formula for the Taylor approximation of the function $f(x) = \|x\|$ near a given nonzero vector z. You can express the approximation in the form $\hat{f}(x) = a^T(x-z) + b$.

Solution:

We know Taylor approximation of $f(x)$ near z is written as $\hat{f}(x) = f(z) + \nabla f(z)^T(x-z)$. First, we derive what is $\dfrac{\partial f}{\partial x_i}$?

$$\begin{aligned}\frac{\partial f}{\partial x_i} &= \frac{\partial \sqrt{x_1^2 + \cdots + x_n^2}}{\partial x_i} \\ &= \frac{x_i}{\sqrt{x_1^2 + \cdots + x_n^2}} \\ &= \frac{x_i}{\|x\|}\end{aligned}$$

make $a = \nabla f(z) = \dfrac{z}{\|z\|}$, $b = \|z\|$, we have

$$\hat{f}(x) = a^T(x-z) + b = \frac{z^T}{\|z\|}(x-z) + \|z\|\,.$$

3.7 Chebyshev inequality. Suppose x is a 100-vector with $\mathbf{rms}(x) = 1$. What is the maximum nunber of entries of x that can satisy $|x_i| \geq 3$? If your answer is k, explain why no such vector can have $k+1$ entries with absolute values at least 3, ans give an example of a specific 100-vector that RMS

value 1, with k of its entries larger than 3 in absolute value.

Solution:

At first let's derive formular (3.2) on page 47. which is

$$\frac{k}{n} \leq (\frac{\mathbf{rms}(x)}{a})^2$$

As fact it quiet easy, with $\sqrt{n}\mathbf{rms}(x) = \|x\|$ and Chebyshev inequality $k \leq \|x\|^2 / a^2$.

Let $a = 3, n = 100, \mathbf{rms}(x) = 1$, we have $k \leq 11.1$, as k is an integer, so max $k = 11$.

If vector has $k+1 = 12$ entries with absolute value at least 3, the $\|x\|^2 = x_1^2 + \cdots + x_n^2 \geq 9(k+1) = 108$, then $\mathbf{rms}(x) = \|x\| / \sqrt{n} \geq \sqrt{108}/\sqrt{100} > 1$, it is wrong because we have $\mathbf{rms}(x) = 1$.

Example: $(3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 0, \ldots, 0)$.

3.8 Converse Chebyshev inequality. Show that at least one entry of a vector has absolute value at least as large as the RMS value of the vector.

Solution:

Suppose for all $i \in [1, \ldots, n]$, $|x_i| < \mathbf{rms}(x)$. Then $\|x\|^2 = x_1^2 + \cdots + x_n^2 < n\mathbf{rms}(x)^2$, with definition of RMS we have $\|x\|^2 = n\mathbf{rms}(x)^2$, then we get $\mathbf{rms}(x) < \mathbf{rms}(x)$, which cannot be happend, so at least one entry of a vector has absolute value as least as large as the RMS value of the vector.

3.9 Difference of squared distances. Determine whether the difference of the squared distances to two fixed vectors c and d, defined as

$$f(x) = \|x - c\|^2 - \|x - d\|^2,$$

is linear, affine, or neither. If it is linear, give its linear product representation, *i.e.*, an n-vector a for which $f(x) = a^T x$. If it is affine, give a and b for which $f(x) = a^T x + b$ holds for all x. If it is neither linear nor affine, give specific x, y, α, and β for which superposition fails. *i.e.*,

$$f(\alpha x + \beta y) \neq \alpha f(x) + \beta f(y).$$

(Provided $\alpha + \beta = 1$, this shows the function is neither linear line nor affine.)

Solution:

We have

$$\begin{aligned} f(x) &= \|x - c\|^2 - \|x - d\|^2 \\ &= \|x\|^2 - 2c^T x + \|c\|^2 - \|x\|^2 + 2d^T x - \|d\|^2 \\ &= 2(d^T - c^T)x + \|c\|^2 - \|d\|^2, \end{aligned}$$

let

$$a = 2(d - c), \qquad b = \|c\|^2 - \|d\|^2,$$

we have

$$f(x) = 2a^T x + b.$$

3.10 Nearest neighbor document. Consider the 5 Wikipedia pages in table 3.1 on page 51. What is the nearest neighbor of (the word count histogram vector of) 'Veterans Day' among the others? Does the answer make sense?

Solution:

The nearest neighbor of 'Veterans Day' is 'Memorial Day'.

3.11 neighboring electronic health records. Let $x_1, \ldots x_N$ be n-vectors that contain n features extracted from a set of N electronic health records (EHRs), for a population of N patients. (The features might involve patient attributes and current and past symptoms, diagnoses, test rusults, hospitalizations, procedures, and medications.) Briefly describe in words a practical use for identifying the 10 nearest neighbors of a given EHR (as measured by their associated feature vectors), among the other EHRs.

Solution:

The 10 nearest neighbors of a given EHR, if we know the what disease that this patient has, it may be true that other 10 patients may have the same disease.

3.12 Nearest point to a line. Let a and b be different n-vectors. The line passing through a and b is given by the set of vectors of the form

$(1-\theta)a+\theta b$, where θ is a scalar that determines the particular point on the line. (See page 18.)

Let x be any n-vector. Find a formula for the point p on the line that is closest to x. The point p is called the *projection* of x onto the line. Show that $(p-x) \perp (a-b)$, and draw a simple picture illustrating this in 2-D. *Hint.* Work with the square of the distance between a point on the line and x, *i.e.*, $\|(1-\theta)a+\theta b-x\|^2$. Expand this, and minimize over θ.

Solution:

Square distance between p and x is

$$\begin{aligned}\|p-x\|^2 &= \|(1-\theta)a+\theta b-x\|^2 \\ &= \|\theta(b-a)+(a-x)\|^2 \\ &= \|\theta(b-a)\|^2+2\theta(b-a)^T(a-x)+\|a-x\|^2 \\ &= \|b-a\|^2\theta^2+2(b-a)^T(a-x)\theta+\|a-x\|^2\end{aligned}$$

is a quadratic function about θ. In order to minimize the $\|p-x\|^2$, need

$$\theta = -\frac{2(b-a)^T(a-x)}{2\|b-a\|^2} = \frac{(b-a)^T(x-a)}{\|b-a\|^2}.$$

We have

$$1-\theta = \frac{\|b-a\|^2-(b-a)^T(x-a)}{\|b-a\|^2} = \frac{(b-a)^T(b-a)-(b-a)^T(x-a)}{\|b-a\|^2} = \frac{(b-a)^T(b-x)}{\|b-a\|^2}$$

So the formula of p is

$$p = (1-\theta)a+\theta b = \frac{(b-a)^T(b-x)a+(b-a)^T(x-a)b}{\|b-a\|^2}.$$

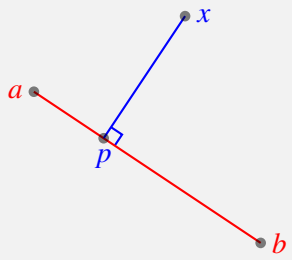
Now I will prove that $(p-x) \perp (a-b)$, consider following equation:

$$(p-x)^T(a-b) = \left(\frac{(b-a)^T(b-x)a+(b-a)^T(x-a)b-\|b-a\|^2 x}{\|b-a\|^2}\right)^T(a-b),$$

Let $v = b - a$, we have

$$\begin{aligned}
(p-x)^T(a-b) &= -\frac{v^T(b-x)a^Tv + v^T(x-a)b^Tv - \|b-a\|^2 x^Tv}{\|b-a\|^2} \\
&= -\frac{v^Tba^Tv - v^Txa^Tv + v^Txb^Tv - v^Tab^Tv - v^Tvx^Tv}{\|b-a\|^2} \\
&= -\frac{v^Tba^Tv + v^Tx(b^Tv - a^Tv) - a^Tvv^Tb - v^Tvx^Tv}{\|b-a\|^2} \\
&= -\frac{v^Tba^Tv - v^Tba^Tv + v^Tx(b-a)^Tv - v^Tvx^Tv}{\|b-a\|^2} \\
&= -\frac{v^Tba^Tv - v^Tba^Tv + v^Tvx^Tv - v^Tvx^Tv}{\|b-a\|^2} \\
&= 0.
\end{aligned}$$

which means $(p-x) \perp (a-b)$. Following is an example.



3.13 Nearest Nonnegative vector. Let x be an n-vector and define y as the nonnegative vector (*i.e.*, the vector with nonnegative entries) closest to x. Give an expression for the elements of y. Show that the vector $z = y - x$ is also nonnegative and the $z^Ty = 0$.

Solution:

Square distance between x and y is $\|x-y\|^2 = (x_1-y_1)^2 + \cdots + (x_n-y_n)^2$, in order to minimize its value, it's same to minimize evry single $(x_i-y_i)^2$. Because $y_i \geq 0$, if $x_i \geq 0$, then $y_i = x_i$ minimize $(x_i-y_i)^2$ to value 0; and if $x_i < 0$, then $y_i = 0$ minimize $(x_i-y_i)^2$ to value x_i^2. So expression of entries in y is:

$$\begin{cases} y_i = x_i, & if \quad x_i \geq 0 \\ y_i = 0, & if \quad x_i < 0 \end{cases}.$$

Therefore, expression of entries in z is:

$$\begin{cases} z_i = 0, & if \quad x_i \geq 0 \\ z_i = -x_i, & if \quad x_i < 0 \end{cases},$$

it's a nonnegative vector, and $z^T y = z_1 y_1 + \cdots + z_n y_n = 0$.

3.14 Nearest unit vector. What is the nearest neighbor of the n-vector x among the unit vectors $e_1, \ldots, e_n$?

Solution:

Square distance between x and i-the unit vector e_i is

$$\|x - e_i\|^2 = x_1^2 + \cdots + (x_i - 1)^2 + \cdots + x_n^2 = \|x\|^2 - 2x_i + 1,$$

suppose k-th entry x_k is the maximum value among entries in x, then $\|x - e_k\|^2$ has the minimum value.

3.15 Average, RMS value, and standard deviation. Use the formula (3.5) to show that for any vector x, the following two inequalities hold:

$$|\mathbf{avg}(x)| \leq \mathbf{rms}(x), \qquad \mathbf{std}(x) \leq \mathbf{rms}(x).$$

Is it possible to have equality in these inequalities? If $|\mathbf{avg}(x)| = \mathbf{rms}(x)$ is possible, give the conditions on x under which it holds. Repeat for $\mathbf{std}(x) = \mathbf{rms}(x)$.

Solution:

The formula (3.5) is

$$\mathbf{rms}(x)^2 = \mathbf{avg}(x)^2 + \mathbf{std}(x)^2.$$

So $\mathbf{rms}(x)^2 - \mathbf{avg}(x)^2 = \mathbf{std}(x)^2 \geq 0 \Rightarrow |\mathbf{avg}(x)^2| \leq \mathbf{rms}(x)^2$. Only when $std(x) = 0$ that equality holded, means all entries in x have same value.

The same $\mathbf{std}(x)^2 \leq \mathbf{rms}(x)$, equality holded only when $\mathbf{avg}(x) = 0$.

3.16 Effect of scaling and offset on average and standard deviation. Suppose x is an n-vector and α and β are scalars.

(a) Show that $\mathbf{avg}(\alpha x + \beta \mathbf{1}) = \alpha \mathbf{avg}(x) + \beta$

(b) Show that $\mathbf{std}(\alpha x + \beta \mathbf{1}) = |\alpha|\mathbf{std}(x)$

Solution:

(a)

$$\begin{aligned}\mathbf{avg}(\alpha x + \beta \mathbf{1}) &= \frac{\alpha x_1 + \beta + \cdots + \alpha x_n + \beta}{n} \\ &= \frac{\alpha x_1 + \cdots + \alpha x_n}{n} + \frac{\beta n}{n} \\ &= \alpha \mathbf{avg}(x) + \beta\end{aligned}$$

(b)

$$\begin{aligned}\mathbf{std}(\alpha x + \beta \mathbf{1}) &= \sqrt{\frac{(\alpha x_1 + \beta - \mathbf{avg}(\alpha x + \beta \mathbf{1}))^2 + \cdots + (\alpha x_n + \beta - \mathbf{avg}(\alpha x + \beta \mathbf{1}))^2}{n}} \\ &= \sqrt{\frac{(\alpha x_1 + \beta - \alpha \mathbf{avg}(x) - \beta)^2 + \cdots + (\alpha x_n + \beta - \alpha \mathbf{avg}(x) - \beta)^2}{n}} \\ &= \sqrt{\frac{\alpha^2((x_1 - \mathbf{avg}(x))^2 + \cdots + (x_n - \mathbf{avg}(x))^2)}{n}} \\ &= |\alpha| \sqrt{\frac{(x_1 - \mathbf{avg}(x))^2 + \cdots + (x_n - \mathbf{avg}(x))^2}{n}} \\ &= |\alpha| \mathbf{std}(x)\end{aligned}$$

3.17 Acerage and standard deviation of linear combination. Let $x_1, \ldots, x_k$ be n-vectors, and $\alpha_1, \ldots, \alpha_k$ be numbers, and consider the linear combination $z = \alpha_1 x_1 + \cdots + \alpha_k x_k$.

(a) Show that $\mathbf{avg}(z) = \alpha_1 \mathbf{avg}(x_1) + \cdots + \alpha_k \mathbf{avg}(x_k)$.

(b) Now suppose the vectors are uncorrelated, which means that for $i \neq j$, x_i and x_j are uncorrelated. Show that $\mathbf{std}(z) = \sqrt{\alpha_1^2 \mathbf{std}(x_1)^2 + \cdots + \alpha_k^2 \mathbf{std}(x_k)^2}$.

Solution:

(a)

$$\begin{aligned}\mathbf{avg}(z) &= z^T \mathbf{1}/n \\ &= (\alpha_1 x_1 + \cdots + \alpha_k x_k)^T \mathbf{1}/n \\ &= \alpha_1 x_1^T \mathbf{1}/n + \ldots \alpha_k x_k^T \mathbf{1}/n \\ &= \alpha_1 \mathbf{avg}(x_1) + \cdots + \alpha_k \mathbf{avg}(x_k)\end{aligned}$$

(b)

$$\begin{aligned} n\mathbf{std}(z)^2 &= \|z - \mathbf{avg}(z)\mathbf{1}\|^2 \\ &= \|\alpha_1(x_1 - \mathbf{avg}(x_1)\mathbf{1}) + \cdots + \alpha_k(x_k - \mathbf{avg}(x_k)\mathbf{1})\|^2 \\ &= \|\alpha_1\tilde{x_1} + \cdots + \alpha_k\tilde{x_k}\|^2 \\ &= \sum_1^k \alpha_i^2 \|\tilde{x_i}\|^2 + \sum_{i<j}^{i,j\in k} 2\rho_{ij}\alpha_i\alpha_j \|\tilde{x_i}\| \left\|\tilde{x_j}\right\| \\ &= n\alpha_1^2\mathbf{std}(x_1)^2 + \cdots + n\alpha_k^2\mathbf{std}(x_k)^2. \end{aligned}$$

Here we make $\tilde{x_i} = x_i - \mathbf{avg}(x)\mathbf{1}$ and ρ_{ij} is the correlated coefficients between x_i and x_j which allway holds $\rho_{ij} = 0$. Both sides divede n and make squar root, have

$$\mathbf{std}(z) = \sqrt{\alpha_1^2\mathbf{std}(x_1)^2 + \cdots + \alpha_k^2\mathbf{std}(x_k)^2}$$

.

3.18 Triangle euqality. When does the triangle inequality hold with equality, *i.e.*, what are the conditions on a and b to have $\|a+b\| = \|a\| + \|b\|$?

Solution:

With the verification on page 57, we know only when Cauchy-Schwarz inequality $a^T b \leq \|a\|\,\|b\|$ hold with equality that the Triangle inequality hold with equality. And the condition on which Cauchy-Schwarz inequality hold with equality is $a = 0$ or $b = 0$, or a is a scalar multiply b.

3.19 Norm of sum. Use the formula (3.1) and (3.6) to show the following:

(a) $a \perp b$ if and only if $\|a+b\| = \sqrt{\|a\|^2 + \|b\|^2}$.

(b) Nonzero vectors a and b make en acute angle if and only if $\|a+b\| > \sqrt{\|a\|^2 + \|b\|^2}$.

(c) Nonzero vectors a and b make en obtuse angle if and only if $\|a+b\| < \sqrt{\|a\|^2 + \|b\|^2}$.

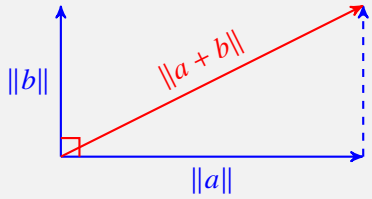
Draw a picture each case in 2-D.

Solution:

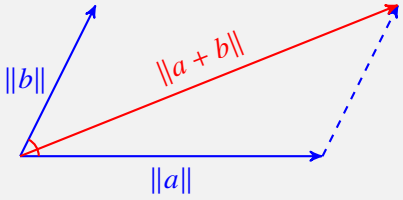
Suppose θ is the angle between a and b, according to the definition if angle, we have $a^T b = \|a\| \|b\| \cos\theta$, and we have

$$\|a+b\|^2 = \|a\|^2 + 2a^T b + \|b\|^2 = \|a\|^2 + 2\|a\| \|b\| \cos\theta + \|b\|^2 \tag{3.1}$$

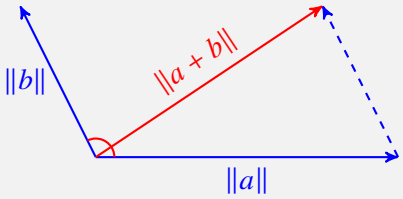
(a) If $a \perp b$ means $\theta = \pi/2 \Rightarrow \cos\theta = 0$, according to (3.1) we have $\|a+b\| = \sqrt{\|a\|^2 + \|b\|^2}$. Inverse make same sense.



(b) If $\theta < \pi/2 \Rightarrow \cos\theta > 0$, therefor $\|a\|^2 + 2\|a\| \|b\| \cos\theta + \|b\|^2 > \|a\|^2 + \|b\|^2$, Inverse too.



(c) Same as above questions.



3.20 Regression model sensitivity. Consider the regression model $\hat{y} = x^T\beta + v$, where $\hat{y}$ is the prediction, x is a feature vector, β is a coefficient vector, and v is the offset term. If x and $\tilde{x}$ are feature vectors with corresponding predictions $\hat{y}$ and $\tilde{y}$, show that $|\hat{y} - \tilde{y}| \leq \|\beta\| \|x - \tilde{x}\|$. This means that when $\|\beta\|$ is small, the prediction is not very sensitive to a change in the feature vector.

Solution:

$|\hat{y}-\tilde{y}| = |x^T\beta+v-\tilde{x}^T\beta-v| = |(x-\tilde{x})^T\beta|$, according to Cauchy-Schwarz inequality,

$$(x-\tilde{x})^T\beta| \leq \|\beta\|\,\|x-\tilde{x}\|\,,$$

therefore, we have

$$|\hat{y}-\tilde{y}| \leq \|\beta\|\,\|x-\tilde{x}\|\,.$$

3.21 Dirichlet energy of a signal. Suppose the T-vector x represents a time series or signal. The quantity

$$D(x) = (x_1 - x_2)^2 + \cdots + (x_{T-1} - x_T)^2,$$

the sum of the differences of adjacent values of the signal, is called the *Dirichlet energy* of the signal, named after the mathmatician Peter Gustav Lejeune Dirichlet. The Dirichlet energe is a measure of the roughness or wiggliness of the time series. It is sometimes divided by $T-1$, to give the mean square differece if the adjacent values.

(a) Express $D(x)$ in vector notation. (You can use the vector slicingm, vector addition or substraction, inner product, norm, and angle.)

(b) How small can $D(x)$ be? What signals x have the minimum value of the Dirichlet energy?

(c) Find a signal x with entries no more than one in absolute value that has the largest possible value of $D(x)$. Give the value of the Dirichlet energy achieved.

Solution:

(a) $D(x) = \|x_{1:T-1} - x_{2:T}\|^2$

(b) $D(x)_{min} = 0$ with $x_1 = x_2 = \cdots = x_T$.

(c) Beacuse $\forall i in [1,\ldots,T]$, $|x_i| \leq 1$, so

$$0 \leq |x_i - x_{i+1}| \leq 2 \Rightarrow |x_i - x_{i+1}|^2 \leq 4,$$

therefore

$$D(x) \leq 4(T-1).$$

We can make all odd entries in x be 1 and all even entries in x

be -1, like $x = [1,-1,1,-1,\dots]$, then $D(x)$ has maximum value $4(T-1)$.

3.22 Distance from Alto to Beijing. The surface of the earth is reasonable approximated as a sphere with radius $R = 6367.5km$. A location on the earth's surface is traditionally given by its latitude θ and its longitude λ, which correspond to angular distance from the equator and prime meridian, respectively. The 3-D coordinates of the location are given by

$$\begin{bmatrix} R\sin\lambda\cos\theta \\ R\cos\lambda\cos\theta \\ R\sin\theta \end{bmatrix}.$$

(In this coordinates system $(0,0,0)$ is the center of the earth, $R(0,0,1)$ is the North pole, and $R(0,1,0)$ is the point on the equator on the prime meridian, due south of the Royal Observatory outside London.)

The distance *Through the earth* between two locations (3-vectors) a and b is $\|a-b\|$. The distance *along the surface of the earth* between point a and b is $R\angle(a,b)$. Find these two distance between Palo Alto and Beijing, with latitudes and longitudes given below.

City	Latitude θ	Longitude λ
Beijign	39.914°	116.392°
Palo Alto	37.429°	−122.138°

Solution:

Let b be the coordinate of Beijing,which $b = (4374.9,-2171.0,4085.6)$.
Let p be the coordinate of Palo Alto, witch $p = (-4281.7,-2689.8,3870.0)$.
Therefore $\|b-p\| = 8674.8km$, and the angle between two vectors

$$\cos(\angle(b,p)) = \cos(\alpha) = \frac{b^T p}{\|b\|\,\|p\|} = 0.072,$$

we have $\alpha \approx 1.5$, so $R\angle(b,p) = 9543.2km$.
By the way, you can use Python module *Numpy.linalg.norm()* function to calculate norm, following are the code.

Listing 1: Python example for 3.22

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
```

```
# Code for Exerises 3.22
# Author: Utoppia
# Date  : 02 May 2020

import numpy as np
from numpy import linalg as LA

class Position:
    def __init__(self, name, latitude, longtitude):
        self.name = name
        self.latitude = latitude
        self.longtitude = longtitude

def creat_coodinate(R, latitude, longtitude):
    return np.array(
        [
            R * np.sin(longtitude * np.pi / 180.0) * np.cos(latitude * np.pi /
                180.0),
            R * np.cos(longtitude * np.pi / 180.0) * np.cos(latitude * np.pi /
                180.0),
            R * np.sin(latitude * np.pi / 180.0)
        ])

def main():
    R = 6367.5
    b = Position('Beijing',   39.914, 116.392)
    p = Position('Palo Alto', 37.429, -122.138)

    bv = creat_coodinate(R, b.latitude, b.longtitude)
    pv = creat_coodinate(R, p.latitude, p.longtitude)

    # Distance through the earth
    dis_1 = LA.norm(bv-pv)
    print('Distance through the earth between %s and %s is %.2f km' % (b.name, p.
        name, dis_1))

    # Angle between two points
    angle = np.arccos(np.dot(bv, pv) / LA.norm(bv) / LA.norm(pv))

    # Distance along the surface of the earth
    dis_2 = R * angle
    print('Distance along the surface of the earth between %s and %s is %.2f km'
```

```
        % (b.name, p.name, dis_2))

main()
```

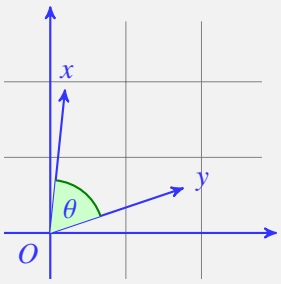
3.23 Angle between two nonnegative vectors. Let x and y be two nonzero n-vectors with nonnegative entries, *i.e.*, echa $x_i \geq 0$ and each $y_i \geq 0$. Show that the angle between x and y lies between 0 and 90°. Draw a picture for the case when $n = 2$, and give a short geometric explanation. When are x and y orthogonal?

Solution:

Use the θ denote the angle between x and y, then

$$\cos(\theta) = \frac{x^T y}{\|x\| \, \|y\|}.$$

Because all $x_i \geq 0$ and $y_i \geq 0$, therefore $x_i y_i \geq 0$, thus $x^T y \geq 0$. Therefore $\cos\theta \geq 0$, and with $-90^\circ \leq \theta \leq 90^\circ$, thus $0 \leq \theta \leq 90^\circ$.



3.24 Distance versus angle nearest neighbor. Suppose $z_1, \ldots, z_m$ is a collection of n-vectors, and x is another n-vector. The vector z_j is the (distance) nearest of x (among the given vectors) if

$$\|x - z_j\| \leq \|x - z_i\|, \quad i = 1, \ldots, m,$$

i.e., x has smallest distance to z_j. We say that z_j is the *angle nearest neighbor* of x if

$$\angle(x, z_j) \leq \angle(x, z)i), \quad i = 1, \ldots, m,$$

i.e., x has smallest angle to z_j.

(a) Given a simple specific numerical example where the (distance) nearest

neighbor is not the same as the angle nearest neighbor.

(b) Now suppose that vectors $z_1, \ldots, z_m$ are normalized, which means that $\|z_i\| = 1, i = 1, \ldots, m$. Show that in this case the distance nearest neighbor and the angle nearest neighbor are always the same. *Hint.* You can use the fact that arccos is a decreasing function, *i.e.*, for any u and v with $-1 \le u < v \le 1$, we have $\arccos(u) > \arccos(v)$.

Solution:

(a) Consider example like this:

$$n = 2, x = (2,2), z_1 = (0.5, 0.5), z_2 = (2, 2\sqrt{3}),$$

then

$$\|x - z_1\| = 1.5\sqrt{2} \approx 2.12, \qquad \|x - z_2\| = 2(\sqrt{3} - 1) \approx 1.46$$
$$\angle(x, z_1) = 0, \qquad \angle(x, z_2) = 30^\circ$$

You may notice that according to distance, z_2 is more nearest to x, but according to angle z_1 is the nearest neighbor of x.

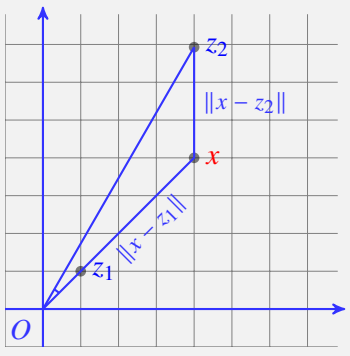
(b) Distance between x and z_i is

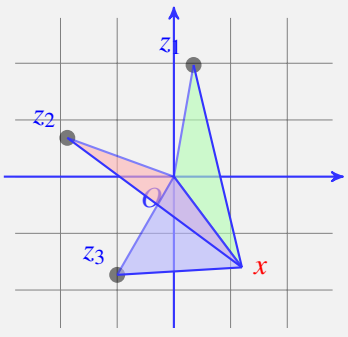
$$\|(\| x - z_i) = \sqrt{\|x\|^2 - 2\cos(\theta_i)\,\|x\|\,\|z_i\| + \|z_i\|^2} = \sqrt{-2\,\|x\|\cos(\theta_i) + \|x\|^2 + 1}$$

which means the distance between x and z is a function with parameter as their angle θ, as

$$\|(\| x - z) = f(\theta) = \sqrt{-2\,\|x\|\cos(\theta) + \|x\|^2 + 1}.$$

It is easy to know by the expression that function $f(\theta)$ an increasing function, which means the minimum θ hold the minimum value of $f(\theta)$.





3.25 Leveraging. Consider an asset with return time series over T periods given by the T-vector r. This asset has mean return μ and rish σ, which we assume is positive. We also consider cash as an asset, with return vector $\mu^{rf}\mathbf{1}$, where μ^{rf} is the cash interest rate per period. Thus, we model cash as an asset with return μ^{rf} and zero risk. (Ths upperscript in μ^{rf} stands for 'rish-free'.) We will create a simple portfolio consisting of the asset and cash. If we invest a fraction θ in the asset, and $1-\theta$ in cash, our portfolio return is given by the time series

$$p = \theta r + (1-\theta)\mu^{rf}\mathbf{1}.$$

We interpret θ as the fraction of our portfolio we hold in the asset. We allow the choices $\theta > 1$, or $\theta < 0$. In the first case we are *borrowing* cash ans using the proceeds to buy more of the asset, which is called *leveraging*. In the second case we are *shorting* the asset. When θ is between 0 ans 1 we are blending out investment in the asset and cash, which is a form of *hedging*.

(a) Derive a formula for the return and risk of the portfolio, *i.e.*, the mean and standard deviation of p. These should be expressed in terms of μ, σ, μ^{rf}, and θ. Check your formulas for the special case $\theta = 0$ and $\theta = 1$.

(b) Explain how to choose θ so the portfolio has a given target risk level σ^{tar} (which is positive). If there multiple values of θ that give the target risk, choose the one that results in the highest portfolio return.

(c) Assume we choose the value of θ as in part (b). When do we use leverage? When do we short the asset? When do we hedge? Your answer should be in English.

Solution:

(a) According to Exercise (3.16),

$$\begin{aligned}\mathbf{avg}(p) &= \arg(\theta r + (1-\theta)\mu^{rf}\mathbf{1}) \\ &= \theta \arg(r) + (1-\theta)\mu^{rf} \\ &= \theta\mu + (1-\theta)\mu^{rf}\end{aligned}$$

$$\begin{aligned}\mathbf{std}(p) &= \mathbf{std}(\theta r + (1-\theta)\mu^{rf}\mathbf{1}) \\ &= |\theta|\mathbf{std}(r) \\ &= |\theta|\sigma\end{aligned}$$

if $\theta = 0$, $p = \mu^{rf}\mathbf{1}$ and $\arg(p) = \mu^{rf}, \mathbf{std}(p) = 0$.
if $\theta = 1$, $p = r$ and $\arg(p) = \mu, \mathbf{std}(p) = \sigma$.

(b) let $\mathbf{std}(p) = \sigma^{tar}$, we get $|\theta|\sigma = \sigma^{tar}$, so

$$|\theta| = \frac{\sigma^{tar}}{\sigma}$$

which means $\theta = \frac{\sigma^{tar}}{\sigma}$ or $\theta = -\frac{\sigma^{tar}}{\sigma}$.
Let's look at the expression of $\arg(p)$, which can be written in $\arg(p) = \theta(\mu - \mu^{rf}) + \mu^{rf}$, if $\mu > \mu^{rf}$, means $\arg(p)$ will increasing while θ is increasing, in order to get highest portfolio return, in this case, we should choose $\theta = \frac{\sigma^{tar}}{\sigma}$.
If $\mu < \mu^{rf}$, means $\arg(p)$ will decreasing while θ is increasing, in order to get highest portfolio return, in this case, we should choose $\theta = -\frac{\sigma^{tar}}{\sigma}$.
If $\mu = \mu^{rf}$, means $\arg(p)$ always be μ^{rf} no matter what θ be, in this case, we can choose either values of θ.

(c) When $\mu > \mu^{rf}$ and $\sigma^{tar} > \sigma$, means $\theta > 1$, in this case, we are leveraging.
When When $\mu > \mu^{rf}$ and $\sigma^{tar} < \sigma$, means $0 < \theta < 1$, in this case, we are hedging.
When $\mu < \mu^{rf}$, means $\theta < 0$, in this case, we are shortting.
Summary, when the return of asset is higher than the return of the cash, and the expected risk level of our portfolio is higher than the risk of the asset, we should leverage. If the expected risk level is lower than the risk of asset, we should hedge. And if the return of asset is lower than the return of the cash, we should short the asset.

3.26 Times series auto-correlation. Suppose the T-vector x is a non-constant time series, with x_t the value at time (or period) t. Let $\mu = (\mathbf{1}^T x)/T$

denote its mean value. The *auto-correlation* of x is the function $R(\tau)$, defined for $\tau = 0, 1, \ldots$ as the correlation coefficient of the vectors $(x, \mu \mathbf{1}_\tau)$ and $(\mu \mathbf{1}_\tau, x)$. (The subscript τ denotes the length of the ones vector.) Both of these vectors also have mean μ. Roughly speaking, $R(\tau)$ tells us how correlated the time series is with a version of itself lagged or shifted by τ periods. (The argument τ is called the lag.)

(a) Explain why $R(0) = 1$, and $R(\tau) = 0$ for $\tau \geq T$.

(b) Let z denote the standardized or z-scored version of x (see page 56). Show that for $\tau = 0, \ldots, T-1$,

$$R(\tau) = \frac{1}{T} \sum_{t=1}^{T-\tau} z_t z_{t+\tau}.$$

(c) Find the auto-correlation for the time series $x = (+1, -1, +1, -1, \ldots, +1, -1)$. You can assume that T is even.

(d) Suppose x denote the number of meals served by a restaurant on day τ. It is observed that $R(7)$ is fairly high, and $R(4)$ is also high, but not as high. Given an English explanation of why this might be.

Solution:

Let $p = (x, \mu \mathbf{1}_\tau)$ and $q = (\mu \mathbf{1}_\tau, x)$. According to definition,

$$R(\tau) = \frac{\tilde{p}^T \tilde{q}}{\|\tilde{p}\| \, \|\tilde{q}\|}.$$

here

$$\tilde{p} = (\tilde{x}, \mathbf{0}_\tau), \qquad \tilde{q} = (\mathbf{0}_\tau, \tilde{x}).$$

(a) if $\tau = 0$, $p = x$ and $q = x$, $R(0) = \dfrac{\tilde{x}^T \tilde{x}}{\|\tilde{x}\| \, \|\tilde{x}\|} = 1$. if $\tau > T$, According to the inner production of block vector, which means

$$\begin{aligned} \tilde{p}^T \tilde{q} &= (\tilde{x}, \mathbf{0}_\tau)^T (\mathbf{0}_\tau, \tilde{x}) \\ &= \tilde{x}^T \mathbf{0}_T + \mathbf{0}_{\tau - T}^T \mathbf{0}_{\tau - T} + \mathbf{0}_T^T \tilde{x} \\ &= 0 \end{aligned}$$

Thus, when $\tau > T$, $R(\tau) = 0$.

(b) Let $u = \tilde{p}/\mathbf{std}(p)$, $v = \tilde{q}/\mathbf{std}(q)$, we can rewrite the expression of $R(\tau)$ as

$$R(\tau) = u^T v/(T + \tau).$$

Now let's look what is the expression of u and v.

$$\begin{aligned}(T+\tau)\mathbf{std}(p)^2 &= \|\tilde{p}\|^2 \\ &= \|(\tilde{x}, \mathbf{0}_\tau)\|^2 \\ &= \|\tilde{x}\|^2 \\ &= T\mathbf{std}(x)^2\end{aligned}$$

So we have $\mathbf{std}(p) = \sqrt{T/(T+\tau)}\mathbf{std}(x)$, use same method we can get $\mathbf{std}(q) = \sqrt{T/(T+\tau)}\mathbf{std}(x)$.

So

$$\begin{aligned}u &= \tilde{p}/\mathbf{std}(p) \\ &= (\tilde{x}, \mathbf{0}_\tau))/(\sqrt{T/(T+\tau)}\mathbf{std}(x)) \\ &= \sqrt{(T+\tau)/T}(z, \mathbf{0}_\tau)\end{aligned}$$

Also, we have $v = \sqrt{(T+\tau)/T}(\mathbf{0}_\tau, z)$, so

$$\begin{aligned}R(\tau) &= u^T v/(T+\tau) \\ &= \frac{1}{T}(z, \mathbf{0}_\tau)^T(\mathbf{0}_\tau, z) \\ &= \frac{1}{T}(\mathbf{0}_\tau, z)^T(z, \mathbf{0}_\tau) \\ &= \frac{1}{T}(z_1 z_{\tau+1} + \cdots + z_{T-\tau} z_T) \\ &= \frac{1}{T}\sum_{t=1}^{T-\tau} z_t z_{t+\tau}\end{aligned}$$

(c) According to part (b), for given time series x, for $0 < \tau < T$:

If τ is even, $z_t z_{t+\tau} = 1$ for all $t = 1, \ldots, T-\tau$, so $R(\tau) = \dfrac{T-\tau}{T}$.

If τ is odd, $z_t z_{t+\tau} = -1$ for all $t = 1, \ldots, T-\tau$, so $R(\tau) = \dfrac{\tau - T}{T}$.

So we can have:

$$R(\tau) = \begin{cases} 1, & \text{if } \tau = 0 \\ \dfrac{T-\tau}{T}, & \text{if } 0 < \tau < T \text{ and } \tau \text{ is even} \\ \dfrac{\tau - T}{T}, & \text{if } 0 < \tau < T \text{ and } \tau \text{ is odd} \\ 0, & \text{if } \tau \geq T \end{cases}$$

(d) As my opinion, the meals served every day in a week have high

correlation, *i.e.*, the meals served in every Monday for different weeks have high correlation, that why $R(7)$ and $R(14)$ is high. And this correlation is more higher when the datas are more 'closer', which mean $R(7)$ is more higher than $R(14)$.

3.27 Another measure of the spread of the entries of a vector. The standard deviation is a measure of how much the entries of a vector differ from their mean value. Another measure of how much the entries of an n-vector x differ from each other, called the *mean square difference*, is defined as

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

(The sum means that you should add up that n^2 terms, as the indices i and j each range from 1 to n.) Show that $MSD(x) = 2\mathbf{std}(x)^2$. *Hint.* First observe that $MSD(\tilde{x}) = MSD(x)$, where $\tilde{x} = x - \mathbf{avg}(x)\mathbf{1}$ is the de-meaned vector. Expand the sum and recall that $\sum_{i=1}^{n}\tilde{x} = 0$.

Solution:

Let $\tilde{x} = x - \mathbf{avg}(x)\mathbf{1}$ be the de-meaned vector of x, we have

$$MSD(\tilde{x}) = \frac{1}{n^2}\sum_{i,j=1}^{n}(x_i - \mathbf{avg}(x) - (x_j - \mathbf{avg}(x)))^2 = \frac{1}{n^2}\sum_{i,j=1}^{n}(x_i - x_j)^2 = MSD(x).$$

In another way, we expand the expression of $MSD(\hat{x})$ as following:

$$\begin{aligned}
n^2 MSD(\tilde{x}) &= \sum_{i,j=1}^{n}(\tilde{x}_i - \tilde{x}_j)^2 \\
&= \sum_{i,j=1}^{n}\tilde{x}_i^2 - 2\tilde{x}_i\tilde{x}_j + \tilde{x}_j^2 \\
&= \sum_{i=1}^{n} n\tilde{x}_i^2 - 2\sum_{i,j=1}^{n}\tilde{x}_i\tilde{x}_j + \sum_{j=1}^{n} n\tilde{x}_j^2 \\
&= n\|\tilde{x}\|^2 - 2\sum_{i=1}^{n}\tilde{x}_i\sum_{j=1}^{n}\tilde{x}_j + n\|\tilde{x}\|^2 \\
&= 2n^2\mathbf{std}(x)^2 - 2\sum_{i=1}^{n}\tilde{x}_i 0 \\
&= 2n^2\mathbf{std}(x)^2
\end{aligned}$$

Both sides divided by n^2, get $MSD(x) = MSD(\tilde{x}) = 2\mathbf{std}(x)^2$. Here we use the fact $\|\tilde{x}\|^2 = n\mathbf{std}(x)^2$ and $\sum_{i=1}^n \tilde{x} = 0$.

3.28 Wieghted norm. On page 51 we discuss the importance of choosing the units or scaling for the individual entries of vectors, when they represent heterogeneous quantities. Anthor approach is to use a *weighted norm* of a vector x, defined as

$$\|x\|_w = \sqrt{w_1 x_1^2 + \cdots + w_n x_n^2},$$

where $w_1, \ldots, w_n$ are given positive *weights*, used to assign more or less importance to the different elements of the n-vector x. If all the weights are one, the weighted norm reduces to the usual ('unweighted') norm. It can be shown that the weighted norm is a general norm, *i.e.*, it satisfies the four norm properties listed on page 46. Following the discussion on page 51, one common rule of thumb is to choose the weight w_i as the inverse of the typical value of x_i^2 in the application.

A version of the Cauchy-Schwarz inequality holds for weighted norms: For any n-vector x and y, we have

$$|w_1 x_1 y_1 + \cdots + w_n x_n y_n| \leq \|x\|_w \, \|y\|_w \, .$$

(The expression inside the absolute value on the left-hand side is sometimes called the weighted inner product of x and y.) Show that this inequality holds. *Hint.* Consider the vector $\tilde{x} = (x_1\sqrt{w_1}, \ldots, x_n\sqrt{w_n})$ and $\tilde{y} = (y_1\sqrt{w_1}, \ldots, y_n\sqrt{w_n})$, and use the (standard) Cauchy-Schwarz inequality.

Solution:

Consider the vector $\tilde{x} = (x_1\sqrt{w_1}, \ldots, x_n\sqrt{w_n})$ and $\tilde{y} = (y_1\sqrt{w_1}, \ldots, y_n\sqrt{w_n})$. With the Cauchy-Schwarz inequality, we have

$$|\tilde{x}^T \tilde{y} \leq \|\tilde{x}\| \, \|\tilde{y}\| \, .$$

And

$$\tilde{x}^T \tilde{y} = w_1 x_1 y_1 + \cdots + w_n x_n y_n \, \|\tilde{x}\| \, \|\tilde{y}\| = \|x\|_w \, \|y\|_w \, .$$

Thus

$$|w_1 x_1 y_1 + \cdots + w_n x_n y_n| \leq \|x\|_w \, \|y\|_w \, .$$

4 Clustering

4.1 Minimizing mean sqaure distance to a set of vectors. Let $x_1, \ldots, x_L$ be a collection of n-vectors. In this exercise you will fill in the missing parts of the argument to show that the vector z which minimize the sum-square distance to the vectors,

$$J(z) = \|x_1 - z\|^2 + \cdots + \|x_L - z\|^2,$$

is the average or centroid of the vectors, $\bar{x} = (1/L)(x_1 + \cdots + x_L)$. (This result is used in one of the steps in the k-means algorithm. But here we have simplified the notation.)

(a) Explain why, for any z, we have

$$J(z) = \sum_{i=1}^{L} \|x_i - \bar{x} - (z - \bar{x})\|^2 = \sum_{i=1}^{L} (\|x_i - \bar{x}\|^2 - 2(x_i - \bar{x})^T (z - \bar{x})) + L \, \|z - \bar{x}\|^2 \, .$$

(b) Explain why $\sum_{i=1}^{L} (x_i - \bar{x})^T (z - \bar{x}) = 0$. *Hint.* Write the left-hand side as

$$(\sum_{i=1}^{L} (x_i - \bar{x}))^T (z - \bar{x}),$$

and argue that the left-hand side vector is 0.

(c) Combine the results of (a) and (b) to get $J(z) = \sum_{i=1}^{L} \|x_i - \bar{x}\|^2 + L \, \|z - \bar{x}\|^2$. Explain why for any $z \neq \bar{x}$, we have $J(z) > J(\bar{x})$. This shows that the choice $z = \bar{x}$ minimizes $J(z)$.

Solution:

(a) In order to simplify the notation, we use $u_i = x_i - \bar{x}$, and $v = z - \bar{x}$.

Then

$$\begin{aligned} J(z) &= \sum_{i=1}^{L} \|x_i - \bar{x} - (z - \bar{x})\|^2 \\ &= \sum_{i=1}^{L} \|u_i - v\|^2 \\ &= \sum_{i=1}^{L} (u_i - v)^T (u_i - v) \\ &= \sum_{i=1}^{L} u_i^T u_i - 2u_i^T v + v^T v \\ &= \sum_{i=1}^{L} (\|u_i\|^2 - 2u_i^T v) + L\|v\|^2 \\ &= \sum_{i=1}^{L} (\|x_i - \bar{x}\|^2 - 2(x_i - \bar{x})^T (z - \bar{x})) + L\|z - \bar{x}\|^2 \end{aligned}$$

(b) Vector $z - \bar{x}$ is a fixed vector, so we have

$$\begin{aligned} \sum_{i=1}^{L} (x_i - \bar{x})^T (z - \bar{x}) &= (\sum_{i=1}^{L} (x_i - \bar{x}))^T (z - \bar{x}) \\ &= (\sum_{i=1}^{L} x_i - L\bar{x})^T (z - \bar{x}) \\ &= (\sum_{i=1}^{L} x_i - \sum_{i=1}^{l} x_i)^T (z - \bar{x}) \\ &= \mathbf{0}^T (z - \bar{x}) \\ &= 0 \end{aligned}$$

(c) Now we have

$$J(z) = \sum_{i=1}^{L} \|x_i - \bar{x}\|^2 + L\|z - \bar{x}\|^2,$$

and

$$J(\bar{x} = \sum_{i=1}^{L} \|x_i - \bar{x}\|^2 .)$$

For any $z \neq \bar{x}$,

$$J(z) - J(\bar{x}) = \|z - \bar{x}\|^2 > 0.$$

i.e., for any $z \neq \bar{x}$, we have $J(z) > J(\bar{x})$, so the $z = \bar{x}$ minimize the $J(z)$.

4.2 k-means with nonnegative, proportions, or Boolean vectors. Suppose that the vectors $x_1, \ldots, x_N$ are clustered using k-means, with gourp representations $z_1, \ldots, z_k$.

(a) Suppose the original vectors x_i are nonnegative, *i.e.*, their entries are nonnegative. Explain why the representations z_j are also nonnegative.

(b) Suppose the original vector x_i represent proportions, *i.e.*, their entries are nonnegative and sum to one. (This is the case when x_i are word count histograms, for example.) Explain why the representations z_j also represent proportions, *i.e.*, their entries are nonnegative and sum to one.

(c) Suppose the original vector x_i are Boolean, *i.e.*, their entries are either 0 or 1. Give an interpretation of $(z_j)_i$, the $i$the entry of the jth gourp representative.

Solution:

Recall how we determine the $j$the group G_j's representation z_j in k-means algorithm, we use the formula

$$z_j = (1/|G_j|) \sum_{k \in G_j} x_k.$$

(a) If x_i are nonnegative, so $\sum_{k \in G_j} x_k$ are nonnegative, with $|G_j|$ are nonnegative, thus z_j are nonnegative.

(b) If x_i are proportions, according to part (a), we know z_j are nonnegative. The sum of entries of z_j is

$$\begin{aligned} \mathbf{1}^T z_j &= \mathbf{1}^T (1/|G_j|) \sum_{k \in G_j} x_k \\ &= (1/|G_j|) \sum_{k \in G_j} \mathbf{1}^T x_k \\ &= (1/|G_j|) \sum_{k \in G_j} 1 \\ &= (1/|G_j|)|G_j| \\ &= 1 \end{aligned}$$

Thus, z_j are proportions.

(c) $(z_j)_i = (1/|G_j|)\sum_{k\in G_j}(x_k)_i$ means the ith entry of vectors which assigned to jth group has $(z_j)_i$ possibility to be one.

4.3 Linear separation in 2-way partitioning. Clustering a collection of vectors into $k = 2$ groups is called 2-way partitioning, since we are partitioning the vectors into 2 groups, with index sets G_1 and G_2. Suppose we run k-means, with $k = 2$, on the n-vectors $x_1, \ldots, x_N$. Show that there is a nonzero vector w and a scalar v which satisfy

$$w^T x_i + v \geq 0 \text{ for } i \in G_1, \qquad w^T x_i + v \leq 0 \text{ for } i \in G_2.$$

In other words, the affine function $f(x) = w^T x + v$ is greater than or equal to zero on the first group, and less than or equal to zero on the second group. This is called *linear separation* of the two groups (although *affine separation* would be more accurated.)

Hint. Consider the function $\|x - z_1\|^2 - \|x - z_2\|^2$, where z_1 and z_2 are the group representatives.

Solution:

Consider function $f(x) = \|x - z_2\|^2 - \|x - z_1\|^2$, it is easy to verify

- for $i \in G_1$, $f(x_i) \geq 0$;
- for $i \in G_2$, $f(x_i) \leq 0$;

Now we expand $f(x)$ and rewrite it in the format of affine function:

$$\begin{aligned} f(x) &= \|x - z_2\|^2 - \|x - z_1\|^2 \\ &= (x - z_2)^T(x - z_2) - (x - z_1)^T(x - z_1) \\ &= \|x\|^2 - 2z_2^T x + \|z_2\|^2 - \|x\|^2 + 2z_1^T x - \|z_1\|^2 \\ &= 2(z_1 - z_2)^T x + \|z_2\|^2 - \|z_1\|^2 \end{aligned}$$

Let $w = 2(z_1 - z_2), v = \|z_2\|^2 - \|z_1\|^2$, rewrite $f(x)$ as

$$f(x) = w^T x + v.$$

And we had verify this function satisfy the requests.

4.4 Pre-assigned vectors. Suppose that some of the vectors $x_1, \ldots, x_N$ are assigned to specific groups. For example, we might insist that x_{27} be assigned to group 5. Suggest a simple modification of the k-means algorithm that respects this requirement. Describe a practical example where this might arise, when each vector represents n features of a medical patients.

Solution:

We can use the initialization with $z_5 = x_{27}$, and do not update the group of x_{27} in every iteration.

5 Linear Independence

5.1 Linear independence of the stacked vectors. Consider the stacked vectors

$$c_1 = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix}, \ldots, c_k = \begin{bmatrix} a_k \\ b_k \end{bmatrix},$$

(a) where $a_1, \ldots, a_k$ are linear independent. (We make no assumptions about the vectors $b_1, \ldots, b_k$.) Can we conclude that the stacked vectors $c_1, \ldots, c_k$ are linearly independent?

(b) Now suppose that $a_1, \ldots, a_k$ are linearly dependent. (Again, with no assumptions about $b_1, \ldots, b_k$.) Can we conclude that the stacked vectors $c_1, \ldots, c_k$ are linearly dependent?

Solution:

(a) ✓ **YES.** Vectors $c_1, \ldots, c_k$ are linearly independent. Suppose $\alpha_1, \ldots, \alpha_k$ satisfy

$$\alpha_1 c_1 + \cdots + \alpha_k c_k = 0$$

we have

$$\alpha_1 a_1 + \cdots + \alpha_k a_k = 0.$$

Due to vectors $a_1, \ldots, a_k$ are linearly independent, that

$$\alpha_1 = \cdots = \alpha_k = 0,$$

means vectors $c_1, \ldots, c_k$ are linearly independent.

(b) × **NO.** We give an example which $k = 2$ and

$$a_1 = (1), \quad a_2 = (1), \quad b_1 = (1), \quad b_2 = (2),$$

then we have

$$c_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad c_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

In this case, vectors a_1, a_2 are linearly dependent, but vectors c_1, c_2 are linearly independent.

5.2 A superising discovery. An intern at a quantitative hedge fund examines the daily returns of around 400 stocks over one year (which has 250 trading days). She tells her surervisor that she has discoverd that the returns of one of the stocks, Google(GOOG), can be expressed as a linear combination if the others, which many stocks that are unrelated to Google (say, in a different type of business or sector).

Her supervisor then say:" It is overwhelmingly unlikely that a linear combination of the returns of unrelated companies can reproduce the daily return of GOOG. So you've made a mistake in your calculations."

Is the supervisor right? Did the intern make a mistake? Give a very brief explanation.

Solution:

I think the supervisor is wrong.

5.3 Replicating a cash flow with single-period loans. We continue the example described on page 93. Let c be any n-vector representing a cash flow over n period. Find the coefficients $\alpha_1, \ldots, \alpha_n$ of c in its expansion in basis $e_1, l_1, \ldots, l_{n-1}$, *i.e.*,

$$c = \alpha_1 e_1 + \alpha_2 l_1 + \cdots + \alpha_{n-1} l_{n-1}.$$

Verify that α_1 is the net present value (NPV) of the cash flow c, defined on page 22, with interest rate r. *Hint.* Use the same type of argument that was used to show that $e_1, l_1, \ldots, l_{n-1}$ are linear independent. Your method will find α_n first, then α_{n-1}, and so on.

Solution:

Recall that

$$l_i = \begin{bmatrix} \mathbf{0}_{i-1} \\ \mathbf{1} \\ -(1+r) \\ \mathbf{0}_{n-i-1} \end{bmatrix}, \qquad i = 1, \ldots, n-1,$$

Suppose that

$$c = \alpha_1 e_1 + \alpha_2 l_1 + \cdots + \alpha_{n-1} l_{n-1}.$$

We can express this as

$$\begin{bmatrix} \alpha_1 + \alpha_2 \\ \alpha_3 - (1+r)\alpha_2 \\ \vdots \\ \alpha_n - (1+r)\alpha_{n-1} \\ -(1+r)\alpha_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix}$$

The last entry is $-(1+r)\alpha_n = c_n$, which implies that $\alpha_n = -c_n/(1+r)$. Using $\alpha_n = -c_n/(1+r)$, the second to last entry becomes $-c_n/(1+r) - (1+r)\alpha_{n-1} = c_{n-1}$, so we conclude that $\alpha_{n-1} = -c_{n-1}/(1+r) - c_n/(1+r)^2$. continuing this we will find

$$\alpha_i = -c_i/(1+r) - c_{i-1}/(1+r)^2 - \cdots - c_n/(1+r)^{n-i+1}, \quad \text{for } i = 2, \ldots, n.$$

and

$$\alpha_1 = c_1 + \frac{c_2}{1+r} + \cdots + \frac{c_n}{(1+r)^{n-1}},$$

which is the net present value (PNV) of the cash flow c.

5.4 Norm of linear combination of orthonormal vectors. Suppose $a_1, \ldots, a_k$ are orthonormal n-vectors, and $x = \beta_1 a_1 + \cdots + \beta_k a_k$, where $\beta_1, \ldots, \beta_k$ are scalars. Express $\|x\|$ in terms of $\beta = (\beta_1, \ldots, \beta_k)$.

Solution:

For any $j = 1, \ldots, k$, we have $a_j^T x = \beta_j$, with the fact $a_j^T a_i = 0$ if $i \neq j$ and $a_j^T a_j = 1$. Therefore

$$\begin{aligned} \|x\|^2 &= x^T x \\ &= (\beta_1 a_1 + \cdots + \beta_k a_k)^T x \\ &= \sum_{i=1}^{k} \beta_i a_i^T x \\ &= \sum_{i=1}^{k} \beta_i^2 \\ &= \|\beta\|^2 . \end{aligned}$$

Thus, $\|x\| = \|\beta\|$.

5.5 Orthogonalizing vectors. Suppose that a and b are any n-vectors. Show that we can always find a scalar γ so that $(a - \gamma b) \perp b$, and that γ is unique if $b \neq \mathbf{0}$. (Give a formula for the scalar γ.) In other words, we can always substract a multiple of a vector from another one, so that the result is orthonormal to the original vector. The orthogonalizing step in the Gram-Schmidt algorithm is an application of this.

Solution:

$(a - \gamma b) \perp b$ means $(a - \gamma b)^T b = 0$, means

$$a^T b = \gamma \|b\|^2 .$$

If $b \neq \mathbf{0}$,

$$\gamma = \frac{a^T b}{\|b\|^2}.$$

5.6 Gram-Schmidt algorithm. Consider the list of n-vectors

$$a_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad a_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad \ldots, \qquad a_n = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

(The vector a_i has its first i entries equal to one, and the remaining entries zero.) Describe what happens when you run the Gram-Schmidt algorithm on this list of vectors, *i.e.*, say what $q_1,\ldots,q_n$ are. Is $a_1,\ldots,a_n$ a bisis?

Solution:

Easy to know $\tilde{q}_1 = a_1$ and $\|\tilde{q}_1\| = 1$, there for $q_1 = a_1 = e_1$.
In the second iteration, $\tilde{q}_2 = a_2 - (q_1^T a_2)q_1 = e_2$, therefore $q_2 = e_2$.
Suppose for $i = 1,\ldots,k-1$, we have $q_i = e_i$, we will derive $q_k = e_k$:

$$\begin{aligned}\tilde{q}_k &= a_k - (q_1^T a_k)q_1 - \cdots - (q_{k-1}^T a_k)q_{k-1} \\ &= a_k - (e_1^T a_k)e_1 - \cdots - (e_{k-1}^T a_k)e_{k-1} \\ &= a_k - e_1 - \cdots - e_{k-1} \\ &= e_k\end{aligned}$$

Therefore

$$q_k = \tilde{q}_k/\|\tilde{q}_k\| = e_k$$

Summary, $\forall i = 1,\ldots,n \quad q_i = e_i$.4 And because the Gram-Schmidt algorithm will run to end, means $a_1,\ldots,a_n$ are linearly independent, So, yes, they are bisis.

5.7 Running Gram-Schmidt algorithm twice. We run the Gram-Schmidt algorithm once on a given a set of vectors $a_1,\ldots,a_k$ (we assume this is successful), which gives the vectors $q_1,\ldots,q_k$. Then we run the Gram-Schmidt algorithm on the vectors $q_1,\ldots,q_k$, which produces the vectors $z_1,\ldots,z_k$. What can you say about $z_1,\ldots,z_k$?

Solution:

The vectors $z_1,\ldots,z_k$ are equal to $q_1,\ldots,q_k$, *i.e.*, $z_i = q_i$ for $i = 1,\ldots,k$.
It is easy to know $\tilde{z}_1 = q_1$, with fact $\|q_i\| = 1$, therefore $z_1 = \tilde{z}_1 = q_1$.
Suppose for all $i = 1,\ldots,n-1, \quad n \geq 2$, we have $z_i = q_i$. Then

$$\tilde{z}_n = p_n - (z_1^T p_n)z_1 - \cdots - (z_{n-1}^T p_n)z_{n-1} = p_n,$$

with the fact that $z_j^T p_n = p_j^T p_n = 0$ for all $j = 1,\ldots,n-1$. Thus

$$z_n = \tilde{z}_p/\|\tilde{z}_n\| = p_n.$$

Summary, for all $i = 1,\ldots,k$, we have $z_i = p_i$.

5.8 Early termination of Gram-Schmidt algorithm. When the Gram-Schmidt algorithm is run on a particular list of 10 15-vectors, it terminates in the iteration 5 (since $\tilde{q}_5 = 0$). Which of the following must be true?

(a) a_2, a_3, a_4 are linearly independent.

(b) a_1, a_2, a_5 are linearly dependent.

(c) a_1, a_2, a_3, a_4, a_5 are linearly dependent.

(d) a_4 is nonzero.

Solution:

(a) ✓ **YES.**

(b) × **NO.**

(c) ✓ **YES.**

(d) ✓ **YES.** If a_4 is zero, which means $\tilde{q}_4 = 0$, and the Gram-Schmidt algorithm would terminates in iteration 4, not will be 5.

5.9 Aparticular computer can carry out the Gram-Schmidt algorithm on a list of $k = 1000$ n-vectors, with $n = 1000$, in around 2 seconds. About how long would you expect it to take to carry out the Gram-Schmidt algorithm with $\tilde{k} = 500$ $\tilde{n}$-vectors, with $\tilde{n} = 1000$?

Solution:

The complexity of the Gram-Schmidt algorithm is $2nk^2$. So it would take about $2 \times 5^2 \div 10^2 \div 10 = 0.05$ second to carry out.

6 Matrices

6.1 Matrix and vector notation. Suppose $a_1,\ldots,a_n$ are m-vectors. Determine whether each expression below makes sense (*i.e.*, uses valid notation). If the expression does make sense, give its dimensions.

(a) $\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$ ✓ **YES**. This is an mn-vector, and it is a block vector.

(b) $\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$ ✓ **YES**. This is a $n \times m$ matrix.

(c) $\begin{bmatrix} a_1 & \dots & a_n \end{bmatrix}$ ✓ **YES**. This is an $m \times n$ matrix.

(d) $\begin{bmatrix} a_1^T & \dots & a_n^T \end{bmatrix}$ × **NO**.

6.2 Matrix notation. Suppose the block matrix

$$\begin{bmatrix} A & I \\ I & C \end{bmatrix}$$

makes sense, where A is a $p \times q$ matrix. What are the dimensions of C?

Solution:

Upper right square unit matrix I is $p \times p$, below left square unit I is $q \times q$. So the dimensions of C is $q \times p$.

6.3 Block matrix. Assuming the matrix

$$K = \begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix}$$

makes sense, which of the following statements must be true? ('Must be true' means that it follows with no conditional assumptions.)

(a) K is square.

(b) A is square or wide.

(c) K is symmetric, *i.e.*, $K^T = K$.

(d) The identity and zero submatrices in K have the same dimensions.

(e) The zero submatrix is square.

Solution:

Suppose the size of the identity matrix is $n \times n$, size of A is $p \times q$. Look at the left part of matrix K, we derive that identity matrix I and A have same columns, means $q = n$. And we can derive that zero matrix has same columns with A^T which is p, and has same rows with A which is p, too. So zero matrix is square.

For part (a) if $p \neq n$, K cannot be square.

For part (b) if $p > q$ means $p > n$, A could be tall.

For part (c) $K^T = \begin{bmatrix} I^T & A^T \\ (A^T)^T & 0^T \end{bmatrix} = K$.
For part (d) if $p \neq n$, The identity and zero submatrices have different dimensions.
For part (e) yes the zero submatrix is square with size $p \times p$.
Conclusion, parts (a) (b) (d) are wrong, and parts (c) (e) are right.

6.4 Adjacency matrix row and column sums. Suppose A is the adjacency matrix of a directed graph (see page 112). What are the entries of the vector $A\mathbf{1}$? What are the entries of the vector $A^T\mathbf{1}$?

Solution:

The i-th entry in $A\mathbf{1}$ means how many path are came out from i-th node.
The i-th entry in $A^T\mathbf{1}$ means how many path are came in to i-th node.

6.5 Adjacency matrix of reversed graph. Suppose A is the adjacency matrix of a directed graph (see page 112). The *reversed graph* is obtained by reversing the directions of all the edges of the original graph. What is the adjacency matrix of the reversed graph? (Express your answer in terms of A.)

Solution:

The answer is the transposion of A, *i.e.*, A^T.

6.6 Matrix-vector multiplication. For each of the following matrices, describe in words how x and $y = Ax$ are related. In each case x and y are n-vectors, with $n = 3k$.

(a) $A = \begin{bmatrix} 0 & 0 & I_k \\ 0 & I_k & 0 \\ I_k & 0 & 0 \end{bmatrix}$.

(b) $A = \begin{bmatrix} E & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & E \end{bmatrix}$, where E is the $k \times k$ matrix with all entries $1/k$.

Solution:

(a) $y = (x_{2k+1}, x_{2k+2}, \ldots, x_{3k}, x_{k+1}, \ldots, x_{2k}, x_1, \ldots, x_k) = (x_{2k+1:3k}, x_{k+1:2k}, x_{1:k})$

(b) $y = (\mathbf{avg}(x_{1:k})\mathbf{1}_k, \mathbf{avg}(x_{k+1:2k})\mathbf{1}_k, \mathbf{avg}(x_{2k+1:3k})\mathbf{1}_k)$

6.7 Currency exchange matrix. we consider a set of n currencies, labeled $1,\ldots,n$. (these might correspond to USD, RMB, EUR, and so on.) At a particular time the exchange or conversion rates among the n currencies are given bu an $n \times n$ (exchange rate) matrix R, where R_{ij} is the amount of currency i that you can buy for one unit of currence j. (All entries of R are positive.) The exchange rates include commission charges, so we have $R_{ji}R_{ij} < 1$ for all $i \neq j$. You can assume that $R_{ii} = 1$.

Suppose $y = Rx$ where x is a vector (with nonnegative entries) that represents the amounts of the currencies that we hold. What is y_i? Your answer should be in English.

Solution:

We know $y_i = b_i x$, b_i is the i-th row-vector of matrix R. Thus y_i means how much amount of i-th currency we can get by all currencies we hold.

6.8 Cash flow to bank account balance. The T-vector c represents the cash flow for an interest bearing bank account over T time periods. Positive values of c indicate a deposit, and negative values indicate a withdrawal. The T-vector b denotes the bank account balance in the T peirod. We have $b_1 = c_1$ (the initial deposit or withdrawal) and

$$b_t = (1 + r)b_{t-1} + c_t, \qquad t = 2,\ldots,T,$$

where $r > 0$ is the (per-period) intrest rate. (The first term is the previous balance plus the interest, and the second term is the deposit or withdrawal.)

Find the $T \times T$ matrix A for which $b = Ax$. That is, the matrix A maps a cash flow sequence into a bank account balance sequence. Your description must take clear what all entries of A are.

Solution:

Suppose the T-vectors $a_1,\ldots,a_T$ is the row-vectors of matrix A. We have $b_1 = a_1 c = c_1$ means $a_1 = (1,0,\ldots,0)^T$. $b_2 = (1 + r)b_1 + c_2 = (1 + r)c_1 + c_2 = a_2 c$, means

$a_2 = (1+r, 1, 0, \ldots, 0)^T$. Suppose $b_i = \sum_{j=1}^{i} (1+r)^{i-j} c_j$ hold for all $i = 1, \ldots, k-1$

$$\begin{aligned} b_k &= (1+r) b_{k-1} + c_k \\ &= (1+r) \sum_{j=1}^{k-1} (1+r)^{k-j-1} c_j + c_k \\ &= \sum_{j=1}^{k-1} (1+r)^{k-j} c_j + (1+r)^{k-k} c_k \\ &= \sum_{j=1}^{k} (1+r)^{k-j} c_j \end{aligned}$$

with $b_k = a_k c$ we have $a_k = ((1+r)^{k-1}, (1+r)^{k-2}, \ldots, 1, 0, \ldots, 0)^T$. Thus,

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ (1+r) & 1 & & & \\ (1+r)^2 & (1+r) & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ (1+r)^{n-1} & (1+r)^{n-2} & \cdots & \cdots & 1 \end{bmatrix}$$

6.9 Multiple channel marketing campaign. Potential customers are divided into m marked segments, which are groups of customers with similar demographics, *e.g.*, college educated women aged 25-29. A company markets its products by purchasing advertising in a set of n channel, *i.e.*, specific TV or radio shows, magazines, web sites, blogs, direct mail, and so on. The ability of each channel to deliver impressions or views by potential customers is characterized by the *reachmatrix*, the $m \times n$ matrix R, where R_{ij} is the number of views of customers in segment i for each market dollar spent on channel j. (We assume that the total nunber of views in each segment is the sum of the views from each channel, and that the views from each channel scale linearly with spending.) The n-vector c will denote the company's purchases of advertising, in dollar, in the n channels. The m-vector v gives the total number of impressions in the m market segments due to the advertising in all channels. Finally, we introduce the m-vector a, where a_i gives the profit in dollars per impression in market segment i. The entries of R, c, v, and a are all nonnegative.

(a) Express the total amount of money the company spends on advertising using vector/matrix notation.

(b) Express v using vector/matrix notation, in terms of the other vectors and matrices.

(c) Express teh total profit from all market segments using vector/matrix notation.

(d) How would you find the single channel most effective at reaching market segment 3, in terms of impressions per dollar spent?

(e) What does it mean if R_{35} is very small (compared to other entries of R)?

Solution:

(a) $\mathbf{1}^T c$

(b) $v = Rc$

(c) $v^T a$

(d) Assume b_3 is an n-row-vector obtained from the third row vector of R. Find the largest value in b_3.

(e) The 5th channel has less effective on market segment 3.

6.10 Resource requirements. We consider an application with n different job (types), each of which consumes m different resources. We define the $m \times n$ resurce matrix R, with entry R_{ij} giving the amount of resource i that is needed to run one unit of job j, for $i = 1,\ldots,m$ and $j = 1,\ldots,n$. (These numbers are typically positive.) The number (or amount) of each of the different jobs to be processed or run is given by the entries of the n-vector x. (These entries are typically nonnegative integers, but they can be fractional of the jobs are divisible.) The entries of the m-vector p give the price per unit of each of the resources.

(a) Let y be the m-vector whose entries give the total of each of the m resources needed to process the jobs given by x. Express y in terms of R and x using matrix and vector notation.

(b) Let c be an n-vector whose entries gives the cost per unit for each of job type. (This is the total cost of the resources required to run one unit of the job type.) Express c in terms of R and p using matrix and vector notation.

Remark. One example is data center, which runs many instances of each of n types of application programs. The resources include number of cores, amount of memory, disk, and network bandwidth.

Solution:

(a) $y = Rx$

(b) $c = R^T p$

6.11 Let A and B be two $m \times n$ matrices. Under each of the assumptions below, determine whether $A = B$ must always hold, or whether $A = B$ holds only sometimes.

(a) Suppose $Ax = Bx$ holds for all n-vectors x.

(b) Suppose $Ax = Bx$ for some nonzero n-vector x.

Solution:

(a) ✓ **YES.** Because $Ax = Bx$, we have $(A-B)x = 0$, let $C = A - B$, we have $Cx = 0$ holds for all n-vector x. Let $c_1, \ldots, c_n$ be the column vectors of C, Consider $\tilde{x} = x + e_1$ we also have $C\tilde{x} = 0$. But $C\tilde{x} = C(x + e_1) = Cx + Ce_1 = c_1 = 0$. Use the same method we can work out that for all $i = 1, \ldots, n$, $c_i = 0$, thus $C = 0$, means $A - B = 0$, same with $A = B$.

(b) × **NO.** For example $A = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & 1 \end{bmatrix}$, $x = (k, 0)$ in which k is a nonzero real number. We have $Ax = Bx$ but $A \neq B$.

6.12 Skew-symmetric matrices. An $n \times n$ matrix A is called *skew-symmetric* if $A^T = -A$, *i.e.*, its transpose is its negative. (A symmetric matrix satisfies $A^T = A$.)

(a) Find all 2×2 skew-symmetric matrices.

(b) Explain why the diagonal entries of a skew-symmetric matrix must be zero.

(c) Show that for a skew-symmetric matrix A, and any n-vector x, $(Ax) \perp x$. This means that Ax and x are orthogonal. *Hint.* First show that for any $n \times n$ matrix A and n-vector x, $x^T(Ax) = \sum_{i,j=1}^{n} A_{ij} x_i x_j$.

(d) Now suppose A is any matrix for which $(Ax) \perp x$ for any n-vector x. Show

that A must be skew-symmetric. *Hint.* You might find the formula

$$(e_i + e_j)^T (A(e_i + e_j)) = A_{ii} + A_{jj} + A_{ij} + A_{ji},$$

valid for any $n \times n$ matrix A, useful. For $i = j$, this reduces to $e_i^T (Ae_i) = A_{ii}$.

Solution:

(a) $\begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}$.

(b) At first we show that for a skew-symmetric metrix A, $A_{ij} = -A_{ji}$. We have $A^T = -A$, means $A_{ij} = A_{ji}^T = -A_{ji}$. Let $i = j$, we have $A_{ii} = -A_{ii}$, thus $A_{ii} = 0$, thus the diagonal entries of skew-symmetric metric must be zero.

(c) In order to show that $(Ax) \perp x$, it is same to show that $x^T(Ax) = 0$. and

$$\begin{aligned} x^T(Ax) &= \sum_{i=1}^{n} x_i (\sum_{j=1}^{n} A_{ij} x_j) \\ &= \sum_{i,j=1}^{n} A_{ij} x_i x_j \\ &= \sum_{i,j=1, i<j}^{n} (A_{ij} x_i x_j + A_{ji} x_j x_i) + \sum_{i=1}^{n} A_{ii} x_i x_i \\ &= \sum_{i,j=1, i<j}^{n} (A_{ij} x_i x_j - A_{ij} x_j x_i) \\ &= 0 \end{aligned}$$

(d) Consider $x = e_i$, we have $0 = x^T(Ax) = e_i^T(Ae_i) = e_i^T a_i = A_{ii}$ Consider $x = (e_i + e_j)$, we have $0 = (e_i + e_j)^T (A(e_i + e_j)) = (e_i + e_j)(a_i + a_j) = A_{ii} + A_{ij} + A_{ji} + A_{jj} = A_{ij} + A_{ji}$, therefore $A_{ij} = -Aji$, thus A is a skew-symmetric matrix.

6.13 Polynomial differentiation. Suppose p is a polynomial of degree $n-1$ or less, given by $p(t) = c_1 + c_2 t + \cdots + c_n t^{n-1}$. Its dericative (with respect to p) $p'(t)$ is a polynomial of degree $n-2$ or less, given by $p'(t) = d_1 + d_2 t + \cdots + d_{n-1} t^{n-2}$. Find a matrix D for which $d = Dc$. (Give the entries of D, and be sure to specific its dimensions.)

Solution:

$d_i = (i-1)c_i = (i-1)e_i^T c$, thus

$$D = \begin{bmatrix} e_2^T \\ 2e_3^T \\ \vdots \\ (n-1)e_n^T \end{bmatrix}$$

Here e_i is an n-vector with ith entry be one.

6.14 Norm of matric-vector product. Suppose A is an $m \times n$ matrix and x is an n-vector. A famous inequality relates $\|x\|$, $\|A\|$, and $\|Ax\|$:

$$\|Ax\| \leq \|A\| \, \|x\| \, .$$

The left-hand side is the (vector) norm of the matrix-vector product; the right-hand side is the (scalar) product of the matrix and vector norms. Show this inequality. *Hint.* Let a_i^T be the ith row of A. Use the Cauchy-Schwarz inequality to get $(a_i^T x)^2 \leq \|a_i\|^2 \, \|x\|^2$. The add the resulting m inequalities.

Solution:

Denote the ith row-vector of A is b_i^T, and $y = Ax$, so we have $y_i = b_i^T x$, we have

$$\begin{aligned} \|Ax\|^2 &= \sum_{i=1}^{m} y_i^2 \\ &= \sum_{i=1}^{m} \left\| b_i^T x \right\|^2 \\ &\leq \sum_{i=1}^{m} \|b_i\|^2 \, \|x\|^2 \\ &= \|x\|^2 \sum_{i=1}^{m} \sum_{j=1}^{b} A_{ij}^2 \\ &= \|x\|^2 \, \|A\|^2 \end{aligned}$$

Thus $\|Ax\| \leq \|A\| \, \|x\|$.

6.15 Distance between adjacency matrices. Let A and B be the $n \times n$ adjacency matrices of two directed graphs with n vertices (see page 112). The squared

distance $\|A-B\|^2$ can be used to express how different the two graphs are. Show that $\|A-B\|^2$ is the total number of directed edges that are in one of the two graphs but not in the other.

Solution:

For convenience, we define four edge set as below:

$\mathcal{R}_{A^-B^-}$ means for any edge $(i,j) \in \mathcal{R}_{A^-B^-}$, it is a edge not in both A and B.

$\mathcal{R}_{A^+B^-}$ means for any edge $(i,j) \in \mathcal{R}_{A^+B^-}$, it is a edge in A but not in B.

$\mathcal{R}_{A^-B^+}$ means for any edge $(i,j) \in \mathcal{R}_{A^-B^+}$, it is a edge not in A but in B.

$\mathcal{R}_{A^+B^+}$ means for any edge $(i,j) \in \mathcal{R}_{A^+B^+}$, it is a edge both in A and B.

then, the entries of $A-B$ can be write as

$$(A-B)_{ij} = \begin{cases} 0, & (i,j) \in \mathcal{R}_{A^-B^-} \\ 1, & (i,j) \in \mathcal{R}_{A^+B^-} \\ 1, & (i,j) \in \mathcal{R}_{A^-B^+} \\ 0, & (i,j) \in \mathcal{R}_{A^+B^+} \end{cases}$$

Therefore

$$(A-B)^2_{ij} = \begin{cases} 0, & (i,j) \in \mathcal{R}_{A^-B^-} + \mathcal{R}_{A^+B^+} \\ 1, & (i,j) \in \mathcal{R}_{A^+B^-} + \mathcal{R}_{A^-B^+} \end{cases}$$

Thus

$$\|A-B\|^2 = |\mathcal{R}_{A^+B^-} + \mathcal{R}_{A^-B^+}|$$

means the total number of directed edges that are in one of the two graphs but not in the other.

6.16 Columns of difference matrix. Are the columns of the difference matrix D, defined in (6.5), linearly independent?

Solution:

× **NO**. The different matrix has n columns, each of these columns is an $n-1$-vector. Accoding to independent-dimension inequality, if they are linearly independent, mean $k \le n$, here $n = k$, thus $n \le n-1$, which is not right, so they are not linearly independent.

Or, it is easy to verify that $D\mathbf{1} = 0$, *i.e.*, there is nonzero vector $x = \mathbf{1}$ satisfies $Dx = 0$, implies the columns of D are linearly dependent.

6.17 Stacked matrix. Let A be an $m \times n$ matrix, and consider the stacked matrix S defined by

$$S = \begin{bmatrix} A \\ I \end{bmatrix}.$$

When does S have linearly independent columns? When does S have linearly independent rows? Your answer can depend on m, n, or whether or not A has linearly independent columns or rows.

Solution:

Assume $a_1, \ldots, a_n$ are m-vectors obtained from A matrix's columns. Consider the linear combination

$$\alpha_1 \begin{bmatrix} a_1 \\ e_1 \end{bmatrix} + \cdots + \alpha_n \begin{bmatrix} a_n \\ e_n \end{bmatrix} = 0.$$

We have $\alpha_1 e_1 + \cdots + \alpha_n e_n = 0$, and $e_1, \ldots, e_n$ are linearly dependent, means $\alpha_1 = \cdots = \alpha_n = 0$, thus S always have linearly independent columns.

Matrix S has $n + m$ rows, each of the rows is an n-vector, if $m = 0$, they are linearly independent. (The row are $e_1^T, \ldots, e_n^T$.) If $m > 0$, means $m + n > n$, they cannot be linearly independent because of the independent-dimension inequality.

6.18 Vandermonde matrices. A Vandermonde matrix is an $m \times n$ matrix of the form

$$V = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^{n-1} \end{bmatrix}$$

where $t_1, \ldots, t_m$ are numbers. Multiplying an n-vector c by the Vandermonde matrix V is the same as evaluating the polynomial of degree less than n, with coefficients $c_1, \ldots, c_n$, at the points $t_1, \ldots, t_m$; see page 120. Show that the columns of a Vandermonde matrix are linearly independent if the numbers $t_1, \ldots, t_m$ are distinct, *i.e.*, different from each other. *Hint.* Use the following fact from algebra: If a polynomial p with degree less than n has n or more roots (points t for which $p(t) = 0$) then all its coefficients are zero.

Solution:

As we want the columns of V to be linearly independent, V must be wide or square, *i.e.*, $m \geq n$. Let $Vx = 0$, for $j = 1,\ldots,m$, we have

$$x_1 + x_2 t_j + \ldots, x_n t_j^{n-1} = 0,$$

consider the polynomial function

$$f(t) = x_1 + x_2 t + \ldots + x_n t^{n-1},$$

we have

$$f(t_1) = 0, \ldots, f(t_m) = 0,$$

i.e., $t_1,\ldots,t_m$ are the roots of polynomial function f, if $t_1,\ldots,t_m$ are distinct, means the number of root of f is larger than its order n, implies the coefficients of the polynomial function f are all equal to zero, *i.e.*, $x = 0$, which means the columns of V are linearly independent.s

6.19 Back-test timing. The $T \times n$ asset returns matrix R gives the returns of n assets over T periods. (See page 120.) When the n-vector w gives a set of portfolio weights, the T-vector Rw gives the time series of portfolio return over the T time periods. Evaluating portfolio return with past returns data is called *back-testing*.

Consider a specific case with $n = 5000$ assets, and $T = 2500$ returns. (This is 10 year of daily returns, since there are around 250 trading days in each year.) About how long would it take to carry out this back-test, on a 1 Gflop/s computer?

Solution:

It requires about

$$2Tn = 2500000$$

flops to compute the back-testing, therefore it needs about

$$2.5ms$$

to carry out the result in this computer.

6.20 Complexity of matrix-vector multiplication. On page 123 we worked out the complexity of computing the m-vector Ax, where A is an $m \times n$ matrix and x is an n-vector, when each entry of Ax is computed as an inner product of a row of A and the vector x. Suppose instead that we compute Ax as linear combination of the columns of A, with coefficients $x_1, \ldots, x_n$. How many flops does this method require? How does it compare to the method described on page 123?

Solution:

We consider the result of Ax as a linear combination of the columns of A, it requires to compute n number of vector-scalar multiplication, which requires nm flops, then $n-1$ vector addtion, which requires $(n-1)m$, *i.e.*, totally $2mn-m$ flops, and it is same to the method described in page 123.

6.21 Complexity of matrix-sparse-vector multiplication. On page 123 we consider the complexity of computing Ax, where A is a sparse $m \times n$ matrix and x is an n-vector. (x not assumed to be sparse). Now consider the complexity of computing Ax when the $m \times n$ matrix A is not sparse, but the n-vector x is sparse, with $\mathbf{nnz}(x)$ nonzero entries. Give the total number of flops in terms of m, n, and $\mathbf{nnz}(x)$, and simplify it by dropping terms that are dominated by others when the dimensions are large. *Hint.* The vector Ax is a linearly combination of $\mathbf{nnz}(x)$ columns A.

Solution:

To compute the ith entry of Ax, we need $\mathbf{nnz}(x)$ multiplies and $\mathbf{nnz}(x)-1$ additions, therefore the totally it require $m*(2\mathbf{nnz}(x)-1)$ which can be simplified to be $2m\mathbf{nnz}(x)$ flops.

6.22 Distribute or not? Suppose you need to compute $z = (A+B)(x+y)$, where A and B are $m \times n$ matrices and x and y are n-vectors.

(a) What is the approximate flop count if you evaluate z as expressed, *i.e.*, by adding A and B, adding x and y, and then carring out the matrix-vector multiplication?

(b) What is the approximate flop count if you evaluate z as $z = Ax+Ay+Bx+By$, *i.e.*, with four matrix-vector multiplies and three vector additions?

(c) Which method requires fewer flops? Your answer can depend on m and n. *Remark.* When comparing two computation methods, we usually do not consider a factor of 2 or 3 in flop counts to be significant, but in this exercise you can.

Solution:

(a) It need nm flops to compute $A+B$ and n flops to compute $x+y$ and $2mn$ flops to compute $(A+B)(x+y)$, totally

$$3mn+n$$

flops.

(b) It need $8mn$ to compute the four matrix-vector multiplication and $3m$ for vector addition, totally

$$8mn+3m$$

flops.

(c) As we always have

$$3mn+n<8mn+3m,$$

the first method rquires fewer flops.

7 Matrix Example

7.1 Projection on a line. Let $P(x)$ denote the projection of the 2-D point (2-vector) x onto the line that passes through $(0,0)$ and $(1,3)$. (This means that $P(x)$ is the point on the line that is closest to x; see exercise 3.12.) Show that P is a linear function, and give the maxtrix A for which $P(x)=Ax$ for any x.

Solution:

Use the conclusion of exercise 3.12, with $a=(0,0)$ and $b=(1,3)$. The projection point p can be expressed as

$$p=\frac{b^T xb}{\|b\|^2}=\frac{b_1x_1b}{\|b\|^2}+\frac{b_2x_2b}{\|b\|^2}$$

There for matrix A is

$$A = \begin{bmatrix} \frac{b_1}{\|b\|^2} b & \frac{b_2}{\|b\|^2} b \end{bmatrix}.$$

as $\|b\|^2 = 10$, we have

$$A = \begin{bmatrix} \frac{1}{10} & \frac{3}{10} \\ \frac{3}{10} & \frac{9}{10} \end{bmatrix}$$

7.2 3-D rotation. Let x and y be 3-vectors representing positions in 3-D. Suppose that the vector y is obtained by rotating the vector x about the vertical axie (*i.e.*, e_3) by 45° (counterclockwise, *i.e.*, from e_1 toward e_2). Find 3×3 matrix A for which $y = Ax$. *Hint.* Determine the three columns of A by finding the results of the transformation on the unit vectors e_1, e_2, e_3.

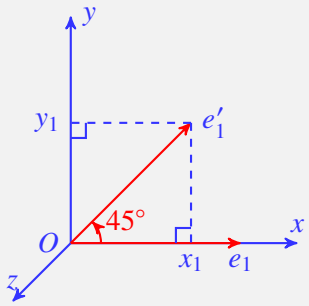
Solution:

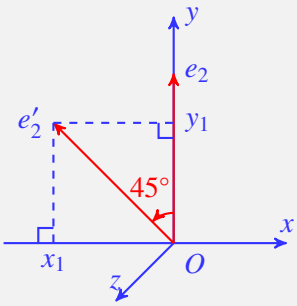
Consider the unit vector e_1, after rotating its changed to $a_1 = (\cos(45°), \sin(45°), 0) = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0)$.

And the unit vector e_2 after rotating changed to $a_2 = (-\sin(45°), \cos(45°), 0) = (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0)$.

Last the unit vector e_3 after rotating changed to $a_3 = (0, 0, 1)$. Thus

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$





7.3 Trimming a vector. Find a matrix A for which $Ax = (x_2, \ldots, x_{n-1})$, where x is an n-vector. (Be sure to specific the size of A, and describe all its entries.)

Solution:

A is an $(n-2)\times n$ matrix, assume n-vector e_k denotes the k-th unit vector. We can express A as

$$A = \begin{bmatrix} e_2^T \\ e_3^T \\ \vdots \\ e_{n-1}^T \end{bmatrix}.$$

Or

$$A = \begin{bmatrix} \mathbf{0}_{(n-2)\times 1} & I_{(n-2)\times(n-2)} & \mathbf{0}_{(n-2)\times 1} \end{bmatrix}.$$

7.4 Down-sampling and up-conversion. We consider n-vector x that represent signals, with x_k the value of the signal at time k for $k = 1,\ldots,n$. Below we describe two functions of x that produce new signals $f(x)$. For each function, give a matrix A such that $f(x) = Ax$ for all x.

(a) $2\times$ *downsampling*. We assume n is even and define $f(x)$ as the $n/2$-vector y with elements $y_k = x_{2k}$. To simplify your notation you can assume that $n = 8$, *i.e.*,

$$f(x) = (x_2, x_4, x_6, x_8).$$

(On page 131 we describe a different type of down-sampling, that uses the average of pairs of original values.)

(b) $2\times$ *up-conversion with linear interpolation*. We define $f(x)$ as the $(2n-1)$-vector y with elements $y_k = x_{(k+1)/2}$ if k is odd and $y_k = (x_{k/2} + x_{k/2+1})/2$ if k is even. To simplify your notation you can assume that $n = 5$, *i.e.*,

$$f(x) = (x_1, \frac{x_1+x_2}{2}, x_2, \frac{x_2+x_3}{2}, x_3, \frac{x_3+x_4}{2}, x_4, \frac{x_4+x_5}{2}, x_5).$$

Solution:

(a)

$$A = \begin{bmatrix} e_2^T \\ e_4^T \\ e_6^T \\ e_8^T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(b)

$$A = \begin{bmatrix} e_1^T \\ (e_1^T + e_2^T)/2 \\ e_2^T \\ (e_2^T + e_3^T)/2 \\ e_3^T \\ (e_3^T + e_4^T)/2 \\ e_4^T \\ (e_4^T + e_5^T)/2 \\ e_5^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

7.5 Transpose of selector matrix. Suppose the $m \times n$ matrix A is a selector matrix. Describe the relation between the m-vactor u and the n-vector $v = A^T u$.

Solution:

Suppose $A = \begin{bmatrix} e_{k_1}^T \\ \vdots \\ e_{k_m}^T \end{bmatrix}$, where k_i belongs to $1,\ldots,n$, e_{k_i} is k_i-th unit n-vector.

Therefore $A^T = \begin{bmatrix} e_{k_1}, \ldots, e_{k_m} \end{bmatrix}$, thus $v = u_1 e_{k_1} + \cdots + u_m e_{k_m}$, mean $v_i = \sum_{k_j = i} u_j$.

7.6 Rows of incidence matrix. Show that the rows of the incidence matrix of a graph are always linearly dependent. *Hint.* Consider the sum of the rows.

Solution:

Suppose the size of incidence matrix is $m \times n$, r_i^T is the n-row-vectors of the matrix, which $i = 1,\ldots,m$, and l_i is the m-vectors obtained by the columns of the matrix, which $i = 1,\ldots,n$. With the definition of incidence matrix, the entries of columns vector l_i has only one 1 and one -1, and otherwise 0, so the express $\mathbf{1}^T l_i = 0$ hold for all $i = 1,\ldots,n$.

Consider the sum of all row-vectors: $r_1^T + \cdots + r_m^T = \begin{bmatrix} \mathbf{1}^T l_1 \\ \vdots \\ \mathbf{1}^T l_n \end{bmatrix} = 0$, means we can get out the 0 vector by linear combiantion of r_i^T, with coefficients are 1, *i.e.*, they are linearly dependent.

7.7 Incidence matrix of reserved graph. (See exercise 6.5.) Suppose A is the incidence matrix of a graph. The reserved graph is obtained by reversing the directions of all the edges of the original graph. What is the incident matrix of the reserved graph? (Express your answer in terms of A.)

Solution:

Recall the definition of incidence matrix A of a graph:

$$A_{ij} = \begin{cases} 1 & \text{edge } j \text{ points to node } i \\ -1 & \text{edge } j \text{ points from node } i \\ 0 & \text{otherwise} \end{cases},$$

after reversed, the edge j which point to node i change to point from i, and for that edge which point from a node change to point to the node, *i.e.*, the incidence matrix of the reserved matrix is

$$-A.$$

7.8 Flow conservation with sources. Suppose A is the incidence matrix of a graph, x is the vector of edge flows, and s is the external source vector, as described in 7.3. Assuming that flow is conserved, *i.e.*, $Ax + s = 0$, show that $\mathbf{1}^T s = 0$. This means that the total amount injected into the network by the sources ($s_i > 0$) must exactly balance the total removed from the netwoerk at the sink nodes ($s_i < 0$). For example if the network is a (lossless) electrical power grid, the total amount of the electrical power generated (and injected into the grid) must exactly balance the total electrical power consumed (from the gird).

Solution:

Suppose a_i is the column vectors of A. we have

$$\mathbf{1}^T(Ax) = \mathbf{1}^T(x_1 a_1 + \cdots + x_m a_m) = x_1 \mathbf{1}^T a_1 + \cdots + x_m \mathbf{1}^T a_m = 0.$$

Here we use the fact $\mathbf{1}^T a_i = 0$ we show in the previous exercise. Thus

$$0 = \mathbf{1}^T(Ax + s) = \mathbf{1}^T(Ax) + \mathbf{1}^T s = \mathbf{1}^T s.$$

7.9 Social network graph. Consider a group of n people or users, and some symmetric social relation among them. This means that some pairs of users are *connected*, or *friend*(say). We can create a directed graph by associating a node with each user, and an edge between each pair of friends, arbitrarily choosing the direction of the edge. Now consider an n-vector v, where v_i is some quantity for user i, for example, age or education level (say, given in years). Let $\mathcal{D}(v)$ denotes the Dirichlet energy associated with the graph and v, thought of as a potential on the nodes.

(a) Explain why the number $\mathcal{D}(v)$ does not depend on the choice of directions for the edges of the graph.

(b) Would you guess that $\mathcal{D}(v)$ is small or large? This is an open-ended, vague question; there is no right answer. Just make a guess as to what you might expect, and give a short English justification of your guess.

Solution:

(a) $(v) = \|Av\|^2 = \sum_{edges(i,j)}(v_i - v_j)^2$, in other words, for a edge (i,j), its contribution to $\mathcal{D}(v)$ is $(v_i - v_k)^2$ is not depend on the direction of the edge.

(b) I think it might be small. My reason is the total differences of all paris, or relations. And people are often make relation with who has similar age or education levels, so the differences between them would be small, that why I say it might be small.

7.10 Circle graph. A *circle graph* (also called a *cycle graph*) has n vertices, with edges pointing from vertix 1 to vertix 2, from vertix 2 to vertix 3, ..., from vertix $n-1$ to vertix n, and finally, from vertix n to vertix 1. (This last edge complete the circle.)

(a) Draw a diagram of a circle graph, and give its incidence mateix A.

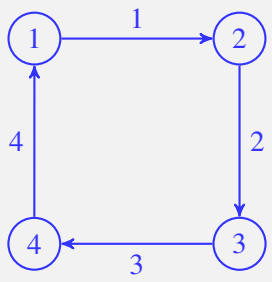
(b) Suppose that x is a circulation for a circle graph. What can you say about x ?

(c) Suppose the n-vector v is a potential on a circle graph. What is the Dirichlet energy $\mathcal{D}(v) = \|Ax\|^2$?

Remark. The circle graph arises when an n-vector v represents a periodic time series. For example, v_1 could be the value of some quantity on Monday, v_2 its value on Tuesday, and v_7 its value on Sunday. The Dirichlet energy is a measure of the roughness of such an n-vector v.

Solution:

(a) $A = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$



(b) Means $Ax = 0$, we have

$$\begin{cases} -x_1 + x_n &= 0 \\ x_1 - x_2 &= 0 \\ \vdots & \\ x_{n-1} - x_n &= 0 \end{cases}$$

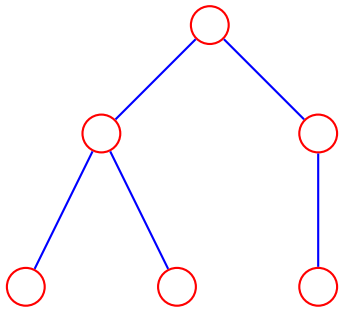
We hve $x_1 = \ldots = x_n$.

(c) According to the definition of Dirichlet energy,

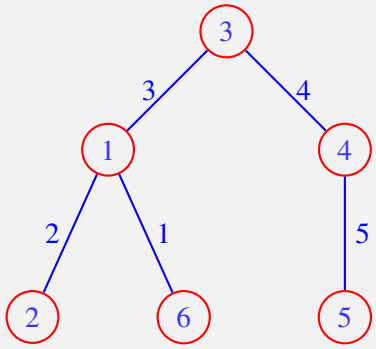
$$\mathcal{R}(v) = (v_1 - v_2)^2 + \cdots + (v_n - v_1)^2.$$

7.11 Tree. An undirected graph is called a tree if it is connected (there is a path from every vertex to every other vertex) and contains no cycle. *i.e.*, there is no path that begins and ends at the same vertex. **Figure 7.1** shows a tree with six vectices. For the tree in the figure, find a numbering of the vectices and edges, and an orientation of the edges, so that the incidence matrix A of the resulting directed graph satisfies $A_i = 1$ for $i = 1, \ldots, 5$ and $A_i j = 0$ for $i < j$. In other words, the first 5 rows of A form a lower triangle matrix with ones on the diagonal.

Figure 7.1: Tree with six vectices.



Solution:



$$A = \begin{bmatrix} 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

7.12 Some properties of convolution. Suppose that a is an n-vector.

(a) *Convolution with* 1. What is $1 * a$? (Here we interpret 1 as a 1-vector.)

(b) *Convolution with a unit vector.* What is $e_k * a$, where e_k is the kth unit vector of dimension q? Describe this vector mathematically (*i.e.*, give its entries), and via a brief English description. You might find vector slice notation useful.

Solution:

(a) $1 * a = a * 1 = a$.

(b) Suppose $v = e_k * a$, according to the definition of convolution, $v_m = \sum_{i+j-1=m} e_{k_i} a_j$, with the fact that $a_{k_i} = 1$ when $i = k$ and $a_{k_i} = 0$ if $i \neq k$, we have $v_m = a_{m+1-k}$, *i.e.*, $v = \begin{bmatrix} \mathbf{0}_k \\ a \\ \mathbf{0}_{p-k} \end{bmatrix}$.

We can also say $e_k * a$ shift the vector a forward by k entries.

7.13 Sum property of convolation. Show that for any vectors a and b, we have $\mathbf{1}^T(a * b) = (\mathbf{1}^T a)(\mathbf{1}^T b)$. In words: The sum of the coefficients of the

convolution of two vectors is the product of the sums of the coefficients of the vectors. *Hint.* If the vector a represents the coefficients of a polynoamial p, $\mathbf{1}^T a = p(1)$.

Solution:

Suppose vectors a and b represent the coefficients of polynomial p and q, respectivly. And Let $f(x) = p(x)q(x)$, means vector $a * b$ reprents the coefficients of f. we have

$$\mathbf{1}^T(a * b) = f(1) = p(1)q(1) = (\mathbf{1}^T a)(\mathbf{1}^T b).$$

7.14 Rainfall and river height. The T-vector r gives the daily rainfall in some region over a period of T days. The vector h gives the daily height of a river in the region (above its normal height). By careful modeling of water flow, or by fitting a model to past data, it is found that these vectors are (approximately) related by convolution: $h = g * r$, where

$$g = (0.1, 0.4, 0.5, 0.2).$$

Give a short in English (with no mathematical terms) to approximately describe this relation. For example, you might mention how many days after a one day heavy rainfull the river height is most affected. Or how many days it takes for the river height to return to the normal heigth once the rain stops.

Solution:

The height of river is related with the past four days' rainfall. And the height of river is affected most by the rainfall in two-days ago with fractional 0.5. And after rainfall, the affection of it will disappeared four days later.

7.15 Channel equalization. We suppose that $u_1, \ldots, u_m$ is signal (time series) that is transmitted (for example by radio). A receiver recieves the signal $y = c * u$, where the n-vector c is called the channel impluse response. (see page 138.) In most applications n is small, *e.g.*, under 10, and m is much larger. An *equalizer* is a k-vector h that satisfies $h * c \approx e_1$, the first unit vector of length $n + k - 1$. The reciever equalizes the recieved signal

y by convolving it with the equalizer to obtain $z = h * y$.

(a) How are z (the equalized recieved signal) and u (the original transmitted signal) related ? *Hint.* Recall that $h*(c*u) = (h*c)*u$.

(b) *Numerical example.* Generate a signal u of length $m = 50$,with each entry a random value that is eight -1 or $+1$. Plot u and $y = c*u$, with $c = (1, 0.7, -0.3)$. Also plot the equalized signal $z = h*y$, with

$$h = (0.9, -0.5, 0.5, -0.4, 0.3, -0.3, 0.2, -0.1)$$

.

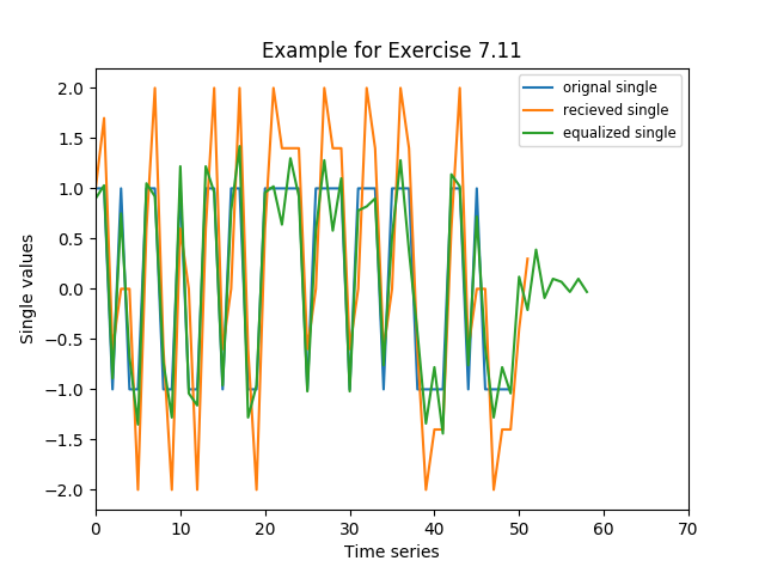
Solution:

(a) $z = h*y = h*(c*u) = h*c*u \approx e_1 * u = (u, 0)$.

Listing 2: Python Example

(b)
```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exerises 7.11
# Author: Utoppia
# Date  : 06 May 2020

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

class Single:
    def __init__(self, m, impluse, equalizer):
        self.impluse = impluse
        self.equalizer = equalizer
        np.random.seed(int(time.time()))
        self.single = np.random.choice([1,-1], m)
        self.recieved = np.convolve(self.impluse, self.single)

    def plot(self):
        self.fig = plt.figure()

        plt.plot(self.single, label='orignal single')
        plt.plot(self.recieved, label='recieved single')
        plt.plot(np.convolve(self.recieved, self.equalizer), label='
            equalized single')
```

```
        plt.xlim((0, 70))
        plt.xlabel('Time series')
        plt.ylabel('Single values')
        plt.title('Example for Exercise 7.11')
        plt.legend(fontsize='small')
        plt.savefig('11.png')

def main():
    impluse = np.array([1, 0.7, -0.3])
    equalizer = np.array([0.9, -0.5, 0.5, -0.4, 0.3, -0.3, 0.2, -0.1])
    m = 50
    single = Single(m, impluse, equalizer)
    single.plot()

main()
```



8 Linear equations

8.1 Sum of linear functions. Suppose $f : \mathrm{R}^n \to \mathrm{R}^m$ and $g : \mathrm{R}^n \to \mathrm{R}^m$ are linear functions. Their *sum* is the function $h(x) = f(x) + g(x)$ for any n-vector x. The sum function is often denoted as $h = f + g$. (This is another case overloading the + symbol, in this case to the sum of functions.) if f has matrix representation $f(x) = Fx$, and g has matrix representation $g(x) = Gx$,

where F and G are $m \times n$ matrices, what is the matrix representation of the sum function $h = f + g$? Be sure to identify any + symbol appearing in your justification.

Solution:

$h = f + g = Fx + Gx = (F + G)x$.

The first + is sum of two function, sencond one is sum of two m-vectors, last one is sum of two $n \times m$ matrices.

8.2 Average and affine functions. Suppose that $g : \mathrm{R}^n \to \mathrm{R}^m$ is an affine function. Let $x_1, \ldots, x_k$ be n-vector, and define the m-vector $y_1 = G(x_1), \ldots, y_k = G(x_k)$. Let

$$\bar{x} = (x_1 + \cdots + x_k)/k, \qquad \bar{y} = (y_1 + \cdots + y_k)/k$$

be the averages of these two lists of vectors. (Here $\bar{x}$ is n-vector and $\bar{y}$ is an m-vector.) Show that we always have $\bar{y} = G(\bar{x})$. In words: The average of an affine function applied to a list of vectors is the same as the affine function applied to the average of the list of the vector.

Solution:

At first, I will show you for affine function $f : \mathrm{R}^n \to \mathrm{R}^m$, The equation

$$f(\alpha_1 x_1 + \cdots + \alpha_k x_k) = \alpha_1 f(x_1) + \cdots + \alpha_k f(x_k)$$

allways hold if $\alpha_1 + \cdots + \alpha_k = 1$. We know when $k = 1, 2$ it holds, Suppose for it holds for $k = 1, \ldots, p-1$, Let $C = \sum_{i=1}^{p-1} \alpha_i$,we have $C + \alpha_p = 1$

$$\begin{aligned} f(\alpha_1 x_1 + \cdots + \alpha_p x_p) &= f(C(\frac{\alpha_1 x_1 + \cdots + \alpha_{p-1} x_{p-1}}{C}) + \alpha_p x_p) \\ &= C f(\frac{\alpha_1}{C} x_1 + \cdots + \frac{\alpha_{p-1}}{C} x_{p-1}) + \alpha_p f(x_p) \\ &= C \frac{\alpha_1}{C} f(x_1) + \cdots + C \frac{\alpha_{p-1}}{C} f(x_{p-1}) + \alpha_p f(x_p) \\ &= \alpha_1 f(x_1) + \cdots + \alpha_p f(x_p) \end{aligned}$$

Now

$$\begin{aligned} G(\bar{x}) &= G(\frac{1}{k}x_1 + \cdots + \frac{1}{k}x_k) \\ &= \frac{1}{k}G(x_1) + \cdots + \frac{1}{k}G(x_k) \\ &= (y_1 + \cdots + y_k)/k \\ &= \bar{y} \end{aligned}$$

8.3 Cross-product. The cross product of two 3-vectors $a = (a_1, a_2, a_3)$ and $x = (x_1, x_2, x_3)$ is defined as the vector

$$a \times x = \begin{bmatrix} a_2x_3 - a_3x_2 \\ a_3x_1 - a_1x_3 \\ a_1x_2 - a_2x_1 \end{bmatrix}$$

The cross product comes up in physics, for example in electricity and magnetism, and in dynamics of mechanical systems like robots or satellites. (You do not need to know this for this exercise.)

Assume a is fixed. Show that the function $f(x) = a \times x$ is a linear function of x, by giving a matrix A that satisfies $f(x) = Ax$ for all x.

Solution:

$a_2x_3 - a_3x_2 = (0, -a_3, a_2)^T x$

$a_3x_1 - a_1x_3 = (a_3, 0, -a_1)^T x$

$a_1x_2 - a_2x_1 = (-a_2, a_1, 0)^T x$

Therefore

$$A = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

8.4 Linear functions of images. In this problem we consider several linear functions of a monochrome image with $N \times N$ pixels. To keep the matrices small enough to work out by hand, we will consider the case with $N = 3$ (which would hardly qualify as an image). We represent a 3×3 image as a 9-vector using the ording of pixels shown below.

1	4	7
2	5	8
3	6	9

(This ordering is called *column-major*.) Each of the operations or transformations below defines a function $y = f(x)$, where the 9-vector x represents the original image, and the 9-vector y represents the resulting or transformed image. For each of these operations, give the 9×9 matrix A for which $y = Ax$.

(a) Turn the original image x upside-down.

(b) Rorate the original image x clockwise 90°.

(c) Translate the image up by 1 pixel and to the right by 1 pixel. In the translated image, assign the value $y_i = 0$ to the pixels in the first column and the last row.

(d) Set each pixel value y_i to be the average of the neighbors of pixel i in the original image. By neighbors, we mean the pixels immediately above and below, and immediately to the left and right. The center pixel has 4 neighbors; corner pixels have 2 neighbors, and the remaining pixels have 3 neighbors.

Solution:

(a) After turn the image, it changed into

3	6	9
2	5	8
1	4	7

Thus

$$A = \begin{bmatrix} e_3^T \\ e_2^T \\ e_1^T \\ e_6^T \\ e_5^T \\ e_4^T \\ e_9^T \\ e_8^T \\ e_7^T \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(b) After rotating, the image changed into

3	2	1
6	5	4
9	8	7

Thus

$$A = \begin{bmatrix} e_3^T \\ e_6^T \\ e_9^T \\ e_2^T \\ e_5^T \\ e_8^T \\ e_1^T \\ e_4^T \\ e_7^T \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(c) After translating, the image changed into

0	2	5
0	3	6
0	0	0

Thus

$$A = \begin{bmatrix} 0^T \\ 0^T \\ 0^T \\ e_2^T \\ e_3^T \\ 0^T \\ e_5^T \\ e_6^T \\ 0^T \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(d)

$$A = \begin{bmatrix} (e_2+e_4)^T/2 \\ (e_1+e_3+e_5)^T/3 \\ (e_2+e_6)^T/2 \\ (e_1+e_5+e_7)^T/3 \\ (e_2+e_4+e_6+e_8)^T/4 \\ (e_3+e_5+e_9)^T/3 \\ (e_4+e_8)^T/2 \\ (e_5+e_7+e_9)^T/3 \\ (e_6+e_8)^T/2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

8.5 Symmetric and anti-sysmmetric part. An n-vector x is *symmetric* if $x_k = x_{n-k+1}$ for $k = 1,\ldots,n$. It is *anti-symmetric* if $x_k = -x_{n-k+1}$ for $k = 1,\ldots,n$.

(a) Show that every vector x can be decomposed in a unique way as a sum $x = x_s + x_a$ of a symmetric vector x_s and an anti-symmetric vector x_a.

(b) Show that the symmetric and anti-symmetric parts x_s and x_a are linear functions of x. Give matrices A_s and A_a such that $x_s = A_s x$ and $x_a = A_a x$ for all x.

Solution:

(a) We have $x_k + x_{n-k+1} = (x_s)_k + (x_s)_{n-k+1} + (x_a)_k + (x_a)_{n-k+1} = 2(x_s)_k$, we have $(x_s)_k = \dfrac{x_k + x_{n-k+1}}{2}$ and $(x_a)_k = \dfrac{x_k - x_{n-k+1}}{2}$.

(b) Let

$$A_s = \frac{1}{2}\begin{bmatrix} (e_1+e_n)^T \\ \vdots \\ (e_n+e_1)^T \end{bmatrix}, \qquad A_a = \frac{1}{2}\begin{bmatrix} (e_1-e_n)^T \\ \vdots \\ (e_n-e_1)^T \end{bmatrix}.$$

we have $x_s = A_s x$ and $x_a = A_a x$.

8.6 Linear functions. For each description of y below, express it as $y = Ax$ for some A. (You should specify A.)

(a) y_i is the difference between x_i and the average of $x_1, \ldots, x_{i-1}$. (We take $y_1 = x_1$.)

(b) y_i is the difference between x_i and the average value of all other x_js, *i.e.*, the average of $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$.

Solution:

(a) A is a below-triangular matrix, with the ith row a_i^T to be

$$a_i^T = -\frac{1}{i-1}\sum_{j=0}^{i-1} e_j^T + e_i^T.$$

for $i = 1, \ldots, n$, here we take $a_1^T = e_1^T$.

(b) The ith row of A a_i^T should be

$$a_i^T = -\frac{1}{n-1}\mathbf{1}^T + \frac{n}{n-1}e_i^T.$$

8.7 Interpolation of polynomial values and derivatives. The 5-vector c represents the coefficients of a quartic polynomial $p(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3 + c_5 x^4$. Express the conditions

$$p(0) = 0, p'(0) = 0, p(1) = 1, p'(1) = 0,$$

as a set of linear equations of the form $Ac = b$. Is the system of equations under-determined, over-determined, or square?

Solution:

$p'(x) = c_2 + 2c_3 x + 3c_4 x^2 + 4c_5 x^3$. we have

$$\begin{aligned} c_1 &= 0 \\ c_2 &= 0 \\ c_1 + c_2 + c_3 + c_4 + c_5 &= 1 \\ c_2 + 2c_3 + 3c_4 + 4c_5 &= 0 \end{aligned}$$

let A and b be

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

we have $Ax = b$. Because the number of equations is less than the number of variables, according to the definition, this is **under-determined.**

8.8 Interpolation of rational functions. A *rational function* of degree two has the form

$$f(t) = \frac{c_1 + c_2 t + c_3 t^2}{1 + d_1 t + d_2 t^2},$$

where c_1, c_2, c_3, d_1, d_2 are coeficients. ('Rational' refers to the fact that f is a ratio of polynomials. Another names for f is *bi-quadratic*.) Consider the interpolation conditions

$$f(t_i) = y_i, \qquad i = 1, \ldots, K,$$

where t_i and y_i are given numbers. Express the interpolation conditions as a set of linear equations in the vector of coefficients $\theta = (c_1, c_2, c_3, d_1, d_2)$, as $A\theta = b$. Given A and b, and their dimensions.

Solution:

For (t_i, y_i), we have

$$c_1 + c_2 t_i + c_3 t_i^2 = y_i + d_1 t_i y_i + d_2 t_i^2 y_i,$$

which can be formed as

$$\begin{bmatrix} 1 & t_i & t_i^2 & -t_i y_i & -t_i^2 y_i \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ d_1 \\ d_2 \end{bmatrix} = y_i.$$

Let A and b to be

$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & -t_1 y_1 & -t_1^2 y_1 \\ \vdots & & & & \\ 1 & t_K & t_K^2 & -t_K y_K & -t_K^2 y_K \end{bmatrix}, \qquad b = \begin{bmatrix} y_1 \\ \vdots \\ y_K \end{bmatrix}$$

The dimension of A is $K \times 5$ and for b is K-vector.

8.9 Required nutrients. We consider a set of n basic foods (such as rice, beans, apples) and a set of m nutrients or components (such as protein, fat, sugar, vitamin C). Food j has a cost given c_j (say, in dollars per gram), and contains an amount N_{ij} of nutrient i (per gram). (The nutrients

are given in some appropriate units, which can depend on the particular nutrient.) A dailly diet is represented by an n-vector d, with d_i the daily intake (in grams) of food i. Express the condition that a diet d contains the total nutrient amounts given by the m-vector n^{des}, and has a total cost B (the budget) as a set of linear equations in the variables $d_1, \ldots, d_n$. (The entries of d must be nonnegative, but we ignore this issue here.)

Solution:

We have

$$Nd = n^{des}, \qquad c^T d = B.$$

Combine these we can have

$$\begin{bmatrix} N \\ c^T \end{bmatrix} d = \begin{bmatrix} n^{des} \\ B \end{bmatrix}.$$

8.10 Blending crude oil. A set of K different types of crude oil are blended (mixed) together in proportions $\theta_1, \ldots, \theta_K$. There numbers sum to one; they must also be nonnegative, but we will ignore that requirement here. Associated with crude oil type k is an n-vector c_k that gives its concentration of n different constituents, such as specific hydrocarbons. Find a set of linear equations on the blending coefficients, $A\theta = b$, that expresses the requirement that the blended crude oil achieves a set of constituent concentrations, given by the n-vector c^{tar}. (Include the condition that $\theta_1 + \cdots + \theta_K = 1$ in your equations.)

Solution:

we have

$$\theta_1 c_1 + \cdots + \theta_K c_K = c^{tar},$$

let $C = \begin{bmatrix} c_1 & \ldots & c_K \end{bmatrix}$, we have $C\theta = c^{tar}$. combine with $\theta_1 + \cdots + \theta_K = \mathbf{1}^T\theta = 1$, we have

$$\begin{bmatrix} C \\ \mathbf{1}^T \end{bmatrix} \theta = \begin{bmatrix} c^{tar} \\ 1 \end{bmatrix}$$

let $A = \begin{bmatrix} C \\ \mathbf{1}^T \end{bmatrix}$ and $b = \begin{bmatrix} c^{tar} \\ 1 \end{bmatrix}$, we have

$$A\theta = b.$$

8.11 Location from range measurements. The 3-vector x represents a location in 3-D. We measure the distance (also called the range) of x to four points at known locations a_1, a_2, a_3, and a_4:

$$\rho_1 = \|x - a_1\|, \qquad \rho_2 = \|x - a_2\|, \qquad \rho_3 = \|x - a_3\|, \qquad \rho_4 = \|x - a_4\|.$$

Express theses distance conditions as a set of three linear equations in the vector x. *Hint.* Square the distance equations, and subtract one from the others.

Solution:

We have

$$\begin{aligned}
\rho_1^2 &= \|x\|^2 - 2x^T a_1 + \|a_1\|^2 \\
\rho_2^2 &= \|x\|^2 - 2x^T a_2 + \|a_2\|^2 \\
\rho_3^2 &= \|x\|^2 - 2x^T a_3 + \|a_3\|^2 \\
\rho_4^2 &= \|x\|^2 - 2x^T a_4 + \|a_4\|^2
\end{aligned}$$

or

$$\begin{aligned}
2(a_1 - a_2)^T x &= \rho_2^2 - \rho_1^2 + \|a_1\|^2 - \|a_2\|^2 \\
2(a_1 - a_3)^T x &= \rho_3^2 - \rho_1^2 + \|a_1\|^2 - \|a_3\|^2 \\
2(a_1 - a_4)^T x &= \rho_4^2 - \rho_1^2 + \|a_1\|^2 - \|a_4\|^2
\end{aligned}$$

which can be written as

$$\begin{bmatrix} 2(a_1 - a_2)^T \\ 2(a_1 - a_3)^T \\ 2(a_1 - a_4)^T \end{bmatrix} x = \begin{bmatrix} \rho_2^2 - \rho_1^2 + \|a_1\|^2 - \|a_2\|^2 \\ \rho_3^2 - \rho_1^2 + \|a_1\|^2 - \|a_3\|^2 \\ \rho_4^2 - \rho_1^2 + \|a_1\|^2 - \|a_4\|^2 \end{bmatrix}$$

8.12 Quadrature. Consider a function $f : \mathrm{R} \to \mathrm{R}$. We are interested in *estimating* the definite integral $\alpha = \int_{-1}^{1} f(x)dx$ based on the value of f at some points $t_1, \ldots, t_n$. (We typically have $-1 \leq t_1 < t_2 < \cdots < t_n \leq 1$, but this is not needed here.) The standard method for estimating α is to form a weighted sum of the value $f(t_i)$:

$$\hat{\alpha} = w_1 f(t_1) + \cdots + w_n f(t_n),$$

where $\hat{\alpha}$ is our estimate of α, and $w_1, \ldots, w_n$ are the weights. This method of estimating the value of an integral of a function from its values at some points is a classical method in applied mathematics called *quadrature*. There are many quadratrue methods (*i.e.*, choices of the points t_i and weight w_i). The most famous one is due to the mathematician Carl Friedrich Gauss, and bears his name.

(a) A typical requirement in quadrature is that the approximation should be exact (*i.e.*, $\hat{\alpha} = \alpha$) when f is any polynominal up to degree d, where d is given. In this case, we say that the quadrature method has *order* d. Express this condition as a set of linear equations on the weight, $Aw = b$, assuming the point $t_1, \ldots, t_n$ are given.
Hint. If $\hat{\alpha} = \alpha$ holds for the specific cases $f(x) = 1, f(x) = x, \ldots, f(x) = x^d$, then it holds for any polynomial of degree up to d.

(b) Show that the following quadrature methods have order 1, 2, and 3 respectively.

- *Trapezoid rule*: $n = 2$, $t_1 = -1$, $t_2 = 1$, and

$$w_1 = 1, \qquad w_2 = 1.$$

- *Simpson's rule*: $n = 3$, $t_1 = -1$, $t_2 = 0, t_3 = 1$, and

$$w_1 = 1/3, \qquad w_2 = 4/3, \qquad w_3 = 1/3.$$

  (Named after the mathematician Thomas Simpson.)
- *Simpson's 3/8 rule*: $n = 4$, $t_1 = -1$, $t_2 = -1/3$, $t_3 = 1/3$, $t_4 = 1$,

$$w_1 = 1/4, \qquad w_2 = 3/4, \qquad w_3 = 3/4, \qquad w_4 = 1/4.$$

Solution:

(a) First, we show that if $\hat{\alpha} = \alpha$ holds for specific cases $f(x) = 1, f(x) = x, \ldots, f(x) = x^d$, then it holds for any polynomial of degree up to d:
For convenient, we denote $f_i(x) = x^i$ for $i = 0, \ldots, d$. And with the fact that any polynomial functions f with degree up to d can be expressed as linear combination of $f_i(x)$, $i = 0, \ldots, d$, *i.e.*,

$$f(x) = s_0 f_0(x) + \cdots + s_d f_d(x).$$

therefore

$$\begin{aligned}
\int_{-1}^{1} f(x)dx &= \int_{-1}^{1} s_0 f_0(x) + \cdots + s_d f_d(x)dx \\
&= s_0 \int_{-1}^{1} f_0(x) + \cdots + s_d \int_{-1}^{1} f_d(x)dx \\
&= s_0 \sum_{j=0}^{n} w_j f_0(t_j) + \cdots + s_d \sum_{j=0}^{n} w_j f_d(t_j) \\
&= \sum_{i=0}^{d} \sum_{j=0}^{n} w_j s_i f_i(t_j) \\
&= \sum_{j=0}^{n} \sum_{i=0}^{d} w_j s_i f_i(t_j) \\
&= \sum_{j=0}^{n} w_j \sum_{i=0}^{d} s_i f_i(t_j) \\
&= \sum_{j=0}^{n} w_j f(t_j)
\end{aligned}$$

Thus, $\hat{\alpha} = \alpha$ holds for any polynomial of degree up to d.
And now we know $\int_{-1}^{1} f_i(x) = \int_{-1}^{1} x^i = \frac{1-(-1)^{i+1}}{i+1}$ and $f_i(t_j) = (t_j)^i$ for $i = 0, \ldots, d$, with assumption, we have $\int_{-1}^{1} f_i(x)dx = \alpha = \hat{\alpha} = w_1 f_i(t_1) + \cdots + w_n f_i(t_n) = \sum_{j=0}^{n} w_j f_i(t_j)$ holds. The equations of $\hat{\alpha} = \alpha$ can be expressed as:

$$\begin{aligned}
w_1 + \cdots + w_n &= 2 \\
t_1 w_1 + \cdots + t_n w_n &= 0 \\
&\vdots \\
t_1^d w_1 + \cdots + t_n^d w_n &= \frac{1-(-1)^{d+1}}{d+1}
\end{aligned}$$

let

$$A = \begin{bmatrix} 1 & \ldots & 1 \\ t_1 & \ldots & t_n \\ \vdots & & \\ t_1^d & \ldots & t_n^d \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ 0 \\ \vdots \\ (1-(-1)^{d+1})/(d+1) \end{bmatrix}$$

we can form the condition as a set of linear equations as

$$Aw = b.$$

(b) We can use the condition we get in part (a) to check these

questions. We check each of these rules from $d = 1$, if for some specific $\bar{d}$, the condition didn't hold, mean $\bar{d} - 1$ is the order of this rule.

- For *Trapezoid rule*, we have

$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2 \neq \frac{2}{3}$$

  Thus, its order is 1.

- For *Simpson's rule*, we have

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 4/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 2/3 \end{bmatrix}, \qquad \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 4/3 \\ 1/3 \end{bmatrix} = 0$$

  I think its order should be 3, too.

- For *Simpson's 3/8 rule*, we have

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1/3 & 1/3 & 1 \\ 1 & 1/9 & 1/9 & 1 \\ -1 & -1/27 & 1/27 & 1 \end{bmatrix} \begin{bmatrix} 1/4 \\ 3/4 \\ 3/4 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 2/3 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 1/81 & 1/81 & 1 \end{bmatrix} \begin{bmatrix} 1/4 \\ 3/4 \\ 3/4 \\ 1/4 \end{bmatrix} = 14/27 \neq 2/5$$

  its order is $d = 3$.

8.13 Portfolio sector exposures. (See exercise 1.14.) The n-vector h denotes a portfilio of omvestments in n assets, with h_i the dollar value invested in asset i. We consider a set of m insudtry, such as pharmaceuticals or consumer electronics. Each asset is assigned to one of these sectors. (More complex models allow for an asset to be assigned to more than one sector.) The *exposure* of the portfilio to sector i is defined as the sum of incestments in the assets in that sector. We denote the sector exposure using the m-vector s, where s_i is the portfilio exposure to sector i. (When $s_i = 0$, the portfilio is said to be *neutral* to sector i.) An investment advisor specifies a set of desired sector exposures, given as the m-vector s^{des}. Express the requirement $s = s^{des}$ as a set of linear equations of the form $Ah = b$. (You must describe the matrix A and the vector b.)

Remark. A typical case involves $n = 1000$ assets and $m = 50$ sectors. An advisor might specify $s_i^{des} = 0$ if she does not have an opinion as how

companies in that sector will do in the future; she might specify a positive value for s_i^{des} if she thinks that the companies in that sector will do well (*i.e.*, generate positive returns) in the future, and a negative value if she thinks they will do poorly.

Solution:

We define a $m \times n$ matrix A as :

$$A_{ij} = \begin{cases} 1, & \text{if asset } j \text{ is assigned to sector } i \\ 0, & \text{if asset } j \text{ is not assigned to sector } i \end{cases}$$

Then the ith row of A denotes which assets are assigned to sector i, therefore $s = Ah$. let $b = s^{des}$, we can form the $s = s^{des}$ as

$$Ah = b.$$

8.14 Affine combinations of solutions of linear equations. Consider the set of m linear equations in n variables $Ax = b$, where A is an $m \times n$ matrix, b is an m-vector, and x is the n-vector of variables. Suppose that the n-vectors $z_1, \ldots, z_k$ are solutions of this set of equations, *i.e.*, satisfy $Az_i = b$. Show that if the coefficients $\alpha_1, \ldots, \alpha_k$ satisfy $\alpha_1 + \cdots + \alpha_k = 1$, then the affine combination

$$w = \alpha_1 z_1 + \cdots + \alpha_k z_k$$

is a solution of the linear equations, *i.e.*, satisfies $Aw = b$. In words: Any affine combination of solutions of a set of linear equations is also a solution of the equations.

Solution:

$$\begin{aligned} Aw &= A(\alpha_1 z_1 + \cdots + \alpha_k z_k) \\ &= \alpha_1 Az_1 + \cdots + \alpha_k Az_k \\ &= \alpha_1 b + \cdots + \alpha_k b \\ &= (\alpha_1 + \cdots + \alpha_k) b \\ &= b \end{aligned}$$

8.15 Stoichiometry and equilibrium reaction rates. We consider a system (such as a single cell) containing m metabolites (chemical species), with n reactions among the metabolites occurring at rates given by the n-vector r. (A negative reaction rate means the reaction runs in reverse.) Each reaction consumes some metabolites and produces others, in known rates proportional to the reaction rate. This is specified in the $m \times n$ *stoichiometry matrix* S, where S_{ij} is the rate of matabolite i production by reaction j, running at rate one. (When S_{ij} is negative, it means that when reaction j runs at rate one, metabolite i is consumed.) The system is said to be in equilibrium if the total production rate of each metabolite, due to all the reactions, is zero. This means that for each metabolite, the total production rate balances the total consumption rate, so the total quantities of the metabolites in the system do not change. Express the condition that the system is in equilibrium as a set of linear equations in the reaction rates.

Solution:

$Sr = 0$.

8.16 Bi-linear interpolation. We are given a scalar value at each of the four corners of a square in 2-D, (x_1, y_1), (x_1, y_2), (x_2, y_1), and (x_2, y_2), where $x_1 < x_2$ and $y_1 < y_2$. We refer to these four values as F_{11}, F_{12}, F_{21}, and F_{22}, respectively. A *bi-linear interpolation* is a function of the form

$$f(u, v) = \theta_1 + \theta_2 u + \theta_3 v + \theta_4 uv,$$

where $\theta_1, \ldots, \theta_4$ are coefficients, that satisfies

$$f(x_1, y_1) = F_{11}, \qquad f(x_1, y_2) = F_{12}, \qquad , f(x_2, y_1) = F_{21}, \qquad f(x_2, y_2) = F_{22},$$

i.e., it agrees with (or interpolates) the given values on the four corners of the square. (The function f is usually evaluated only for point (u, v) inside the square. It is called bi-linear since it is affine in u when v is fixed, and affine in v when u is fixed.)

Express the interpolation conditions as a set of linear equations of the form $A\theta = b$, where A is a 4×4 matrix and b is a 4-vector. Give the entries of A and b in terms of x_1, x_2, y_1, y_2, F_{11}, F_{12}, F_{21}, and F_{22}.

Remark. Bi-linear interpolation is used in many applications to guess or approximate the values of a function at an arbitrary point in 2-D, given the function values on a grid of points. To approximate the value at a point (x, y), we first find the square of grid points that the point lies in. Then we use bi-linear interpolation to get the approximate value at (x, y).

Solution:

We have

$$\theta_1 + \theta_2 x_1 + \theta_3 y_1 + \theta_4 x_1 y_1 = F_{11}$$
$$\theta_1 + \theta_2 x_1 + \theta_3 y_2 + \theta_4 x_1 y_2 = F_{12}$$
$$\theta_1 + \theta_2 x_2 + \theta_3 y_1 + \theta_4 x_2 y_1 = F_{21}$$
$$\theta_1 + \theta_2 x_2 + \theta_3 y_2 + \theta_4 x_2 y_2 = F_{22}$$

let

$$A = \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_1 & y_2 & x_1 y_2 \\ 1 & x_2 & y_1 & x_2 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \end{bmatrix}, \qquad b = \begin{bmatrix} F_{11} \\ F_{12} \\ F_{21} \\ F_{22} \end{bmatrix}$$

we can write the condition as

$$A\theta = b.$$

9 Linear dynamicaln system

9.1 Compartmental system. A *compartmental system* is a model used to describe the movement of some material over time among a set of n compartments of a system, and the outside world. It is widely used in *pharmaco-kinetics*, the study of how the concentration of a drug varies over time in the body. In this application, the material is a drug, and the compartments are the bloodstream, lungs, heart, liver, kidneys, and so on. Compartmental systems are special cases of linear dynamical systems.

In this problem we will consider a very simple compartmental system with 3 compartments. We let $(x_t)_i$ denote the amount of the material (say, a drug) in compartment i at time period t. Between period t and period $t+1$, the material moves as follow.

- 10% of the material in compartment 1 moves to compartment 2. (This decreases the amount in compartment 1 and increases the amount in compartment 2.)
- 5% of the material in compartment 2 moves to compartment 3.
- 5% of the material in compartment 3 moves to compartment 1.
- 5% of the material in compartment 3 is eliminated.

Express this compartmental system as a linear dynamical system, $x_{t+1} = Ax_t$. (Give the matrix A.) Be sure to account for all the material entering and leaving each compartment.

Solution:

- $(x_{t+1})_1 = 0.9(x_t)_1 + 0.05(x_t)_3$
- $(x_{t+1})_2 = 0.1(x_t)_1 + 0.95(x_t)_2$
- $(x_{t+1})_3 = 0.05(x_t)_2 + 0.9(x_t)_3$

Therefore

$$x_{t+1} = \begin{bmatrix} 0.9 & 0 & 0.05 \\ 0.1 & 0.95 & 0 \\ 0 & 0.05 & 0.9 \end{bmatrix} x_t$$

9.2 Dynamics of economy. An economy (of a country or region) is described by an n-vector a_t, where $(a_t)_i$ is the economic output in sector i in year t (measured in billions of dollars, say.) The total output of the economy in year t is $\mathbf{1}^T a_t$. A very simple model of how the econimic output changes over time is $a_{t+1} = Ba_t$, where B is an $n \times n$ matrix. (This is closely related to the Leontief input-output model described on page 157 of the book. But the Leontief model is static, *i.e.*, doesn't consider how an economy changes over time.) The entries of a_t and B are positive in general.

In this problem we will consider the specific model with $n = 4$ sectors and

$$B = \begin{bmatrix} 0.10 & 0.06 & 0.05 & 0.70 \\ 0.48 & 0.44 & 0.10 & 0.04 \\ 0.00 & 0.55 & 0.52 & 0.04 \\ 0.04 & 0.01 & 0.42 & 0.51 \end{bmatrix}$$

(a) Briefly interpret B_{23}, in English.

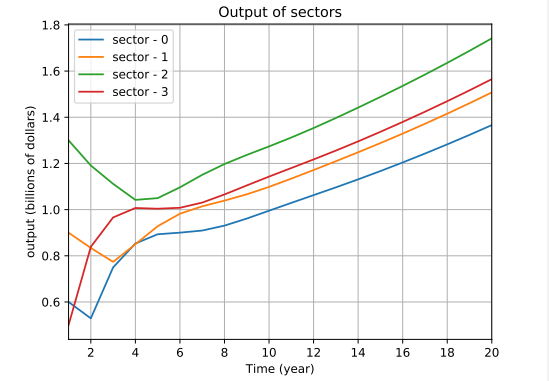
(b) *Simulation.* Suppose $a_1 = (0.6, 0.9, 1.3, 0.5)$. Plot the four sector output (*i.e.*, $a(t)_i$ for $i = 1, \ldots, 4$) and total economic output (*i.e.*, $\mathbf{1}^T a_t$) versus

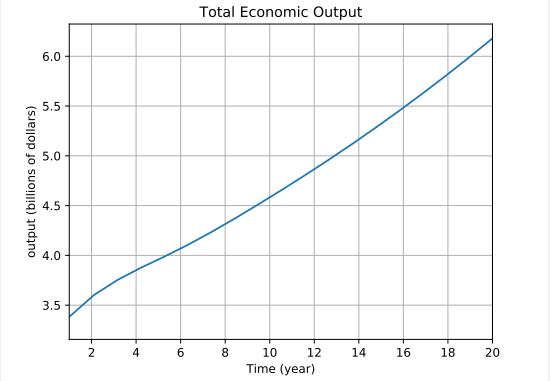
t, for $t = 1, \ldots, 20$.

Solution:

(a) B_{23} percent of economic output of sector 3 will change to sector 2 in the next year.

(b) The output are





Codes are

Listing 3: Python code for 9.2

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exerises 9.2
# Author: Utoppia
# Date  : 06 May 2020

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

class Economy:
    def __init__(self, B, a):
        self.a = a
        self.B = B

    def simulation(self, T):
        prediction = np.zeros((T, len(self.a)))
        prediction[0] = self.a

        for i in range(1, T):
            prediction[i] = self.B.dot(prediction[i-1])
```

```
        return prediction

def plot_sectors(T, prediction):
    fig = plt.figure()

    x = np.linspace(1, 20, 20)
    (m, n) = prediction.shape
    for i in range(n):
        plt.plot(x, prediction[:, i], label='sector - {}'.format(i))

    plt.xlabel('Time (year)')
    plt.ylabel('output (billions of dollars)')
    plt.title('Output of sectors')
    plt.xticks(np.linspace(0,20,11))
    plt.xlim(1, 20)
    plt.grid(True)
    plt.legend()

    fig.savefig('9-2-1.pdf')

def plot_total_output(T, prediction):
    fig = plt.figure()
    x = np.linspace(0,20,20)
    (m,n) = prediction.shape
    total = [item.sum() for item in prediction]
    plt.plot(x, total)

    plt.xlabel('Time (year)')
    plt.ylabel('output (billions of dollars)')
    plt.title('Total Economic Output')
    plt.xticks(np.linspace(0,20,11))
    plt.xlim(1, 20)
    plt.grid(True)

    plt.savefig('9-2-2.pdf')

def main():
    B = np.array([
        [0.10, 0.06, 0.05, 0.70],
        [0.48, 0.44, 0.10, 0.04],
        [0.00, 0.55, 0.52, 0.04],
        [0.04, 0.01, 0.42, 0.52]
```

```
    ])
    a_init = np.array([0.6, 0.9, 1.3, 0.5])
    T = 20

    economy = Economy(B, a_init)
    prediction = economy.simulation(T)
    plot_sectors(T, prediction)
    plot_total_output(T, prediction)

main()
```

9.3 Equilibrium point for linear dynamical system. Consider a time-invariant linear dynamical system with offset, $x_{t+1} = Ax_t + c$, where x_t is the state n-vector. We say that a vector z is an *equilibrium point* of the linear dynamical system if $x_1 = z$ implies $x_2 = z, x_3 = z, \ldots$. (In words: If the system starts in state z, is stays in state z.)

Find a matrix F and vector g for which the set of linear equation $Fz = g$ characterizes equilibrium points. (This means: If z is an equilibrium point, then $Fz = g$; conversely if $Fz = g$, then z is an equilibrium point.) Express F and g in terms of A, c, any standard matrices or vector (*e.g.*, I, $\mathbf{1}$, or 0), and matrix and vector operations.

Remark. Equilibrium points often have interesting interpretations. For example, if the linear dynamical system describe the population dynamics of a country, with the vector c denoting immigration (emigration when entries of c are negative), and equilibrium point is a population distribution that does not change, year to year. In other words, immigration exactly cancels the changes in population distribution caused by aging, births, and deaths.

Solution:

It is easy to figure out that if z is a equilibrium point, we have

$$z = Az + c,$$

let $F = A - I$ and $g = -c$, we conclude that equilibrium point z satisfies

$$Fz = g.$$

And it is clearly that if a vector $\bar{z}$ satisfies $F\bar{z} = g$, *i.e.*$A\bar{z} + c = \bar{z}$, it is an equilibrium point.

9.4 Reducing a Markov model to a linear dynamical system. Consider the 2-Markov model

$$x_{t+1} = A_1 x_t + A_2 x_{t-1}, \qquad t = 2, 3, \ldots,$$

where x_t is an n-vector. Define $z_t = (x_t, x_{t-1})$. Show that z_t satisfies the linear dynamical system equation $z_{t+1} = Bz_t$, for $t = 2, 3, \ldots$, where B is a $(2n) \times (2n)$ matrix. This idea can be used to express any K-Markov model as a linear dynamical system, with state $(x_t, \ldots, x_{t-K+1})$.

Solution:

It is easy to figure out that

$$z_{t+1} = \begin{bmatrix} x_{t+1} \\ x_t \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ I & 0 \end{bmatrix} \begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ I & 0 \end{bmatrix} z_t,$$

let B be the $(2n) \times (2n)$ matrix

$$B = \begin{bmatrix} A_1 & A_2 \\ I & 0 \end{bmatrix},$$

we have

$$z_{t+1} = Bz_t.$$

9.5 Fibonacci sequence. The Fibonacci sequence $y_0, y_1, y_2, \ldots$ starts with $y_0 = 0, y_1 = 1$, and for $t = 2, 3, \ldots$, y_t is the sum of the previous two entries, *i.e.*, $y_{t-1} + y_{t-2}$. (Fibonacci is the name used by the 13th century mathematician Leonardo of Pisa.) Express this as a time-invariant linear dynamical system with state $x_t = (y_t, y_{t-1})$ and output y_t, for $t = 1, 2, \ldots$. Use your linear dynamical system to simulate (compute) the Fibonacci sequence up to $t = 20$. Also simulate a modified Fibonacci sequence $z_0, z_1, z_2, \ldots$,which start with the same value $z_0 = 0$ and $z_1 = 1$, but for $t = 2, 3, \ldots$, z_t is the differece of the two previous values, *i.e.*, $z_{t-1} - z_{t-2}$.

Solution:

The dynamics of the state z_t can be formed as

$$z_{t+1} = \begin{bmatrix} y_{t+1} \\ y_t \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_t \\ y_{t-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} z_t,$$

the output is

$$y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} z_t.$$

The first 20 Fibonacci sequence are

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765,$$

and the first 20 entries of modified Fibonacci sequence are

$$1, 1, 0, -1, -1, 0, 1, 1, 0, -1, -1, 0, 1, 1, 0, -1, -1, 0, 1, 1.$$

9.6 Recursive averaging. Suppose that $u_1, u_2, \ldots$ is a sequence of n-vectors. Let $x_1 = 0$, and for $t = 2, 3, \ldots$, let x_t be the average of $u_1, \ldots, u_{t-1}$, *i.e.*, $x_t = (u_1 + \cdots + u_{t-1})/(t-1)$. Express this as a linear dynamical system with input, *i.e.*, $x_{t+1} = A_t x_t + B_t u_t$, $t = 1, 2, \ldots$ (with initial state $x_1 = 0$). *Remark.* This can be used to compute the average of an extremely large collection of vectors, by accessing then one-by-one.

Solution:

We have

$$\begin{aligned} x_{t+1} &= \frac{u_1 + \cdots + u_t}{t} \\ &= \frac{t-1}{t} \times \frac{u_1 + \cdots + u_{t-1}}{t-1} + \frac{1}{t} u_t \\ &= \frac{t-1}{t} x_t + \frac{1}{t} u_t. \end{aligned}$$

Let $A_t = (t-1)/t$ and $B_t = 1/t$, we have

$$x_{t+1} = A_t x_t + B_t u_t.$$

9.7 Complexity of linear dynamical system simulation. Consider the time-invariant linear dynamical system with n-vector state x_t and m-vector input u_t, and dynamics $x_{t+1} = Ax_t + Bu_t$, $t = 1, 2, \ldots$. You are given the matrices A and B, the initial state x_1, and the input $u_1, \ldots, u_{T-1}$. What is the

complexity of carraying out a simulation, *i.e.*, computing $x_2, \ldots, x_T$? About how long would it take to carry out a simulation with $n = 15$, $m = 5$, and $T = 10^5$, using a 1 Gflop/s computer?

Solution:

For every iteration, it need $2n^2 + 2mn + n$ to compute the next state, therefore the total complexity are

$$(T-1)(2n^2 + 2mn + n)$$

flops.

With the numerial given by the exercise, the complexity is about $(T-1)(2n^2 + 2mn + n) = 61499385$, it need about $61ms$ to take the simulation.

10 Matrix multiplication

10.1 Scalar-row-vector multiplication. Suppose a is a number and $x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$ is an n-row-vector. The scalar-row-vector product ax is the n-row-vector $\begin{bmatrix} ax_1 & \cdots & ax_n \end{bmatrix}$. Is this a special case of matrix-matrix multiplication? That is, can you interoret scalar-row-vector multiplication as matrix multiplication? (Recall that scalar-vector multiplication, with the scalar on the left, is *not* a special case of matrix-matrix multiplication; see page 177.)

Solution:

✓ **YES.** We can interpret scalar a as an 1×1 matrix, and n-row-matrix x as an $1 \times n$ matrix. They are compatible.

10.2 Ones matrix. There is no special notation for an $m \times n$ matrix all of whose entries are one. Give a simple expression for this matrix in term of matrix multiplication, transpose, and the ones vector $\mathbf{1}_m$, $\mathbf{1}_n$ (where the subscripts denote the dimension).

Solution:

The Ones matrix is the outer product of two ones vector, as $\mathbf{1}_m^T \mathbf{1}_n$ is an $m \times n$ ones matrix.

10.3 Matrix sizes. Suppose A, B, and C are matrices that satisfy $A+BB^T = C$, Determine which of the following statements are necessarily true. (There may be more than one true statement.)

(a) A is square.

(b) A and B have the same dimensions.

(c) A, B, and C have the same number of rows.

(d) B is tall matrix.

Solution:

We know BB^T is a sqaure matrix, that why A and C must be square.

(a) ✓ **YES.**

(b) × **NO.**

(c) ✓ **YES.**

(d) × **NO.**

10.4 Block matrix notation. Consider the block matrix

$$A = \begin{bmatrix} I & B & 0 \\ B^T & 0 & 0 \\ 0 & 0 & BB^T \end{bmatrix}$$

where B is 10×5. What are the dimensions of the five zero matrices and the identity matrix in the definition of A? What are the dimensions of A?

Solution:

The identity matrix I has same rows with B, which is 10, so its dimension is 10×10. The zero matrix in the right upper has same row with B as 10, and same columns with BB^T as 10, therefore its dimensions is 10×10. The zero matrix in the right middle has same row with B^T as 5, and same columns with BB^T as 10, therefor its dimensions is 5×10. The zero matrix in the center has same row with B^T as 5, and same columns with B as 5, its dimensions is 5×5. The zero matrix in the below left has same rows with BB^T as 10, and same columns with B^T as 10, its dimensions is 10×10. The zero matrix in the below middle has same rows with BB^T as 10, and same columns with

B as 5, its dimensions is 10×5.
The dimensions of matrix A is 25.

10.5 When is the outer product symmetric? Let a and b be n-vectors. The inner product is symmetric, *i.e.*, we have $a^T b = b^T a$. The outer product of the two vectors is generally *not* symmetric; that is, we generally have $ab^T \neq ba^T$. What are the conditions on a and b under which $ab^T = ba^T$? You can assume that all the entries of a and b are nonzero. (The conclusion you come to will hold even when some entries of a and b are zero.) *Hint.* Show that $ab^T = ba^T$ implies that a_i/b_i is a constant (*i.e.*, independent of i).

Solution:

ab^T and ba^T are $n \times n$ matrices, the i,j index of ab^T and ba^T are $a_i b_j$ and $b_i a_j$, respectively. we have $a_i b_j = b_i a_j$ hold for all i and j. Assume all entries in a and b are nonzero, we have a_i/b_i is a constant.

10.6 Product of rotation matrices. Let A be the 2×2 matrix that corresponds to rotation by θ radians, defined in (7.1), and let B be the 2×2 matrix that corresponds to rotation by ω radians. Show that AB is also a rotation matrix, and give the angle by which it rotates vectors. Verify that $AB = BA$ in this case, and give a simple English explanation.

Solution:

According to the definition of rotation matrix, we can interpret it as an function map a vector to a new vector by rotation the vector θ radians. So AB means rotating the vector ω radian first, then θ radian, the results is by rotating it $\theta + \omega$ radians. Now let use definition of matrix multiplication to verify our conclusion.

$$\begin{aligned} AB &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta\cos\omega - \sin\theta\sin\omega & -\cos\theta\sin\theta - \sin\theta\cos\omega \\ \sin\theta\cos\omega + \cos\theta\sin\omega & -\sin\theta\sin\omega + \cos\theta\cos\omega \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta+\omega) & -\sin(\theta+\omega) \\ \sin(\theta+\omega) & \cos(\theta+\omega) \end{bmatrix} \end{aligned}$$

It is easy to verify that $AB = BA$.

10.7 Two rotations. Two 3-vectors x and y are related as follows. First, the vector x is rorated 40° around the e_3 axis, conterclockwise (from e_1 toward e_2), to obtain the 3-vector z. Then, z is rotated 20° around the e_1 axis, counterclockwise (from e_2 toward e_3), to form y. Find the 3×3 matrix A for which $y = Ax$. Verify that A is an orthogonal matrix. *Hint.* Express A as a product of two matrices, which carry out the two rotations described above.

Solution:

For convenient, denote $\theta = 40^\circ$ and $\omega = 20^\circ$. we have

$$z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} x = Px$$

and

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix} z = Qz = QPx$$

therefore

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \cos\omega\sin\theta & \cos\omega\cos\theta & -\sin\omega \\ \sin\omega\sin\theta & \sin\omega\cos\theta & \cos\omega \end{bmatrix}$$

There are two methods to derive that A is orthogonal matrix; in the first method, we show that both Q and P are orthogonal matrix:

$$\begin{aligned} P^T P &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos^2\omega + \sin^2\omega & 0 \\ 0 & 0 & \sin^2\omega + \cos^2\omega \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I \end{aligned}$$

Use same method we have $Q^T Q = I$, therefore

$$A^T A = (QP)^T QP = p^T Q^T QP = P^T IP = P^T P = I.$$

Thus, A is orthogonal matrix.

Another method you can use the expression of A and to calculate the $A^T A$.

Remark that the result is not depend on the specific values of θ and ω.

10.8 Entries of matrix triple product. (See page 182.) Suppose A has dimensions $m \times n$, B has dimensions $n \times p$, C has dimensions $p \times q$, and let $D = ABC$. Show that

$$D_{ij} = \sum_{k=1}^{n} \sum_{l=1}^{p} A_{ik} B_{kl} C_{lj}.$$

This is the formula analogous to (10.1) for the product of two matrices.

Solution:

Let $E = BC$ then we have

$$E_{kj} = \sum_{l=1}^{p} B_{kl} C_{lj}.$$

now,

$$\begin{aligned} D_{ij} &= \sum_{k=1}^{n} A_{ik} E_{kj} \\ &= \sum_{k=1}^{n} A_{ik} \sum_{l=1}^{p} B_{kl} C_{lj} \\ &= \sum_{k=1}^{n} \sum_{l=1}^{p} A_{ik} B_{kl} C_{lj} \end{aligned}$$

10.9 Multiplication by a diagonal matrix. Suppose that A is an $m \times n$ matrix, D is a diagonal matix, and $B = DA$. Describe B in terms of A and the entries of D. You can refer to the rows or columns or entries of A.

Solution:

Let $d_1, \ldots, d_m$ be the diagonal value of matrix D. Then D can be expressed in form of

$$D = \begin{bmatrix} d_1 e_1 & \ldots & d_m e_m \end{bmatrix}.$$

Let $a_1^T, \ldots, a_m^T$ be the rows of A, we have

$$\begin{aligned} B &= \begin{bmatrix} d_1 e_1 & \ldots & d_m e_m \end{bmatrix} \begin{bmatrix} a_1^T \\ \vdots \\ a_m^T \end{bmatrix} \\ &= d_1 e_1 a_1^T + \cdots + d_m e_m a_m^T \\ &= \begin{bmatrix} d_1 a_1^T \\ \vdots \\ d_m a_m^T \end{bmatrix} \end{aligned}$$

10.10 Converting from purchase quantity matrix to purchase dollar matrix. An $n \times N$ matrix Q gives the purchase history of a set of n products by N customers, over some period, with Q_{ij} being the quantity of product i bought by customer j. The n-vector p gives the product prices. A data analyst needs the $n \times N$ matrix D, where D_{ij} is the total dollar value that customer j spent on product i. Express D in terms of Q and p, using compact matrix/vector notation. You can use any notation or ideas we have encountered, *e.g.*, stacking, slicing, block matricesm transpose, matrix-vector product, matrix-matrix product, inner product, norm, correlation, **diag**(), and so on.

Solution:

It is easy to know that

$$D_{ij} = p_i Q_{ij}.$$

and the ith row-vector of D

$$d_i^T = p_i q_i^T$$

Therefore

$$D = \begin{bmatrix} p_1 q_1^T \\ \vdots \\ p_n q_n^T \end{bmatrix}$$

Recall the previous exercise, we can denote

$$P = \mathbf{diag}(p)$$

then we have

$$D = \mathbf{diag}(p)Q$$

10.11 Trace if matrix-matrix product. The sum of the diagonal entries of a square matrix is called the *trace* of the matrix, denoted $\mathbf{tr}(A)$.

(a) Suppose A and B are $m \times n$ matrices. Show that

$$\mathbf{tr}(A^T B) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}.$$

What is the complexity of calculating $\mathbf{tr}(A^T B)$?

(b) The number $\mathbf{tr}(A^T B)$ is sometimes referred to as the inner product of the matrices A and B. (This allows us to extend concepts like angle to matrices.) Show that $\mathbf{tr}(A^T B) = \mathbf{tr}(B^T A)$.

(c) Show that $\mathbf{tr} A^T A = \|A\|^2$. In other words, the sqaure of the norm of a matrix is the trace of its Gram matrix.

(d) Show that $\mathbf{tr}(A^T B) = \mathbf{tr}(BA^T)$, even though in general $A^T B$ and BA^T can have different dimensions, and even when they have the same dimensions, they need not to be equal.

Solution:

(a) First we find out the expression of the diagonal entries of $A^T B$, from the multiplication of matrix, we can form the ith diagonal entry of $A^T B$ as

$$(A^B)_{ii} = \sum_{j=1}^{m} (A^T)_{ij} B_{ji} = \sum_{j=1}^{m} A_{ji} B_{ji}$$

therefore the trace of the $A^T B$ is

$$\mathbf{tr}(A^T B) = \sum_{i=1}^{n} (A^T B)_{ii} = \sum_{i=1}^{n} \sum_{j=1}^{m} A_{ji} B_{ji}.$$

Or we can change the interpretation of i and j, and reform the expression above to be

$$\mathbf{tr}(A^T B) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}.$$

(b) Replace $A = B$ and $B = A$ in the formula we proved in part (a), we

have

$$\mathbf{tr}(B^T A) = \sum_{i=1}^{m}\sum_{j=1}^{n} B_{ij}A_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}B_{ij} = \mathbf{tr}(A^T B).$$

(c) Let $B = A$ in the formula of part (a), we have

$$\mathbf{tr}(A^T A) = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}A_{ij} = \|A\|^2 .$$

(d) Use the formula we proved in part (a), we have

$$\mathbf{tr}(BA^T) = \mathbf{tr}\left((B^T)^T(A^T)\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ji}B_{ji} = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}B_{ij} = \mathbf{tr}(A^T B).$$

10.12 Norm of matrix product. Suppose A is an $m \times p$ matrix and B is a $p \times n$ matrix. Show that $\|AB\| \leq \|A\|\,\|B\|$, *i.e.*, the (matrix) norm of the matrix product is no more than the product of the norms of the matrices. *Hint.* Let $a_1^T, \ldots, a_m^T$ be the rows of A, and $b_1, \ldots, b_n$ be the columns of B. Then

$$\|AB\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(a_i^T b_j)^2.$$

Now use the Cauchy-Schawarz inequality.

Solution:

Let $a_1^T, \ldots a_m^T$ be the rows of A, and $b_1, \ldots, b_n$ be the columns of B, we have

$$\|AB\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(a_i^T b_j)^2,$$

with Cauchy-Schwarz inequality, we have

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(a_i^T b_j)^2 \leq \sum_{i=1}^{m}\sum_{j=1}^{n}\|a_i\|^2\|b_j\|^2 = \sum_{i=1}^{m}\|a_i\|^2 \sum_{j=1}^{n}\|b_j\|^2 = \|A\|^2\|B\|^2,$$

i.e.,

$$\|AB\| \leq \|A\|\,\|B\|.$$

10.13 Laplacian matrix fo a graph. Let A be the incidence matrix of a directed graph with n nodes and m edges (see 7.3). The *Laplacian matrix* associated with the graph is defined as $L = AA^T$, which is the Gram matrix of A^T. It is named after the mathematician Pierre-Simon Laplace.

(a) Show that $\mathcal{D}(v) = v^T L v$, where $\mathcal{D}(v)$ is the Dirichlet energy defined on page 135.

(b) Describe the entries of L. *Hint.* The following two quantities might be useful: The *degree* of a node, which is the number of edges that connect to the edge (in either direction), and the number of edges that connect a pair of distinct nodes (in either direction).

Solution:

(a) It is easy to find out that

$$v^T L v = v^T A A^T v = \left\| A^T v \right\|$$

which is the definition of the Dirichlet energy.

(b) Let $a_1^T, \ldots, a_n^T$ denote the rows of A, therefore

$$L_{ii} = a_i^T a_i = \sum_{j=1}^{m} (A_{ij})^2$$

means the *degree* of node i.

And for $i \neq j$, the

$$L_{ij} = a_i^T a_j = \sum_{k=1}^{m} A_{ik} A_{jk},$$

when edge k is the edge connected to i and j we have $A_{ik} A_{jk} = -1$, otherwise we have $A_{ik} = 0$ or $A_{jk} = 0$, which will implies $A_{ik} A_{jk} = 0$, therefore L_{ij} means the negative of the number of the edge that connect to node i and j.

10.14 Gram matrix. Let $a_1, \ldots, a_n$ be the columns of the $m \times n$ matrix A. Suppose that the columns all have norm one, and for $i \neq j$, $\angle(a_i, a_j) = 60^\circ$. What can you say about the Gram matrix $G = A^T A$? Be as specific as you can be.

Solution:

We have $\|a_i\| = 1$ for $i = 1, \ldots, n$ and $\angle(a_i, a_j) = 60^\circ$, *i.e.*, $a_i^T a_j = 0.5$ for $i \neq j$. And we know $G_{ii} = \|a_i\|^2 = 1$ and $G_{ij} = a_i^T a_j = 0.5$ for $i \neq j$. Thus the diagonal of G are one, and others are 0.5.

10.15 Pairwise distances from Gram matrix. Let A be an $m \times n$ matrix with columns $a_1, \ldots, a_n$, and associated Gram matrix $G = A^T A$. Express $\left\| a_i - a_j \right\|$ in

terms of G, specifically G_{ii}, G_{ij}, and G_{jj}.

Solution:

With the face that

$$G_{ij} = a_i^T a_j, \qquad G_{ii} = a_i^T a_i = \|a_i\|^2$$

therefore

$$\|a_i - a_j\|^2 = \|a_i\|^2 - 2a_i^T a_j + \|a_j\|^2 = G_{ii} - 2G_{ij} + G_{jj},$$

thus

$$\|a_i - a_j\| = \sqrt{G_{ii} + G_{jj} - 2G_{ij}}.$$

10.16 Covariance matrix. Consider a list of k n-vectors $a_1, \ldots, a_k$, and define the $n \times k$ matrix $A = \begin{bmatrix} a_1 & \cdots & a_k \end{bmatrix}$.

(a) Let the k-vector μ give the means of the columns, *i.e.*, $u_i = \mathbf{avg}(a_i)$, $i = 1, \ldots, k$. (Ths symbol μ is a traditional one to denote an average value.) Give an expression for μ in terms of the matrix A.

(b) Let $\tilde{a}_1, \ldots, \tilde{a}_k$ be the de-meaned versions of $a_1, \ldots, a_k$, and define $\tilde{A}$ as the $n \times k$ matrix $A = \begin{bmatrix} \tilde{a}_1 & \cdots & \tilde{a}_k \end{bmatrix}$. Give a matrix expression for $\tilde{A}$ in terms of A and μ.

(c) The *covariance matrix* of the vector $a_1, \ldots, a_k$ is the $k \times k$ matrix $\sum = (1/N)\tilde{A}^T \tilde{A}$, the Gram matrix of $\tilde{A}$ multiplied with $1/N$. Show that

$$\sum_{ij} = \begin{cases} \mathbf{std}(a_i)^2 & i = j \\ \mathbf{std}(a_i)\mathbf{std}(a_j)\rho_{ij} & i \neq j \end{cases}$$

where ρ_{ij} is the correlation coefficient of a_i and a_j. (The expression for $i \neq j$ assumes that ρ_{ij} is defined, *i.e.*, $\mathbf{std}(a_i)$ and $\mathbf{std}(a_j)$ are nonzero. If not, we interpret the formula as $\sum_{ij} = 0$.) Thus the covariance matrix encodes the standard deviations of the vectors, as well as correlation between all pairs. The covarianve matrix is widely used in probability and statistics.

(d) Let $z_1, \ldots, z_k$ be the standardized versions of $a_1, \ldots, a_k$. (We assume the de-meaned vectors are nonzero.) Derive a matrix expression for $Z = \begin{bmatrix} z_1 & \cdots & z_k \end{bmatrix}$, the matrix of standardized vectors. Your expression should use A, μ, and the numbers $\mathbf{std}(a_1), \ldots, \mathbf{std}(a_k)$.

Solution:

(a) Let k-vector p with all entries be $1/n$, *i.e.*, $p = \begin{bmatrix} 1/n \\ \vdots \\ 1/n \end{bmatrix}$ we have $\mu_i = a_i^T p$, therefore, $\mu = A^T p$.

(b) According to the definition of de-meaned vector, we have $\tilde{a} = a_i - u_i = a_i - u_i \mathbf{1}_n$, (here the subscript of $\mathbf{1}_k$ means the dimension of the one vector.) we have

$$\begin{aligned} \tilde{A} &= \begin{bmatrix} \tilde{a}_1 & \dots & \tilde{a}_k \end{bmatrix} \\ &= \begin{bmatrix} a_1 - u_1 \mathbf{1}_n & \dots & a_k - u_k \mathbf{1}_n \end{bmatrix} \\ &= A - \begin{bmatrix} u_1 \mathbf{1}_n & \dots & u_k \mathbf{1}_n \end{bmatrix} \\ &= A - \mathbf{1}_n u^T \end{aligned}$$

(c) $\sum_{ij} = (1/n)\tilde{a}_i^T \tilde{a}_j$, if $i = j$, we have

$$\sum_{ii} = (1/n)\tilde{a}_i{}^T \tilde{a}_i = \mathbf{std}(a_i)^2,$$

if $i \neq j$, we have

$$\begin{aligned} \sum_{ij} = (1/n)\tilde{a}_i^T \tilde{a}_j &= \frac{\|\tilde{a}_i\|}{\sqrt{n}} \frac{\|\tilde{a}_j\|}{\sqrt{n}} \frac{\tilde{a}_i^T \tilde{a}_j}{\|\tilde{a}_i\| \|\tilde{a}_j\|} \\ &= \mathbf{std}(a_i)\mathbf{std}(a_j)\rho_{ij} \end{aligned}$$

(d) We have $z_i = \tilde{a}_i / \mathbf{std}(a_i)$ for $i = 1, \dots, k$. Let

$$q = \begin{bmatrix} 1/\mathbf{std}(a_1) & & \\ & \ddots & \\ & & 1/\mathbf{std}(a_k) \end{bmatrix}$$

i.e., q is an $k \times k$ diagonal matrix for $1/\mathbf{std}(a_1), \dots, 1/\mathbf{std}(a_k)$, or $\mathbf{diag}(1/\mathbf{std}(a_i))$. we have $z = \tilde{A}q = (A - \mathbf{1}_n u^T)q = Aq - \mathbf{1}_n u^T q$

10.17 Patients and symptoms. Each of a set of N patients can exhibit any number of a set of n symptoms. We express this as an $N \times n$ matrix S, with

$$S_{ij} = \begin{cases} 1 & \text{patient } i \text{ exhibits symptom } j \\ 0 & \text{patient } i \text{ does not exhibit symptom } j \end{cases}$$

Give simple English descriptions of the following expressions. Including the dimensions, and describe the entries.

(a) $S\mathbf{1}$.

(b) $S^T\mathbf{1}$.

(c) S^TS.

(d) SS^T.

Solution:

Let N n-vector a_i^T denote the rows of S, and n N-vector s_i denote the columns of S.

(a) $S\mathbf{1}$ is an N-vector, Its entry $(S\mathbf{1})_i = \sum_{j=1}^{n} S_{ij}$ means the total symptoms that patient i exhibits.

(b) $S^T\mathbf{1}$ is an n-vector, it entry $(S^T\mathbf{1})_i = \sum_{j=1}^{N} S_{ji}$ means the total patients who exhibit symptom i.

(c) S^TS is an $n\times n$ matrix, it entry $(S^TS)_{ij} = \sum_{k=1}^{N} S_{ki}S_{kj}$ means the total patient who exhibit both symptom i and j.

(d) SS^T is an $N\times N$ matrix, it entry $(SS^T)_{ij} = \sum_{k=1}^{n} S_{ik}S_{jk}$ means the total symptoms that both patient i and patient j exhibit.

10.18 Students, classes, and majors. We consider m students, n classes, and p majors. Each student can be in any number of the classes (although we'd expect the number to range from 3 to 6), and can have any number of the majors (although the common values would be 0, 1, or 2). The data about the students' classes and majors are given by an $m\times n$ matrix c and an $m\times p$ matrix M, where

$$C_{ij} = \begin{cases} 1 & \text{student } i \text{ is in class } j \\ 0 & \text{student } i \text{ is not in class } j, \end{cases}$$

and

$$M_{ij} = \begin{cases} 1 & \text{student } i \text{ is in major } j \\ 0 & \text{student } i \text{ is not in major } j. \end{cases}$$

(a) Let E be the n-vector with E_i being the enrollment in class i. Express E using matrix notation, in terms of the matrix C and M.

(b) Define the $n \times p$ matrix S where S_{ij} is the total number of students in class i with major j. Express S using matrix notation, in terms of the matrices C and M.

Solution:

(a) $E = C^T \mathbf{1}$

(b) $S = C^T M$

10.19 Student group membership. Let $G \in \mathbf{R}^{m \times n}$ represent a contingency matrix of m students who are members of n groups:

$$G_{ij} = \begin{cases} 1 & \text{student } i \text{ is in group } j \\ 0 & \text{student } i \text{ is not in group } j. \end{cases}$$

(A student can be in any number of the groups.)

(a) What is the meaning of the 3rd columns of G?

(b) What is the meaning of the 15th row of G?

(c) Give a simple formula (using matrices, vector, etc.) for the n-vector M, where M_i is the total membership (*i.e.*, number of students) in group i.

(d) Interpret $(GG^T)_{ij}$ in simple English.

(e) Interpret $(G^T G)_{ij}$ in simple English.

Solution:

(a) The 3rd column of G means which students are in group 3.

(b) The 15th row of G means what groups are student 15 in.

(c) Use $G^T \mathbf{1}_m$.

(d) $(GG^T)_{ij}$ means the total groups that student i and j both are in.

(e) $(G^T G)_{ij}$ means the total students that both in group i and j.

10.20 Products, materials, and locations. P different products each require some amount of different materials, and are manufactured in L different locations, which have different material costs. We let G_{lm} denote the cost of material m in location l, for $l = 1, \ldots, L$ and $m = 1, \ldots, M$.

We denote Q_{mp} denote the amount of material m required to manufacture on unit of product p, for $m = 1,\ldots,M$ and $p = 1,\ldots,P$. Let T_{pl} denote the total cost manufacture product p in location l, for $p = 1,\ldots,P$ and $l = 1,\ldots,L$. Give an expression for the matrix T.

Solution:

We have

$$T_{ij} = \sum_{k=1}^{M} Q_{ki} C_{kj},$$

therefore

$$T = Q^T C^T.$$

10.21 Integral of product of polynomials. Let p and q be two quadratic polynomials, given by

$$p(x) = c_1 + c_2 x + c_3 x^2, \qquad q(x) = d_1 + d_2 x + d_3 x^2.$$

Express the integral $J = \int_0^1 p(x)q(x)dx$ in the form $J = c^T G d$, where G is a 3×3 matrix. Give the entries of G (as number).

Solution:

Integral of a polynomial is also a polynomial. Suppose the coefficients of $p(x)q(x)$ is 5-vector a, a is convolution of c and d, *i.e.*,

$$a = c * d = \begin{bmatrix} c_1 & & \\ c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \\ & & c_3 \end{bmatrix} d.$$

It is easy to find out that the integral J is the sums of its coifficients a multiply correspond index, *i.e.*,

$$J = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} a.$$

Thus

$$J = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} c_1 & & \\ c_2 & c_1 & \\ c_3 & c_2 & c_1 \\ & c_3 & c_2 \\ & & c_3 \end{bmatrix} d$$
$$= \begin{bmatrix} c_1 & 2c_1 & 3c_1 \\ 2c_2 & 3c_2 & 4c_2 \\ 3c_3 & 4c_3 & 5c_3 \end{bmatrix} d$$
$$= c^T \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix} d$$

10.22 Composition of linear dynamical systems. We consider two time-invariant linear dynamical systems with outputs. The first one is given by

$$x_{t+1} = Ax_t + Bu_t, \qquad y_t = Cx_t, \quad t = 1, 2, \ldots,$$

with state x_t, input u_t, and output y_t. The second is given by

$$\tilde{x}_{t+1} = \tilde{A}\tilde{x}_t + \tilde{B}w_t, \qquad v_t = \tilde{C}\tilde{x}_t, \quad t = 1, 2, \ldots,$$

with state $\tilde{x}_t$, input w_t, and output v_t. We now connect the output of the first linear dynamical system to the input of the second one, which means we take $w_t = y_t$. (This is called the *composition* of the two system.) Show that this composition can also be expressed as a linear dynamical system with state $z_t = (x_t, \tilde{x}_t)$, input u_t and output v_t. (Give the state transimiton matrix, input matrix, and output matrix.)

Solution:

State transimition equation is

$$z_{t+1} = \begin{bmatrix} x_{t+1} \\ \tilde{x}_{t+1} \end{bmatrix} = \begin{bmatrix} A & 0 \\ \tilde{B}C & \tilde{A} \end{bmatrix} \begin{bmatrix} x_t \\ \tilde{x}_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t = \begin{bmatrix} A & 0 \\ \tilde{B}C & \tilde{A} \end{bmatrix} z_t + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t,$$

Output equation is

$$v_t = \begin{bmatrix} 0 & \tilde{C} \end{bmatrix} z_t.$$

Let

$$A' = \begin{bmatrix} A & 0 \\ \tilde{B}C & \tilde{A} \end{bmatrix}, \qquad B' = \begin{bmatrix} B \\ 0 \end{bmatrix}, \qquad C' = \begin{bmatrix} 0 & \tilde{C} \end{bmatrix},$$

we have

$$z_{t+1} = A'z_t + B'u_t, \qquad v_t = C'z_t, \quad t = 1,2,\ldots.$$

10.23 Suppose A is an $n \times n$ matrix that satisfies $A^2 = 0$. Does this imply that $A = 0$? (This is the case when $n = 1$.) If this is (always) true, explain why. If it is not, give a specific counterexample, *i.e.*, a matrix A that is nonzero but satisfies $A^2 = 0$.

Solution:

It is not always true.

Consider the matrix $A = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$, we have

$$A^2 = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

but $A \neq 0$.

10.24 Matrix power identity. A student says that for any square matrix A,

$$(A + 1)^3 = A^3 + 3A^2 + 3A + I.$$

Is she right? If she is, explain why; if she is wrong, give a specific counterexample, *i.e.*, a square matrix A for which it does not hold.

Solution:

She is right. For

$$\begin{aligned} (A + I)^3 &= (A + I)^2(A + I) \\ &= (A^2 + AI + IA + I^2)(A + I) \\ &= (A^2 + 2A + I)(A + I) \\ &= A^3 + 2A^2 + IA + A^2I + 2AI + I^2 \\ &= A^3 + 3A^3 + 3A + I \end{aligned}$$

10.25 Squareroots of the identity. The number 1 has tow squareroots (*i.e.*, numbers who square is 1), 1 and -1. The $n \times n$ identity matrix I_n has many more squareroots.

(a) Find all diagonal squareroots of I_n. How many are there? (For $n = 1$, you should get 2.)

(b) Find a nondiagonal 2×2 matrix A that satisfies $A^2 = I$. This means that in general there are even more squreroots of I_n than you found in part (a).

Solution:

(a) Suppose the diagonal matrix is $A = \mathbf{diag}(a_i)$, we have $A^2 = \mathbf{diag}(a_i^2)$, if $A^2 = I$, means $a_i^2 = 1$ for $i = 1, \ldots, n$, every a_i have two choices (*i.e.*, 1 or -1). there for total number of the choices of A is 2^n.

(b) Consider the matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

10.26 Circular shift matrices. Let A be the 5×5 matrix

$$A = \begin{matrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{matrix}$$

(a) How is Ax related to x? Your answer should be in English. *Hint.* See exercise title.

(b) What is A^5? *Hint.* The answer should make sense, given you answer to part (a).

Solution:

(a) $Ax = (e_5^T x, e_1^T x, e_2^T x, e_3^T x, e_4^T x) = (x_5, x_1, x_2, x_3, x_4)$. Means Ax shift the series x right one index.

(b) A^5 means shift the series tight with five index, which will be the series itself, therefore $A^5 = I$.

10.27 Dynamics of an econimy. Let $x_1, x_2, \ldots$ be n-vectors that give the level of economic activity of a country in year $1, 2, \ldots$, in n different sectors (like energy, defense, manufacturing). Specifically, $(x_t)_i$ is the level of economic activity in economic sector i (say, in billions of dollar) in year t. A common model that connects these economic activity vector is $x_{t+1} = Bx_t$, where B is an $n \times n$ matrix. (see exercise 9.2.)

Give a matrix expression for the total economic activity across all sectors in year $t = 6$, in terms of the matrix B and the power of intial activity levels x_1. Suppose you can increase economic activity in year $t = 1$ by some fixed amount (say, one billion dollars) in *one* sector, by government spending. How should you choose which sector to stimulate so as to maximize the total economic output in year $t = 6$?

Solution:

Total economic activity across all sectors in year $t = 6$ is $\mathbf{1}^T A^5 x_1$. In order to maximize the total economic activity output in year $t = 6$, we should find out the maximum value in a $\mathbf{1}^T A^5$, and change the correspond index in x_1.

10.28 Controllability matrix. Consider the time-invariant linear dymanical system $x_{t+1} = Ax_t + Bu_t$, with n-vector state x_t and m-vector input u_t. Let $U = (u_1, u_2, \ldots, u_{T-1})$ denote the sequence of input, stacked in one vector. Find the maxtrix C_T for which

$$x_T = A^{T-1}x_1 + C_T U$$

holds. The first term is what x_T would be if $u_1 = \cdots = x_{T-1} = 0$; the second term shows how the sequence of input $u_1, \ldots, u_{T-1}$ affect x_T. The matrix C_T is called the *controllability matrix* of linear dynamical system.

Solution:

We have $x_T = A^{T-1}x_1 + A^{T-2}Bu_1 + \cdots + Bu_{T-1}$, let

$$C_T = \begin{bmatrix} A^{T-2}B & A^{T-3}B & \ldots & B \end{bmatrix}$$

then $x_T = A^{T-1}x_1 + C_T U$.

10.29 Linear dynamical system with 2× down-sampling. We consider a linear dynamical system with n-vector state x_t, m-vector input u_t, and dynamics given by

$$x_{t+1} = Ax_t + bu_t, \qquad t = 1, 2, \ldots,$$

where A is an $n \times n$ matrix and B is $n \times m$. Define $z_t = x_{2t-1}$ for $t = 1, 2, \ldots$, *i.e.*,

$$z_1 = x_1, \quad z_2 = x_3, \quad z_3 = x_5, \ldots.$$

(The sequence z_t is the original state sequence x_t 'dowm-sampling' by 2×.) Define the $(2m)$-vectors w_t as $w_t = (u_{2t-1}, u_{2t})$ for $t = 1, 2, \ldots$, *i.e.*,

$$w_1 = (u_1, u_2), \quad w_2 = (u_3, u_4), \quad w_3 = (u_5, u_6), \ldots .$$

(Each entry of the sequence w_t is a stack of two consecutive original inputs.) Show that z_t, w_t satisfy the linear dynamics equation $z_{t+1} = Fz_t + Gw_t$, for $t = 1, 2, \ldots$. Give the matrices f and G in terms of A and B.

Solution:

$$\begin{aligned} z_{t+1} &= x_{2t+1} \\ &= Ax_{2t} + Bu_{2t} \\ &= A(Ax_{2t-1} + Bu_{2t-1}) + Bu_{2t} \\ &= A^2 x_{2t-1} + ABu_{2t-1} + Bu_{2t} \\ &= A^2 z_t + \begin{bmatrix} AB & B \end{bmatrix} \begin{bmatrix} u_{2t-1} \\ u_{2t} \end{bmatrix} \\ &= A^2 z_t + \begin{bmatrix} AB & B \end{bmatrix} w_t \end{aligned}$$

Let

$$F = A^2, \qquad G = \begin{bmatrix} AB & B \end{bmatrix},$$

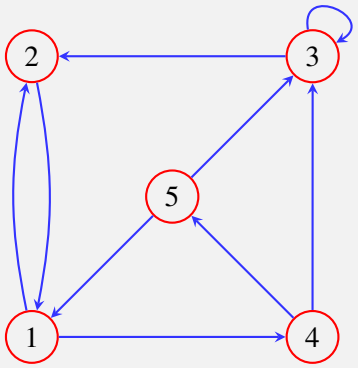
we have

$$z_{t+1} = Fz_t + Gw_t.$$

10.30 Cycles in a graph. A *cycle* of length ℓ in a directed graph is a path of length ℓ that starts and ends at the same vertex. Determine the total number of cycles of length $\ell = 10$ for the directed graph given in the example on page 187. Break this number down into the number of cycles that begin (and end) at vertex 1, vertex 2, ..., vertex 5. (These should add up to the total.) *Hint.* Do not count the cycles by hand.

Solution:

The graph and its adjacency matrix is



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

And we know A_{ij}^{ℓ} means the number of path length ℓ from vertex i to vertex j, therefore A_{ii}^{ℓ} means the number of cycles of length ℓ of vertex i. After calculation, we have

$$A^{10} = \begin{bmatrix} 53 & 31 & 44 & 59 & 57 \\ 72 & 42 & 60 & 81 & 77 \\ 70 & 41 & 58 & 78 & 75 \\ 31 & 18 & 26 & 35 & 33 \\ 18 & 11 & 15 & 20 & 20 \end{bmatrix}$$

The total number of cycles of length $\ell = 10$ is $\mathbf{1}^T \mathbf{diag}(A^{10}) = 53 + 42 + 58 + 35 + 20 = 208$.

10.31 Diameter of a graph. A directed graph with n vertices is described by its $n \times n$ adjacency matrix A (see 10.3).

(a) Derive an expression P_{ij} for the total number of paths, with length no more than k, from vectex j to vertex i. (We include in this total the number of paths of length zero, with go from each vertex j to itself.) *Hint.* You can derive an expression for the matrix P, in terms of A.

(b) The *diameter* D of a graph is the smallest number for which there is a path of length $\leq D$ from node j to node i, for every pair of vertex j and vertex i. Using part (a), explain how the diameter of a graph using matrix operations (such as addition, multiplication).

Remark. Suppose the vertices represent all people on earth, and the graph edges represent acquaintance, *i.e.*, $A_{ij} = 1$ if person j and person i are acquainted. (This graph is symmetric.) Even though n is measured in billions, the diameter of this acquaintance graph is thought to be quite

small, perhaps 6 or 7. In other words, any two people on earth can be connected though a set of 6 or 7 (or fewer) acquaintances. This idea, originally conjectured in the 1920s, is sometimes called *six degrees of separation*.

Solution:

(a) Since A_{ij}^{ℓ} means the toatl of path of length ℓ from vertex j to i, let

$$P = A^0 + \cdots + A^k.$$

here we have P_{ij} denotes the total number of paths, with length no more than k, from vertex j to i.

(b) We denote $P^{(\ell)} = A^0 + \cdots + A^{\ell}$, we try $\ell = 0, 1, \ldots,$ if when $\ell = D$, the all entries of $P^{(D)}$ are nonzero in the first time, means D is the **diameter** of this graph.

10.32 Matrix exponential. You may know that for any real number a, the sequence $(1 + a/k)^k$ converges as $k \to \infty$ to the exponential of a, denoted exp a or e^a. The *matrix exponential* of a square matrix A is defined as the limit of the matrix sequence $(I + A/k)^k$ as $k \to \infty$. (It can shown that this sequence always converges.) The matrix exponential arise in many applications, and is covered in more advanced courses on linear algebra.

(a) Find exp 0 (the zero matrix) and exp I.

(b) Find exp A, for $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$.

Solution:

(a) exp $0 = I^k = I$ as $k \to \infty$. exp $I = (1 + 1/k)^k I = eI$ as $k \to \infty$.

(b) We can observe that for any x and k,

$$\begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix}^k = \begin{bmatrix} 1 & kx \\ 0 & 1 \end{bmatrix}.$$

It is true when $k = 1$, suppose it is true for $1, \ldots, k-1$, we have

$$\begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix}^k = \begin{bmatrix} 1 & (k-1)x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & kx \\ 0 & 1 \end{bmatrix}.$$

now exp $A = \begin{bmatrix} 1 & k/k \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ as $k \to \infty$.

10.33 Matrix equations. Consider two $m \times n$ matrices A and B. Suppose that for $j = 1, \ldots, n$, the jth column of A is a linear combination of the first j columns of B. How do we express this as a matrix equation? Choose one of the matrix equantions below and justify your choice.

(a) $A = GB$ for some upper triangular matrix G.

(b) $A = BH$ for some upper triangular matrix H.

(c) $A = FB$ for some lower triangular matrix F.

(d) $A = BJ$ for some lower triangular matrix J.

Solution:

we have

$$a_j = H_{1j} b_1 + \cdots + H_{jj} b_j,$$

define $H_{ij} = 0$ for $i > j$, we can write the equations above in compact matrix form as

$$A = BH.$$

Here J is upper matrix. The answer is part (b).

Recall the QR factorization in page 190.

10.34 Choose one of the response always, never, or sometimes for each of the statements below. 'Always' means the statement is always true, 'never' means it is never true, and 'Sometimes' means it can be true or false, depending on the particular values of the matrix or matrices. Give a brief justification of each answer.

(a) An upper triangular matrix has linearly independent columns.

(b) The rows of a tall matrix are linearly dependent.

(c) The columns of A are linearly independent, and $AB = 0$ for some nonzero matrix B.

Solution:

(a) Always. Suppose $Ax = 0$, for A is $n \times m$ an upper triangular matrix and x is an m-vector, means

$$x_1 a_1 + \cdots + x_m a_m = 0.$$

We have

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1m}x_m &= 0 \\ A_{21}x_2 + \cdots + A_{2m}x_m &= 0 \\ &\vdots \\ A_{nm}x_m &= 0 \end{aligned}$$

solve the equations begin with the last equation, which is $A_{nm}x_m = 0$, as $A_{nm} \neq 0$, we have $x_m = 0$. Use $x_m = 0$ to the last second equation we have $x_{m-1} = 0$. Using same method we will have $x = 0$, means a_i are linearly independent.

(b) Always. For a $n \times m$ tall matrix, means $n > m$. The dimension of its row-vector is m, and the number of it row-vector is n, as $n > m$, with independent-dimension inequality, we can derive that the rows are linearly dependent.

(c) Never. Assume exist nonzero B, $AB = 0$ holde. Consider the ith column of B, we have $Ab_i = 0$, and because the columns of A are linearly independent, $Ab_i = 0$ hold only if $b_i = 0$, *i.e.*, $B = 0$, conflict with our assumption.

10.35 Orthogonal matrices. Let U and V be two orthogonal $n \times n$ matrices. Show that the matrix UV and $(2n) \times (2n)$ matrix

$$\frac{1}{\sqrt{2}} \begin{bmatrix} U & U \\ V & -V \end{bmatrix}$$

are orthogonal.

Solution:

The jth column of UV is Uv_j, we observe that

$$(Uv_i)^T(Uv_j) = v_i^T U^T U v_j = v_i^T I v_j = v_i^T v_j.$$

Therefore

$$\begin{cases} (Uv_i)^T(Uv_i) = 1 \\ (Uv_i)^T(Uv_j) = 0, \quad i \neq j \end{cases}$$

Thus, UV is orthogonal matrix.

Denote a_i as the columns of $\frac{1}{\sqrt{2}}\begin{bmatrix} U & U \\ V & -V \end{bmatrix}$.

We have

$$\begin{cases} a_i = \dfrac{1}{\sqrt{2}}\begin{bmatrix} u_i \\ v_i \end{bmatrix}, & i \leq n \\ a_i = \dfrac{1}{\sqrt{2}}\begin{bmatrix} u_i \\ -v_i \end{bmatrix}, & i > n \end{cases}$$

If $i \leq n$,

$$a_i^T a_i = (1/2)\begin{bmatrix} u_i & v_i \end{bmatrix}\begin{bmatrix} u_i \\ v_i \end{bmatrix} = (1/2)(u_i^T u_i + v_i^T v_i) = 1.$$

If $i > n$,

$$a_i^T a_i = (1/2)\begin{bmatrix} u_i & -v_i \end{bmatrix}\begin{bmatrix} u_i \\ -v_i \end{bmatrix} = (1/2)(u_i^T u_i + v_i^T v_i) = 1.$$

Thus for all $i = 1, \ldots, 2n$, we have $a_i^T a_i = 1$.

If $i \leq n$ and $j \leq n$ and $i \neq j$, we have

$$a_i^T a_j = (1/2)\begin{bmatrix} u_i & v_i \end{bmatrix}\begin{bmatrix} u_j \\ v_j \end{bmatrix} = (1/2)(u_i^T u_j + v_i^T v_j) = 0.$$

If $i \leq n$ and $j > n$ and $i \neq j$, we have

$$a_i^T a_j = (1/2)\begin{bmatrix} u_i & v_i \end{bmatrix}\begin{bmatrix} u_j \\ -v_j \end{bmatrix} = (1/2)(u_i^T u_j - v_i^T v_j) = 0.$$

If $i > n$ and $j \leq n$ and $i \neq j$, we have

$$a_i^T a_j = (1/2)\begin{bmatrix} u_i & -v_i \end{bmatrix}\begin{bmatrix} u_j \\ v_j \end{bmatrix} = (1/2)(u_i^T u_j - v_i^T v_j) = 0.$$

If $i > n$ and $j > n$ and $i \neq j$, we have

$$a_i^T a_j = (1/2)\begin{bmatrix} u_i & -v_i \end{bmatrix}\begin{bmatrix} u_j \\ -v_j \end{bmatrix} = (1/2)(u_i^T u_j + v_i^T v_j) = 0.$$

Thus for all $i, j = 1, \ldots, 2n$ and $i \neq j$, we have $a_i^T a_j = 0$.
i.e., It is orthogonal matrix.

10.36 Quadratic form. Suppose A is an $n \times n$ matrix and x is an n-vector. The triple product $x^T A x$, a 1×1 matrix which we consider to be a scalar (*i.e.*, number), is called a *quadratic form* of the vector x, with coefficient matrix A. A quadratic form is the vector analog of a quadratic function αu^2, where α and u are both numbers. Quadratic forms arise in many fields and applications.

(a) Show that $x^T A x = \sum_{i,j=1}^n A_{ij} x_i x_j$.

(b) Show that $x^T (A^T) x = x^T A x$. In other words, the quadratic form with the transposed coefficient matrix has the same value for any x. *Hint.* Take the transpose of the triple product $x^T A x$.

(c) Show that $x^T ((A + A^T)/2) x = x^T A x$. In other words, the quadratic form with coefficient matrix equal to the symmetric part of a matrix (*i.e.*, $(A + A^T)/2$) has the same value as the original quadratic form.

(d) Express $2x_1^2 - 3x_1 x_2 - x_2^2$ as a quadratic form, with symmetric coefficient matrix A.

Solution:

(a) $x^T A x = x^T \sum_{j=1}^n a_j x_j = \sum_{j=1}^n x^T a_j x_j = \sum_{j=1}^n \sum_{i=1}^n x_i A_{ij} x_j = \sum_{i,j=1}^n A_{ij} x_i x_j$

(b) $(x^T A x)^T = x^T (A^T) x$, because $x^T A x$ is a scalar, we have $(x^T A x)^T = x^T A x$, thus $x^T (A^T) x = x^T A x$.

(c) We have $2x^T A x = x^T A x + x^T (A^T) x = x^T (A + A^T) x$, both sides divide 2, we have $x^T ((A + A^T)/2) x = x^T A x$.

(d) $2x_1^2 - 3x_1 x_2 - x_2^2 = 2x_1^2 - 1.5 x_1 x_2 - 1.5 x_2 x_1 - x_2^2$ we have

$$2x_1^2 - 3x_1 x_2 - x_2^2 = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1.5 & -1.5 \\ -1.5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

10.37 Orthogonal 2×2 matrices. In this problem, you will show that every 2×2 orthogonal matrix is either a rotation or reflection (see 7.1).

(a) Let

$$Q = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

be an orthogonal 2×2 matrix. Show that following equantions hold:

$$a^2+c^2=1, \qquad b^2+d^2=1, \qquad ab+cd=0.$$

(b) Define $s=ad-bc$. Combine the three equations in part (a) to show that

$$|s|=1, \qquad b=-sc, \qquad d=sa.$$

(c) Suppose $a=\cos\theta$. Show that there are two possible matrices Q: A rotation (conterclockwise over θ radians), and a reflection (through the line that passes through the origin at an angle of $\theta/2$ radians with respect to horizontal).

Solution:

(a) With definition of orthogonal matrix we have

$$\|(a,c)\|=1, \qquad \|(b,d)\|=0, \qquad (a,c)^T(b,d)=0,$$

which is

$$a^2+c^2=1, \qquad b^2+d^2=1, \qquad ab+cd=0.$$

(b) Because $ab+cd=0$, we have $(ab+cd)^2=0\Rightarrow a^2b^2+c^2d^2=-2abcd$, Consider

$$s^2=a^2d^2+b^2c^2-2abcd=a^2d^2+b^2c^2+a^2b^2+c^2d^2=(a^2+c^2)(b^2+d^2)=1,$$

therefore

$$|s|=1.$$

For sc, we have

$$sc=adc-bc^2=a(-ab)-bc^2=-b(a^2+c^2)=-b,$$

Therefore

$$b=-sc.$$

For sa, we have

$$sa=a^2d-abc=a^2d+cdc=d(a^2+c^2)=d.$$

Thus

$$d=sa.$$

(c) Suppose $b=\sin\phi$, we can have $c=\sin\theta$ or $c=-\sin\theta$, $d=\cos\phi$ or

$d = -\cos\phi$.

- If $c = \sin\theta$ and $d = \cos\phi$: with $ab + cd = 0$, we have

$$\cos\theta\sin\phi + \sin\theta\cos\phi = \sin(\theta + \phi) = 0.$$

  Therefore $\sin\phi = -\sin\theta$, $\cos\phi = \cos\theta$, *i.e.*,

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

- If $c = -\sin\theta$ and $d = \cos\phi$: with $ab + cd = 0$, we have

$$\cos\theta\sin\phi - \sin\theta\cos\phi = \sin(\phi - \theta) = 0.$$

  Therefore $\sin\phi = \sin\theta$, $\cos\phi = \cos\theta$, *i.e.*,

$$Q = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

- If $c = \sin\theta$ and $d = -\cos\phi$: with $ab + cd = 0$, we have

$$\cos\theta\sin\phi - \sin\theta\cos\phi = \sin(\phi - \theta) = 0.$$

  Therefore $\sin\phi = \sin\theta$, $\cos\phi = \cos\theta$, *i.e.*,

$$Q = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}.$$

- If $c = -\sin\theta$ and $d = -\cos\phi$: with $ab + cd = 0$, we have

$$\cos\theta\sin\phi + \sin\theta\cos\phi = \sin(\phi + \theta) = 0.$$

  Therefore $\sin\phi = -\sin\theta$, $\cos\phi = \cos\theta$, *i.e.*,

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ -\sin\theta & -\cos\theta \end{bmatrix}.$$

10.38 Orthogonal matrix with nonnegative entries. Suppose the $n \times n$ matrix A is orthogonal, and all of its entries are nonnegative, *i.e.*, $A_{ij} > 0$ for $i, j = 1, \ldots, n$. Show that A must be a permutation matrix, *i.e.*, each entry is either 0 or 1, each row has exactly one entry with value one, and each column has exactly one entry with value one. (See page 132.)

Solution:

Because A is orthogonal matrix, therefore the columns of A are orthonormal basis, and rows of A are orthonormal basis, too.

Suppose ith row has two entries with nonzero value, suppose they are the kth and jth entries of ith row. Consider the kth column a_k and jth column a_j, we have $a_k^T a_j = \sum_{p=0}^{n} A_{pk} A_{pj}$, and with the condition that $A_{ij} \geq 0$, we have $\sum_p^n A_{pk} A_{pj} \geq A_{ik} A_{ij} > 0$, *i.e.*, $a_k^T a_j > 0$, but as a_k and a_j are orthogonal, we have $a_k^T a_j = 0$, confliction. Thus, **The rows of A have excactly one entry with value nonzero.** Consider that for any row b_i^T, we have $b_i^T b_i = 1$, therefore, **the only one entry with nonzero value of the rows of A are one.**

Use the same method, we can derive that the columns of A have exactly one entry with value one.

Thus, A is a permutation matrix.

10.39 Gram matrix and QR factorization. Suppose the matrix A has linear independent columns and QR factorization $A = QR$. What is the relationship between the Gram matrix of A and the Gram matrix of R? What can you say about the angles between the columns of A and the angles between the columns of R?

Solution:

$A^T A = (QR)^T QR = R^T Q^T QR = R^T IR = R^T R$. The Gram matrix of A if **same** with the Gram matrix of R.

Suppose a_i denote the columns of A, and r_i donate the columns of R. Consider the diagonal entries of $A^T A$ and $R^T R$, we have $a_i^T a_i = r_i^T r_i$, with $a_i^T a_i = \|a_i\|^2$ and $r_i^T r_i = \|r_i\|^2$, we have $\|a_i\| = \|r_i\|$.

The i,j entry of $A^T A$ is $a_i^T a_j = \cos\angle(a_i, a_j) \|a_i\| \left\|a_j\right\|$, the i,j entry of $R^T R$ is $r_i^T r_j = \cos\angle(r_i, r_j) \|r_i\| \left\|r_j\right\|$, with $a_i^T a_j = r_i^T r_j$, $\|a_i\| = \|r_i\|$, and $\left\|a_j\right\| = \left\|r_j\right\|$, we have

$$\cos\angle(a_i, a_j) = \cos\angle(r_i, r_j).$$

means

$$\angle(a_i, a_j) = \angle(r_i, r_j) \text{ or } \angle(a_i, a_j) + \angle(r_i, r_j) = 2\pi.$$

10.40 QR factorization of first i columns of A. Suppose the $n \times k$ matrix A has QR factorization $A = QR$. We define the $n \times i$ matrix

$$A_i = \begin{bmatrix} a_1 & \cdots & a_i \end{bmatrix}, \qquad Q_i = \begin{bmatrix} q_1 & \cdots & q_i \end{bmatrix},$$

for $i = 1, \ldots, k$. Define the $i \times i$ matrix R_i as the submatrix of R containing its first i rows and columns, for $i = 1, \ldots, k$. Using index range notation, we have

$$A_i = A_{i:n,1:i}, \qquad Q_i = Q_{1:n,1:i}, \qquad R_i = R_{i:i,1:i}.$$

Show that $A_i = Q_i R_i$ is the QR factorization of A_i. This means that when you compute the QR factorization of A, you are also computing the QR factorization of all submatrices $A_1, \ldots, A_k$.

Solution:

Because R is an upper-triangle matrix, therefore

$$a_i = B_{1i} q_1 + \cdots + b_{ii} q_i.$$

10.41 Clustering via k-means as an approximate matrix factorization. Suppose we run the k-means algorithm on the N n-vectors $x_1, \ldots, n_N$, to obtain the group representatives $z_1, \ldots, z_k$. Define the matrix

$$X = \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix}, \qquad Z = \begin{bmatrix} z_1 & \cdots & z_k \end{bmatrix}.$$

X has size $n \times N$ and Z has size $n \times k$. We encode the assignment of vectors to groups by the $k \times N$ clustering matrix C, with $C_{ij} = 1$ if x_j is assigned to group i, and $C_{ij} = 0$ otherwise. Each column of C is a unit vectors; its transpose is a selector matrix.

(a) Give an interpretation of the columns of matrxi $X - ZC$, and the square norm (matrix) $\|X - ZC\|^2$.

(b) Justify the following statement: The goal of the k-means algorithm is to find an $n \times k$ matrix Z, and a $k \times N$ matrix C, which is the transpose of a selector matrix, so that $\|X - ZC\|$ is small, *i.e.*, $X \approx ZC$.

Solution:

(a) First we give a interpretation of ZC, observe that

$$ZC = \begin{bmatrix} Zc_1 & \cdots Zc_N \end{bmatrix},$$

it is easy to understand that Zc_j means the representative of the group that x_j belonged. Therefore the jth columns of $X - ZC$ means the residual vector between x_j and its group representative. And $\|X - ZC\|$ means the squres of the sum of all the distance between the vectors x_j and its group representatives.

(b) Like what we argued in part (a), as the k-mean algorithm is to find a group representatives which can be expressed as Z and a groups set which can be expressed as C, for which the sum of distance of the vector to its group representatives is smallest, which can be described as $\|X - ZC\|$. Thus they are the same problem with different interpretations.

10.42 A matrix-vector multiplication Ax of an $n \times n$ matrix A and an n-vector x take $2n^2$ flops in genral. Formulate a faster method, with complexity linear in n, for matrix-vector multiplication with the matrix $A = I + ab^T$, where a and b are given n-vectors.

Solution:

Let expand the Ax for $A = I + ab^T$:

$$\begin{aligned} Ax &= (I + ab^T)x \\ &= \left(\begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} + \begin{bmatrix} a_1 b_1 & \dots & a_1 b_n \\ \vdots & \ddots & \vdots \\ a_n b_1 & \dots & a_b b_n \end{bmatrix} \right) \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} a_1(x_1 b_1 + \cdots + x_n b_n) \\ \vdots \\ a_n(x_1 b_1 + \cdots + x_n b_n) \end{bmatrix} \\ &= \begin{bmatrix} x_1 + a_1(x_1 b_1 + \cdots + x_n b_n) \\ \vdots \\ x_n + a_n(x_1 b_1 + \cdots + x_n b_n) \end{bmatrix} \qquad = \begin{bmatrix} x_1 + a_1 x^T b \\ \vdots \\ x_n + a_n x^T b \end{bmatrix} \end{aligned}$$

We now formulate the fast method, first, calculate the value of $x^T b$, which need n flops of multipication and $n - 1$ addition, total $2n - 1$. We store the value of $x^T b$, for the ith entry if Ax, it is $x_i + a_i x^T b$, as we have store the value of $x^T b$, here we need 1 flop for scalar multiplication and 1 flop for scalar addtion, total 2 flops, for

$i = 1, \ldots, n$, we need $2n$ to get the Ax with know the value of $x^T b$.
Thus, the total complexity is $2n - 1 + 2n = 4n$ is linear in n.

10.43 A particular computer takes about 0.2 seconds to multiply two 1500×1500 matrices. About how long would you guess the computer takes to muliply two 3000×3000 matrices? Give your prediction (*i.e.*, the time in seconds), and your (very brief) reasoning.

Solution:

The complexity for multiplying two $n \times n$ matrix requires about $2n^3$ flops, therefore it is about $8 \times 0.2 = 1.6$ second for computer to take the multiply two 3000×3000 matrices.

10.44 Complxity of matrix quadruple product. (See page 182.) We wish to computer the product $E = ABCD$, where A is $m \times n$, B is $n \times p$, C is $p \times q$, and D is $q \times r$.

(a) Find all methods for computeing E using three matrix-matrix multiplications. For example, you can computer AB, CD, and then the product $(AB)(CD)$. Give total number of flops required for each of these methods. *Hint.* There are four other methods.

(b) Which method requires the fewest flops, with dimensions $m = 10$, $n = 1000$, $p = 10$, $q = 1000$, $r = 100$.

Solution:

(a) There are 5 method to compute $ABCD$.

(a) Compute AB, which need $2mnp$ flops, then $(AB)C$, which need $2mpq$ flops, then $(ABC)D$ which need $2mqr$ flops, totally

$$2m(np + pq + qr)$$

flops.

(b) Compute AB which need $2mnp$ flops, compute CD which need $2pqr$ flops, then $(AB)(CD)$ which need $2mpr$ flops, totally

$$2p(mn + qr + mr)$$

flops.

(c) Compute BC which need $2npq$ flops, then $A(BC)$ which need $2mnq$ flops, then $(ABC)D$ which need $2mqr$ flops, totally

$$2q(np + mn + mr)$$

flops.

(d) Compute BC which need $2npq$ flops, then $(BC)D$ which need $2nqr$ flops, then $A(BCD)$ which need $2mnr$ flops, totally

$$2n(pq + qr + mr)$$

flops.

(e) Compute CD which need $2pqr$ flops, then $B(CD)$ which need $2mpr$ flops, then $A(BCD)$ which need $2mnr$ flops, totally

$$2r(pq + mp + mn)$$

flops.

(b) I list the complexity of all method below, from which we can find out the second method requires the fewest flops.

Method Number	Complexity	Value
1	$2m(np + pq + qr)$	$2,400,000$
2	$2p(nm + mr + rq)$	$2,220,000$
3	$2q(rm + mn + np)$	$42,000,000$
4	$2n(pq + qr + rm)$	$222,000,000$
5	$2r(nm + mp + pq)$	$4,020,000$

11 Matrix inverses

11.1 Affine combinations of left inverses. Let Z be a tall $m \times n$ matrix with linearly independent columns, and let X and Y be left inverses of Z. Show that for any scalars α and β satisfying $\alpha + \beta = 1$, $\alpha X + \beta Y$ is also left inverse of Z. It follows that if a matrix has two different left inverses, it has an infinite number of different left inverses.

Solution:

$(\alpha X + \beta Y)Z = \alpha XZ + \beta YZ = \alpha I + \beta I = (\alpha + \beta)I = I$, *i.e.*, $\alpha X + \beta Y$ is left inverse of Z.

11.2 Left and right inverses of a vector. Suppose that x is a nonzero n-vector with $n > 1$.

(a) Does x have a left inverse ?

(b) Does x have a right inverse ?

Solution:

A vector can be consider as an $n \times 1$ tall matrix, therefore it can have a left inverse, but can not have a right inverse.

Consider that there are only one column of this matrix, and its nonzero, therefore it is linearly independent. *i.e.*, it must have a left inverse.

(a) ✓ **YES.**

(b) × **NO.**

11.3 Matrix cancellation. Suppose the scalar a, x, and y satisfy $ax = ay$. When $a \neq 0$ we can conclude that $x = y$; that is, we can cancel the a on the left of the equation. In this exercise we explore the matrix analog of the cancellation, specifically, what properties of A are needed to conclude $X = Y$ from $AX = AY$, for matrices A, X, and Y?

(a) Give an example showing that $A \neq 0$ is not enough to conclude that $X = Y$.

(b) Show that if A is left-invertible, we can conclude from $AX = AY$ that $X = Y$.

(c) Show that if A is not left-invertible, there are matrices X and Y with $X \neq Y$, and $AX = AY$.

Remark. Part (b) and part (c) show that you can cancel a matrix on the left when, and only when, the matrix is left-invertible.

Solution:

(a) Consider $A = \begin{bmatrix} 1 & 1 \end{bmatrix}$ and $X = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $Y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, we have $AX = AY = 1$ but $X \neq Y$.

(b) If A is left-invertible, soppose it is $A^\dagger$, *i.e.*, $A^\dagger A = I$, both side of $AX = AY$ left mutiply $A^\dagger$, we have $A^\dagger AX = A^\dagger AY$, *i.e.*, $X = Y$.

(c) Suppose for any $X \neq Y$, $AX \neq AY$, consider $Y = 0$ and X to be any nonzero vector, we have $AX \neq 0$ if $X \neq 0$, mean $Ax = 0$ only when $x = 0$, *i.e.*, the columns of A are linearly independent, thus, A has left-inverse.

11.4 Transpose of the orthogonal matrix. Let U be an orthogonal $n \times n$ matrix. Show that its transpose U^T is also orthogonal.

Solution:

Because U is orthogonal matrix, it has inverse matrix U^T, we have $U^T U = UU^T = I$. Here we see U^T is left-invertible, mean its columns are linear independent, and its transpose matrix is its left inverse, means U^T is orthogonal.

11.5 Inverse of a block matrix. Consider the $(n+1) \times (n+1)$ matrix

$$A = \begin{bmatrix} I & a \\ a^T & 0 \end{bmatrix},$$

where a is an n-vector.

(a) When if A invertible? Give your answer in terms of a. Justify your answer.

(b) Assuming the condition you found in part (a) holds, give an expression for the inverse matrix A^{-1}.

Solution:

(a) If A is invertible, the columns or rows of it must be linear

independent, Suppose $Ax = 0$, we expand this equation:

$$\begin{cases} x_1 + a_1 x_{n+1} & = 0 \\ \vdots & \\ x_n + a_n x_{n+1} & = 0 \\ a_1 x_1 + \cdots + a_n x_n & = 0 \end{cases}$$

From the first n equations we have $x_i = -a_i x_{n+1}$, for $i = 1, \ldots, n$, use these in the $n+1$ equation, we have $x_{n+1} \sum_{i=1}^{n} a_i = 0$, if $\sum_{i=1}^{n} a_i \neq 0$, implies $x_{n+1} = 0$, with $x_i = a_i x_{n+1}$, we have $x_i = 0$. therefore, if $\sum_{i=1}^{n} a_i \neq 0$, the equation $Ax = 0$ holds only when $x = 0$, *i.e.*, the columns of A are linearly independent. As we know $\sum_{i=1}^{n} a_i = \|a\|^2$, $\sum_{i=1}^{n} a_i \neq 0$ means $\|a\| \neq 0$.

So the condition we found is when $\|a\| \neq 0$, A is invertible.

(b) As A is square, if it is invertible, then its left-inverse is same with right-inverse. (see page 202.) Let's try to find its right-inverse, suppose its right inverse is B, for which $AB = I$ holds. Consider the ith columns of B, for $i = 1, \ldots, n$, we have $Ab_i = e_i$, expand this we have

$$\begin{cases} B_{1,i} + a_1 B_{n+1,i} & = 0 \\ \vdots & \\ B_{i,i} + a_i B_{n+1,i} & = 1 \\ \vdots & \\ B_{n,i} + a_n B_{n+1,i} & = 0 \\ a_1 B_{1,i} + \cdots + a_n B_{n,i} & = 0 \end{cases}$$

use the same method in part (a) as $B_{1,j} = -a_j B_{n+1,i}$ for $j = 1, \ldots, n$ and $j \neq i$, $B_{1,j} = 1 - a_j Bn + 1, i$ for $j = i$, use these in the $n+1$ equation, we have $a_i = B_{n+1,i} \|a\|^2$, means $B_{n+1,i} = \dfrac{a_i}{\|a\|^2}$, then we have $B_{1,j} = -\dfrac{a_j a_i}{\|a\|^2}$, for $j = 1, \ldots, n$ and $j \neq i$, $B_{i,j} = 1 - \dfrac{a_j a_i}{\|a\|^2}$ for $j = i$, we can rewrite this in

vector notation:

$$b_i = \begin{bmatrix} -\frac{a_i}{\|a\|^2}a_1 \\ \vdots \\ 1-\frac{a_i}{\|a\|^2}a_i \\ \vdots \\ -\frac{a_i}{\|a\|^2}a_n \\ \frac{d_i}{\|a\|^2} \end{bmatrix} = \begin{bmatrix} e_i - \frac{a_i}{\|a\|^2}a \\ \frac{d_i}{\|a\|^2} \end{bmatrix}$$

here e_i is an n-vector unit vector. Thus we can rewrite the first n columns of B as

$$B_{1:n+1,1:n} = \begin{bmatrix} I - \frac{aa^T}{\|a\|^2} \\ \frac{a^T}{\|a\|^2} \end{bmatrix}$$

Now let look at the $n+1$ columns of B, we should satisfy

$$\begin{cases} B_{1,i} + a_1 B_{n+1,i} & = 0 \\ \vdots & \\ B_{n,i} + a_n B_{n+1,i} & = 0 \\ a_1 B_{1,i} + \cdots + a_n B_{n,i} & = 1 \end{cases}$$

means $-B_{n+1,n+1}\|a\|^2 = 1 \Rightarrow B_{n+1,n+1} = -\frac{1}{\|a\|^2}$, we hve

$$b_{n+1} = \begin{bmatrix} \frac{a}{\|a\|^2} \\ -\frac{1}{\|a\|^2} \end{bmatrix}$$

Totally, we have

$$B = \begin{bmatrix} I - \frac{aa^T}{\|a\|^2} & \frac{a}{\|a\|^2} \\ \frac{a^T}{\|a\|^2} & -\frac{1}{\|a\|^2} \end{bmatrix}$$

It is also easily to verify that $AB = I_{n+1}$, as

$$AB = \begin{bmatrix} I & a \\ a^T & 0 \end{bmatrix} \begin{bmatrix} I - \frac{aa^T}{\|a\|^2} & \frac{a}{\|a\|^2} \\ \frac{a^T}{\|a\|^2} & -\frac{1}{\|a\|^2} \end{bmatrix} = \begin{bmatrix} I - \frac{Iaa^T}{\|a\|^2} + \frac{aa^T}{\|a\|^2} & \frac{Ia}{\|a\|^2} - \frac{a}{\|a\|^2} \\ a^T I - \frac{a^T aa^T}{\|a\|^2} & \frac{a^T a}{\|a\|^2} \end{bmatrix} = \begin{bmatrix} I_n & 0 \\ 0 & 1 \end{bmatrix} = I_{n+1}$$

11.6 Inverse of a block upper triangular matrix. Let B and D be invertible matrices of sizes $m \times m$ and $n \times n$, respectively, and let C be any $m \times n$ matrix. Find the inverse of

$$A = \begin{bmatrix} B & C \\ 0 & D \end{bmatrix}$$

in terms of B^{-1}, C, and D^{-1}. (This matrix A is called *block upper tianglar.*)

Hint. First get an idea of what the solution should look like by considering the case when B, C, and D are scalars. For the matrix case. Your goal is to find matrices W, X, and Y, Z (in terms of B^{-1}, C, and D^{-1}) that satisfy

$$A \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = I.$$

Use block matrix multiplication to express this as a set of four matrix equations that you can then solve. The method you will find is sometimes called the *block back substitution.*

Solution:

Suppose the inverse of A is $\begin{bmatrix} W & X \\ Y & Z \end{bmatrix}$, then we have

$$\begin{bmatrix} B & C \\ 0 & D \end{bmatrix} \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = \begin{bmatrix} BW + CY & DY \\ BX + CZ & DZ \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

we can get matrix equations

$$\begin{cases} BW + CY = I \\ DY = 0 \\ BX + CZ = 0 \\ DZ = I \end{cases}$$

From the second equation $DY = 0$, we have

$$Y = D^{-1}0 = 0,$$

from the last equation $DZ = 0$, we have

$$Z = D^{-1}I = D^{-1},$$

from the first equation $BW + CY = BW = I$, we have

$$W = B^{-1},$$

from the third equation $BX + CZ = BX + CD^{-1} = 0$, we have

$$X = -B^{-1}CD^{-1}.$$

Thus the inverse of A is

$$\begin{bmatrix} B^{-1} & -B^{-1}CD^{-1} \\ 0 & D^{-1} \end{bmatrix}$$

11.7 Inverse of an upper triangular matrix. Suppose the $n \times n$ matrix R is upper triangular and invertible, *i.e.*, its diagonal entries are all nonzero. Show that R^{-1} is also upper triangular. *Hint.* Use the back substitution to solve $Rs_k = e_k$, for $k = 1, \ldots, n$, and arange that $(s_k)_i = 0$ for $i > k$.

Solution:

Suppose the inverse of R is S, and denote the columns of S as $s_1, \ldots, s_n$, we have $RS = I$, or $Rs_k = e_k$, expand this equation, we have

$$\begin{aligned} R_{11}(s_k)_1 + \ldots + R_{1n}(s_k)_n &= 0 \\ &\vdots \\ R_{kk}(s_k)_k + \ldots + R_{kn}(s_k)_n &= 1 \\ &\vdots \\ R_{nn}(s_k)_n &= 0 \end{aligned}$$

From the last equation, we can get $(s_k)_n = 0$, use the back substitution, we have $(s_k)_i = 0$ for $i > k$, and $(s_k)_k = 1/R_{kk} \neq 0$. Therefore, S is an upper triangular matrix.

11.8 If a matrix is samll, its inverse is large. If a number a is small, its inverse $1/a$ (assuming $a \neq 0$) is large. In this exercise you will explore a matrix analog of this idea. Suppose the $n \times n$ matrix A is invertible.

Show that $\|A^{-1}\| \geq \sqrt{n}/\|A\|$. This implies that if a matrix is small, its inverse is large. *Hint.* You can use the inequality $\|AB\| \leq \|A\|\,\|B\|$, which holds for any matrices for which the product makes sense. (See exercise 10.12.)

Solution:

We have $\|A^{-1}A\| \leq \|A^{-1}\|\,\|A\|$, and $\|A^{-1}A\| = \|I\| = \sqrt{n}$, therefore $\sqrt{n} \leq \|A^{-1}\|\,\|A\|$.

11.9 Push-through identity. Suppose A is $m \times n$, B is $n \times m$, and the $m \times m$ matrix $I + AB$ is invertible.

(a) Show that the $n \times n$ matrix $I + BA$ is invertible. *Hint.* Show that $(I + BA)x = 0$ implies $(I + AB)y = 0$ where $y = Ax$.

(b) Establish the identity

$$B(I + AB)^{-1} = (I + BA)^{-1}B.$$

This is sometimes called the *push-through identity* since the matrix B appearing on the left 'moves' into the inverse, and 'pushs' the B in the inverse out to the right side. *Hint.* Start with the identity

$$B(I + AB) = (I + BA)B,$$

and multiply on the right by $(I + AB)^{-1}$, and on the left by $(I + BA)^{-1}$.

Solution:

(a) Let x satisfies

$$(I + BA)x = 0,$$

multiply both side by A, we have

$$Ax + ABAx = 0,$$

i.e.,

$$(I + AB)Ax = 0,$$

as $I + AB$ is invertible, we have

$$Ax = 0,$$

i.e.,

$$x = 0,$$

therefore $I + BA$ is invertible.

(b) In order to prove $B(I + AB)^{-1} = (I + BA)^{-1}B$, multiply on the left by $I + BA$, it is same to prove

$$(I + BA)B(I + AB)^{-1} = B,$$

multiply on the right by $I + AB$, it is same to prove

$$(I + BA)B = B(I + AB),$$

which is always holds, thus we have

$$B(I + AB)^{-1} = (I + BA)^{-1}B.$$

11.10 Reverse-time linear dynamical system. A linear dynamical system has the form

$$x_{t+1} = Ax_t,$$

where x_t in the (n-vector) state in period t, and A is the $n \times n$ dynamical matrix. This formula gives the state in the next period as a function of the current state.

We want to derive a recursion of the form

$$x_{t-1} = A^{rev}x_t,$$

which gives the previous state as a function of the current state. We call this ths *reverse time linear dynamical system.*

(a) When is this possible? When it is possible, what is A^{rev}?

(b) For the specific linear dynamical system with dynamical matrix

$$A = \begin{bmatrix} 3 & 2 \\ -1 & 4 \end{bmatrix},$$

find A^{rev}, or explain why the reverse time linear dynamical system doesn't exist.

Solution:

(a) When A is invertible, this is possible. First, we show that when A is invertible, this is possible: suppose A^{-1} is the inverse of A, *i.e.*, $A^{-1}A = I$, we have $A^{-1}x_{t+1} = A^{-1}Ax_t$, means $x_t = A^{-1}x_{t+1}$, we have $A^{rev} = A^{-1}$.

Second, we show that only when A is invertible, this is possible: Suppose A is not invertible, but exist A^{rev} that $x_{t-1} = A^{rev}x_t$, with $x_t = Ax_{t-1}$, we have $x_{t-1} = A^{rev}Ax_{t-1}$, is the state x_t is nonzero we have $A^{rev}A = I$, means A^{rev} is the inverse of A.

(b) For $A = \begin{bmatrix} 3 & 2 \\ -1 & 4 \end{bmatrix}$, its inverse is]{

11.11 Interpolation of rational functions. (Continuation of exercise 8.8.) Find a ratinal function

$$f(t) = \frac{c_1 + c_2 t + c_3 t^2}{1 + d_1 t + d_2 t^2}$$

that satisfies the following interpolation conditions:

$$f(1) = 2, \qquad f(2) = 5, \qquad f(3) = 9, \qquad f(4) = -1, \qquad f(5) = -4.$$

In exercise 8.8 these conditions were expressed as a set of linear quations in the coeficients c_1, c_2, c_3, d_1 and d_2; here we are asking you to form and (numerically) solve the system of equations. Plot the rational function you find over the range $x = 0$ to $x = 6$. Your plot should include markers at the interpolation point $(1,2),\ldots,(5,-4)$. (Your ratinal function graph should pass through these points.)

Solution:

We can form the equations as

$$\begin{bmatrix} 1 & 1 & 1 & -2 & -2 \\ 1 & 2 & 4 & -10 & -20 \\ 1 & 3 & 9 & -27 & -81 \\ 1 & 4 & 16 & 4 & 16 \\ 1 & 5 & 25 & 20 & 100 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ 9 \\ -1 \\ -4 \end{bmatrix}$$

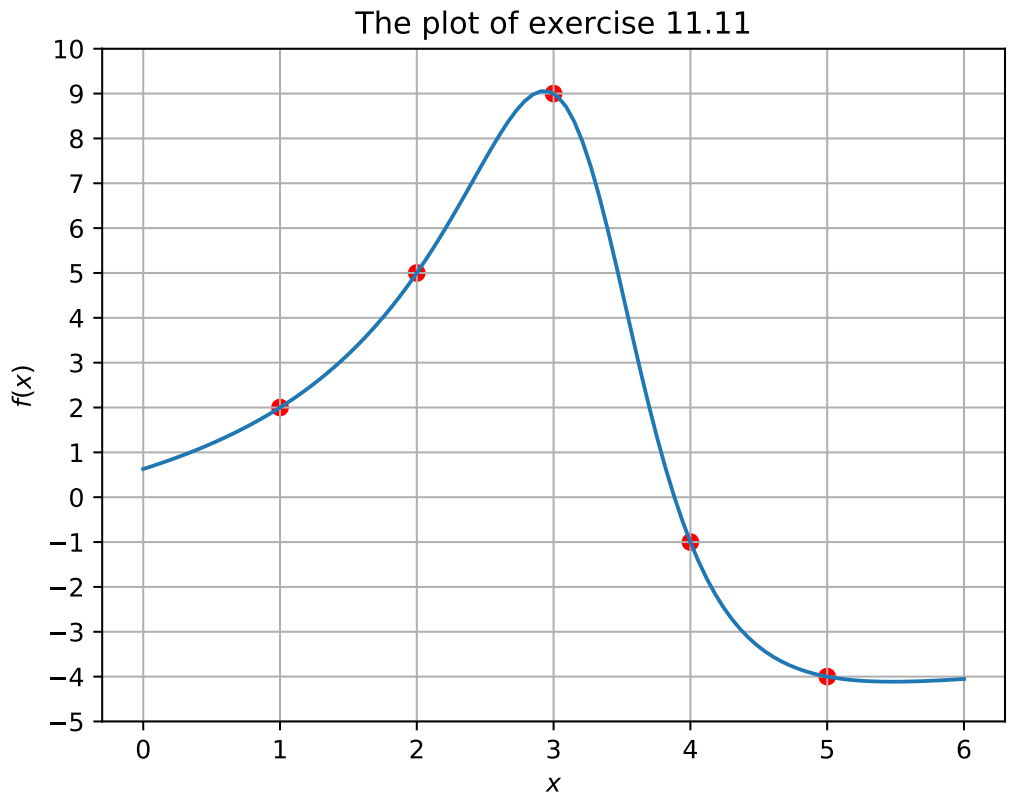
We have

$$A^{-1} = \begin{bmatrix} 2.59259259 & -2.09876543 & 0.74074074 & -0.55555556 & 0.32098765 \\ -1.81790123 & 2.48353909 & -1.04320988 & 0.90740741 & -0.52983539 \\ 0.29938272 & -0.50823045 & 0.22839506 & -0.12962963 & 0.1100823 \\ 0.05864198 & -0.10699588 & -0.0308642 & 0.14814815 & -0.06893004 \\ -0.02160494 & 0.04526749 & -0.00617284 & -0.03703704 & 0.01954733 \end{bmatrix}$$

therefore

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0.62962963 \\ 0.60493827 \\ -0.19753086 \\ -0.56790123 \\ 0.08641975 \end{bmatrix}$$

I plot the function as following:



11.12 Combinations of invertible matrices. Suppose the $n \times n$ matrices A and B are both invertible. Determine whether each of the matrices given below is invertible, without any further assumptions about A and b.

(a) $A + B$.

(b) $\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$.

(c) $\begin{bmatrix} A & A+B \\ 0 & B \end{bmatrix}$.

(d) ABA.

Solution:

(a) × **NO.** For $A = B = \begin{bmatrix} 1 \end{bmatrix}$, $A + B$ is not invertible.

(b) ✓ **YES.** $\begin{bmatrix} A^{-1} & 0 \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = I$.

(c) ✓ **YES.** $\begin{bmatrix} A^{-1} & -B^{-1} - A^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} A & A+B \\ 0 & B \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = I$

(d) ✓ **YES.** $A^{-1}B^{-1}A^{-1}ABA = A^{-1}B^{-1}IBA = A^{-1}B^{-1}BA = A^{-1}IA = A^{-1}A = I$.

11.13 Another left inverse. Suppose the $m \times n$ matrix A is tall and has linearly independent columns. One left inverse of A is the pseudo-inverse $A^\dagger$. In this problem we explore another one. Write A as the block matrix

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where A_1 is $n \times n$. We assume that A_1 is invertible (which need not happend in general). Show that the following matrix is a left inverse of A:

$$\tilde{A} = \begin{bmatrix} A_1^{-1} & 0_{n \times (m-n)} \end{bmatrix}.$$

Solution:

We have

$$\begin{aligned} \tilde{A}A &= \begin{bmatrix} A_1^{-1} & 0_{n \times (m-n)} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \\ &= A_1^{-1}A_1 + 0 \\ &= I \end{aligned}$$

Thus, $\tilde{A}$ is a left-inverse of A.

11.14 Middle inverse. Suppose A is an $n \times p$ matrix and b is a $q \times n$ matrix. If a $p \times q$ matrix X exists that satisfies $AXB = I$, we call it *middle inverse* of the pair A and B. (This is not a standard concept.) Note that when A

or B is an identity matrix, the middle inverse reduces to the right or left inverse, respectively.

(a) Descibe the conditions on A and B under which a middle inverse X exists. Give your answer using only the following four concept: Linear independence of the rows or columns of A, and linear independence of the rows or columns of B. You must justify your answer.

(b) Give an expression for a middle inverse, assuming the conditions in part (a) hold.

Solution:

(a) If $AXB = I$, means A is right-invertible, *i.e.*, rows of A are linearly independent. Similarly, B is left-invertible, *i.e.*, columns of B are linearly independent.

(b) If the conditions of part (a) hold, *i.e.*, A has right inverse A' and B has left inverse B', let $X = A'B'$, the $AXB = AA'B'B = I$.

11.15 Invertibility of population dynamics matrix. Consider the population dynamics matrix

$$A = \begin{bmatrix} b_1 & b_2 & \dots & b_{99} & b_{100} \\ 1-d_1 & 0 & \dots & 0 & 0 \\ 0 & 1-d_2 7\dots & 0 & 0 & \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1-d_{99} & 0 \end{bmatrix},$$

where $b_i \geq 0$ are the birth rates and $0 \leq d_i \leq 1$ are the death rates. What are the conditions on b_i and d_i under which A is invertible? (If the matrix is never invertible or always invertible, say so.) Justify your answer.

Solution:

The condition is $d_i \neq 1$ and $b_{100} \neq 0$. We can use back substitution to justify this. If columns of A are linear independent, A is

invertible. Consider $Ax = 0$, expand this as

$$\begin{aligned} b_1x_1 + b_2x_2 + \cdots + b_{100}x_{100} &= 0 \\ (1 - d_1)x_1 &= 0 \\ (1 - d_2)x_2 &= 0 \\ &\vdots \\ (1 - d_{99})x_{99} &= 0 \end{aligned}$$

As $d_i \neq 0$, we have $x_i = 0$, for $i = 1,\ldots,99$. and if $b_{100} \neq 0$, $x_{100} = 0$. Thus A is invertible.

11.16 Inverse of running sum matrix. Find the inverse of the $n \times n$ running sum matrix,

$$S = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ 1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \ldots & 1 & 0 \\ 1 & 1 & \ldots & 1 & 1 \end{bmatrix}$$

Does your answer make sense?

Solution:

Denote the inverse of A as B, *i.e.*, $BA = AB = I$. Consider the ith column of B, b_i, we have $Ab_i = e_i$, we can expand this as

$$B = I + \begin{bmatrix} 0 & 0 \\ -I & 0 \end{bmatrix}$$

11.17 A matrix identity. Suppose A is a square matrix that satisfies $A^k = 0$ for some k. (Such a matrix is called *nipotent*.) A student guesses that $(I - A)^{-1} = I + A + \cdots + A^{k-1}$, based on the infinite series $1/(1 - a) = 1 + a + a^2 + \ldots$, which holds for numbers a that satisfy $|a| < 1$.

Is the student right or wrong? If right, show that her assertion holds with no further assumptions about A. If she is wrong, give a conterexample, *i.e.*, a matrix A that satisfies $A^k = 0$, but $I + A + \cdots + A^{k-1}$ is not the inverse of $I - A$.

Solution:

She is right.

$$\begin{aligned}&(I-A)(I+A+\cdots+A^{k-1})\\&=I+A+\cdots+A^{k-1}-(A+\cdots+A^{K})\\&=I\end{aligned}$$

11.18 Tall-wide product. Suppose A is an $n\times p$ matrix and B is a $p\times n$ matrix, so $C=AB$ makes sense. Explain why C cannot invertible if A is tall and B is wide, *i.e.*, if $p<n$. *Hint.* First argure that the columns of B must be linearly independent.

Solution:

If C is invertible, *i.e.*, exsist C^{-1}, that $C^{-1}C=CC^{-1}=I$. as $I=C^{-1}C=C^{-1}AB$, means B is left-invertible, as $C^{-1}A$ is its left inverse, thus B must be tall. With same method, we can conclude that A must be wide.

11.19 Control restricted to one time period. A linear dynamical system has the form $x_{t+1}=Ax_t+u_t$, where the n-vector x_t is the state and u_t is the input at time t. Our goal is to choose the input sequence $u_1,\ldots,u_{N-1}$ so as to achieve $x_N=x^{des}$, where x^{des} is a given $n-vector$, and N is given. The input sequence must satisfy $u_t=0$ unless $t=K$, where $K<N$ is given. In other words, the input can only act at time $t=K$. Give a formula for u_K that achieve this goal. Your formula can involve A, N, K, x_1, and x^{des}. You can assume that A is invertible. *Hint.* First derive an expression for x_K, then use the dynamics equation to find x_{K+1}. Form x_{K+1} you can find x_N.

Solution:

As $u_i=0$ for $i=1,\ldots,K-1,K+1,\ldots,N$, we have $x_K=A^{K-1}x_1$, therefore $x_{K+1}=A^Kx_1+u_K$, $x_N=A^{N-K-1}X_{K+1}=A^{N-1}x_1+A^{N-K-1}u_K$, let $u_K=u^{des}$, we have $A^{N-1}x_1+A^{N-K-1}u_K=u^{des}$, we have

$$u_K=A^{K+1-N}u^{des}-A^Kx_1.$$

11.20 Immigration. The population dynamics of a country is given by $x_{t+1} = Ax_t + u$, $t = 1,\ldots,T-1$, where the 100-vector x_t gives the population age distribution in year t, and u gives the immigration age distribution (which negative entries meaning emigration), which we assume is constant (*i.e.*, does nont vary with t). You are given A, x_1, and x^{des}, a level of immigration u that achieves $x_T = x^{des}$.

Give a matrix formula for u. If your formula only makes sense when some conditions hold (for example invertibility of one or more matrices), say so.

Solution:

We have $x_T = A^{T-1}x_1 + A^{T-2}u + \cdots + u$, let $x_T = x^{des}$, we get

$$A^{T-1}x_1 + A^{T-2}u + \cdots + u = x^{des}.$$

i.e.,

$$(A^{T-2} + \cdots + 1)u = x^{des} - A^{T-1}x_1.$$

If $A^{T-2} + \cdots + 1$ has inverse, denoted as B, we have

$$u = Bx^{des} - BA^{T-1}x_1.$$

11.21 Quadrature weights. Consider a quadrature problem (see exercise 8.12) with $n = 4$, with point $t = (-0.6, -0.2, 0.2, 0.6)$. We require that the quadrature rule be exact for all polynomials of degree up to $d = 3$.

Set this up as a square system of linear equations in the weight vector. Numerically solve this system to get the weights. Compute the true value and the quadrature estimate,

$$\alpha = \int_{-1}^{1} f(x)dx, \qquad \hat{\alpha} = w_1 f(-0.6) + w_2 f(-0.2) + w_3 f(0.2) + w_4 f(0.6),$$

for the specific function $f(x) = e^x$.

Solution:

With the solution of 8.12, we have

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -0.6 & -0.2 & 0.2 & 0.6 \\ 0.36 & 0.04 & 0.04 & 0.36 \\ -0.216 & -0.008 & 0.008 & 0.216 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0.54881164 \\ 0.81873075 \\ 1.22140276 \\ 1.8221188 \end{bmatrix}$$

The inverse of square matrix in the left side is:

$$\begin{bmatrix} -0.0625 & 0.10416667 & 1.5625 & -2.60416667 \\ 0.5625 & -2.8125 & -1.5625 & 7.8125 \\ 0.5625 & 2.8125 & -1.5625 & -7.8125 \\ -0.0625 & -0.10416667 & 1.5625 & 2.60416667 \end{bmatrix},$$

we have

$$w = \begin{bmatrix} -2.78567551 \\ 10.33288762 \\ -13.53235815 \\ 6.53395767 \end{bmatrix}$$

Thus we have

$$\alpha = e - e^{-1} \approx 2.35 \qquad \hat{\alpha} \approx 2.30.$$

11.22 Properties of pseudo-inverses. For an $m \times n$ matrix A and its pseudo-inverse $A^\dagger$, show that $A = AA^\dagger A$ and $A^\dagger = A^\dagger AA^\dagger$ in each of the following cases.

(a) A is tall with linearly independent columns.

(b) A is wide with linearly independent rows.

(c) A is square and invertible.

Solution:

(a) Because A has linearly independent columns, $A^\dagger$ is left inverse of A, *i.e.*$A^\dagger A = I$, we have $AA^\dagger A = AI = A$ and $A^\dagger AA^\dagger = IA^\dagger = A$.

(b) If A has linearly independent rows, $A^\dagger$ is right inverse of A, *i.e.*$AA^\dagger = I$, we $AA^\dagger A = IA = A$ and $A^\dagger AA^\dagger = A^\dagger I = A$.

(c) If A has inverse, $A^\dagger$ is its inverse.

11.23 Product of pseudo-inverses. Suppose A and D are right-verertible matrices and the product AD exists. We have seen that if B is a right inverse of A and E is a right inverse of D, the EB is a right inverse of AD. Now suppose B is the pseudo-inverse of A and E is the pseudo-inverse of D. Is EB the pseudo-inverse of AD? Prove that this is always true or given an example for which it is false.

Solution:

We have

$$B = A^\dagger = A^T(AA^T)^{-1}, \qquad E = D^\dagger = D^T(DD^T)^{-1},$$

therefore

$$EB = D^T(DD^T)^{-1}A^T(AA^T)^{-1},$$

the pseudo-inverse of AD is

$$(AD)^\dagger = (AD)^T\left(AD(AD)^T\right)^{-1} = D^TA^T(ADD^TA^T)^{-1},$$

there is no promise that $EB = (AD)^\dagger$, *i.e.*,

$$(AD)^\dagger \neq D^\dagger A^\dagger.$$

For instance, let

$$A = \begin{bmatrix}1 & 2\end{bmatrix}, \qquad D = \begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix},$$

we have

$$B = \begin{bmatrix}0.2\\ 0.4\end{bmatrix}, \qquad E = \begin{bmatrix}1 & 0\\ 0 & 0.5\end{bmatrix},$$

therefore

$$EB = \begin{bmatrix}1 & 0\\ 0 & 0.5\end{bmatrix}\begin{bmatrix}0.2\\ 0.4\end{bmatrix} = \begin{bmatrix}0.2\\ 0.2\end{bmatrix}.$$

And

$$AD = \begin{bmatrix}1 & 2\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix} = \begin{bmatrix}1 & 4\end{bmatrix},$$

therefore

$$(AD)^\dagger = \begin{bmatrix}\frac{1}{17}\\ \frac{4}{17}\end{bmatrix} \neq EB.$$

11.24 Simultaneous left inverse. The two matrices

$$A = \begin{bmatrix}1 & 2\\ 3 & 1\\ 2 & 1\\ 2 & 2\end{bmatrix}, \qquad B = \begin{bmatrix}3 & 2\\ 1 & 0\\ 2 & 1\\ 1 & 3\end{bmatrix}$$

are both left-invertible, and have multiple left inverses. Do they have a common left inverse? Explain how to find a 2×4 matrix C that satisfies $CA = CB = I$, or determine that no such matrix exists. (You can use numerical computing to find C.) *Hint.* There is nothing special about

the particular entries of the two matrices A and B.

Solution:

There is no such C.

11.25 Checking the conputed solution of linear equations. One of your colleagues says that whenever you compute the solution x of a square set of n equations $Ax = b$ (say, using QR factorization), you should computer the number $\|Ax - b\|$ and check that it is small. (It is not exactly zero due to the small rounding errors made in floating point cumputations.) Another colleague says that this would be nice to do, but the additional cost of computing $\|Ax - b\|$ is too high. Brief comment on your colleagues' advice. Who is right?

Solution:

The complexity to compute $\|Ax - b\|$ is about $5n^2$, compare to the complextity of solving the $Ax - b = 0$ which is $2n^3$, it is small, *i.e.*, we can check the result and it will not cost too many computing caompared to the solution of QR factorization.

11.26 Sensitivity of solution of linear equations. Let A be an invertible $n \times n$ matrix, and b and x be n-vectors satisfying $Ax = b$. Suppose we perturb the jth entry of b by $\epsilon \neq 0$ (which is a traditional symbol for a small quantity), so b becomes $\tilde{b} = b + \epsilon e_j$. Let $\tilde{x}$ be the n-vector that satisfies $A\tilde{x} = \tilde{b}$, *i.e.*, the solution of the linear equations using the perturbed right-hand side. We are interested in $\|x - \tilde{x}\|$, which is how much the solution changes due to the change in the right-hand side. The ratio $\|x - \tilde{x}\| / |\epsilon|$ gives the sensitivity of the solution to changes (perturbations) of the jth entry of b.

(a) Show that $\|x - \tilde{x}\|$ does not depend on b; it only depends on the matrix A, ϵ, and j.

(b) How would you find the index j that maximizes the value $\|x - \tilde{x}\|$? By part (a), your answer should be in terms of A (or quantities derived from A) and ϵ only.

Remark. If a small change in the right-hand side vector b can lead to a large change in the solution, we say that the linear equaitons $Ax = b$ are

poorly conditioned or *ill-conditioned*. As a practical matter it means that unless you are very confident in what the entries of b are, the solution $A^{-1}b$ may not be useful in practice.

Solution:

(a) We have $x = A^{-1}b$ and $\tilde{x} = A^{-1}\tilde{b} = A^{-1}(b + \epsilon e_j)$, thus

$$x - \tilde{x} = A^{-1}b - A^{-1}(b + \epsilon e_j) = -A^{-1}\epsilon e_j.$$

we have

$$\|x - \tilde{x}\| = \left\|\epsilon A^{-1} e_j\right\|.$$

(b) Consider expression $A^{-1}e_j$ means the jth column of A^{-1}, denote a'_j as the jth column of A^{-1}, we can form the

$$\|x - \tilde{x}\| = |\epsilon| \left\|a'_j\right\|.$$

In order to maximize the value of $\|x - \tilde{x}\|$, we shoul find out j with which has the maximum value of $\left\|a'_j\right\|$. In words: If the jth column of A^{-1} has the maximum norm, then j was what we want.

11.27 Timing test. Generate a random $n \times n$ matrix A and an n-vector b, for $n = 500$, $n = 1000$, and $n = 2000$. For each of these, compute the solution $x = A^{-1}b$ (for example using the backslash operator, if the software you are using supports it), and verify that $Ax - b$ is (very) small. Report the time it takes to solve each of your processor in Gflop/s, based on the $2n^3$ complexity of solving equations using the QR factorization.

Solution:

Times with solving $x = A^{-1}b$ on a 1 Gflops/s computer

n	$2n^3$ Gflops	time s
500	0.25	0.25
1000	2	2
2000	16	16

11.28 Solving multiple linear equations efficiently. Suppose the $n \times n$ matrix A is invertible. We can solve the system of linear equations $Ax = b$

in around $2n^3$ flops using algorithm 11.2. Once we have done that (specifically, computed the QR factorization of A), we can solve an additional set of linear equations with same matrix but different right-hand side, $Ay = c$, in around $3n^2$ additional flops. Assuming we have solved both of these sets of equations, suppose we want to solve $Az = d$, where $d = \alpha b + \beta c$ is a linear combination of b and c. (We are given the coefficients α and β.) Suggest a method for doing this that is even faster thean re-using the QR factorization of A. Your method should have a complxity that is *linear* in n. Give rough estimates for the time needed to solve $Ax = b$, $Ay = c$ and $Az = d$ (using your method) for $n = 3000$ on a computer capable of carrying out 1 Gflop/s.

Solution:

If x and y are the solution of equations $Ax = b$ and $Ay = c$, respectively, we first show that $z = \alpha x + \alpha y$ is the solution $Az = d$ with $d = \alpha b + \beta c$:

$$Az = A(\alpha x + \alpha y) = \alpha Ax + \beta Ay = \alpha b + \beta c = d.$$

Which means we can use $2n + 1$ flops to solve $Az = d$ if we know about x and y.

Totally, we need around $2n^3$ flops to solve $Ax = b$, and $3n^2$ to solve $Ay = c$ with the calculated QR factorization of A. And $2n + 1$ flops to solve $Az = d$ with the first two solutions. Totally $2n^3 + 3n^2 + 2n$, with $n = 3000$,

$$2n^3 + 3n^2 + 2n + 1 \approx 2 \text{ Gflops.}$$

Need 2 second.

12 Least squares

12.1 Approximationg a vector as a multiple of another one. In the special case $n = 1$, the general least quares problem (12.1) reduces to finding a scalar x that minimizes $\|ax - b\|^2$, where a and b are m-vectors. (We write the matrix A here in lower case, since it is an m-vector.) Assuming a and b are nonzero, show that $\|a\hat{x} - b\|^2 = \|b\|^2 (\sin\theta)^2$, where $\theta = \angle(a, b)$. This shows that optimal relative error in approximating one vector by a multiple of another one depends on their angle.

Solution:

Here we have

$$\hat{x} = a^{\dagger} b = (a^T a)^{-1} a^T b = (\|a\|^2)^{-1} a^T b,$$

since a is a nonzero, we have $\|a\| \neq 0$, implies

$$\hat{x} = \frac{a^T b}{\|a\|^2},$$

let θ be the angle between a and b, *i.e.*

$$\|a\| \, \|b\| \cos\theta = a^T b,$$

we have

$$\hat{x} = \frac{\|b\|}{\|a\|} \cos\theta.$$

Substitute this to $\|a\hat{x} - b\|^2$, we have

$$\begin{aligned} \|a\hat{x} - b\|^2 &= \|a\|^2 \hat{x}^2 - 2\hat{x} a^T b + \|b\|^2 \\ &= \|b\|^2 \cos^2\theta - 2\|b\|^2 \cos^2\theta + \|b\|^2 \\ &= \|b\|^2 (1 - \cos^2\theta) \\ &= \|b\|^2 \sin^2\theta. \end{aligned}$$

12.2 Least squares with orthonormal columns. Suppose the $m \times n$ matrix Q has orthonormal columns and b is an m-vector. Show that $\hat{x} = Q^T b$ is the vector that minimizes $\|Qx - b\|^2$. What is the complexity of computing $\hat{x}$, given Q and b, and how does it compare to the complexity of a general least squares problem with an $m \times n$ coefficient matrix?

Solution:

As Q has orthonormal columns, we have

$$Q^T Q = I.$$

Therefore $Q^{\dagger} = (Q^T Q)^{-1} Q^T = Q^T$, we have

$$\hat{x} = Q^{\dagger} b = Q^T b.$$

12.3 Least angle property of least squares. Suppose the $m \times n$ matrix A has linear independent columns, and b is an m-vector. Let $\hat{x} = A^{\dagger} b$ denote the

least squares approximate solution of $Ax = b$.

(a) Show that for any n-vector x, $(Ax)^T b = (Ax)^T (A\hat{x})$, *i.e.*, the inner product of Ax and b is the same as the inner product of Ax and $A\hat{x}$. *Hint.* Use $(Ax)^T b = x^T (A^T b)$ and $(A^T A)\hat{x} = A^T b$.

(b) Show that when $A\hat{x}$ and b are both nonzero, we have

$$\frac{(A\hat{x})^T b}{\|A\hat{x}\|\,\|b\|} = \frac{\|A\hat{x}\|}{\|b\|}.$$

The left-hand side is the cosine of the angle between $A\hat{x}$ and b. *Hint.* Apply part (a) with $x = \hat{x}$.

(c) *Least angle property of least squares.* The choice $x = \hat{x}$ minimizes the distance between Ax and b. Show that $x = \hat{x}$ also minimizes the angle between Ax and b. (You can assume that Ax and b are nonzero.) *Remark.* For any positive scalar α, $x = \alpha\hat{x}$ also minimizes the angle between Ax and b.

Solution:

(a) With normal equations $A^T(A\hat{x} - b) = 0$, we have $A^T b = A^T A\hat{x}$, therefores

$$(Ax)^T b = x^T A^T b = x^T A^T A\hat{x} = (Ax)^T (A\hat{x}).$$

(b) Let $x = \hat{x}$ for the equations in part (a), we have

$$(A\hat{x})^T b = \|A\hat{x}\|^2,$$

if $A\hat{x}$ and b are nonzero, *i.e.*, $\|A\hat{x}\| \neq 0$ and $\|b\| \neq 0$. both side divided $\|A\hat{x}\|\,\|b\|$, get

$$\frac{(A\hat{x})^T b}{\|A\hat{x}\|\,\|b\|} = \frac{\|A\hat{x}\|}{\|b\|}.$$

(c) In order to prove that $\hat{x}$ minimizing the angle between Ax and b, it's same to prove that for all x, we have

$$\angle(Ax, b) \geq \angle(A\hat{x}, b),$$

as the angle between two vector a and b can be expressed as

$$\theta = \arccos \frac{a^T b}{\|a\|\,b},$$

it is same to improve that

$$\frac{(Ax)^T b}{\|Ax\|\,\|b\|} \le \frac{(A\hat{x})^T b}{\|A\hat{x}\|\,\|b\|},$$

from part (a) and part (b), it is same to prove

$$\frac{(Ax)^T (A\hat{x})}{\|Ax\|\,\|b\|} \le \frac{\|A\hat{x}\|}{\|b\|},$$

which is same to prove

$$(Ax)^T (A\hat{x}) \le \|Ax\|\,\|A\hat{x}\|,$$

with Cauchy-Schwarz inequality, we have

$$(Ax)^T (A\hat{x}) \le |(Ax)^T (A\hat{x})| \le \|Ax\|\,\|A\hat{x}\|,$$

which means the inequality we want to prove hold.

12.4 Weighted least squares. In the least squares, the objective (to be minimized) is

$$\|Ax - b\|^2 = \sum_{i=1}^{m} (\tilde{a}_i^T x - b_i)^2,$$

where $\tilde{a}_i^T$ are the rows of A, and the n-vector x is to chosen. In the *weighted least squares problem*, we minimize the objective

$$\sum_{i=1}^{m} w_i (\tilde{a}_i^T x - b_i)^2,$$

where w_i are given positive weights. The weights allows us to assign different weights to the different components of the residual vector. (The objective of the weighted least squares problem is the square of the weighted norm, $\|Ax - b\|_w^2$, as defined in exercise 3.28.)

(a) Show that the weighted least squares objective can be expressed as $\|D(Ax - b)\|^2$ for an appropriate diagonal matrix D. This allows us to solve the the weighted least squares problem as a standard least squares problem, by minimizing $\|Bx - d\|^2$ where $B = DA$ and $d = Db$.

(b) Show that when A has linear independent columns, so does the matrix B.

(c) The least squares approximate solution is given by $\hat{x} = (A^TA)^{-1}A^Tb$. Give a similar formula for the solution of the weighted least squares problem. You might want to use the matrix $W = \mathbf{diag}(w)$ in your formula.

Solution:

(a) Let D be the diagonal matrix with entries $\sqrt{w_1},\ldots,\sqrt{w_m}$, *i.e.*

$$D = \mathbf{diag}(\sqrt{w}),$$

we have

$$D(Ax+b) = \begin{bmatrix} \sqrt{w_1} & & \\ & \ddots & \\ & & \sqrt{w_m} \end{bmatrix} \begin{bmatrix} \tilde{a}_1^Tx + b_1 \\ \vdots \\ \tilde{a}_m^Tx + b_m \end{bmatrix} = \begin{bmatrix} \sqrt{w_1}(\tilde{a}_1^Tx + b_1) \\ \vdots \\ \sqrt{w_m}(\tilde{a}_m^Tx + b_m) \end{bmatrix}$$

implies

$$\|D(Ax+b)\|^2 = \sum_{i=1}^{m} w_i(\tilde{a}_i^Tx + b_i)^2.$$

(b) Suppose $Bx = 0$, *i.e.*, $DAx = 0$, expand DAx as

$$DAx = \begin{bmatrix} \sqrt{w_1}\tilde{a}_1^Tx_1 \\ \vdots \\ \sqrt{w_m}\tilde{a}_m^Tx_m \end{bmatrix},$$

$DAx = 0$ is same that $\sqrt{w_i}\tilde{a}_i^Tx_i = 0$ for $i = 1,\ldots,m$, implies that $\tilde{a}_i^Tx_i = 0$. If A has linearly independent columns we have $x = 0$, therefore B has linearly independent columns.

(c) Let W denote the diagonal matrix obtained from w, *i.e.*, $W = \mathbf{diag}(w)$, we have

$$W = D^TD,$$

therefore the approximate solution of weighted least squares problem is

$$\hat{x} = (B^TB)^{-1}B^Td = (A^TD^TDA)^{-1}A^TD^TDb = (A^TWA)^{-1}A^TWb.$$

12.5 Approximate right inverse. Suppose the tall $m\times n$ matrix A has linearly independent columns. It does not have a right inverse, *i.e.*, there is no $n\times m$ matrix X for which $AX = I$. So instead we seek the $n\times m$ matrix X for which the residual matrix $R = AX - I$ has the smallest possible matrix norm. We call this matrix the *least squares approximate right inverse*

of A. Show that the least squares right inverse of A is given by $X = A^\dagger$. *Hint.* This is a matrix least squares problem; see page 233.

Solution:

We have $\|AX - I\|^2 = \|Ax_1 - e_1\|^2 + \cdots + \|Ax_m - e_m\|^2$, here x_i is the ith column if X. Minimizing $\|AX - I\|$ is same to minimizing every $\|Ax_i - e_i\|^2$, we have $\hat{x}_i = A^\dagger e_i$, thus

$$X = \begin{bmatrix} A^\dagger \hat{x}_1 & \cdots & A^\dagger \hat{x}_m \end{bmatrix} = A^\dagger I = A^\dagger.$$

12.6 Least squares equalizer design. (see exercise 7.15.) You are given a channel impulse response, the n-vector c. Your job is to find an equalizer impulse response, the n-vector h, that minimizes $\|h * c - e_1\|^2$. You can assume that $c_1 \neq 0$. *Remark.* h is called an equalizer since it approximately inverts, or undoes, convolution by c.

Explain how to find h. Apply your method to find the equalizer h for the channel $c = (1.0, 0.7, -0.3, -0.1, 0.05)$. Plot c, h, and $h * c$.

Solution:

Denote

$$C = \begin{bmatrix} c_1 & 0 & 0 & 0 & 0 \\ c_2 & c_1 & 0 & 0 & 0 \\ c_3 & c_2 & c_1 & 0 & 0 \\ c_4 & c_3 & c_2 & c_1 & 0 \\ c_5 & c_4 & c_3 & c_2 & c_1 \\ 0 & c_5 & c_4 & c_3 & c_2 \\ 0 & 0 & c_5 & c_4 & c_3 \\ 0 & 0 & 0 & c_5 & c_4 \\ 0 & 0 & 0 & 0 & c_5 \end{bmatrix} = \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 \\ 0.7 & 1.0 & 0 & 0 & 0 \\ -0.3 & 0.7 & 1.0 & 0 & 0 \\ -0.1 & -0.3 & 0.7 & 1.0 & 0 \\ 0.05 & -0.1 & -0.3 & 0.7 & 1.0 \\ 0 & 0.05 & -0.1 & -0.3 & 0.7 \\ 0 & 0 & 0.05 & -0.1 & -0.3 \\ 0 & 0 & 0 & 0.05 & -0.1 \\ 0 & 0 & 0 & 0 & 0.05 \end{bmatrix},$$

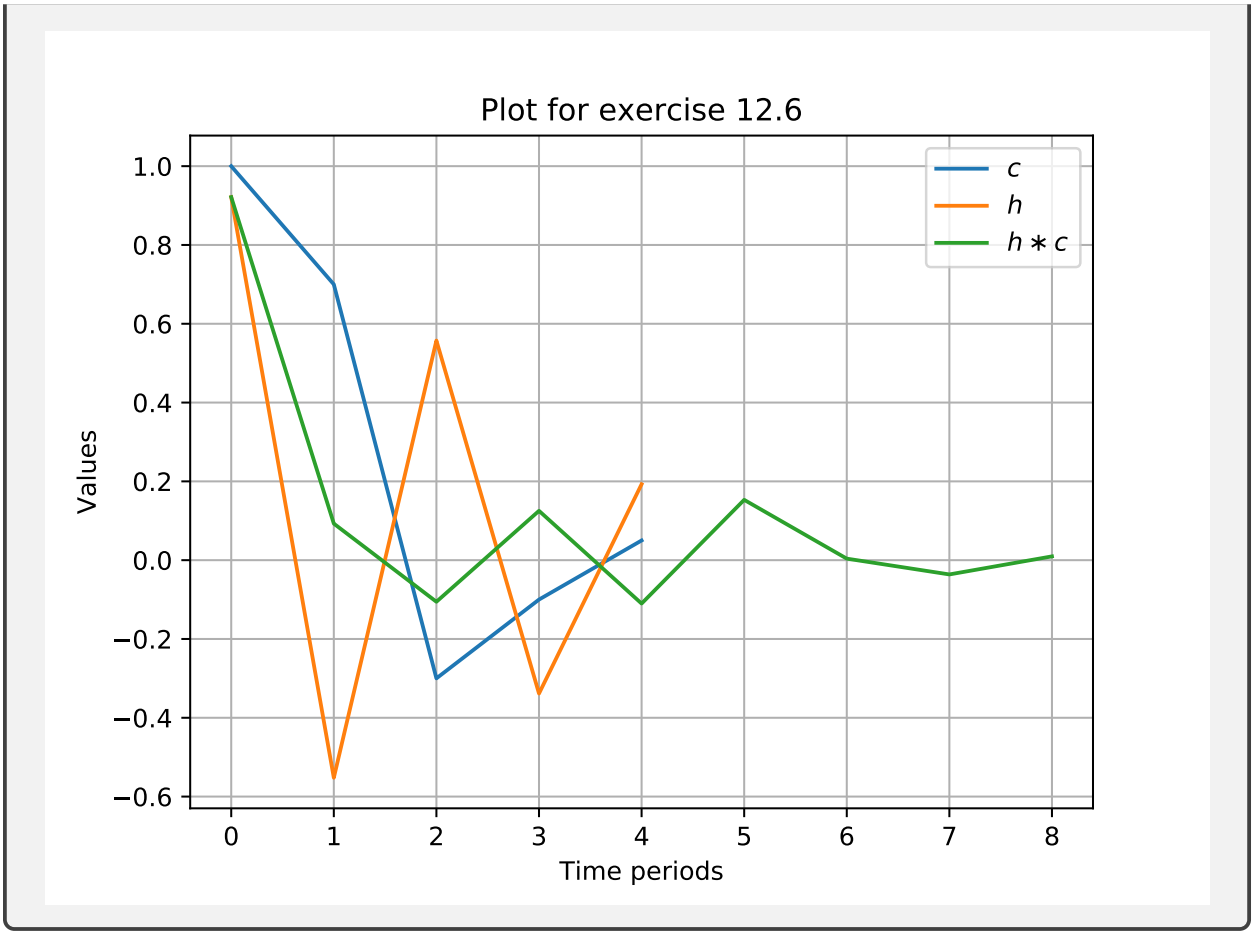
we can form the convolution $h * c$ as Ch, our goals is to find h satisfies $Ch \approx e_1$, with

$$C^\dagger = \begin{bmatrix} 0.921 & 0.093 & -0.106 & 0.125 & -0.110 & 0.153 & 0.004 & -0.036 & 0.010 \\ -0.552 & 0.825 & 0.199 & -0.236 & 0.208 & -0.288 & -0.004 & 0.066 & -0.017 \\ 0.557 & -0.425 & 0.687 & 0.371 & -0.331 & 0.452 & -0.014 & -0.097 & 0.028 \\ -0.338 & 0.407 & -0.263 & 0.482 & 0.460 & -0.636 & 0.004 & 0.116 & -0.028 \\ 0.193 & -0.204 & 0.263 & -0.080 & 0.411 & 0.738 & -0.193 & -0.120 & 0.046 \end{bmatrix},$$

we have

$$h = C^\dagger e_1 = \begin{bmatrix} 0.921 & -0.552 & 0.557 & -0.338 & 0.193 \end{bmatrix}.$$

The plot of c, h and $h * c$ is :



12.7 Network tomography. A network consists of n links, labeled $1,\ldots,n$. A *path* through the network is a subset of the links. (The order of the links on a path does not matter here.) Each link has a (positive) *delay*, which is the time it takes to traverse it. We let d denote the n-vector that gives the link delays. The total travel time of a path if the sum of the delays of the links on the path. Our goal is to estimate the link delays (*i.e.*, the vector d), from a large number of (noisy) measurements of the travel times along different paths. This data is given to you as an $N \times n$ matrix P, where

$$P_{ij} = \begin{cases} 1 & \text{link } j \text{ is on path } i \\ 0 & \text{otherwise,} \end{cases}$$

and the an N-vector t whose entries are the (noisy) travel times along the N paths. You can assume that $N > n$. You will choose your estimate $\hat{d}$ by minimizing the RMS deviation between the measured travel times (t) and

the travel times predicted by the sum of the link delays. Explain how to do this, and give a matrix expression for $\hat{d}$. If your expression requires assumptions about the data P and t, state them explicitly.

Remark. This problem arises in several contexts. The network could be a computer network, and a path gives the sequence of communication links data packets traverse. The network could be a transportation system, with the links representing road segments.

Solution:

Suppose the row of P is p_i^T for $i = 1,\ldots,N$, $(p_i^T)_j$ means the whether link j in on path i, therefore the delay of path i is

$$t_i = p_i^T d,$$

Combinate for all $i = 1,\ldots,N$, we have

$$Pd = t.$$

Our goal is to find d, which minimize $\|Pd - t\|^2$, thus

$$\hat{d} = p^{\dagger} t.$$

The condition is that P must has linearly independent columns.

12.8 Least squares and QR factorization. Suppose A is an $n \times n$ matrix with linearly independent columns and QR factorization $A = QR$, and b is an m-vector. The vector $A\hat{x}$ is the linear combination of the columns of A that is closest to the vector b, *i.e.*, it is the prjection of b onto the set of linear combinations of the columns of A.

(a) Show that $A\hat{x} = QQ^T b$. (The matrix QQ^T is called *projection matrix*.)

(b) Show that $\|A\hat{x} - b\|^2 = \|b\|^2 - \left\|Q^T b\right\|^2$. (This is the square of the distance between b and the closest linear combination of the columns of A.)

Solution:

(a) We have $\hat{x} = R^{-1}Q^T b$, therefore

$$A\hat{x} = QRR^{-1}Q^T b = QQ^T b.$$

(b) Substitute $A\hat{x}$ with the result in part (a) we have

$$\begin{aligned}\|A\hat{x}-b\|^2 &= \left\|QQ^Tb-b\right\|^2\\ &= \|b\|^2+(QQ^Tb)^T(QQ^Tb)-2b^TQQ^Tb\\ &= \|b\|^2+b^TQQ^TQQ^Tb-2b^TQQ^Tb\\ &= \|b\|^2+b^TQQ^Tb-2b^TQQ^Tb\\ &= \|b\|^2-b^TQQ^Tb\\ &= \|b\|^2-\left\|Q^Tb\right\|^2,\end{aligned}$$

where we use the fact that $Q^TQ=I$.

12.9 Invertibility of matrix in sparse least squares formulation. Show that the $(m+n)\times(m+n)$ coefficient matrix appearing in equation (12.11) is invertible if and only if the columns of A are linearly independent.

Solution:

The coefficient matrix in equation (12.11) is

$$\begin{bmatrix}0 & A^T\\ A & I\end{bmatrix}.$$

First, we assume that the coeffeciet matrix is invertible but A has linearly dependent columns, means there is nonzero vector $\bar{x}$ with

$$A\bar{x}=0,$$

implies that

$$\begin{bmatrix}0 & A^T\\ A & I\end{bmatrix}\begin{bmatrix}\bar{x}\\ 0\end{bmatrix}=0,$$

i.e., the coefficient matrix has linearly dependent columns, *i.e.*, it is not invertible, which is constradict.

Second, we assume that A has linearly independent columns, but the coefficient is not invertible, means there is nonzero vector $(\bar{x},\bar{y})$ with

$$\begin{bmatrix}0 & A^T\\ A & I\end{bmatrix}\begin{bmatrix}\bar{x}\\ \bar{y}\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix},$$

from the below block equation $A\bar{x}+\bar{y}=0$, we have

$$\bar{y}=-A\bar{x},$$

substitute it to the upper block equation $A^T\bar{y} = 0$, we have

$$A^T A\bar{x} = 0,$$

as A has linearly independent columns, *i.e.*, $A^T A$ is invertible, implies

$$\bar{x} = 0,$$

as $\bar{y} = -A\bar{x}$, we have

$$\bar{y} = 0,$$

which is a constradic (our assumption that either $\bar{x}$ or $\bar{y}$ is nonzero.). Thus, the coefficient matrix is invertible if and only if the columns of A are linearly independent.

12.10 Numerical check of the least squares approximate solution. Generate a random 30×10 matrix A and a random 30-vector b. Computer the least squares approximate solution $\hat{x} = A^\dagger b$ and the associated rediual norm squared $\|A\hat{x} - b\|^2$. (There may be several ways to do this, depending on the software package you use.) Generate three different random 10-vectors d_1, d_2, d_3, and verify that $\|A(\hat{x} + d_i) - b\|^2 > \|A\hat{x} - b\|^2$ holds. (This show that $x = \hat{x}$ has a smaller associated residual than the choices $x = \hat{x} + d_i$, $i = 1, 2, 3$.)

Solution:

I use the code in listing 4 to implement this exercise.

Listing 4: Code for exercise 12.10

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exerises 12.10
# Author: Utoppia
# Date  : 29 May 2020

import numpy as np
from numpy.linalg import pinv, norm

# generate (30, 10) A
A = np.random.random((30,10))
# generate (30, ) vector b
```

```
b = np.random.random(30)

x = pinv(A) @ b

residual = norm(A @ x - b)

d = [np.random.random(10) for _ in range(3)]

for i in range(3):
    print(norm(A @ (x + d[i]) - b) > residual)
```

12.11 Complexity of matrix least squares problem. Explain how to compute the matrix least squares approximate solution of $AX = B$, given by $\hat{X} = A^{\dagger}B$ (see (12.12)), in no more than $2mn^2 + 3mnk$ flops. (In constrast, solving k vector least squares problems to obtain the columns of $\hat{X}$, in a naïve way, requires $2mn^2k$ flops.)

Solution:

Consider the method below

- *QR Factorization.* Compute the QR factorization $A = QR$;
- Compute $Q^T b_i$, $i = 1, \ldots, k$ is the columns of B;
- *Back Substitution.* Solve the triangular equations $Rx_i = Q^T b_i$, $i = 1, \ldots, k$.

The fisrt step need $2mn^2$ flops; the second step need $2mn \times k = 2mnk$, the last step need $n^2 \times k = kn^2$. Since the problem can be solve, A must have linear independent columns, *i.e.*, A must be tall, implies

$$m > n,$$

therefore the total complexity is

$$2mn^2 + 2mnk + kn^2 < 2mn^2 + 3mnk.$$

12.12 Least squares placement. The 2-vectors $p_1, \ldots, p_N$ represent the locations or positions of N objects, for example, factories, warehouses, and stores. The last K of these locations are fixed and given; the goal in a *placement problem* to choose the locations of the first $N - K$ objects. Our

choice of the locations is guided by an undirected graph; an edge between two objects means we would like them to be close to each other. In *least squares placement*, we choose the locations $p_1, \ldots, p_{N-K}$ so as to minimize the sum if the squares of the distances between objects connected by an edge,

$$\left\|p_{i_1} - p_{j_1}\right\|^2 + \cdots + \left\|p_{i_L} - p_{j_L}\right\|^2,$$

where the L edges of the graph are given by $(i_1, j_1), \ldots, (i_L, j_L)$.

(a) Let $\mathcal{D}$ be the Dirichlet energy of the graph, as defined on page 135. Show that the sum of the squared distances between the N objects can be expressed as $\mathcal{D}(u) + \mathcal{D}(v)$, where $u = ((p_1)_1, \ldots, (p_N)_1)$ and $v = ((p_1)_2, \ldots, (p_N)_2)$ are N-vectors containing the first and second coordinates of the objects, respectively.

(b) Express the least squares placement problem as a least squares problem, with variable $x = (u_{1:(N-K)}, v_{1:(N-K)})$. In other words, express the objective above (the sum of squares of the distances across edges) as $\|Ax - b\|^2$, for an appropriate $m \times n$ matrix A and m-vector b. You will find that $m = 2L$. *Hint.* Recall that $\mathcal{D}(y) = \left\|B^T y\right\|^2$, where B is the incidence matrix of the graph.

(c) Solve the least sqaures placement problem for the specific problem with $N = 10$, $K = 4$, $L = 13$, fixed locations

$$p_7 = (0,0), \quad p_8 = (0,1), \quad p_9 = (1,1), \quad p_{10} = (1,0),$$

and edges

$$(1,3), \quad (1,4), \quad (1,7), \quad (2,3), \quad (2,5), \quad (2,8), \quad (2,9),$$

$$(3,4), \quad (3,5), \quad (4,6), \quad (5,6), \quad (6,9), \quad (6,10).$$

Plot the locations, showing the graph edges as lines connecting the locations.

Solution:

(a) The squares of distance between edge (u, v) can be formed as

$$\|p_u - p_v\|^2 = ((p_u)_1 - (p_v)_1)^2 + ((p_u)_2 - (p_v)_2)^2$$

Therefore the sum of the squares of the distances

$$\begin{aligned}\sum_{i,j\in\text{Edge}} \|p_i - p_j\|^2 &= \sum_{i,j\in\text{Edge}} ((p_i)_1 - (p_j)_1)^2 + ((p_i)_2 - (p_j)_2)^2 \\ &= \sum_{i,j\in\text{Edge}} ((p_i)_1 - (p_j)_1)^2 + \sum_{i,j\in\text{Edge}} ((p_i)_2 - (p_j)_2)^2 \\ &= \mathcal{D}(u) + \mathcal{D}(v)\end{aligned}$$

(b) Suppose B is $n \times L$ incidence matrix of graph. we have

$$\mathcal{D}(u) = \|B^T u\|^2, \qquad \mathcal{D}(v) = \|B^T v\|^2.$$

For convenient, we denote the $C = B_{1:(N-K),1:L}$ and $D = B_{N-k+1:N,1:L}$, in words: C is $(N-K)\times L$ matrix obtained from the first $N-K$ rows of B, D is $K \times L$ matrix obtained from the last K rows of B. *i.e.*,

$$B = \begin{bmatrix} C \\ D \end{bmatrix}, \qquad B^T = \begin{bmatrix} C^T & D^T \end{bmatrix}.$$

Denote the first $N-K$ entries of u and v as $\hat{u}$ and $\hat{v}$, respectively, and the last K entries of u and v as $\tilde{u}$ and $\tilde{v}$, respectively. We can form the u and v as block vectors:

$$u = \begin{bmatrix} \hat{u} \\ \tilde{u} \end{bmatrix}, \qquad v = \begin{bmatrix} \hat{v} \\ \tilde{v} \end{bmatrix}$$

Then we have

$$B^T u = \begin{bmatrix} C^T & D^T \end{bmatrix} \begin{bmatrix} \hat{u} \\ \tilde{u} \end{bmatrix} = C^T \hat{u} + D^T \tilde{u}$$

and

$$B^T v = \begin{bmatrix} C^T & D^T \end{bmatrix} \begin{bmatrix} \hat{v} \\ \tilde{v} \end{bmatrix} = C^T \hat{v} + D^T \tilde{v}$$

and we have

$$x = \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix}.$$

According to part (a), the sum of the squares of the distances can be expressed as:

$$\mathcal{D}(u) + \mathcal{D}(v) = \|B^T u\|^2 + \|B^T v\|^2$$

which is same with the norm of $2L$-vector:

$$\begin{bmatrix} B^T u \\ B^T v \end{bmatrix}.$$

And

$$\begin{bmatrix} B^T u \\ B^T v \end{bmatrix} = \begin{bmatrix} C^T \hat{u} + D^T \tilde{u} \\ C^T \hat{v} + D^T \tilde{v} \end{bmatrix} = \begin{bmatrix} C^T \hat{u} \\ C^T \hat{v} \end{bmatrix} + \begin{bmatrix} D^T \tilde{u} \\ D^T \tilde{v} \end{bmatrix} = \begin{bmatrix} C^T & 0 \\ 0 & C^T \end{bmatrix} \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} + \begin{bmatrix} D^T \tilde{u} \\ D^T \tilde{v} \end{bmatrix} = \begin{bmatrix} C^T & 0 \\ 0 & C^T \end{bmatrix} x + \begin{bmatrix} D^T \tilde{u} \\ D^T \tilde{v} \end{bmatrix}$$

let

$$A = \begin{bmatrix} C^T & 0 \\ 0 & C^T \end{bmatrix}, \qquad b = -\begin{bmatrix} D^T \tilde{u} \\ D^T \tilde{v} \end{bmatrix}$$

we have

$$\mathcal{D}(u) + \mathcal{D}(v) = \|Ax - b\|^2 .$$

And because $x = A^\dagger b$ minimizes the $\|Ax - b\|^2$, therefores $x = A^\dagger b$ is the solution of original question.

(c) For this specific example, we have $\tilde{u} = (0,0,1,1)$ and $\tilde{v} = (0,1,1,0)$, and

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

We can get C and D from B and form A and b as following:

$$A = \begin{bmatrix}
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}, \qquad b = \begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0
\end{bmatrix}$$

we can get

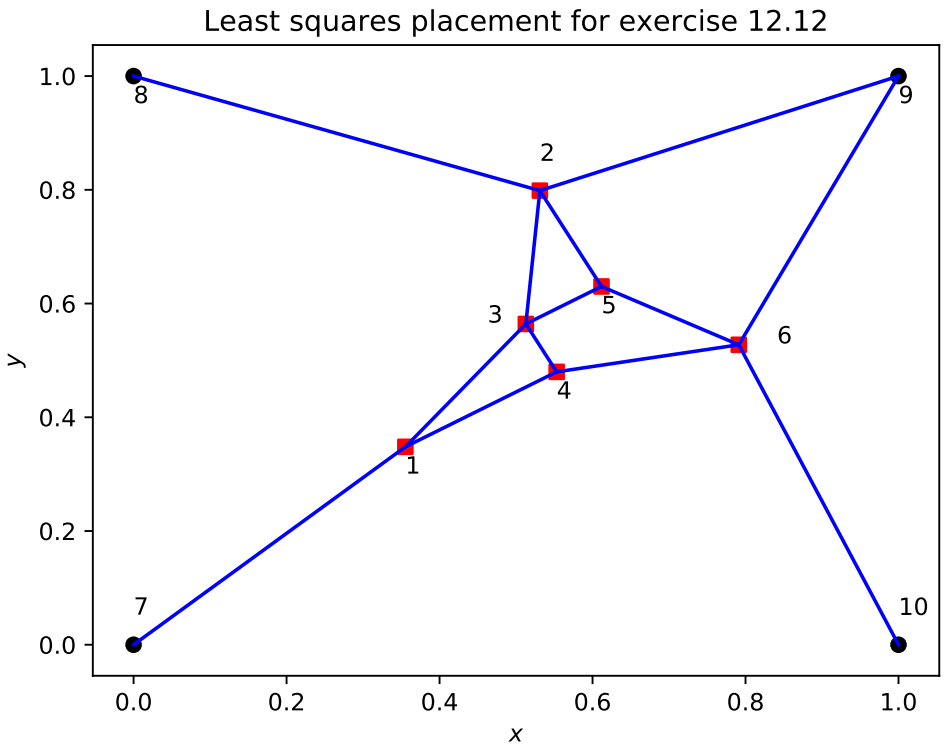
$$x = (0.355, 0.531, 0.512, 0.553, 0.611, 0.791, 0.347, 0.798, 0.564, 0.479, 0.630, 0.527),$$

or

$$p_1 = (0.355, 0.347), \quad p_2 = (0.531, 0.798), \quad p_3 = (0.512, 0.564),$$

$$p_4 = (0.553, 0.479), \quad p_5 = (0.611, 0.630), \quad p_6 = (0.791, 0.527).$$

And the plot is



12.13 Iterative method for least squares problem. Suppose that A has linearly independent columns, so $\hat{x} = A^{\dagger}b$ minimize $\|Ax - b\|^2$. In this exercise we explore an interative method, due to the mathematician Lewis Richardson, that can be used to compute $\hat{x}$. We define $x^{(1)} = 0$ and for $k = 1, 2, \ldots,$

$$x^{(k+1)} = x^{(k)} - \mu A^T(Ax^{(k)} - b),$$

where μ is a positive parameter, and the superscripts denote the ireration number. This defines a sequence of vectors that converge to $\hat{x}$ provided μ is not too large; the choice $\mu = 1/\|A\|^2$, for example, always works. The iteration is terminated when $A^T(Ax^{(k)} - b)$ is small enough, which means the least squares optimality conditions are almost satisfied. To implement the method we only need to multiply vectors A and by A^T. If we have efficient methods for carrying out these two matrix-vector multiplications, this interative method can be faster than algorithm 12.1 (although it does not give the exact solution). Interative methods are often used for very large scale least squares problems.

(a) Show that if $x^{(k+1)} = x^{(k)}$, we have $x^{(k)} = \hat{x}$.

(b) Express the vector sequence $x^{(k)}$ as a linear dynamical system with constant dynamics matrix and offset, *i.e.*, in the form $x^{(k+1)} = Fx^{(k)} + g$.

(c) Generate a random 20×10 matrix A and 20-vector b, and compute $\hat{x} = A^{\dagger}b$. Run the Richardson algorithm with $\mu = 1/\|A\|^2$ for 500 iterators, and plot $\|x^k - \hat{x}\|$ to verify that $x^{(k)}$ appears to be converging to $\hat{x}$.

Solution:

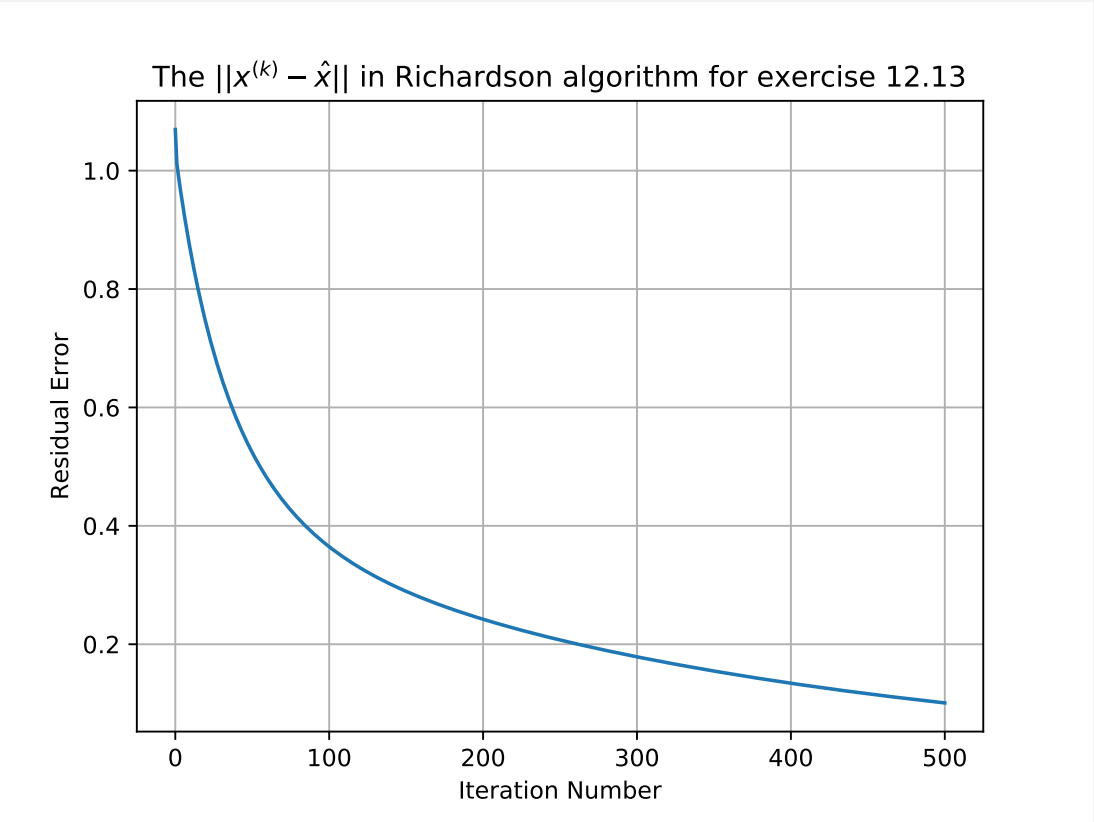
(a) If $x^{(k+1)} = x^{(k)}$, which means $\mu A^T(Ax^{(k)} - b) = 0$, which means $x^{(k)}$ satisfies the normal equations, Thus

$$x^{(k)} = \hat{x}.$$

(b) $x^{(k+1)} = (I - \mu A^T A)x^{(k)} + \mu A^T b$, let $F = I - \mu A^T A$ and $g = \mu A^T b$, we have

$$x^{(k+1)} = Fx^{(k)} + g.$$

(c) The plot is



12.14 Recursive least squares. In some applications of least squares the rows of the coefficient matrix A become available (or are added) sequentially, and we wish to solve the resulting family of growing least squares problems. Define the $k \times n$ matrices and k-vectors

$$A^{(K)} = \begin{bmatrix} a_1^T \\ \vdots \\ a_k^T \end{bmatrix} = A_{1:k,1:n}, \qquad b^{(K)} = \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} = b_{1:k},$$

for $k = 1, \ldots, m$. We wish to compute $\hat{x}^{(k)} = A^{(k)\dagger} b^{(k)}$, for $k = n, n+1, \ldots, m$. We will assume that the columns if $A^{(n)}$ are linearly independent, which implies that the columns of $A^{(k)}$ are linearly independent for $k = n, \ldots, m$. We will also assume that m is much large than n. The naïve method for computing $x^{(k)}$ requires $2kn^2$ flops, so the total cost for $k = n, \ldots, m$ is

$$\sum_{k=n}^{m} 2kn^2 = (\sum_{k=n}^{m} k)(2n^2) = (\frac{m^2 - n^2 + m + n}{2})(2n^2) \approx m^2 n^2 \text{flops}.$$

A simple trick allows us to compute $x^{(k)}$ for $k = n, \ldots, m$ much more efficiently, with a cost that grows linearly with m. The trick also requires memory storage order n^2, which does not depend on m. For $k = 1, \ldots, m$, define

$$G^{(k)} = (A^{(k)})^T A^{(k)}, \qquad h^{(k)} = (A^{(k)})^T b^{(k)}.$$

(a) Show that $\hat{x}^{(k)} = (G^{(k)})^{-1} h^{(k)}$ for $k = n, \ldots, m$. *Hint.* See (12.8)

(b) Show that $G^{(k+1)} = G^{(k)} + a_{k+1} a_{k+1}^T$ and $h^{(k+1)} = h^{(k)} + b_{k+1} a_{k+1}$, for $k = n-1, \ldots, m-1$.

(c) *Recursive least squares* is the following algorithm. For $k = n, \ldots, m$, compute $G^{(k+1)}$ and $h^{(k+1)}$ using (b); then compute $\hat{x}^{(k)}$ using (a). Work out the total flop count for this method, keeping only dominant terms. (You can include the cost of computing $G^{(k)}$ and $h^{(k)}$, which should be negligible in the total.) Compare to the flop count for the naïve method.

Remark. A further trick called the *matrix inversion lemma* (which is beyond the scope of this book) can be used to reduce the complexity of recursive least squares to order mn^2.

Solution:

(a) $\hat{x}^{(k)} = A^{(k)\dagger} b = ((A^{(k)})^T A^{(k)})^{-1} (A^{(k)})^T b^{(k)} = (G^{(k)})^{-1} h^{(k)}$.

(b) We have

$$G^{(k+1)} = (A^{(k+1)})^T A^{(k+1)} = \begin{bmatrix} (A^{(k)})^T & a_{k+1} \end{bmatrix} \begin{bmatrix} A^{(k)} \\ a_{k+1}^T \end{bmatrix} = (A^{(k)})^T A^{(k)} + a_{k+1} a_{k+1}^T = G^{(k)} + a_{k+1} a_{k+1}^T$$

and

$$h^{(k+1)} = (A^{(k+1)})^T b^{(k+1)} = \begin{bmatrix} (A^{(k)})^T & a_{k+1} \end{bmatrix} \begin{bmatrix} b^{(k)} \\ b_{k+1} \end{bmatrix} = (A^{(k)})^T b^{(k)} + b_{k+1} a_{k+1} = h^{(k)} + b_{k+1} a_{k+1}$$

(c) computer $G^{(n)}$ need around n^3 flops, and $h^{(n)}$ need around $2n^2$ flops. For every interation, for compute $G^{(k+1)} = G^{(k)} + a_{k+1} a_{k+1}^T$, we need n^2 flops to compute the outer product, and n^2 for matrix addition, total $2n^2$; for compute $h^{(k+1)} = h^{(k)} + b_{k+1} a_{k+1}$, we need n^2 for outer product and n^2 for matrix addition, total $2n^2$. There are $m - n$ interations, implies $(m-n)(2n^2 + 2n^2) = 4mn^2 - 4n^3$. Thus the total folps is $4mn^2 - 4n^3 + n^3 + 2n^2$, with $m >> n$, $4mn^2 - 4n^3 + n^3 + 2n^2 \approx 4mn^2$. The total flop count for recuresive least squares has order mn^2, far less than the naïve method of which is $m^2 n^2$.

12.15 Minimizing a squared norm plus an affine function. A generalization of the least squares problem (12.1) adds an affine funtion to the least squares objective,

$$\text{minimize } \|Ax - b\|^2 + c^T x + d,$$

where the n-vector x is the variable to be chosen, and the (given) data are the $m \times n$ matrix A, the m-vector b, the n-vector c, and the number d. We will use the same assumption we use in least squares: The columns of A are linearly independent. This generalized problem can be solved by reducing it to a standard least sqaures problem, using a trick call *Completing the square.*

Show that the objective of the problem above can be expressed in the form

$$\|Ax + b\|^2 + c^T x + d = \|Ax - b + f\|^2 + g,$$

for some m-vector f and some constant g. It follows that we can solve the generalized least square problem by minimizing $\|Ax - (b - f)\|$, an ordinary

least square problem with solution $\hat{x} = A^\dagger(b - f)$.

Hint. Express the norm squared term on the right-hand side as $\|(Ax - b) + f\|^2$ and expand it. Then argue that the equality above holds provided $2A^T f = c$. One possible choice is $f = (1/2)(A^\dagger)^T c$. (You must justify these statements.)

Solution:

Let expand $\|Ax - b + f\|^2 + g$ as

$$\begin{aligned}\|Ax - b + f\|^2 + g &= \|Ax - b\|^2 + \|f\|^2 + 2f^T(Ax - b) + g \\ &= \|Ax - b\|^2 + 2f^T Ax + \|f\|^2 - 2f^T b + g\end{aligned}$$

If we can find f and g which satisfy

$$\begin{aligned}2f^T A &= c^T \\ \|f\|^2 - 2f^T b + g &= d\end{aligned}$$

and $2f^T A = c^T$ can be formed as $(2f^T A)^T = (c^T)^T$, *i.e.*, $2A^T f = c$, As the columns of A are linearly independent, implies the pseudo-inverse $A^\dagger$ is the left-inverse of A, *i.e.*$A^\dagger A = I$, let

$$f = (1/2)(A^\dagger)^T c,$$

we have

$$2A^T f = A^T(A^\dagger)^T c = (A^\dagger A)^T c = c.$$

which means $f = (1/2)(A^\dagger)^T c$ satisfies $2A^T f = c$, imply it to the seconde equation, we have

$$g = 2f^T b + d - \|f\|^2 = c^T A^\dagger b + d - (1/4)\left\|(A^\dagger)^T c\right\|^2.$$

12.16 Gram method for computing least squares approximate solution. Algorithm 12.1 uses the QR factorization to compute the least squares approximate solution $\hat{x} = A^\dagger b$, where the $m \times n$ matrix A has the linearly independent columns. It has a complexity of $2mn^2$ flops. In this exercise we consider an alternative method: First, form the Gram matrix $G = A^T A$ and the vector $h = A^T b$; and then compute $\hat{x} = G^{-1}h$ (using algorithm 11.2). What is the complexity of this method? Compare it to algorithm 12.1. *Remark.* You might find that the Gram algorithm appears to be a bit faster than the

QR method, but the factor is not the large enough to have any practical significance. The idea is useful in situations where G is partially available and can be computed more efficiently than by multiplying A and its transpose. An example is exercise 13.21.

Solution:

We need n^2m flops to compute $G = A^TA$, and $2nm$ flops for $h = A^Tb$, and about $2n^3$ to compute $\hat{x} = G^{-1}h$, totally

$$mn^2 + 2mn + 2n^3$$

flops, when $m >> n$, its around to mn^2, is a little faster than QR factorization methods.

13 Least squares data fitting

13.1 Error in straight-line fit. Consider the straight-line fit described on page 249, with data given by the N-vectors x^d and y^d. Let $r^d = x = y^d - \hat{y}^d$ denote the residual or prediction error using the straight-line model (13.3). Show that $\mathbf{rms}(r^d) = \mathbf{std}(y^d)\sqrt{1-\rho^2}$, where ρ is the correlation coefficient of x^d and y^d (assumed non-constant). This shows that the RMS error with straight-line fit is a factor $\sqrt{1-\rho^2}$ smaller than the RMS error with a constant fit, which is $\mathbf{std}y^d$. It follows that when x^d and y^d are highly correlated ($\rho \approx 1$) or anti-correlated ($\rho \approx -1$), the straight-line fit is much better than the constant fit. *Hint.* From (13.3) we have

$$\hat{y}^d - y^d = \rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}(x^d - \mathbf{avg}(x^d)\mathbf{1}) - (y^d - \mathbf{avg}(y^d)\mathbf{1}).$$

Expand the norm squared of this expression, and use

$$\rho = \frac{(x^d - \mathbf{avg}(x^d)\mathbf{1})^T(y^d - \mathbf{avg}(y^d)\mathbf{1})}{\left\|x^d - \mathbf{avg}(x^d)\mathbf{1}\right\|\left\|y^d - \mathbf{avg}(y^d)\mathbf{1}\right\|}.$$

Solution:

We know the least squares fit with straight-line model is

$$\hat{f}(x) = \mathbf{avg}(y^d) + \rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}(x - \mathbf{avg}(x^d)).$$

Therefore

$$\begin{aligned}\hat{y}^d - y^d &= \hat{f}(x^d) - y^d \\ &= \mathbf{avg}(y^d)\mathbf{1} + \rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}(x^d - \mathbf{avg}(x^d)\mathbf{1}) - y^d \\ &= \rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}(x^d - \mathbf{avg}(x^d)\mathbf{1}) - (y^d - \mathbf{avg}(y^d)\mathbf{1}) \\ &= \rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}\tilde{x}^d - \tilde{y}^d\end{aligned}$$

here $\tilde{x}^d = x^d - \mathbf{avg}(x^d)\mathbf{1}$ and $\tilde{y}^d = y^d - \mathbf{avg}(y^d)\mathbf{1}$ are the de-meaned vectors of x^d and y^d, respectively. Also we have

$$\begin{aligned}n\mathbf{rms}(r^d)^2 &= \left\|r^d\right\|^2 \\ &= \left\|y^d - \hat{y}^d\right\|^2 \\ &= \left\|\hat{y}^d - y^d\right\|^2 \\ &= (\hat{y}^d - y^d)^T(\hat{y}^d - y^d) \\ &= (\rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}\tilde{x}^d - \tilde{y}^d)^T(\rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}\tilde{x}^d - \tilde{y}^d) \\ &= \rho^2\frac{\mathbf{std}(y^d)^2}{\mathbf{std}(x^d)^2}\left\|\tilde{x}^d\right\|^2 + \left\|\tilde{y}^d\right\|^2 - 2\rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}(\tilde{x}^d)^T(\tilde{y}^d) \\ &= n\rho^2\mathbf{std}(y^d)^2 + n\mathbf{std}(y^d)^2 - 2\rho\mathbf{std}(y^d)^2\frac{(\tilde{x}^d)^T(\tilde{y}^d)}{\mathbf{std}(x^d)\mathbf{std}(y^d)} \\ &= n\rho^2\mathbf{std}(y^d)^2 + n\mathbf{std}(y^d)^2 - 2n\rho^2\mathbf{std}(y^d)^2 \\ &= n\mathbf{std}(y^d)^2(1-\rho^2)\end{aligned}$$

here we use the fact

$$n = \frac{\left\|\tilde{x}^d\right\|^2}{\mathbf{std}(x^d)^2}, \quad \left\|\tilde{y}^d\right\|^2 = n\mathbf{std}(y^d)^2, \quad \frac{(\tilde{x}^d)^T(\tilde{y}^d)}{\mathbf{std}(x^d)\mathbf{std}(y^d)} = n\frac{(\tilde{x}^d)^T(\tilde{y}^d)}{\left\|\tilde{x}^d\right\|\left\|\tilde{y}^d\right\|} = n\rho.$$

Divide n in both hand-sides of the equation above, we have

$$\mathbf{rms}(r^d)^2 = \mathbf{std}(y^d)^2(1-\rho^2),$$

thus,

$$\mathbf{rms}(r^d) = \mathbf{std}(y^d)\sqrt{1-\rho^2}.$$

13.2 Regression to the mean. Consider a data set in which the (scalar) $x^{(i)}$ is the parent's height (average of mother's and father's height), and $y^{(i)}$ is their child's height. Assume that over the data set the parent and child heights have the same mean value μ, and the same standard deviation σ. We will also assume that the correlation coefficient ρ between parent and child height is (strictly) between zero and one. (These assumptions hold, at least approximately, in real data sets that are large enough.) Consider the simple straight-line fit or regression model given by (13.3), which predicts a child's height from the parent's height. Show that this prediction of the child's height lies (strictly) between the parent's height and the mean height μ (unless the parent's height happens to be exactly the mean μ). For example, if the parents are tall, *i.e.*, have height above the mean, we preidict that their child will be shorter, but still tall. This phenomenon, called *regression to the mean*, was first observed by the early statistician Sir Francis Galton (who indeed, studied a data set of parent's and child's heights).

Solution:

We know the least squares fit with straight-line model is

$$\hat{f}(x) = \mathbf{avg}(y^d) + \rho\frac{\mathbf{std}(y^d)}{\mathbf{std}(x^d)}(x - \mathbf{avg}(x^d)).$$

with

$$\mathbf{avg}(x^d) = \mathbf{avg}(y^d) = \mu, \qquad \mathbf{std}(x^d) = \mathbf{std}(y^d) = \sigma,$$

we have

$$\hat{f}(x) = \mu + \rho(x - \mu).$$

therefore

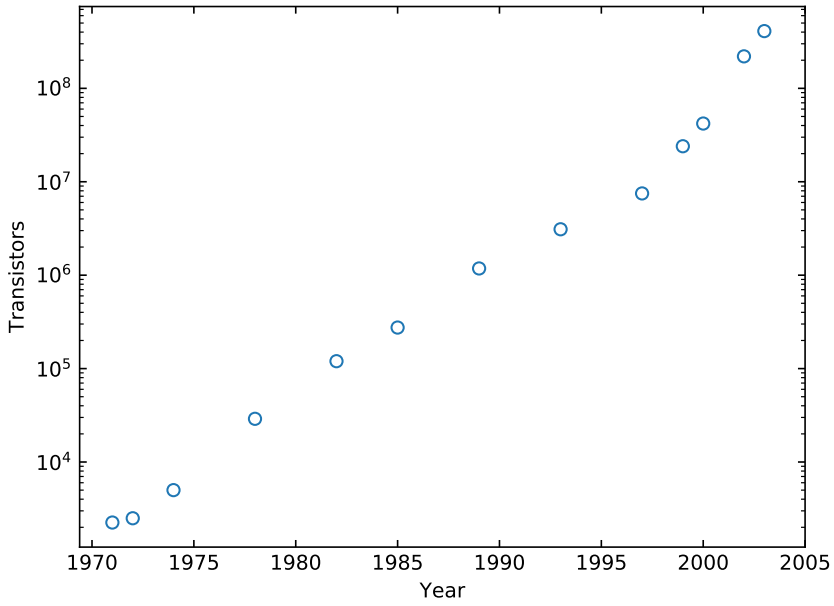
$$\hat{f}(x) - \mu = \rho(x - \mu), \qquad \hat{f}(x) - x = (\rho - 1)(x - \mu).$$

we can imply that

- if $x > \mu$, $\hat{f}(x) > \mu$ and $\hat{f}(x) < x$, *i.e.*, $\mu < \hat{f}(x) < x$.
- if $x < \mu$, $\hat{f}(x) < \mu$ and $\hat{f}(x) > x$, *i.e.*, $x < \hat{f}(x) < \mu$.

13.3 Moore's law. The figure and table below show the number of transistors N in 13 micro-processors, and the year of their introduction.

Year	Transistors
1971	2,250
1972	2,500
1974	5,000
1978	29,000
1982	120,000
1985	275,000
1989	1,180,000
1993	3,100,000
1997	7,500,000
1999	24,000,000
2000	42,000,000
2002	220,000,000
2003	410,000,000



The plot gives the number of transistors on a logarithmic scale. Find the least squares straight-line fit of the data using the model

$$\log_{10} N \approx \theta_1 + \theta_2(t - 1970),$$

where t is the year and N is the number of transistors. Note that θ_1 is the model's prediction of the log of the number of transistors in 1970, and 10^{θ_2} gives the model's preidiction of the fractional increase in number of stransistors per year.

(a) Find the coefficients θ_1 and θ_2 that minimize the RMS error on the data, and give the RMS error on the data. Plot the model you find along with the data points.

(b) Use your model to preidict the number of transistors in microprocessor introduced in 2015. Compare the prediction to the IBM Z13 microprocessor, released in 2015, which has around 4×10^9 transistors.

(c) Compare your result with Moore's law, which states that the number of transistors per integrated circuit roughly doubles every one and a half to two year.

The computer scientist and Intel corporation co-founder Gordon Moore formulated the law that bears his name in a magazine article published in 1965.

Solution:

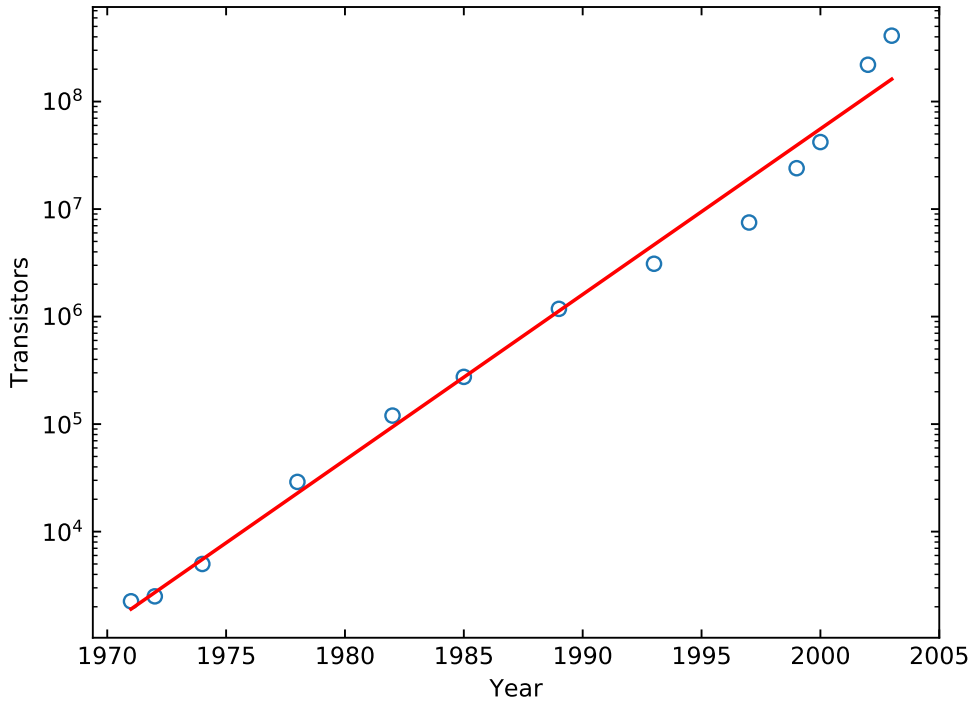
(a) We use vector notaion to denote the data, as $x^{(d)}$ in which $x_i^{(d)}$ means $t_i - 1970$ for $i = 1,\ldots,13$, $y^{(d)}$ in which $y_i^{(d)}$ means $\log_{10} N_i$ for $i = 1,\ldots,13$, $\theta = (\theta_1, \theta_2)$, and matrix $A = \begin{bmatrix} \mathbf{1} & x^{(d)} \end{bmatrix}$, we have

$$\theta = A^{\dagger} y^{(d)}.$$

we have

$$\theta_1 \approx 3.126, \qquad \theta_2 \approx 0.154.$$

RMS is 0.203. plot are



(b) The preiction in 2015 is $\hat{N}_{2015} = 10^{\theta_1 + \theta_2(2015-1970)} \approx 10^{10}$, compare to $N_{2015} = 4 \times 10^9$, as $\hat{N}_{2015}/N_{2015} = 2.5$, which is not that good as we might expected.

13.4 Asset α and β and market correlation. Suppose the T-vectors $r^{ind} = (r_1^{ind},\ldots,r_T^{ind})$ and $r^{mkt} = (r_1^{mkt},\ldots,r_T^{mkt})$ are return time series for a specific asset and the whole market, as described on page 251. We let r^{rf} denote

the risk-free interest rate, μ^{mkt} and σ^{mkt} the market return and risk (*i.e.*, $\mathbf{avg}(r^{mkt})$ and $\mathbf{std}(r^{mkt})$), and μ and σ the return and risk of the asset (*i.e.*, $\mathbf{avg}(r^{ind})$ and $\mathbf{std}(r^{ind})$). Let ρ be the correlation coefficient between the market and asset return time series r^{mkt} and r^{ind}. Express the asset α and β in terms of r^{rf}, μ, σ, μ^{mkt}, σ^{mkt}, and ρ.

Solution:

The model is

$$\hat{f}(x) = (\alpha + r^{rf}) + \beta(x - \mu^{mkt}).$$

Compare it with the straight line mode $\hat{f}(x) = \theta_1 + \theta_2 x$, we have $\beta = \theta_2$ and $\alpha = \theta_1 - r^{rf} + \beta\mu^{mkt}$. We use the solution of the straight-line model, which has

$$\theta_2 = \rho\frac{\mathbf{std}(r^{ind})}{\mathbf{std}(r^{mkt})} = \rho\frac{\sigma}{\sigma^{mkt}}, \qquad \theta_1 = \mathbf{avg}(r^{i}nd) - \theta_2\mathbf{avg}(r^{mkt}) = \mu - \rho\mu^{mkt}\frac{\sigma}{\sigma^{mkt}},$$

therefore

$$\beta = \theta_2 = \frac{\rho\sigma}{\sigma^{mkt}},$$

$$\alpha = \theta_1 - r^{rf} + \beta\mu^{mkt} = \mu - \rho\mu^{mkt}\frac{\sigma}{\sigma^{mkt}} - r^{rf} + \mu^{mkt}\frac{\rho\sigma}{\sigma^{mkt}} = \mu - r^{rf}.$$

13.5 Polynomial model with multiple feature. The idea of polynomial models can be expressed from the case discussed on page 255 where there is only one feature. In this exercise we consider a quadratic (degree two) model with 3 features, *i.e.*, x is a 3-vector. This has the form

$$\hat{f}(x) = a + b_1x_1 + b_2x_2 + b_3x_3 + c_1x_1^2 + c_2x_2^2 + c_3x_3^2 + c_4x_1x_2 + c_5x_1x_3 + c_6x_2x_3,$$

where the scalar a, 3-vector b, and 6-vector c are the zeroth, first, and second order coefficients in the model. Put this model into our general linear in the parameter form, by giving p, and the basis function $f_1, \ldots, f_p$ (which map 3-vector to scalars).

Solution:

In this case, we have

$$f_1(x) = 1, \qquad f_2(x) = x_1, \qquad f_3(x) = x_2, \qquad f_4(x) = x_3, \qquad f_5(x) = x_1^2,$$

$$f_6(x) = x_2^2, \qquad f_7(x) = x_3^2, \qquad f_8(x) = x_1x_2, \qquad f_9(x) = x_1x_3, \qquad f_{10}(x) = x_2x_3.$$

There are $p = 10$ basis functions, denote the parameters vector as

$$w = (a, b, c)$$

we can change this fitting problem to a least squares problem.

13.6 Average prediction error. Consider a data fitting problem, with first basis function $\phi_1(x) = 1$, and the data set $x^{(1)}, \ldots, x^{(N)}$, $y^{(1)}, \ldots, y^{(N)}$. Assume the matrix A in (13.1) has linearly independent columns and let $\hat{\theta}$ denote the parameter values that minimize the mean sqaure prediction error over the data set. Let the N-vector $\hat{r}^d$ denote the prediction errors using the optimal model parameter $\hat{\theta}$. Show that $\mathbf{avg}(\hat{r}^d) = 0$. In other words: With the least squares fit, the mean of the prediciton errors over the data set is zero. *Hint.* Use the orthogonality principle (12.9), with $z = e_1$.

Solution:

With the orthogonality principle: Which declears that if $A\hat{x}$ is closest to vector b, *i.e.*, $\hat{x}$ is the solution of least squares problem

$$\text{minimize} \quad \|Ax - b\|^2,$$

and $\hat{r} = A\hat{x} - b$ is the optimal residual, then for any vector z, we have

$$Az \perp \hat{r}.$$

Let $z = e_1$, Az is the first column of A, for which entries are all one, *i.e.*, $Ae_1 = \mathbf{1}$, due to $Az \perp \hat{r}^d$, we have $\mathbf{1} \perp \hat{r}^d$, *i.e.*$\mathbf{1}^T r^d = 0$, with

$$\mathbf{1}^T \hat{r}^d = \mathbf{avg}(\hat{r}^d),$$

we have

$$\mathbf{avg}(\hat{r}^d) = 0.$$

13.7 Data matrix in auto-regressive time series model. An auto-regressive model woth memory M is fit by minimizing the sum of the squares of the predictions errors on a data set with T samples, $z_1, \ldots, z_T$, as described on page 259. Find the matrix A and vector y for which $\|A\beta - y\|^2$ gives the sum of the squares of the prediction errors. Show that A is a Toeplitz

matrix (see page 138), *i.e.*, entries of A_{ij} with the same value of $i-j$ are the same.

Solution:

The zuto-regressice model with memory M is

$$\hat{z}_{t+1} = \theta_1 z_t + \cdots + \theta_M z_{t-M+1}, \qquad t = M, M+1, \ldots,$$

let a_i^T denote the ith row of A, we have

$$a_i^T = \begin{bmatrix} Z_{i+M-1} & \cdots & Z_i \end{bmatrix},$$

therefore

$$A_{ij} = Z_{i+M-j}.$$

which means for all pair of i, j that have the same value of $i-j$, A_{ij} are same, *i.e.*, it is a Toeplitz matrix.

13.8 Fitting an input-output convolution system. Let $u_1, \ldots, u_T$ and $y_1, \ldots, y_T$ be observed input and output time series for a system that is thought to be an input-output convolution system, meaning

$$y_t \approx \hat{y}_t = \sum_{j=1}^{n} h_j u_{t-j+1}, \qquad t = 1, \ldots, T,$$

where we interpret u_t as zero for $t \leq 0$. Here the n-vector h is the system impluse response; see page 140. This model of the relation between the input and the output time series is also called a *moving average* (MA) model. Find a matrix A and vector b for which

$$\|Ah - b\|^2 = (y_1 - \hat{y}_2)^2 + \cdots + (y_T - \hat{y}_T)^2.$$

Show that A is Toeplitz. (See page 138.)

Solution:

This is like auto-regressive model, we can denote a_i^T to be row of A, for $i = 1, \ldots, T$, which we have

$$a_i^T = \begin{bmatrix} u_i & \cdots & u_{i-n+1} \end{bmatrix},$$

therefore

$$A_{ij} = (a_i)_j^T = u_{i-j+1}, \qquad j = 1, \ldots, n.$$

Thus, for any pair (i, j) that have the same value of $i - j$, the value of A_{ij} are same, which means $T \times n$ matrix A is Toeplitz matrix.

13.9 Conclusions from 5-fold cross-validation. You have developed a regression model for predicting a scalar outcome y from a feature vector x of dimension 20, using a collection of $N = 600$ data points. The mean of the outcome variable y across the givem data is 1.85, and its standard deviation is 0.32. After running 5-fold across-validation we get the following RMS test errors (based on forming a model based on the data excluding fold i, and testing it on fold i).

Fold excluded	RMS test error
1	0.13
2	0.11
3	0.09
4	0.11
5	0.14

(a) How would you expect your model to do on new, unseen (but similar) data? Respond brief and justify your response.

(b) A co-worker observes that the regression model parameters found in the 5 different folds are quite close, but not the same. He says that for the production system, you should use the regression model parameters found when you excluded fold 3 from the original data, since it achieved the best RMS test error. Comment brief.

Solution:

(a) We can see that the RMS test errors for all 5 folds are small than the standard deviation, means our model is good than a constant model. As we do not know the RMS train error, we can't make a conclusion that this model is over-fit or not, but we find out that all RMS test error are similar, which means this model has good stability.

(b) × **NO.** We can not choose the parameters for which have the minimum RSM test error among cross-validation, because the parameters

in the 3th fold achieved the smallest RMS error maybe due to the specific data, and didn't have good generalization. Usually, after cross-validation, we specify the model, and fit the model to all data set to get the parameters, or we can choose the average of the parameters in all folds data fitting.

13.10 Augmenting feature with the average. You are fitting a regression model $\hat{y} = x^T\beta + v$ to data, computing the model coefficient β and v using least squares. A friend suggests adding a new feature, which is the average of the original features. (That is, he suggests using the new feature vector $\tilde{x} = (x, \mathbf{avg}(x))$.) He explains that by adding this new feature, you might end up with a better model. (Of course, you would test the new model using validation.) Is this a good idea?

Solution:

It is not a good idea, because the average of x is a linear combination of x, which will not effect the result campared with our original model. We use mathematical notation to show this:
Suppose the new model with additional feature $\mathbf{avg}(x)$ is

$$\hat{f}^{new}(x) = x^T\beta^{new} + \mu\mathbf{avg}(x) + v^{new}.$$

denote n-vector a for which the entries $a_i = 1/n$, then

$$\mathbf{avg}(x) = x^T a.$$

Therefore

$$\hat{f}^{new}(x) = x^T\beta^{new} + \mu x^T a + v^{new} = x^T(\beta^{new} + \mu a) + v^{new},$$

it has the same formula with origianl mode $\hat{f} = x^T\beta + v$ as

$$\beta = \beta^{new} + \mu a, \qquad v = v^{new}.$$

Thus they are the same model!

13.11 Interpreting model fitting results. Five different models are fit using the same training data set, and tested on the same (separate) test set (which has the same size as the training set). The RMS prediction

errors for each model, on the training and test sets, are reported below. Commnet briefly on the results for each model. You might mention whether the model's predictions are good or bad, whether it is likely to generalize to unseen data, or whether it is over-fit. You are also welcome to say that you don't believe the results, or think the reported numbers are fishy.

Model	Train RMS	Test RMS
A	1.355	1.423
B	9.769	9.165
C	5.033	0.889
D	0.211	5.072
E	0.633	0.633

Solution:

For model A, B, E, they all have similar train RMS and test RMS, means they all have good generalization, among them model E has the best prediction performace, model B has the worst prediciton performance.
For model D it seems to have good prediciton performace on train data, but bad on test data, means it is over-fitting.
For model C, the test RMS is far smaller than train RMS, which I believe it is a fishy, or wrong number.

13.12 Standardizing Boolean features. (See page 269.) Suppose that the N-vector x gives the value of a (scalar) Boolean feature across a set of N examples. (Boolean means that each x_i has the value 0 and 1. This might represent the presence or absence of a symptom, or whether or not a day is a holiday.) How do we standardize such a feature? Express your answer in terms of p, the fraction of x_i that have the vaule 1. (You can assume that $p > 0$ and $p < 1$; otherwise the feature is constant.)

Solution:

The average of feature is

$$\mathbf{avg}(x) = p,$$

where p is the fraction of x_i that have the value 1. and the standard deviation is

$$\mathbf{std}(x) = \sqrt{\frac{\sum_{i=1}^n (x_i - p)^2}{n}} = \sqrt{\frac{\sum_{i=1}^n (x_i^2 - 2px_i + p^2)}{n}} = \sqrt{\frac{np - 2np^2 + np^2}{n}} = \sqrt{p - p^2}.$$

Therefore the standardized feture of x are

$$x' = \frac{x - p}{\sqrt{p - p^2}}.$$

13.13 Interaction model with Boolean features. Consider a data fitting problem in which all n original features are Boolean, *i.e.*, entries of x have the value 0 or 1. These features could be the results of Boolean tests for patient (or absence or present of symptoms), or a person's response to a survey with yes/no questions. We wish to use those to predict an outcome, the number y. Our model will include a constant feature 1, the original n Boolean features, and all *Interaction terms*, which have the form $x_i x_j$ where $1 \leq i < j \leq n$.

(a) What is p, the total number of basis functions, in this model? Explicity list the basis functions for the case $n = 3$. You can decide their order. *Hint.* To count the number of pairs i, j that satisfy $1 \leq i < j \leq n$, use equation (5.7).

(b) Interpret (together) the following three coefficients of θ: the one associated with the original feature x_3; the one associated with original feature x_5; and the one associated with the new product feature $x_3 x_5$. *Hint.* Consider the four possible values of the pair x_3, x_5.

Solution:

(a) $p = 1 + n + \sum_{i=1}^n n - i = 1 + n + n(n-1)/2$.

(b) We assume the features be the results of Boolean test for patient, coefficient of feature x_3 means how would the sympton 3 affect the result if the test for sympton 3 is positive (or say, to be true). It is same with the coefficient of feature x_5, and the coefficient of $x_3 x_5$ means how it will affecet the result, if the

test for sympton 3 and sympton 5 are both positive (or say, to be true).

13.14 Least squares timing. A computer takes around one second to fit a regression model (using least squares) with 20 parameters using 10^6 data points.

(a) About how long do you guess it will take the same computer to fit the same 20-parameter model using 10^7 data points (*i.e.*, $10\times$ more data points)?

(b) About how long do you guess it will take the same computer to fit a 200-parameter model using 10^6 data points (*i.e.*, $10\times$ more model parameters)?

Solution:

The complexity to solve the least squares problem with an $m\times n$ matrix A of the form $\|Ax-b\|^2$ is around $2mn^2$.

(a) The data point corresponded to the row of the matrix A, which has linear related with the complexity, so the time may be 10 seconde.

(b) The model parameters corresponded to the column of the matrix A, which has squared related with the complexity, so the time may be 100 seconds.

13.15 Estimating a matrix. Suppose that the n-vector x and the m-vector y are thought to be approximately related by a linear function, *i.e.*, $y = Ax$, where A is an $m\times n$ matrix. We do not know the matrix A, but we do have observed data,

$$x^{(1)},\ldots,x^{(N)}, \qquad y^{(1)},\ldots,y^{(N)}.$$

We can estimate or guess the matrix A by choosing it to minimize

$$\sum_{i=1}^{N}\left\|Ax^{(i)}-y^{(i)}\right\| = \|AX-Y\|^2,$$

where $X = \begin{bmatrix}x^{(1)} & \cdots & x^{(N)}\end{bmatrix}$ and $Y = \begin{bmatrix}y^{(1)} & \cdots & y^{(N)}\end{bmatrix}$. We denote this *least squares estimate* as $\hat{A}$. (This notation here can be confusing, since X and Y are

known, and A is to be found; it is more conventional to have symbols near the begining of the alphabet, like A, denote known quantities, and symbols near the end, like X and Y, denote variables or unknowns.)

(a) Show that $\hat{A} = YX^\dagger$, assuming the rows of X are linearly independent.
Hint. Use $\|Ax - T\|^2 = \left\|X^T A^T - Y^T\right\|^2$, whic turns the problem into a matrix least squares problem; see page 233.

(b) Suggest a good way to compute $\hat{A}$, and give the complexity in terms of n, m, and N.

Solution:

(a) We know that for a matrix A, $\|A\|^2 = \left\|A^T\right\|^2$, therefore

$$\|AX - Y\|^2 = \left\|(AX - Y)^T\right\|^2 = \left\|X^T A^T - Y^T\right\|^2.$$

As the row of X are linearly independent, we know the column of X^T are linearly independent. Therefore

$$\hat{A}^T = (X^T)^\dagger Y^T = (X^\dagger)^T Y^T,$$

Thus

$$\hat{A} = YX^\dagger.$$

(b) As X^T can be QR factorized, we assume $X^T = QR$, then we have

$$X^\dagger = Q^T R^{-T},$$

therefore

$$\hat{A} = YX^\dagger = YQ^T R^{-T}.$$

To solve $\hat{A}$, we first compute multiply Q^T by Y, then we compute $(YQ^T)R^{-T}$ by forward-substitution, with the algorithm below:

- *QR factorization.* Compute the QR factorization $X^T = QR$.
- Compute YQ^T.
- *Forward substitution.* Solve the triangular equation $AR^T = YQ^T$.

13.16 Relative fitting error and error fitting the logarithm. (See page 259.) The relative fitting error between a positive outcome y and its prediction $\hat{y}$ is given by $\eta = \max\{\hat{y}/y, y/\hat{y}\} - 1$. (This is often given as a

percentage.) Let r be the residual between their logarithms, $r = \log y - \log \hat{y}$. Show that $\eta = e^{|r|} - 1$.

Solution:

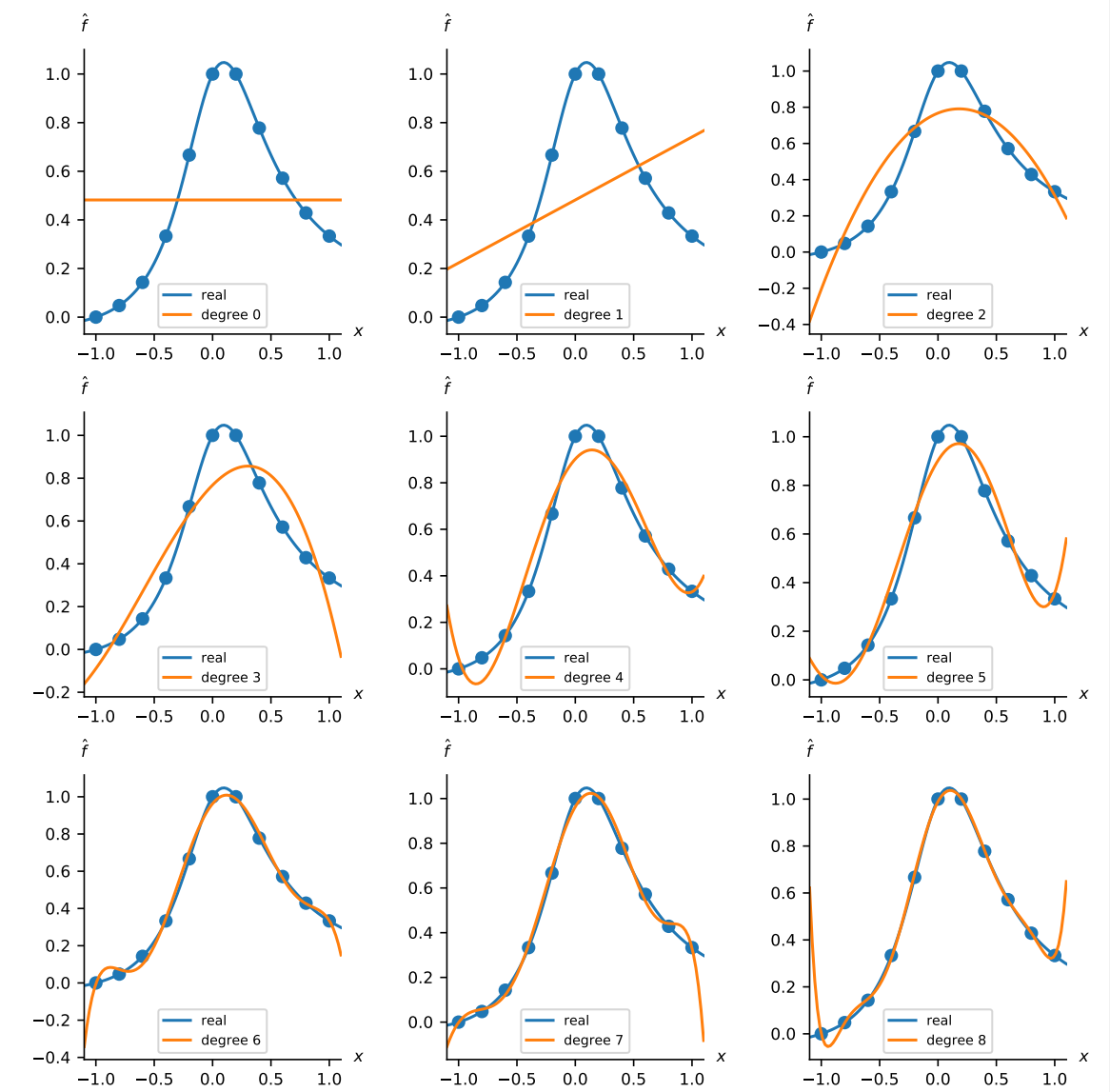
If $y \geq \hat{y}$, $|r| = \log y - \log \hat{y} = log(y/\hat{y})$, therefore $e^{|r|} - 1 = y/\hat{y} - 1$.
If $y < \hat{y}$, $|r| = \log \hat{y} - \log y = log(\hat{y}/y)$, therefore $e^{|r|} - 1 = \hat{y}/y - 1$.
Combine these, we have $e^{|r|} - 1 = \max\{\hat{y}/y, y/\hat{y}\} - 1 = \eta$.

13.17 Fitting a rational function with a polynomial. Let $x_1, \ldots, x_{11}$ be 11 points uniformly spaced in the interval $[-1, 1]$. (This means $x_i = -1.0 + 0.2(i-1)$ for $i = 1, \ldots, 11$.) Take $y_i = (1 + x_i)/(1 + 5x_i^2)$, for $i = 1, \ldots, 11$. Find the least squares fit of polynomials of degree $0, 1, \ldots, 8$ to these points. Plot the fitting polynomials, and the true function $y = (1 + x)/(1 + 5x^2)$, over the interval $[-1.1, 1.1]$ (say, using 100 points). Note that the interval for the plot, $[-1.1, 1.1]$, extends a bit outside the range of the data used to fit the polynomials, $[-1, 1]$; this gives us an idea of how well the polynomial fits can extrapolate.

Generate a test data set by choosing $u_1, \ldots, u_{10}$ uniformly spaced over $[-1.1, 1.1]$, with $v_i = (1 + u_i)/(1 + 5u_i^2)$. Plot the RMS error of the polynomial fits above on this test data set. Suggest a reasonable value for the degree of the polynomial fit, based on the RMS fits on the training and test data. *Remark.* There is no practical reason to fit a rational function with a polynomial. This exercise is only meant to illustrate the ideas of fitting with different basis functions, over-fit, and validation with a test data set.

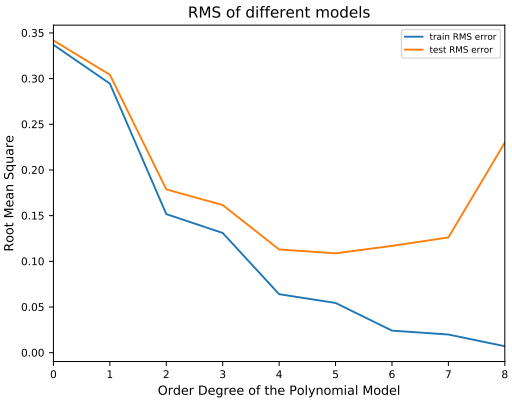
Solution:

Follow are the plot:



With the test data u we generated, the test RMS and train test are listed and plot followed:

Degree	Train RMS	Test RMS
0	0.337	0.342
1	0.295	0.304
2	0.152	0.179
3	0.131	0.162
4	0.064	0.113
5	0.054	0.109
6	0.024	0.117
7	0.020	0.126
8	0.007	0.230



13.18 Vector auto-regressive model. The auto-regressive time series model (see page 259) can be extended to handle times series $z_1, z_2, \ldots$ where z_t are n-vectors. This arises in applications such as econometrics, where the entries of the vector give different economic quantities. The vector auto-regressive model (already mentioned on page 164) has the same form as a scalar auto-regressive model,

$$\hat{z}_{t+1} = \beta_1 z_t + \cdots + \beta_M z_{t-M+1}, \qquad t = M, M+1, \ldots$$

where M is the *memory* of the model; the difference here is that the model parameters $\beta_1, \ldots, \beta_M$ are all $n \times n$ matrices. The total number of (scalar) parameters in the vector auto-regressive model is Mn^2. These parameters are chosen to minimize the sum of the squared norms of the prediction errors over a data set $z_{M+1}, \ldots, z_T$,

$$\|z_{M+1} - \hat{z}_{M+1}\|^2 + \cdots + \|z_T - \hat{z}_T\|^2 .$$

(a) Give an interpretation of $(\beta_2)_{13}$. What would it mean if this coefficient is very small?

(b) Fitting a vector auto-regressive model can be expressed as a matrix least squares problem (see page 233), *i.e.*, the problem of choosing the matrix X to minimize $\|AX - B\|^2$, where A and B are matrices. Find the matrices A and B. You can consider the case $M = 2$. (It will be clear how to generalize your answer to larger values of M.)

Hint. Use the $(2n) \times n$ matrix variable $X = \begin{bmatrix} \beta_1 & \beta_2 \end{bmatrix}^T$, and right-hand side

($(T-2)\times n$ matrix) $B = \begin{bmatrix} z_3 & \cdots & z_T \end{bmatrix}^T$. Your job is to find the matrix A so that $\|Ax - B\|^2$ is the sum of the squared norms of the prediction errors.

Solution:

(a) The $(\beta_2)_{13}$ is the 13th columns in matrix β_2, it means how the 13th feature's value at 2 times periods ago affecet the next state.

(b) Consider z_{M+i}, for $i = 1,\ldots,T-M$, the prediciton $\hat{z}_{M+i}$ is

$$\hat{z}_{M+i} = \beta_1 z_{M+i-1} + \cdots + \beta_M z_i,$$

therefore the residual of the real value and prediciton value is

$$\begin{aligned} \|z_{M+i} - \hat{z}_{M+i}\|^2 &= \left\|\hat{z}_{M+i}^T - z_{M+i}^T\right\|^2 \\ &= \left\|(z_{M+i-1}^T \beta_1^T + \cdots + z_i^T \beta_M^T) - z_{M+i}^T\right\|^2 \\ &= \left\| \begin{bmatrix} z_{M+i-1}^T & \cdots & z_i^T \end{bmatrix} \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_M^T \end{bmatrix} - z_{M+i}^T \right\|^2 \end{aligned}$$

let

$$a_i^T = \begin{bmatrix} z_{M+i-1}^T & \cdots & z_i^T \end{bmatrix}, \quad X = \begin{bmatrix} \beta_1 & \cdots & \beta_M \end{bmatrix}^T,$$

the equaiton above can be formed as

$$\|z_{M+i} - \hat{z}_{M+i}\|^2 = \left\|a_i^T X - z_{M+i}^T\right\|^2.$$

Now let

$$A = \begin{bmatrix} a_1 & \cdots & a_{T-M} \end{bmatrix}^T, \qquad B = \begin{bmatrix} z_{M+1} & \cdots & z_T \end{bmatrix}^T,$$

the objective

$$\|z_{M+1} - \hat{z}_{M+1}\|^2 + \cdots + \|z_T - \hat{z}_T\|^2$$

can be expressed as

$$\|AX - B\|^2.$$

13.19 Sum of sinusoid time series model. Suppose that $z_1, z_2, \ldots$ is a time series. A very common approximation of the time series is as a sum of K

sinusoids

$$z_t \approx \hat{z}_t = \sum_{k=1}^{K} a_k \cos(\omega_k t - \phi_k), \qquad t = 1, 2, \ldots.$$

The kth term in this sum is called *sinusoid signal*. The coefficient $a_k \geq 0$ is called the *amplitude*, $\omega_k > 0$ is called *frequency*, and ϕ_k is called the *phase* of the kth sinusoid. (The phase is usually chosen to lie in the range from $-\pi$ to π.) In many applications the frequencies are multiples of ω_1, *i.e.*, $\omega_k = k\omega_1$ for $k = 2, \ldots, K$, in which case the approximation is called a *Fourier approximation*, named for the mathematician Jean-Baptiste Joseph Fourier.

Suppose you have observed the values $z_1, \ldots, z_T$, and wish to choose the sinusoid amplitudes $a_1, \ldots, a_K$ and phases $\phi_1, \ldots, \phi_K$ so as to minimize the RMS value of the approximation error $(\hat{z}_1 - z_1, \ldots, \hat{z}_T - z_T)$. (We assume that the frequencies are given.) Explain how to solve this using least squares model fitting.

Hint. A sinusoid with amplitude a, frequency ω, and phase ϕ can be described by its cosine and sine coefficients α and β, where

$$a\cos(\omega t - \phi) = \alpha\cos(\omega t) + \beta\sin(\omega t),$$

where (using the cosine of sum formula) $\alpha = a\cos(\phi)$, $\beta = a\sin(\phi)$. We can recover the amplitude and phase from the cosine and sine coefficients as

$$a = \sqrt{\alpha^2 + \beta^2}, \qquad \phi = \arctan(\beta/\alpha).$$

Express the problem in terms of the cosine and sine coefficients.

Solution:

Our purpose is to find the parameters K-vectors a and θ, that minimize the value

$$\|\hat{z} - z\|^2 .$$

Consider our modle

$$\hat{f}(t) = \sum_{k=1}^{K} a_k \cos(\omega_k t - \phi_k),$$

as we know

$$a\cos(\omega t - \phi) = \alpha\cos(\omega t) + \beta\sin(\omega t),$$

we can rewrite our model in the form below:

$$\hat{f}(t) = \sum_{k=1}^{K} (\alpha_k \cos(\omega_k t) + \beta_k \sin(\omega_k t)),$$

this is a **linear in the parameters model**, as we consider $2K$-vector $\theta = (\alpha, \beta)$, and the basis function

$$\hat{f}_i(t) = \begin{cases} \cos(\omega_i t), & i = 1, \ldots, K \\ \sin(\omega_{i-K} t) & i = k+1, \ldots, 2K \end{cases}$$

Use least squares model fitting we can get $\hat{\alpha}$ and $\hat{\beta}$, Thus

$$\hat{a} = \sqrt{\hat{\alpha}^2 + \hat{\beta}^2}, \qquad \hat{\phi} = \arctan(\hat{\beta}/\hat{\alpha}).$$

13.20 Fitting with continuous and discontinuous piecewise-linear functions. Consider a fitting problem with $n = 1$, so $x^{(1)}, \ldots, x^{(N)}$ and $y^{(1)}, \ldots, y^{(N)}$ are numbers. We consider two types of closely related models. The first is a piecewise-linear model with knot points at -1 and 1, as described on page 256, and illustrated in figure 13.8. The second is a stratified model (see page 272), with three independent affine models, one for $x < -1$, one for $-1 \leq x \leq 1$, and one for $x > 1$. (In other words, we stratify on x taking low, middle, or high values.) Are these two models the same? Is one more general than the other? How many parameters does each model have? *Hint.* See problem title.

What can you say about the training set RMS error and test RMS error that would be achieved using least squares with these tow models?

Solution:

The basis function of first model, *i.e.*, peicewise-linear model is

$$f_1(x) = 1, \qquad f_2(x) = x, \qquad f_3(x) = (x+1)_+, \qquad f_4(x) = (x-1)_+,$$

therefore the model is

$$\tilde{f}(x) = \theta_1 + \theta_2 x + \theta_3 (x+1)_+ + \theta_4 (x-1)_+.$$

The straight model can be described as

$$\tilde{f}'(x) = \begin{cases} \theta_1 + \theta_2(x+1), & x < -1 \\ \theta_3 + \theta_4 x, & -1 \le x \le 1 \\ \theta_5 + \theta_6(x-1), & x > 1 \end{cases},$$

it is same with the piecewise-linear model while

$$\lim_{x\to -1^+}(\theta_1 + \theta_2(x+1)) = \lim_{x\to -1^-}(\theta_3 + \theta_4 x), \qquad \lim_{x\to 1^+}(\theta_3 + \theta_4 x) = \lim_{x\to 1^-}(\theta_5 + \theta_6(x-1)),$$

i.e., piecewise-linear model is straight model with above constants, which make the model continuous at the knot points.
As the straight model is more compliment than piecewise-linear model (according to number of the parameters), we guess the straight model has less train set RMS error.

13.21 Efficient cross-validation. The cost of fitting a model with p basis functions and N data points (say, using QR factorization) is $2Np^2$ flops. In this exercise we explore the complexity of carrying out 10-fold cross-validation on the same data set. We divide the data set into 10 folds, each with $N/10$ data points. The naïve method is to fit 10 different models, each using 9 of folds, using the QR factorization, which requires $10 \cdot 2(0.9)Np^2 = 18Np^2$ flops. (To evalute each of these models on the remaining fold requires $2(N/10)p$ flops, which can be ignored conpared to the cost of fitting the models.) So the naïve method of carrying out 10-fold cross-validation requires, not surprisingly, around $10\times$ the number of flops as fitting a single model.

The method below outlines another method to carry out 10-fold cross-validation. Give the total flop count for each step, keeping only the dominant terms, and compare the total cost of the method to that of the naïve method. Let $A_1, \ldots, A_{10}$ denote the $(N/10) \times p$ blocks of the data matrix associated with the folds, and let $b_1, \ldots, b_{10}$ denote the right-hand sides in the least squres fitting problem.

(a) Form the Gram matrices $G_i = A_i^T A_i$ and the vectors $c_i = A_i^T b_i$.

(b) Form $G = G_1 + \cdots + G_{10}$ and $c = c_1 + \cdots + c_{10}$.

(c) For $k = 1, \ldots, 10$, compute $\theta_k = (G - G_k)^{-1}(c - c_k)$.

Solution:

For fold k as the test data set, the coefficient matrix is $(A_1, \ldots, A_{k-1}, A_{k+1}, A_N)$, the Gram matrix of this matrix is

$$A_1^T A_1 + \cdots + A_{k-1}^T A_{k-1} + A_{k+1}^T A_{k+1} + \cdots + A_N^T A_N = G - G_k,$$

the transform of matrix multiplying right-hide side vector is

$$A_1^T b_1 + \cdots + A_{k-1}^T b_{k-1} + A_{k+1}^T b_{k+1} + \cdots + A_N^T b_N = c - c_k,$$

therefore we can get model parameters

$$\theta_k = (G - G_k)^{-1}(c - c_k).$$

Now let derive the complexity of this method, first we need $10 \times (N/10)p^2 = Np^2$ flops to compute G_i, and $10 \times 2(N/10)p = 2Np$ flops to compute c_i, and $10p^2$ flops for computing G and $10p$ flops to compute c. Last, we use QR factorization to solve $(G - G_k)\theta_k = (c - c_k)$ which need $10 \times 2p^3 = 20p^3$ flops, totally

$$Np^2 + 2Np + 10p^2 + 10p + 20p^3$$

flops, when $N >> p$, this methods is more efficent than the naïve method.

13.22 Prediction contests. Several companies have run prediction contests open to the public. Netflix ran the best known contest, offering a $1M prize for the first prediction of user movie rating that beat their existing method RMS prediction error by 10% on a test set. The contests generally work like this (although there are several variations on this format, and most are more complicated). The company posts a public data set, that includes the regressions or features and the outcome for a large number of examples. They also post the features, but not the outcomes, for a (typically smaller) test data set. The contestants, usually teams with obscure names, submit predictions for the outcomes in the test set. usually there is a limit on how many times, or how frequently, each team can submit a prediction on the test set. The company computes the RMS test

set prediction error (say) for each submission. The teams' prediction performance is shown on a *leaderboard*, which lists the 100 or so best predictions in order.

Discuss such contests in terms of model validation. How should a team check a set of predictions before submitting it? What would happen if there were no limits on the number of predictions each team can submit? Suggest an obvious method (typically disallowed by the contest rules) for a team to get around the limit on prediction submissions. (And yes, it has been done.)

Solution:

Because we know the features of the test data set, but not the outcome, the test data set is more like a real furture data set for us, and the RMS the company compute can interpret how our model performance in real uknown data set.

If there is no limits for the number of submition, we suppose we make N submit, the submitted outputs are denoted by $y^{(1)},\dots,y^{(N)}$, and the RMS system feedback are $r_1,\dots,r_N$, denote the real output be y, we have

$$\left\|y^{(i)}-y\right\|^2 = nr_i^2, \qquad i=1,\dots,N,$$

where n is the dimension of y. Substract ith$(i=2,\dots,N)$ equation to the first quation, we can get

$$2y^T(y^{(1)}-y^{(i)}) = n(r_i^2-r_1^2)+\left\|y^{(1)}\right\|^2-\left\|y^{(i)}\right\|^2, \qquad i=2,\dots,N,$$

as we know $y^{(i)}$ and r_i, we can reform $N-1$ equtions above as

$$\begin{bmatrix} 2(y^{(1)}-y^{(2)})^T \\ \vdots \\ 2(y^{(N)}-y^{(2)})^T \end{bmatrix} y = \begin{bmatrix} n(r_2^2-r_1^2)+\left\|y^{(1)}\right\|^2-\left\|y^{(2)}\right\|^2 \\ \vdots \\ n(r_N^2-r_1^2)+\left\|y^{(1)}\right\|^2-\left\|y^{(N)}\right\|^2 \end{bmatrix},$$

when we have enough $y^{(i)}$ and r_i, we can compute out y directly.

14 Least squares classification

14.1 Chebyshev bound. Let $\tilde{f}(x)$ denote the continuous prediction of the Boolean outcome y, and $\hat{f}(x) = \mathbf{sign}(\tilde{f}(x))$ the actual classifier. Let σ denote

the RMS error in the continuous prediction over some set of data, *i.e.*,

$$\sigma^2 = \frac{(\tilde{f}(x^{(1)}) - y^{(1)})^2 + \cdots + (\tilde{f}(x^{(N)}) - y^{(N)})^2}{N}.$$

Use the Chebyshev bound to argure that the error rate over this data set, *i.e.*, the fraction of data points for which $\tilde{f}(x^{(i)}) \neq y^{(i)}$, is no more than σ^2, assuming $\sigma < 1$.

Remark. This bound on the error is usually quite bad; that is, the actual error rate is often much lower than this bound. But it does show that if the continuous prediction is good, then the classifier must perform well.

Solution:

We know $\hat{f}(x) = \mathbf{sign}(\tilde{f}(x))$, which means that $\hat{f}(x) \neq y$ means $|\tilde{f}(x) - y| \leq 1$, suppose there are k entries satisfy $|\tilde{f}(x) - y| \leq 1$, according to Chebyshev inequality, we have

$$k \leq \left\|\tilde{f}(x) - y\right\|^2,$$

divede by N in both hand-sides, we have

$$k/N \leq \sigma^2.$$

14.2 Interpreting the parameters in a regression classifier. Consider a claasifier of the form $\hat{y} = \mathbf{sign}(x^T\beta + v)$, where $\hat{y}$ is the prediction, the n-vector x is the feature vector, and the n-vector β and scalar v are the claasifier parameters. We will assume here that $v \neq 0$ and $\beta \neq 0$. Evidently $\hat{y} = \mathbf{sign}(v)$ is the prediction when the feature vector x is zero. Show that when $\|x\| < |v|/\|\beta\|$, we have $\hat{y} = \mathbf{sign}(v)$. This shows that when all the features are small (*i.e.*, near zero), the classifier makes the prediction $\hat{y} = \mathbf{sign}(v)$.
Hint. If two number a and b satisfy $|a| < |b|$, then $\mathbf{sign}(a + b) = \mathbf{sign}(b)$.

This means that we can interpret $\mathbf{sign}(v)$ as the classifier prediction when the feature are small. The ratio $|v|/\|b\|$ tells us how big the feature vector must be before the classifier 'changes its mind' and predict $\hat{y} = -\mathbf{sign}(v)$.

Solution:

With Cauchy-Schwarz inequality, we have

$$|x^T b| \leq \|x\| \, \|b\|,$$

if $\|x\| < |v|/\|b\|$, we have

$$|x^T b| \leq \|x\| \, \|b\| < |v| \, \|b\| \, / \|b\| = |v|.$$

And with the fact that if $|a| < |b|$, then $\mathbf{sign}(a+b) = \mathbf{sign}(b)$, we have

$$\mathbf{sign}(x^T \beta + v) = \mathbf{sign}(v).$$

14.3 Likert classifier. A response to a question has the options *Strongly Desagree*, *Disagree*, *Neutral*, *Agree*, *Strongly Agree*, encode as $-2, -1, 0, 1, 2$, respectively. You wish to build a multi-class classifier that takes a feature vector x and predicts the response. A multi-class least squares classifier builds a separate (continuous) predictor for each response versus the others. Suggest a simple classifier, based on one continuous regression model $\tilde{f}(x)$ that is fit to the numbers that code the response, using least squares.

Solution:

We can consider the classifier defined below

$$\hat{f}(x) = \begin{cases} -2, & \tilde{f}(x) < -1.5 \\ -1, & -1.5 \leq \tilde{f}(x) < -0.5 \\ 0, & -0.5 \leq \tilde{f}(x) < 0.5 \\ 1 & 0.5 \leq \tilde{f}(x) < 1.5 \\ 2 & \tilde{f}(x) \geq 1.5 \end{cases}.$$

14.4 Multi-class classifier via matrix least squares. Consider the least squares multi-class classifier described in 14.3, with a regression model $\tilde{f}_k(x) = x^T \beta_k$ for the one-versus-others classifiers. (We assume that the offset term is included using a constant feature.) Show that the coefficient vector $\beta_1, \ldots, \beta_K$ can be found by solving the matrix least squares problem of minimizing $\left\|X^T \beta - Y\right\|^2$, where β is the $n \times K$ matrix with columns $\beta_1, \ldots, \beta_K$, and Y is an $N \times K$ matrix.

(a) Give Y, *i.e.*, described its entries. What is the ith row of Y?

(b) Assuming the rows of X (*i.e.*, the data feature vectors) are linearly

independent, show that the least squares estimate is given by $\hat{\beta} = (X^T)^\dagger Y$.

Solution:

(a) The jth entry of ith row of Y can be expressed as

$$Y_{ij} = \begin{cases} 1, & \text{if } i\text{th sample is labeled } j \\ -1, & \text{otherwise} \end{cases},$$

the ith row can be expressed as

$$(2e_j - \mathbf{1})^T,$$

where e_j is the jth unit vector.

(b) Let $y_1, \ldots, y_K$ denote the columns of Y, the objective $\left\|X^T\beta - Y\right\|^2$ can be formed as

$$\left\|X^T\beta - Y\right\|^2 = \|X\beta_1 - y_1\|^2 + \cdots + \|X\beta_K - y_K\|^2.$$

To find β to minimize the objective is same to find β_i minimizing $\|X\beta_i - y_i\|$ independently for $i = 1, \ldots, K$, which is $\hat{\beta}_i = X^\dagger y_i$, therefore

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_1 & \ldots & \hat{\beta}_K \end{bmatrix} = \begin{bmatrix} X^\dagger y_1 & \ldots & X^\dagger y_K \end{bmatrix} = X^\dagger Y.$$

14.5 List classifier. Consider a multi-class classification problem with K classes. A standard multi-class classifier is a function $\hat{f}$ that returns a class (one of the label $1, \ldots, K$), when given a feature n-vector x. We interpret $\hat{f}(x)$ as the classifier's guess of the class that corresponds to x. A *list classifier* returns a list of guesses, typically in order from 'most likely' to 'least likely'. For example, for a specific feature vector x, a list classifier might return $3, 6, 2$, meaning (roughly) that its top guess is class 3, its next guess is class 6, and its third guess is class 2. (This lists can have diffirent lengths for different values of the feature vector.) How would you modify the least squares multi-class classifier described in 14.3.1 to create a list classifier? *Remark.* List classifier are widely used in electronic communication systems, where the feature vector x is the received signal, and the class correspons to which of k messages was sent. In this context they are called *list*

decoders. List decoders produce a list of probable or likely messages, and allow a later processing stage to make the final decision or guess.

Solution:

It is similar with the standard multi-class classifier, as in the standard multi-class classification we use

$$\hat{f}(x) = \underset{k=1,\ldots,K}{\operatorname{argmax}} \tilde{f}_k(x),$$

as the classifier, it choose the index of the largest value among the numbers $\tilde{f}_k(x)$.

In list classification, we consider a theshold α, say, for index k which satisfies $\tilde{f}_k(x) > \alpha$ means the sample with feature x can be classified as label k. (We can see that generally $0 \le \alpha \le 1$.), we denote $\text{filter}(x,;\alpha) \in [1,\ldots,K]$ as the set of index k which satisfies $\tilde{f}_k(x) > \alpha$, in math:

$$\text{filter}(x,;\alpha) = \{k : \tilde{f}_k(x) > \alpha\}, \qquad k = 1,\ldots,K.$$

Then we sort the set by decreasing the values of $\tilde{f}_k(x)$, means the head of the set have more likely to be the label of the sample.

14.6 Ploynomial classifier with one feature. Generate 200 points $x^{(1)},\ldots,x^{(200)}$, uniformly spaced in the interval $[-1,1]$, and take

$$y^{(i)} = \begin{cases} +1 & -0.5 \le x^{(i)} < 0.1 \text{ or } 0.5 \le x^{(i)} \\ -1 & \text{otherwise} \end{cases}$$

for $i = 1,\ldots,200$. Fit polynomial least squares classifiers of degrees $0,\ldots,8$ to this training data set.

(a) Evaluate the error rate on the training data set. Does the error decrease when you inncrease the degree?

(b) For each degree, plot the polynomial $\tilde{f}(x)$ and the classifier $\hat{f}(x) = \mathbf{sign}(\tilde{f}(x))$.

(c) It is possible to classify this data set perfectly using a claasifier $\hat{f}(x) = \mathbf{sign}(\tilde{f}(x))$ and a cubic polynomial

$$\tilde{f}(x) = c(x + 0.5)(x - 0.1)(x - 0.5),$$

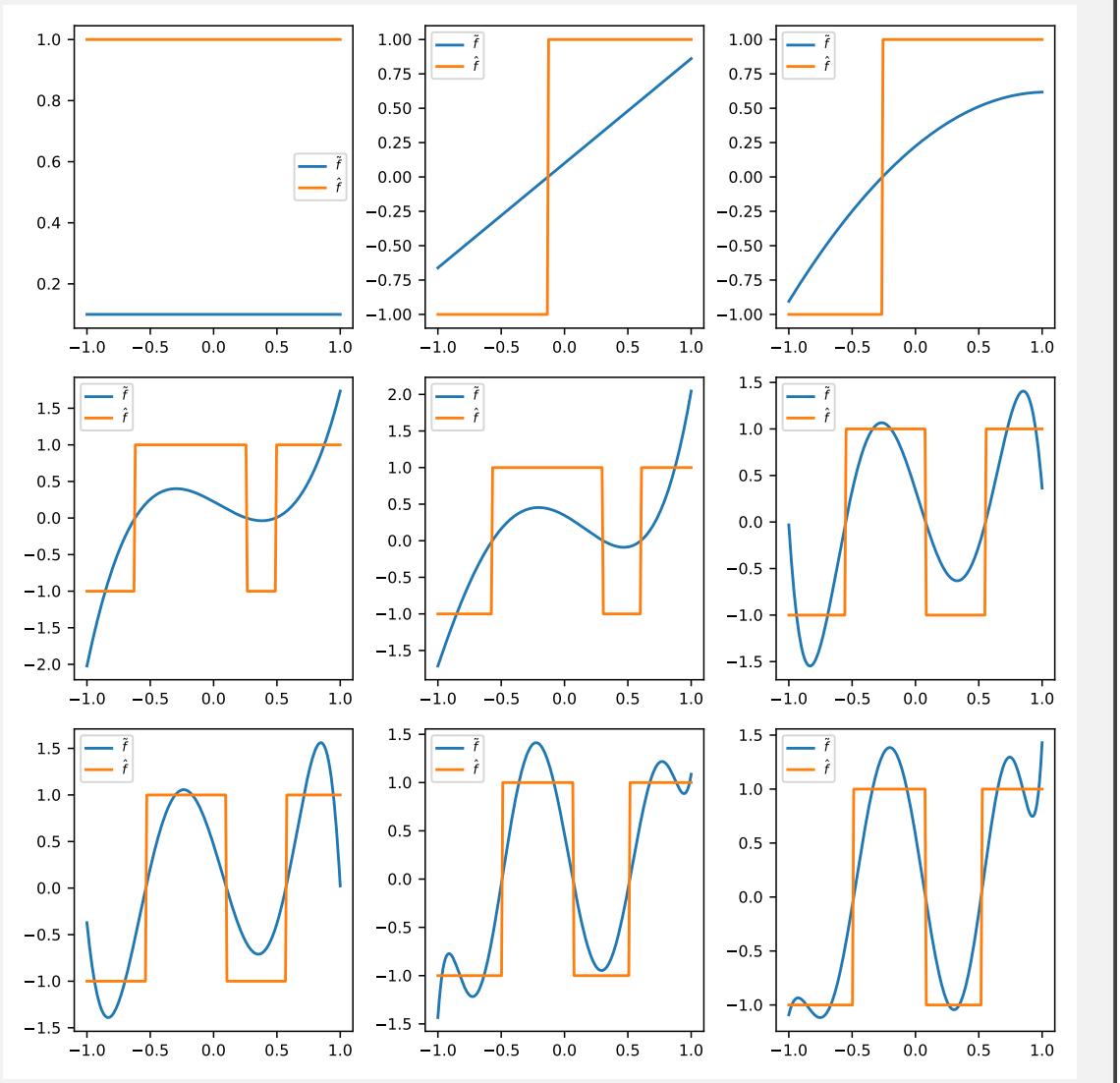
for any positive c. Compare this classifier with the least squares classifier of degree 3 that you found and explain why there is a difference.

Solution:

(a) The error rate are displayed in the follow table:

Degree	Error rate
0	0.9
1	0.77
2	0.64
3	0.29
4	0.37
5	0.12
6	0.1
7	0.05
8	0.05

(b) The plots are



14.7 Polynomial classifier with two features. Generate 200 random 2-vectors $x^{(1)},\ldots,x^{(200)}$ in a plane, from a standard norm distribution. Define

$$y^{(i)} = \begin{cases} +1 & x_1^{(i)} x_2^{(i)} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

for $i = 1,\ldots,200$. In other words, $y^{(i)}$ is $+1$ when $x^{(i)}$ is in the first or third quadrant, and -1 otherwise. Fit a polynomial least squares classifier of degree 2 to the data set, *i.e.*, use a polynomial

$$\tilde{y}(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_1^2 + \theta_5 x_1 x_2 + \theta_6 x_2^2.$$

Give the error rate of the classifier. Show the regions in the plane where $\tilde{f}(x) = 1$ and $\tilde{f}(x) = -1$. Also compare the computed coefficients with the polynomial $\tilde{f}(x) = x_1 x_2$, which classifies the data points with zero error.

Solution:

We use the method described above and use the least squares get the parameters are

$$\theta_1 = 0.02, \quad \theta_2 = -0.10, \quad \theta_3 = -0.02, \quad \theta_4 = 0.06, \quad \theta_5 = 0.89, \quad \theta_6 = 0.01,$$

(The results are around to 2 digits.) and the confusion matrix are

	Prediciton		
Outcome	$\hat{y} = +1$	$\hat{y} = -1$	Total
$y = +1$	99	3	102
$y = -1$	9	89	98
All	108	92	200

The error rate is $(3+9)/200 = 6\%$, its true positive rate is $99/102 = 97.1\%$, its false positive rate is $9/98 = 9.2\%$ (means it incorrectly labeled around 9.2% of the points in second and fourth quadrant as in first and third quadrant.)

Compare with the real classifier $\tilde{f}(x) = x_1 x_2$, the coefficient of $x_1 x_2$ in our model is $\theta_5 = 0.89$ is larger than other features, means our model is more depended on the feature $x_1 x_2$, it is same with the real classifier, and grows confident on our model.

The codes are 5. (In the code the x are generated randomly, means you may get different results every time, but it will be similar.)

Listing 5: Python code for 14.7

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exerises 14.7
# Author: Utoppia
# Date  : 17 May 2020

import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
```

```
from sklearn.metrics import confusion_matrix

def main():
    # Generate 200 examples with standard normal distribution
    N, p = 200, 2
    x = np.random.normal(size=(N,p))
    func = lambda x, y: 1 if x*y >= 0 else -1
    real_fun = np.frompyfunc(func, 2, 1)
    real_y = real_fun(x[:,0], x[:,1]).astype(np.int8)

    # Define the classifier
    classifier = np.frompyfunc(lambda x: 1 if x>=0 else -1, 1, 1)

    # Define model
    model = Pipeline([
        ('poly', PolynomialFeatures(degree=2)), # feature engineering
        ('linear', LinearRegression(fit_intercept=False)) # linear model
    ])

    model.fit(x, real_y) # modle fit on data
    tilde_y = model.predict(x) # prediction on original data
    pred_y = classifier(tilde_y).astype(np.int8)

    print("The parameters are: ")
    print(model.named_steps['linear'].coef_)

    print('Confusion matrix is:')
    print(confusion_matrix(real_y, pred_y))

if __name__ == '__main__':
    main()
```

14.8 Author attribution. Suppose that the N feature n-vectors $x^{(1)},\ldots,x^{(N)}$ are word count histograms, and the labels $y^{(1)},\ldots,y^{(N)}$ give the document authors (as one of $1,\ldots,K$). A classifier guesses which of the K authors wrote an unseen document, which is called *author atrribution.* A least squares classifier using regression is fit on the data, resulting in the classifier

$$\hat{f}(x) = \underset{k=1,\ldots,K}{\mathbf{argmax}}(x^T\beta_k + v_k).$$

For each author (*i.e.*, k=1,…,K) we find the ten largest (most positive) entries in the n-vector β_k and the ten smallest (most negative) entries. These correspond to two sets of ten words in the dictionary, for each author. Interpret these words, brief, in English.

Solution:

For the words corresponded to the ten largest value of β_k, it means these words have more weight to judge whether this document is writen by author k or not, in other word, the author k has more likely to use these words. For the words that corresponded to the smallest entries, say, is most not likely used by author k.

14.9 Nearest neighbor interpretation of multi-class classifier. We consider the least squares K-class classifier of 14.3.1. We associate with each data point the n-vector x, and the label or class, which is one of $1,\ldots,K$. If the class of the data point is k, we associate it with a K-vector y, whose entries are $y_k = +1$ and $y_j = -1$ for $j \neq k$. (We can write this vector as $y = 2e_k - \mathbf{1}$.) Define $\tilde{y} = (\tilde{f}_1(x),\ldots,\tilde{f}_K(x))$, which is our (real value or continuous) prediction of the label y. Our multi-class prediction is given by $\hat{f}(x) = \mathbf{argmax}_{k=1,\ldots,K} \tilde{f}_k(x)$. Show that $\hat{f}(x)$ is also the index of the nearest neighbor of the nearest neighbor of our continuous prediction $\tilde{y}$, among the vectors that encode the class label.

Solution:

The question is same as to show that

$$\operatorname*{argmin}_{k=1,\ldots,K} \|\tilde{y} - y_k\|^2 = \operatorname*{argmax}_{k=1,\ldots,K} \tilde{f}_k(x).$$

Expand $\|\tilde{y} - y_k\|^2$ as following:

$$\begin{aligned}
\|\tilde{y} - y_k\|^2 &= \|\tilde{y}\|^2 + \|y_k\|^2 - 2\tilde{y}^T y_k \\
&= \|\tilde{y}\|^2 + \|2e_k - \mathbf{1}\| - 2\tilde{y}^T(2e_k - \mathbf{1}) \\
&= \|\tilde{y}\|^2 + K - 4\tilde{y}^T e_k + 2\tilde{y}^T \mathbf{1} \\
&= \|\tilde{y}\|^2 + K + 2\tilde{y}^T \mathbf{1} - 4\tilde{f}_k(x)
\end{aligned}$$

Because the term $\|\tilde{y}\|^2$, K, and $2\tilde{y}^T\mathbf{1}$ are not depended on k, therefore

$$\underset{k=1,\ldots,K}{\operatorname{argmin}} \|\tilde{y} - y_k\|^2 = \underset{k=1,\ldots,K}{\operatorname{argmin}} -4\tilde{f}_k(x) = \underset{k=1,\ldots,K}{\operatorname{argmax}} \tilde{f}_k(x).$$

14.10 One-versus-one multi-class classifier. In 14.3.1 we construct a K-class classifier from K Boolean classifier that attempt to distinguish each class from the others. In this exercise we describe another method for constructing a K-class classifier. We first develop a Boolean classifier for every *pair* of class i and class j, $i < j$. There are $K(K-1)/2$ such pairs of classifiers, called *one-versus-one* classifiers. Given a feature vector x, we let $\hat{y}_{ij}$ be the prediction of the i-versus-j classifier, with $\hat{y}_{ij} = 1$ meaning that the one-vertsus-one classifier guessing that $y = i$. We consider $\hat{y}_{ij} = 1$ as one 'vote' for class i, and $\hat{y}_{ij} = -1$ as one 'vote' for class j. We obtain the final estimated class by *majority voting*: We take $\hat{y}$ as the class that has the most votes. (We can break ties in some simple way, like taking the samllest index that achieves the largest nunber of votes.)

(a) Construct the largest squares classifier, and the one-versus-one classifier, for a multi-class (trining) data set. Find the the confusion matrices, and the error rates, of the two classifiers on both the training data set and a separate test data set.

(b) Compare the complexity of computing the one-versus-one multi-class classifier with the complexity of the least squares multi-class classifier (see page 300). Assume the training set contains N/K exmaples of each class and that N/K is much greater than the number of features p. Distinguish two methods for the one-versus-one multi-class classifier.The first, naïve, method solves $K(K-1)/2$ least squares problem with N/K rows and p columns. The second, more efficient, method precomputes the Gram matrices $G_i = A_i A_i^T$ for $i = 1, \ldots, K$, where the rows of the $(N/K) \times p$ matrix A_i are the training example for class i, and uses the pre-computed Gram matrices to speed up the solution of the $K(K-1)/2$ least squares problems.

Solution:

(a) Denote $\tilde{f}_{ij}$ to be the least squares regression model for class i versus class j, $\tilde{f}_{ij}(x)$ is the real-valued prediction for Boolean classifier for class i versus class j, the classifier for class i versus class j is defeind as

$$\hat{f}_{ij}(x) = \mathbf{sign}(\tilde{f}_{ij}(x)).$$

Let $f_i^*(x)$ denote the 'vote' for feature x predicted as class i, then we have

$$f_i^*(x) = K/2 + \sum_{j=1}^{K} \hat{f}_{ij}(x).$$

Thus the K-class classifier $\hat{f}(x)$ can be formed as

$$\hat{f}(x) = \operatorname*{argmax}_{k=1,\ldots,K} f_i^*(x).$$

14.11 Equalizer design from training message. We consider an electronic communication system, with message to be sent given by an N-vector s, whose entries are -1 or 1, and received signal y, where $y = c*s$, where c is an n-vector, the channel impulse response. The receiver applies equalization to receive signal, which means that it computers $\tilde{y} = h*y = h*c*s$, where h is an n-vector, the equalizer impulse response. The receiver then estimates the original message using $\hat{s} = \mathbf{sign}(\tilde{y}_{1:N})$. This works well if $h*c \approx e_1$. (See exercise 7.15.) If the channel impulse response c is known or can be measured, we can design or choose h using least squares, as in exercise 12.6.

In this exercise we explore a method for choosing h directly, withoud estimating or measuring c. The sender first sends a message that is *known* to the receiver, called the *training message*, s^{train}. (From the point of view of communications, this is wasted transmission, and called *overhead*.) The recevier receives the signal $y^{\text{train}} = c*s^{\text{train}}$ from the training message, and then chooses h to minimize $\left\|(h*y^{\text{train}})_{1:N} - s^{\text{train}}\right\|^2$. (In practice, this equalizer is used until the bit error rate increases, which means the channle has changed, at which point another training message is sent.) Explain how this method is the same as least squares classification. What

are the training data $x^{(i)}$ and $y^{(i)}$? What is the least squares problem that must be solved to determine the equalizer impulse response h?

Solution:

First we expand $(h * y^{\text{train}})_{1:N}$, we have

$$(h * y^{\text{train}})_{1:N} = \begin{bmatrix} h_1 y_1 \\ h_1 y_2 + h_2 y_1 \\ \vdots \\ h_1 y_n + \cdots + h_n y_1 \end{bmatrix}$$

let $\tilde{f} : \mathrm{R}^N \to R$ to a linear function defined as

$$\tilde{f}(x) = x^T h,$$

and let $x^{(i)}$ be N-vector which is

$$x^{(i)} = (y_i, y_{i-1}, \ldots, y_1, 0, \ldots, 0), \qquad i = 1, \ldots, N,$$

therefore $(h * y^{\text{train}})_{1:N}$ can be formed as

$$(h * y^{\text{train}})_{1:N} = \begin{bmatrix} \tilde{f}(x^{(1)}) \\ \vdots \\ \tilde{f}(x^{(N)}) \end{bmatrix},$$

let $y^{(i)} = s_i^{\text{train}}$ for $i = 1, \ldots, N$, the objective can be write as

$$\left\| y^{(1)} - \tilde{f}(x^{(1)}) \right\|^2 + \cdots + \left\| y^{(N)} - \tilde{f}(x^{(N)}) \right\|^2,$$

with classifier

$$f(x) = \mathbf{sign}(\tilde{f}(x)),$$

we conclude that the problem is same as the least squares classification problem. Let $n \times n$ matrix A to be

$$A = \begin{bmatrix} y_1 & & & \\ y_2 & y_1 & & \\ \vdots & & \vdots & \\ y_N & y_{N-1} & \cdots & y_1 \end{bmatrix},$$

we can solve the classifier problem by find h which minimize

$$\left\| Ah - s^{\text{train}} \right\|^2.$$

15 Multi-objective least squares

15.1 A scalar multi-objective least squares problem. We consider the special case of the multi-objective least squares problem in which the variable x is a scalar, and the k matrices A_i are all 1×1 matrices with value $A_i = 1$, so $J_i = (x - b_i)^2$. In this case our goal is to choose a number x that is simultaneously close to all the numbers $b_1, \ldots, b_k$. Let $\lambda_1, \ldots, \lambda_k$ be positive weights, and $\hat{x}$ the minimizer of the weighted objective (15.1). Show that $\hat{x}$ is a weighted average (or convex combination; see page 17) of the number $b_1, \ldots, b_k$, *i.e.*, it has the form

$$x = w_1 b_1 + \cdots + w_k b_k,$$

where w_i are nonegative and sum to one. Give an explicit formular for the combination weights w_i in terms of the multi-objective least squares weights λ_i

Solution:

According to (15.1), which is

$$J = \lambda_1 J_1 + \cdots + \lambda_k J_k.$$

Our goal is to find the $\hat{x}$ that minimize the weighted sum objective

$$J = \lambda_1 (x - b_1)^2 + \cdots + \lambda_k (x - b_k)^2.$$

Denote $k \times 1$ matrix A as

$$A = (\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_k}),$$

and k-vector b as

$$b = (\sqrt{\lambda_1} b_1, \ldots, \sqrt{\lambda_k} b_k).$$

Our objective can be form as

$$J = \|Ax - b\|^2.$$

This is a least squares problem, we know

$$\hat{x} = A^{\dagger} b$$

is the solution to minimize the objective J, with

$$A^TA = \lambda_1 + \cdots + \lambda_k, \qquad A^Tb = \lambda_1 b_1 + \cdots + \lambda_k b_k,$$

Here we denote λ^{sum} as the sum of all λ_i for $i = 1,\ldots,k$, *i.e.*, $A^TA = \lambda^{\text{sum}}$. Therefore

$$\hat{x} = A^\dagger b = (A^TA)^{-1}A^Tb = (\lambda^{\text{sum}})^{-1}(\lambda_1 b_1 + \cdots + \lambda_k b_k),$$

let $w_i = \lambda_i/\lambda^{\text{sum}}$ for $i = 1,\ldots,k$, we have

$$\hat{x} = w_1 b_1 + \cdots + w_k b_k,$$

and

$$w_1 + \cdots + w_k = 1.$$

15.2 Consider the regularized data fitting problem (15.7). Recall that the elements in the first column of A are one. Let $\hat{\theta}$ be the solution of (15.7), *i.e.*, the minimizer of

$$\|A\theta - y\|^2 + \lambda(\theta_2^2 + \cdots + \lambda_p^2),$$

and let $\tilde{\theta}$ be the minimizer of

$$\|A\theta - y\|^2 + \lambda\|\theta\|^2 = \|A\theta - y\|^2 + \lambda(\theta_1^2 + \theta_2^2 + \cdots + \lambda_p^2),$$

in which we also penalize θ_1. Suppose columns 2 through p of A have mean zero (for example, because features $2,\ldots,p$ have been standardized on the data set; see page 269). Show that $\hat{\theta}_k = \tilde{\theta}_k$ for $k = 2,\ldots,p$.

Solution:

Denote

$$A = \begin{bmatrix}\mathbf{1} & X\end{bmatrix}, \qquad \theta = (v,\beta),$$

here X is $n\times(p-1)$ matrix, and v is scalar with $v = \theta_1$, β is a $(p-1)$-vector with $\beta = (\theta_2,\ldots,\theta_p)$. with

$$\lambda(\theta_2^2 + \cdots + \lambda_p^2) = \left\|\begin{bmatrix}0 & 0\\ 0 & \sqrt{\lambda}I\end{bmatrix}\begin{bmatrix}v\\ \beta\end{bmatrix}\right\|^2,$$

here I is a $(p-1)\times(p-1)$ unit matrix, Let

$$\hat{A} = \begin{bmatrix} \mathbf{1} & X \\ 0 & 0 \\ 0 & \sqrt{\lambda} I \end{bmatrix}, \qquad \hat{y} = (y, 0, 0).$$

The first objective can be write in form as

$$\left\| \hat{A} \begin{bmatrix} v \\ \beta \end{bmatrix} - \hat{y} \right\|^2.$$

Therefore we have

$$\begin{bmatrix} \hat{v} \\ \hat{\beta} \end{bmatrix} = \hat{A}^{\dagger} \hat{y},$$

with $\hat{A}^{\dagger} = (\hat{A}^T \hat{A})^{-1} \hat{A}^T$, we have

$$\hat{A}^T \hat{A} \begin{bmatrix} \hat{v} \\ \hat{\beta} \end{bmatrix} = \hat{A}^T \hat{y}.$$

Now we expand $\hat{A}^T \hat{A}$ and $\hat{A}^T \hat{y}$,

$$\hat{A}^T \hat{A} = \begin{bmatrix} \mathbf{1}^T & 0 & 0 \\ X^T & 0 & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} \mathbf{1} & X \\ 0 & 0 \\ 0 & \sqrt{\lambda} I \end{bmatrix} = \begin{bmatrix} n & \mathbf{1}^T X \\ X^T \mathbf{1} & X^T X + \lambda I \end{bmatrix}.$$

We know the columns 2 through p of matrix A have mean zero, means the columns of X have mean zero, *i.e.*,

$$\mathbf{1}^T X = 0, \qquad X^T \mathbf{1} = 0,$$

notice that the 0 in left equation is an $1\times(p-1)$ matrix, and the 0 in right eqaution is a $(p-1)\times 1$ matrix. therefore

$$\hat{A}^T \hat{A} = \begin{bmatrix} n & 0 \\ 0 & X^T X + \lambda I \end{bmatrix}.$$

Also we have

$$\hat{A}^T \hat{y} = \begin{bmatrix} \mathbf{1}^T & 0 & 0 \\ X^T & 0 & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix} = \mathbf{1}^T y + X^T y = A^T y.$$

Thus we imply the equation

$$\begin{bmatrix} n & 0 \\ 0 & X^T X + \lambda I \end{bmatrix} \begin{bmatrix} \hat{v} \\ \hat{\beta} \end{bmatrix} = A^T y.$$

Now we consider the second objective, let

$$\tilde{A} = \begin{bmatrix} \mathbf{1} & X \\ \sqrt{\lambda} & 0 \\ 0 & \sqrt{\lambda} I \end{bmatrix}, \qquad \tilde{y} = (y, 0, 0).$$

The second objective can be formed as

$$\left\| \tilde{A} \begin{bmatrix} v \\ \beta \end{bmatrix} - \tilde{y} \right\|^2,$$

we have

$$\tilde{A}^T \tilde{A} \begin{bmatrix} \tilde{v} \\ \tilde{\beta} \end{bmatrix} = \tilde{A}^T \tilde{y}.$$

Expand $\tilde{A}^T \tilde{A}$:

$$\tilde{A}^T \tilde{A} = \begin{bmatrix} \mathbf{1}^T & \sqrt{\lambda} & 0 \\ X^T & 0 & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} \mathbf{1} & X \\ \sqrt{\lambda} & 0 \\ 0 & \sqrt{\lambda} I \end{bmatrix} = \begin{bmatrix} n+\lambda & \mathbf{1}^T X \\ X^T \mathbf{1} & X^T X + \lambda I \end{bmatrix} = \begin{bmatrix} n+\lambda & 0 \\ 0 & X^T X + \lambda I \end{bmatrix},$$

and

$$\tilde{A}^T \tilde{y} = \begin{bmatrix} \mathbf{1}^T & \sqrt{\lambda} & 0 \\ X^T & 0 & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} y & 0 & 0 \end{bmatrix} = \mathbf{1}^T y + X^T y = A^T Y,$$

thus we have

$$\begin{bmatrix} n+\lambda & 0 \\ 0 & X^T X + \lambda I \end{bmatrix} \begin{bmatrix} \tilde{v} \\ \tilde{\beta} \end{bmatrix} = A^T y.$$

Consider the first equation we got, we have

$$\begin{bmatrix} n & 0 \\ 0 & X^T X + \lambda I \end{bmatrix} \begin{bmatrix} \hat{v} \\ \hat{\beta} \end{bmatrix} = \begin{bmatrix} n+\lambda & 0 \\ 0 & X^T X + \lambda I \end{bmatrix} \begin{bmatrix} \tilde{v} \\ \tilde{\beta} \end{bmatrix},$$

which is same as

$$n\hat{v} = (n+\lambda)\tilde{v}(X^T X + \lambda I)(\hat{\beta} - \tilde{\beta}) = 0$$

therefore we have

$$\hat{\beta} = \tilde{\beta}.$$

15.3 Weighted Gram matrix. Consider a multi-objective least squares problems with matrices $A_1, \ldots, A_k$ and positive weights $\lambda_1, \ldots, \lambda_k$. The matrix

$$G = \lambda_1 A_1^T A_1 + \cdots + \lambda_k A_k^T A_k$$

is called *weighted Gram matrix*; it is the Gram matrix of the stacked matrix $\tilde{A}$ (given in (15.2)) associated with the multi-objective problem. Show that G is invertible provided there is no nonzero vector x that

satisfies $A_1 x = 0, \ldots, A_k x = 0$.

Solution:

Suppose $Gx = 0$, implies $x^T G x = 0$, means

$$\lambda_1 x^T A_1^T A_1 x + \cdots + \lambda_k x^T A_k^T A_k x = \lambda_1 \|A_1 x\|^2 + \cdots + \lambda_k \|A_k x\|^2 = 0.$$

implies

$$A_i x = 0, i = 1, \ldots, k,$$

as there is no nonzero x that satisfies $A_1 x = 0, \ldots, A_k x = 0$, thus $x = 0$, means the columns of G is linearly independent, *i.e.*, G is invertible.

15.4 Robust approximate solution of linear equations. We wish to solve the sqaure set of n linear eqautions $Ax = b$ for the n-vector x. If A is invertible the solution is $x = A^{-1}b$. In this exercise we address an issue that comes up frequently: We don't know A exactly. One simple method is to just choose a typical value of A and use it. Another method, which we explore here, takes into account the vatiation in the matrix A. We find a set of K versions of A, and denote them as $A^{(1)}, \ldots, A^{(K)}$. (These could be found by measuring the matrix A at different times, for example.) Then we choose x so as to minimize

$$\left\|A^{(1)}x - b\right\|^2 + \cdots + \left\|A^{(K)}x - b\right\|^2,$$

the sum of the squares of residuals obtained with the K versions of A. This choice of x, which we denote x^{rob}, is called a *robust* (approximate) solution. Give a formula for x^{rob}, in terms of $A^{(1)}, \ldots, A^{(K)}$ and b. (You can assume that a matrix you construct has linearly independent columns.) Verify that for $K = 1$ your formula reduces to $x^{\text{rob}} = (A^{(1)})^{-1}b$.

Solution:

Let

$$A = \begin{bmatrix} A^{(1)} \\ \vdots \\ A^{(K)} \end{bmatrix}, \qquad \tilde{b} = \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix},$$

we have $x^{\mathrm{rob}} = A^{\dagger}c$, with $A^{\dagger} = (A^TA)^{-1}A^T$, we have

$$x^{\mathrm{rob}} = ((A^{(1)})^TA^{(1)} + \cdots + (A^{(K)})^TA^{(K)})^{-1}((A^{(1)})^Tb + \cdots + (A^{(K)})^Tb),$$

if $K = 1$,

$$x^{\mathrm{rob}} = ((A^{(1)})^TA^{(1)})^{-1}(A^{(1)})^Tb = (A^{(1)})^{\dagger}b = (A^{(1)})^{-1}b.$$

15.5 Some properties of bi-objective least squares. Consider the bi-objective least squares problem with objectives

$$J_1(x) = \|A_1x - b_1\|^2, \qquad J_2(x) = \|A_2x - b_2\|^2.$$

For $\lambda > 0$, let $\hat{x}(\lambda)$ denote the minimizer of $J_1(x)+\lambda J_2(x)$. (We assume the columns of the stacked matrix are linearly independent.) We define $J_1^{\star}(\lambda) = J_1(\hat{x}(\lambda))$ and $J_2^{\star}(\lambda) = J_2(\hat{x}(\lambda))$, the values of the two objectives as functions of the weight parameter. The optimal trade-off curve is the set of points $(J_1^{\star}(\lambda), J_2^{\star}(\lambda))$, as λ varies over all positive numbers.

(a) *Bi-objective problems without trade-off.* Suppose μ and γ are different positive weights, and $\hat{x}(\mu) = \hat{x}(\gamma)$. Show that $\hat{x}(\lambda)$ is constant for all $\lambda > 0$. Therefore the point $(J_1^{\star}(\lambda), J_2^{\star}(\lambda))$ is same for all λ and the trade-off curve collapses to a single point.

(b) *Effec of weight on objectives in a bi-objective problem.* Suppose $\hat{x}(\lambda)$ is not constant. Show that following: for $\lambda > \mu$, we have

$$J_1^{\star}(\lambda) < J_1^{\star}(\mu), \qquad J_2^{\star}(\lambda) > J_2^{\star}(\mu).$$

This means that if you increase the weight (on the second objective), the second objective goes down, and the first objective goes up. In other words that trade-off curve slopes downward.

Hint. resist the urge to write out any equations or formulas. Use the fact that $\hat{x}(\lambda)$ is the unique minimizer of $J_1(x)+\lambda J_2(x)$, and similarly for $\hat{x}(\mu)$, to deduce the inequalities

$$J_1^{\star}(\mu) + \lambda J_2^{\star}(\mu) > J_1^{\star}(\lambda) + \lambda J_2^{\star}(\lambda), \qquad J_1^{\star}(\lambda) + \mu J_2^{\star}(\lambda) > J_1^{\star}(\mu) + \mu J_2^{\star}(\mu).$$

Combine these inequalities to show that $J_1^{\star}(\lambda) < J_1^{\star}(\mu)$ and $J_2^{\star}(\lambda) > J_2^{\star}(\mu)$.

(c) *Slope of the trade-off curve.* The slope of the trade-off curve at

the point $(J_1^\star(\lambda), J_2^\star(\lambda))$ is given by

$$S = \lim_{u \to \lambda} \frac{J_2^\star(\mu) - J_2^\star(\lambda)}{J_1^\star(\mu) - J_1^\star(\lambda)}.$$

(This limit is the same if μ approaches λ from below or from above.) Show that $S = -1/\lambda$. This gives another interpretation of the parameter λ: $(J_1^\star(\lambda), J_2^\star(\lambda))$ is the point on the trade-off curve where the curve has slope $-1/\lambda$.

Hint. First assume that μ approaches λ above (meaning, $\mu > \lambda$) and use the inequalities in the hint for part (b) to show that $S \geq -1/\lambda$. Then assume that μ approaches λ from below and show that $S \leq -1/\lambda$.

Solution:

(a) We have

$$(A_1^T A_1 + \mu A_2^T A_2)\hat{x}(\mu) = A_1^T b_1 + \mu A_2^T b_2,$$
$$(A_1^T A_1 + \gamma A_2^T A_2)\hat{x}(\gamma) = A_1^T b_1 + \gamma A_2^T b_2,$$

the first equation minus the second we get

$$(\mu - \gamma)A_2^T A_2 \hat{x}(\mu) = (\mu - \gamma)A_2^T b_2,$$

therefore

$$A_2^T A_2 \hat{x}(\mu) = A_2^T b_2,$$

and with the result and the first equation, we have

$$A_1^T A_1 \hat{x}(\mu) = A_1^T b_1.$$

Now for λ, we have

$$(A_1^T A_1 + \lambda A_2^T A_2)\hat{x}(\lambda) = A_1^T b_1 + \lambda A_2^T b_2 = (A_1^T A_1 + \lambda A_2^T A_2)\hat{x}(\mu),$$

implies that

$$\hat{x}(\lambda) = \hat{x}(\mu).$$

for any positive λ.

(b) We have

$$J_1^\star(\mu) + \lambda J_2^\star(\mu) > J_1^\star(\lambda) + \lambda J_2^\star(\lambda), \qquad J_1^\star(\lambda) + \mu J_2^\star(\lambda) > J_1^\star(\mu) + \mu J_2^\star(\mu).$$

Or we can write as

$$J_1^\star(\mu) - J_1^\star(\lambda) > \lambda(J_2^\star(\lambda) - J_2^\star(\mu))$$
$$J_1^\star(\lambda) - J_1^\star(\mu) > \mu(J_2^\star(\mu) - J_2^\star(\lambda)).$$

Combine two inequalities, we have

$$0 > (\lambda - \mu)(J_2^\star(\lambda) - J_2^\star(\mu)),$$

as $\lambda - \mu < 0$, therefore $J_2^\star(\lambda) - J_2^\star(\mu) > 0$, *i.e.*,

$$J_2^\star(\lambda) > J_2^\star(\mu).$$

Use same method we have

$$(\frac{1}{\lambda} - \frac{1}{u})(J_1^\star(\mu) - J_1^\star(\lambda) > 0,$$

as $1/\lambda - 1/\mu > 0$, therefore $J_1^\star(\mu) - J_1^\star(\lambda) > 0$, *i.e.*,

$$J_1^\star(\lambda) < J_1^\star(\mu).$$

(c) According to $J_1^\star(\mu) + \lambda J_2^\star(\mu) > J_1^\star(\lambda) + \lambda J_2^\star(\lambda)$, we have

$$J_2^\star(\mu) - J_2^\star(\lambda) > (1/\lambda)(J_1^\star(\lambda) - J_1^\star(\mu)).$$

If $\mu > \lambda$, we have $J_1^\star(\mu) - J_1^\star(\lambda) > 0$, implies

$$\frac{J_2^\star(\mu) - J_2^\star(\lambda)}{J_1^\star(\mu) - J_1^\star(\lambda)} > -1/\lambda,$$

therefore

$$S \geq \lim_{\mu \to \lambda^-} -1/\lambda = -1/\lambda.$$

If $\mu < \lambda$, we have $J_1^\star(\mu) - J_1^\star(\lambda) < 0$, implies

$$\frac{J_2^\star(\mu) - J_2^\star(\lambda)}{J_1^\star(\mu) - J_1^\star(\lambda)} < -1/\lambda,$$

therefore

$$S \leq \lim_{\mu \to \lambda^+} -1/\lambda = -1/\lambda.$$

Thus we have $S = 1/\lambda$.

15.6 Least squares with smoothness regularization. Consider the weighted sum least squares objective

$$\|Ax - b\|^2 + \lambda \|Dx\|^2,$$

where the n-vector x is the variable, A is an $m \times n$ matrix, D is the $(n-1) \times n$ difference matrix, with ith row $(e_{i+1} - e_i)^T$, and $\lambda > 0$. Although it does not matter in this problem, this objective is what we would minimize if we want an x that satisfies $Ax \approx b$, and has entries that are smoothly varying. We can express this objective as a standard least squares objective with a stacked matrix of size $(m + n - 1) \times n$.

Show that the stacked matrix has linearly independent columns if and only if $A\mathbf{1} \neq 0$, *i.e.*, the sum of the columns of A is not zero.

Solution:

The stacked matrix is $\begin{bmatrix} A \\ \sqrt{\lambda} D \end{bmatrix}$, suppose $\begin{bmatrix} A \\ \sqrt{\lambda} D \end{bmatrix} x = 0$, which is same as

$$Ax = 0, \qquad Dx = 0.$$

From $Dx = 0$, implies

$$x_1 = x_2 = \cdots = x_n,$$

here x_i is the ith entry of vector x, in words: all entries of x have same value, suppose the value is x_0, therefore $x = x_0 \mathbf{1}$, therefore we have

$$x_0 A\mathbf{1} = 0.$$

If $A\mathbf{1} \neq 0$, implies $x_0 = 0$, *i.e.*, $x = 0$, means the stacked matrix has linearly independent columns.

Inverstly, if the stacked matrix has linearly independent columns, consider $x = \mathbf{1}$, we have $A\mathbf{1} \neq 1$.

15.7 Greedy regulation policy. Consider a linear dynamical system given by $x_{t+1} = Ax_t + Bu_t$, where the n-vector x_t is the state at time t, and the m-vector u_t is the input at time t. The goal in regulation is to choose the input so as to make the state small. (In applications, the state $x_t = 0$ corresponds to the desired operating point, so small x_t means the state is close to the desired operating point.) One way to achieve this

goal is to choose u_t so as to minimize

$$\|x_{t+1}\|^2 + \rho \|u_t\|^2,$$

where ρ is a (given) positive parameter that trades off using a small input versus making the (next) state small. Show that choosing u_t this way leads to a state feedback policy $u_t = Kx_t$, where K is an $m \times n$ matrix. Give a formula for K (in terms of A, B, and ρ). If an inverse appeares in your formula, state the conditions under which the inverse exists.

Remark. This policy is called *greedy* or *myopic* since it does not take into account the effect of the input u_t on future states, beyond x_{t+1}. It can work very poorly in practice.

Solution:

Our goal is to minimize

$$\|x_{t+1}\|^2 + \rho \|u_t\|^2,$$

which is same as to minimize

$$\|Ax_t + Bu_t\|^2 + \rho \|u_t\|^2.$$

Denote

$$P = \begin{bmatrix} B \\ \sqrt{\rho} I \end{bmatrix}, \qquad b = \begin{bmatrix} -Ax_t \\ 0 \end{bmatrix},$$

Our goal is same as

$$\|Pu_t - b\|^2.$$

which we know has minimizer

$$u_t = P^{\dagger} b = -(B^T B + \rho I)^{-1} B^T A x_t.$$

This is an **Tikhonov regularized inversion,** and $B^T B + \rho I$ is allways invertible without any assumption.

15.8 Estimating the elasticity matrix. In this problem you create a standard model of how demand varies with the prices of a set of products, based on some observed data. There are n different products, with (positive) prices given by the n-vector p. The prices are held constant over some period, say, a day. The (positive) demands for the products over the day are given by the n-vector d. The demand in any particular day

varies, but it is thought to be (approximately) a function of the prices.

The *nominal prices* are given by the n-vector p^{nom}. You can think of these as the prices that have been charged in the past for the products. The *nominal demand* is the n-vector d^{nom}. This is the average value of the demand, when the prices are set to p^{nom}. (The actual daily demand fluctuates around the value d^{nom}.) You know both p^{nom} and d^{nom}. We will describe the prices by their (fractional) variations for the nominal values, and the same for demands. We define δ^p and δ^d as the (vectors of) relative price change and demand change:

$$\delta_i^p = \frac{p_i - p_i^{\text{nom}}}{p_i^{\text{nom}}}, \quad \delta_i^d = \frac{d_i - d_i^{\text{nom}}}{d_i^{\text{nom}}}, \quad i = 1, \ldots, n.$$

So $\delta_3^p = +0.05$ means that the price for product 3 has been increased by 5% over its nominal value, and $\delta_5^d = -0.04$ means that the demand for product 5 in some day is 4% below its nominal value.

Your task is to build a model of the demand as a function of the price, of the form

$$\delta^d \approx E\delta^p,$$

where E is the $n \times n$ elasticity matrix. You don't know E, but you do have the results of some experiments in which the prices were changed a bit from their nominal values for one day, and the day's demands were recorded. This data has the form

$$(p_1, d_1), \ldots, (p_N, d_N),$$

where p_i is the price for day i, and d_i is the observed demand.

Explain how you would estimate E, given this price-demand data. Be sure to explain how you will test for, and (if need) avoid over-fit. *Hint.* Formulate the problem as a matrix least squares problem; see page 233.

Remark. Note the difference between elasticity estimation and demand shaping, discussed on page 315. In demand shaping, we know the elasticity matrix and are choosing prices; in elasticity estimate, we are guessing the elasticity matrix from some observed price and demand data.

Solution:

Denote $(x^{(i)}, y^{(i)})$ be the ith sample, $i = 1, \ldots, N$, where $x^{(i)}$ and $y^{(i)}$ are n-vectors, and define by

$$x_j^{(i)} = \frac{(p_i)_j - p_j^{\text{nom}}}{p_j^{\text{nom}}}, \qquad y_j^{(i)} = \frac{(d_i)_j - d_j^{\text{nom}}}{d_j^{\text{nom}}}.$$

Our goal is to find a matrix E, minimize the objective

$$\left\|Ex^{(1)} - y^{(1)}\right\|^2 + \cdots + \left\|Ex^{(N)} - y^{(N)}\right\|.$$

Let

$$X = \begin{bmatrix} x^{(1)} & \ldots & x^{(N)} \end{bmatrix}, \qquad Y = \begin{bmatrix} y^{(1)}, \ldots, y^{(N)} \end{bmatrix},$$

here X and Y are $n \times N$ matrix, the objective can be formed as

$$\|EX - Y\|^2 = \left\|X^T E^T - Y^T\right\|.$$

This is a matrix least squares problem, Suppose the columns of X^T are linearly independent, *i.e.*, the rows of X are linearly independent, we have

$$\tilde{E}^T = (X^T)^\dagger Y^T = (XX^T)^{-1} XY^T,$$

therefore

$$\tilde{E} = YX^T(XX^T)^{-1}.$$

Here, we can use cross-validation to test our model, and to avoid over-fit.

15.9 Regularizing stratified models. In a *stratified* model (see page 272), we divide the data into different sets, depending on the value of some (often Boolean) feature, and then fit a separate model for each of these two data sets, using the remaining features. As an example, to develop a model of some health outcome we might build a separate model for women and men. In some cases better models are obtained when we encourage the different models in a stratified model to be close to each other. For the case of stratifying on one Boolean feature, this is done by choosing the two model parameters $\theta^{(1)}$ and $\theta^{(2)}$ to minimize

$$\left\|A^{(1)}\theta^{(1)} - y^{(1)}\right\|^2 + \left\|A^{(2)}\theta^{(2)} - y^{(2)}\right\|^2 + \lambda \left\|\theta^{(1)} - \theta^{(2)}\right\|^2,$$

where $\lambda \geq 0$ is a parameter. The first term is the least squares residual for the first model on the first data set (say, women); the second term is the least squares residual for the second model on the second data set (say, men); the third term is a regularization term that encourages the two model parameters to be close to each other. Note that when $\lambda = 0$, we simply fit each model separately; when λ is very large, we are basically fitting one model to all data. Of course the choice of an approximate value of λ is obtained using out-of-sample validation (or cross-validation).

(a) Give a formula for the optimal $(hat\theta^{(1)}, \hat{\theta}^{(2)})$. (If your formula requires one or more matrices to have linearly independent columns, say so.)

(b) *Stratifying across age groups.* Suppose we fit a model with each data point representing a person, and we stratify over the person's *age group*, which is a range of consecutive ages such as $18-24$, $24-32$, $33-45$, and so on. Our goal is to fit a model for each age of k groups, with the parameters for adjacent age groups similar, or not too far, from each other. Suggest a method for doing this.

Solution:

(a) Denote $\theta = (\theta^{(1)}, \theta^{(2)})$, therefore

$$A^{(1)}\theta^{(1)} - y^{(1)} = \begin{bmatrix} A^{(1)} & 0 \end{bmatrix} \theta - y^{(1)}, A^{(2)}\theta^{(2)} - y^{(2)} = \begin{bmatrix} 0 & A^{(2)} \end{bmatrix} \theta - y^{(2)}, \theta^{(1)} - \theta^{(2)} = \begin{bmatrix} I & -I \end{bmatrix} \theta.$$

Let

$$A = \begin{bmatrix} A^{(1)} & 0 \\ 0 & A^{(2)} \\ \sqrt{\lambda} I & -\sqrt{\lambda} I \end{bmatrix}, \qquad b = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ 0 \end{bmatrix},$$

Our goal is same as

$$\|A\theta - y\|^2$$

First we show that matrix A has linearly independent columns if one of $A^{(1)}$ and $A^{(2)}$ has linearly independent columns: let $Ax = 0$, we have

$$A^{(1)}x_1 = 0, \qquad A^{(2)}x_2 = 0, \qquad \sqrt{\lambda} I(x_1 - x_2) = 0,$$

here x_1 and x_2 consist of $x = (x_1, x_2)$. if one of $A^{(1)}$ and $A^{(2)}$ has linearly independent columns, means $x_1 = x_2 = 0$, *i.e.*, A has linearly independent columns.

Now we suppose one of $A^{(1)}$ and $A^{(2)}$ has linearly independent columns, means A has linearly independent colums, therefore

$$\hat{\theta} = A^{\dagger} b = (A^T A)^{-1} A^T b.$$

expand $A^T A$ as

$$A^T A = \begin{bmatrix} (A^{(1)})^T & 0 & \sqrt{\lambda} I \\ 0 & (A^{(2)})^T & -\sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} A^{(1)} & 0 \\ 0 & A^{(2)} \\ \sqrt{\lambda} I & -\sqrt{\lambda} I \end{bmatrix} = \begin{bmatrix} (A^{(1)})^T A^{(1)} + \lambda I & -\lambda I \\ -\lambda I & (A^{(2)})^T A^{(2)} + \lambda I \end{bmatrix},$$

expand $A^T b$ as

$$A^T b = \begin{bmatrix} (A^{(1)})^T & 0 & \sqrt{\lambda} I \\ 0 & (A^{(2)})^T & -\sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ 0 \end{bmatrix} = \begin{bmatrix} (A^{(1)})^T y^{(1)} \\ (A^{(2)})^T y^{(2)} \end{bmatrix}$$

thus

$$\hat{\theta} = \left(\begin{bmatrix} (A^{(1)})^T A^{(1)} + \lambda I & -\lambda I \\ -\lambda I & (A^{(2)})^T A^{(2)} + \lambda I \end{bmatrix} \right)^{-1} \begin{bmatrix} (A^{(1)})^T y^{(1)} \\ (A^{(2)})^T y^{(2)} \end{bmatrix}$$

(b) We can create a object like

$$\left\| A^{(1)}\theta^{(1)} - y^{(1)} \right\|^2 + \left\| A^{(2)}\theta^{(2)} - y^{(2)} \right\|^2 + \left\| A^{(3)}\theta^{(3)} - y^{(3)} \right\|^2 + \lambda(\left\| \theta^{(1)} - \theta^{(2)} \right\|^2 + \left\| \theta^{(2)} - \theta^{(3)} \right\|^2).$$

15.10 Estimating a periodic time series. (See 15.3.2.) Suppose that the T-vector y is a measured time series, and we wish to approximate it with a P-periodic T-vector. For simplelicity, we assume that $T = KP$, where K is an integer. Let $\hat{y}$ be the simple least squares fit, with no regulatization, *i.e.*, the P-periodic vector that minimize $\|\hat{y} - y\|^2$. Show that for $i = 1, \ldots, P-1$, we have

$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} y_{i+(k-1)P}.$$

In other words, each entry of the periodic estimate is the average of the entries of the original vector over the corresponding indices.

Solution:

The periodic vector can be expressed as a P-vector x, therefore

$$\hat{y} = (x, x, \ldots, x),$$

let

$$A = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix},$$

here A is a $T \times P$ matrix, and I is a $¶ \times P$ unit matrix, then

$$\hat{y} = Ax.$$

Our goal is to minimize $\|\hat{y} - y\|^2 = \|Ax - y\|^2$, where we have

$$\hat{x} = A^\dagger y.$$

With

$$A^T A = \begin{bmatrix} I & \cdots & I \end{bmatrix} \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} = KI,$$

we have

$$\hat{x} = (A^T A)^{-1} A^T y = (1/K) A^T y.$$

For $i = 1, \ldots P - 1$,

$$\hat{y}_i = \hat{x}_i = \frac{1}{K} \sum_{k=1}^{K} y_{i+(k-1)P}.$$

15.11 General pseudo-inverse. In chapter 11 we encountered the pseudo-inverse of a tall matrix with linearly independent columns, a wide matrix with linearly independent rows, and a square invertible matrix. In this exercise we describe the pseudo-inverse of a general matrix, *i.e.*, one that does not fit these categories. The general pseudo-inverse can be defined in terms of Tikhonov regularized inversion (see page 317). Let A be any matrix, and $\lambda > 0$. The Tikhonov regularized approximate solution of $Ax = b$, *i.e.*, unique minimizer of $\|Ax - b\|^2 + \lambda \|x\|^2$, is given by $(A^T A + \lambda I)^{-1} A^T b$. The pseudo-inverse of A is defined as

$$A^\dagger = \lim_{\lambda \to 0} (A^T A + \lambda I)^{-1} A^T.$$

In other words, $A^\dagger b$ is the limit of the Tikhonov-regularized approximate solution of $Ax = b$, as the regularization parameter converges to zero. (It can be shown that this limit always exists.) Using the kernel trick

identity (15.10), we can also express the pseudo-inverse as

$$A^\dagger = \lim_{\lambda \to 0} A^T (AA^T + \lambda I)^{-1}.$$

(a) What is the pseudo-inverse of the $m \times n$ zero matrix ?

(b) Suppose A has linearly independent columns. Explain why the limits above reduce to our previous definition, $A^\dagger = (A^T A)^{-1} A^T$.

(c) Suppose A has linearly independent rows. Expalin why the limits above reduce to our previous definition, $A^\dagger = A^T (AA^T)^{-1}$.

Hint. For part (b) and (c), you can use the fact that the matrix inverse is a continuous function, which means that the limit of the inverse of a matrix is the inverse of the limit, provided the limit matrix is invertible.

Solution:

(a) If A is an $m \times n$ zero matrix, then $(A^T A + \lambda I)^{-1} A^T = 0$, which means

$$A^\dagger = \lim_{\lambda \to 0} 0 = 0.$$

(b) If the columns of A are linearly independent, implies that $A^T A$ is invertible, therefore

$$A^\dagger = (\lim_{\lambda \to 0} (A^T A + \lambda I)^{-1}) A^T = (\lim_{\lambda \to 0} (A^T A + \lambda I))^{-1} A^T = (A^T A)^{-1} A^T.$$

(c) If the rows of A are linearly independent, implies that AA^T is invertible, therefore

$$A^\dagger = A^T \lim_{\lambda \to 0} (AA^T + \lambda I)^{-1} = A^T (\lim_{\lambda \to 0} (AA^T + \lambda I))^{-1} = A^T (AA^T)^{-1}.$$

16 Constrained least squares

16.1 Smallest right inverse. Suppose the $m \times n$ matrix A is wide, with linearly independent rows. Its pseudo-inverse $A^\dagger$ is a right inverse of A. In fact, there are many right inverses of A and it turns out that $A^\dagger$ is the smallest one among them, as measured by matrix norm. In orther words, if X satisfies $AX = I$, then $\|X\| > \left\|A^\dagger\right\|$. You will show this in this problem.

(a) Suppose $AX = I$, and let $x_1, \ldots, x_m$ denote the columns of X. Let b_j denote

the jth column of $A^\dagger$. Explain why $\|x_j\|^2 \geq \|b_j\|^2$. *Hint.* Show that $z = b_j$ is the vector of smallest norm that satisfies $Az = e_j$, for $j = 1, \ldots, m$.

(b) Use the inequalities from part (a) to establish $\|X\| \geq \|A^\dagger\|$.

Solution:

(a) Consider the least norm problem:

$$\begin{array}{ll} \text{minimize} & \|z\|^2 \\ \text{subject to} & Az = e_j \end{array}$$

since A has linearly independent rows, this problem has solution

$$\hat{z} = A^\dagger e_j = b_j,$$

due to $Ax_j = e_j$, x_j is also a solution of $Az = e_j$, but $\|x_j\|^2 \geq \|b_j\|^2$.

(b) we have

$$\|X\|^2 = \|x_1\|^2 + \cdots + \|x_m\|^2 \geq \|b_j\|^2 + \cdots + \|b_m\|^2 = \|A^\dagger\|^2,$$

thus

$$\|X\| \geq \|A^\dagger\|.$$

16.2 Matrix least norm problem. The matrix least norm problem is

$$\begin{array}{ll} \text{minimize} & \|X\|^2 \\ \text{subject to} & CX = D \end{array}$$

where the variable to be chosen is the $n \times k$ matrix X; the $p \times n$ matrix C and the $p \times k$ matrix D are given. Show that the solution of this problem is $\hat{X} = C^\dagger D$, assuming the rows of C are linearly independent. *Hint.* Show that we can find the columns of X independently, by solving a least norm problem for each one.

Solution:

Suppose n-vectors $x_1, \ldots, x_k$ denote the columns of X, and p-vectors $d_1, \ldots, d_k$ denote the columns of D. we have

$$\|X\|^2 = \|x_1\|^2 + \cdots + \|x_k\|^2,$$

and $CX = D$ is same with

$$Cx_1 = d_1, \ldots, Cx_k = d_k,$$

our least norm problem can be write as

$$\begin{array}{ll} \text{minimize} & \|x_1\|^2 + \cdots + \|x_k\|^2 \\ \text{subject to} & Cx_1 = d_1, \ldots, Cx_k = d_k, \end{array}$$

which is same to solve k least norm subproblems:

$$\begin{array}{ll} \text{minimize} & \|x_i\|^2 \\ \text{subject to} & Cx_i = d_i. \end{array}$$

We have $\hat{x}_i = C^\dagger d_i$ for $i = 1, \ldots, k$, combine these k equation and written as matrix form, we have

$$\hat{X} = C^\dagger D.$$

16.3 Closet solution to a given point. Suppose the wide matrix A has linearly independent rows. Find an expression for the point x that is closest to a given vector y (*i.e.*, minimizes $\|x - y\|^2$) among all vectors that satisfies $Ax = b$.

Remark. This problem comes up when x is some set of inputs to be found, $Ax = b$ represents some set of requirements, and y is some nominal value of the input. For example, when the input represent actions that are re-calculated each day, (say, because b changes every day), y might be yesterday's action, and the today's action x found as above gives the least change from yesterday's action, subject to meeting today's requirements.

Solution:

The KKT equation for this problem reduces to

$$\begin{bmatrix} 2I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} 2y \\ b \end{bmatrix},$$

as the stacked matrix $\begin{bmatrix} I \\ A \end{bmatrix}$ always has linearly independent columns, and A has linearly independent rows, therefore the matrix in the

left-handside of above equation is invertible. The first block row of this equation is $2\hat{x} + A^T\hat{z} = 2y$, therefore

$$\hat{x} = y - (1/2)A^T\hat{z},$$

we substitute this into the second block row, $A\hat{x} = b$, to obtain

$$Ay - (1/2)AA^T\hat{z} = b,$$

since rows of A are linearly independent, AA^T is invertible, implies

$$\hat{z} = 2(AA^T)^{-1}(Ay - b),$$

substitute this expression for $\hat{z}$ into the formula for $\hat{x}$ above to get

$$\hat{x} = y - A^T(AA^T)^{-1}(Ay - b) = y - A^\dagger(Ay - b).$$

16.4 Nearest vector with a given average. Let a be an n-vector an β a scalar. How would you find the n-vector x that is closest to a among all n-vectors that have average value β? Give a formula for x and describe it in English.

Solution:

Let $A = \begin{bmatrix} 1/n & \cdots & 1/n \end{bmatrix}$ denote a $1\times n$ matrix, the problem can be expressed as to solve the norm least problem:

$$\begin{aligned} \text{minimize} \quad & \|x - a\|^2 \\ \text{subjecto to} \quad & Ax = \beta, \end{aligned}$$

since one-row matrix A has linearly independent rows, according to exercise 16.3, we have

$$\hat{x} = a - A^T(AA^T)^{-1}(Aa - \beta).$$

Since $AA^T = 1/n$, therefore $A^T(AA^T)^{-1} = \mathbf{1}$, we get

$$hat x = a - \mathbf{1}Aa + \mathbf{1}\beta,$$

we have $a - \mathbf{1}Aa = a - \mathbf{avg}(a)\mathbf{1} = \tilde{a}$, where $\tilde{a}$ is the de-meaned vector of a, finally we have

$$\hat{x} = \tilde{a} - \beta\mathbf{1}.$$

The entries of $\hat{x}$ is equal to the corresponded entry of a minus the average of a and plus β:

$$\hat{x}_i = a_i - \mathbf{avg}(a) + \beta.$$

16.5 Checking constrained least squares solution. Generate a random 20×10 matrix A and a random 5×10 matrix C. Then generate random vector b and d of appropriate dimensions for the constrained least squares problem

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|^2 \\ \text{subject to} & Cx = d \end{array}$$

Compute the solution $\hat{x}$ by forming and solving the KKT equations. Verify that the constrained very nearly hold, *i.e.*, $C\hat{x} - d$ is small. Find the least norm solution x^{ln} of $Cx = d$. The vector x^{ln} also satisfies $Cx = d$ (very nearly). Verify that $\left\|Ax^{\text{ln}} - b\right\|^2 > \|A\hat{x} - b\|^2$.

Solution:

If easy to know b is a 20-vector and d is a 5-vector, when C has linearly independent rows, which is a condition that KKT coefficient matrix is invertible, that $x^{\text{ln}} = C^{\dagger}d$ is a solution that has $Cx^{\text{ln}} - d \approx 0$, after implement we have

$$\left\|Ax^{\text{ln}} - b\right\| \approx 2.155, \qquad \|A\hat{x} - b\| \approx 1.584,$$

we conclude that $\left\|Ax^{\text{ln}} - b\right\| > \|A\hat{x} - b\|$

16.6 Modifying a diet to meet nutrient requirements. (Continuation of exercise 8.9.) The current daily diet is specified by the n-vector d^{curr}. Explain how to find the closest diet d^{mod} to d^{curr} that satisfies the nutrient requirements given by the m-vector n^{des}, and has the same cost as the current diet d^{curr}.

Solution:

According to the solution of exercise 8.9, we know the cost of current diet d^{curr} is $c^T d^{\text{curr}}$. Our problem can be expressed as to solve a least norm problem:

$$\begin{array}{ll} \text{minimize} & \|d - d^{\text{ccur}}\|^2 \\ \text{subject to} & \begin{bmatrix} N \\ c^T \end{bmatrix} d = \begin{bmatrix} n^{\text{des}} \\ c^T d^{\text{curr}} \end{bmatrix} \end{array}$$

which can be solve by KKT equation, and we have $d^{\text{mod}} = \hat{d}$, where $\hat{d}$ is the solution of the above least norm problem.

16.7 Minimum cost trading to achieve target sector exposures. A current portfolio is given by the n-vector h^{curr}, with the entries giving the dollar value invested in the n assets. The total value (or net asset value) of the portfolio is $\mathbf{1}^T h^{\text{curr}}$. We seek a new portfolio, given by the n-vector h, with the same total value as h^{curr}. The difference $h - h^{\text{curr}}$ is called the *trade vector*; it gives the amount of each asset (in dollars) that we buy or sell. The n assets are divided into m industry sectors, such as pharmaceuticals or consumer electronics. We let the m-vector s denote the (dollar value) sector exposures to the m sectors. (See exercise 8.13.) These are given by $s = Sh$, where S is the $m \times n$ sector exposure matrix defined by $S_{ij} = 1$ if asset j is in sector i and $S_{ij} = 0$ is asset j is not in sector i. The new portfolio must have a given sector exposure s^{des}. (The given sector exposures are based on forecast of whether companies in the different sectors will do well or poorly in the future.)

Among all portfolio that have the same value as our current portfolio and achieve the desired exposures, we wish to minimize the trading cost, given by

$$\sum_{i=1}^{n} \kappa_i (h_i - h_i^{\text{curr}})^2,$$

a weighted sum of squares of the asset trades. The weights κ_i are positive. (These depend on the daily trading volumes of assets, as well as other quantities. In general, it is cheaper to trade assets that have high trading volumes.)

Explain how to find h using constrained least squares. Give the KKT

equations that you would solve to find h.

Solution:

The objective $\sum_{i=1}^n \kappa_i (h_i - h_i^{\text{curr}})^2$ can be formed as

$$\sum_{i=1}^n \kappa_i (h_i - h_i^{\text{curr}})^2 = \sum_{i=1}^n (\sqrt{\kappa_i} h_i - \sqrt{\kappa_i} h_i^{\text{curr}})^2 = \|Kh - Kh^{\text{curr}}\|^2,$$

where $n \times n$ matrix K is the diagonal matrix if $\sqrt{\kappa_1}, \ldots, \sqrt{\kappa_n}$, it is easy to know that $K^T K = \mathbf{diag}(\kappa)$. Denote $n \times n$ matrix A and n-vector b as

$$A = K, \qquad b = Kh^{\text{curr}},$$

the objective can be expressed as

$$\|Ah - b\|^2.$$

The constraints are

$$\mathbf{1}^T h = \mathbf{1}^T h^{\text{curr}}, \qquad Sh = s^{\text{des}},$$

which can be formed as

$$\begin{bmatrix} \mathbf{1}^T \\ S \end{bmatrix} h = \begin{bmatrix} \mathbf{1}^T h^{\text{curr}} \\ s^{\text{des}} \end{bmatrix},$$

denote $(m+1) \times n$ matrix C and $(m+1)$-vector d as

$$C = \begin{bmatrix} \mathbf{1}^T \\ S \end{bmatrix}, \qquad d = \begin{bmatrix} \mathbf{1}^T h^{\text{curr}} \\ s^{\text{des}} \end{bmatrix},$$

the constraints can be formed as

$$Ch = d,$$

therefore we can expressed the problem as a constrained least squares problem,

$$\begin{aligned} &\text{minimize} && \|Ah - b\|^2 \\ &\text{subject to} && Ch = d. \end{aligned}$$

The KKT equations of this problem is

$$\begin{bmatrix} 2\mathbf{diag}(\kappa) & \mathbf{1} & S^T \\ \mathbf{1}^T & 0 & 0 \\ S & 0 & 0 \end{bmatrix} \begin{bmatrix} h \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 2\mathbf{diag}\kappa h^{\text{curr}} \\ \mathbf{1}^T h^{\text{curr}} \\ s^{\text{des}} \end{bmatrix},$$

where scalar z_1 and n-vector z_2 are Langrage multipliers for the

constraints equalities.

16.8 minimum energy regulator. We consider a linear dynamical system with dynamics $x_{t+1} = Ax_t + bu_t$, where the n-vector x_t is the state at time t and the m-vector u_t is the input at time t. We assume that $x = 0$ represents the desired operating point; the goal is to find an input sequence that takes the state to the desired operating point at time T is called *regulation*.

Find an explicit formula for the sequence of inputs that yields regulation, and minimizes $\|u_1\|^2 + \cdots + \|u_{T-1}\|^2$, in terms of A, B, T, and x_1. This sequence of input is called the *minimum energy regulator*.

Hint. Express x_T in terms of x_1, A, the *controllability matrix*

$$C = \begin{bmatrix} A^{T-2}B & A^{T-3}B & \cdots & AB & B \end{bmatrix},$$

and $(u_1, u_2, \ldots, u_{T-1})$ (which is the input sequence stacked). You may assume that C is wide and has linearly independent rows.

Solution:

Let $u = (u_1, u_2, \ldots, u_{T-1})$, we have

$$x_T = Cu + A^{T-1}x_1,$$

with $x_T = 0$, the problem is same as the least norm squares problem:

$$\begin{array}{ll} \text{minimize} & \|u\|^2 \\ \text{subject to} & Cu + A^{T-1}x_1 = 0 \end{array}$$

the solution of this problem is

$$hatu = -C^{\dagger}A^{T-1}x_1.$$

16.9 Smoothest force sequence to move a mass. We consider the same setup as the example given on page 343, where the 10-vector f represents a sequence of forces applied to a unit mass over 10 1-second intervals. As in the example, we wish to find a force sequence f that achieves zero final velocity and final position one. In the example on page 343, we choose the smallest f, as measured by its norm (squared). Here, though,

we want the *smoothest* force sequence, *i.e.*, the one that minimizes

$$f_1^2 + (f_2 - f_1)^2 + \cdots + (f_{10} - f_9)^2 + f_{10}^2.$$

(This is the sum of the sequence of differences, assuming that $f_0 = 0$ and $f_{11} = 0$.) Explain how to find this force sequence. Plot it, and give a brief comparision with the force sequence found in the example on page 343.

Solution:

According to exercise 2.3, we know the final velocity and position are given by

$$v^{\text{fin}} = f_1 + f_2 + \cdots + f_{10}$$
$$p^{\text{fin}} = (19/2)f_1 + (17/2)f_2 + \cdots + (1/2)f_{10}.$$

And let A be the 11×10 different matrix

$$A = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & 1 \end{bmatrix}$$

with which we have

$$f_1^2 + (f_2 - f_1)^2 + \cdots + (f_{10} - f_9)^2 + f_{10}^2 = \|Af\|^2 .$$

Thus, the problem can be expressed as a constrined least squares problem :

$$\begin{array}{rl} \text{minimize} & \|Af\|^2 \\ \text{subject to} & \begin{bmatrix} 1 & 1 & \dots & 1 \\ 19/2 & 17/2 & \dots & 1/2 \end{bmatrix} f = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{array}$$

Denote 2×10 matrix C as

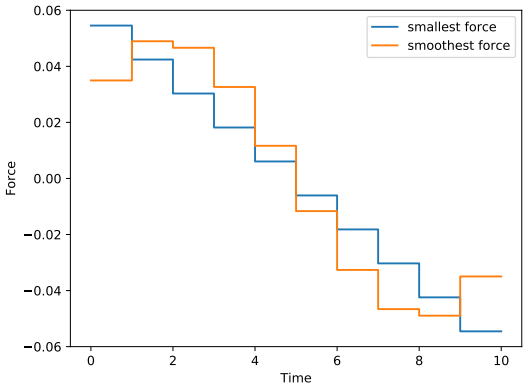
$$C = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 19/2 & 17/2 & \dots & 1/2 \end{bmatrix},$$

the KKT equation can be formed as

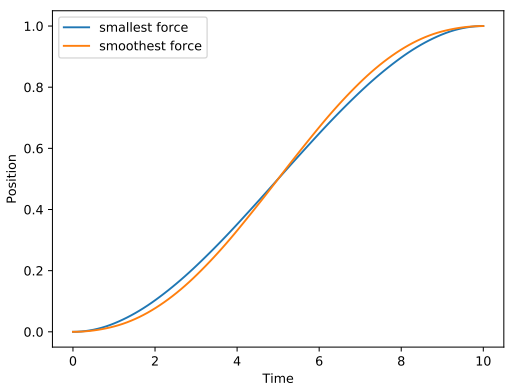
$$\begin{bmatrix} 2A^TA & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{f} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

be careful that the 0 in the right-handside term is an 11-vector with all entries equals to zero.

The Force sequence



The result position



16.10 Smallest force sequence to move a mass to a given position. We consider the same setup as the example given on page 343, where the 10-vector f represents a sequence of forces applied to a unit mass over 10 1-second intervals. In that example the goal is to find the smallest force sequnce (measured by $\|f\|^2$) that acheives zero final velocity and final position one. Here we ask, what is the smallest force sequence that acheives final position one? (We impose no condition on the final velocity.) Explain how to find this force sequence. Compare it to the force sequence found in the example, and give a brief intuitive explaination of the difference. *Remark.* Problems in which the final position of an object is specified, but the final velocity doesn't matter, generally arise in applications that are not socially positive, for example control of missiles.

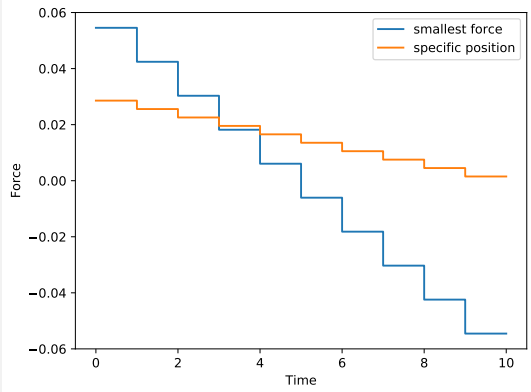
Solution:

Here the problem can be expressed as the least norm problem:

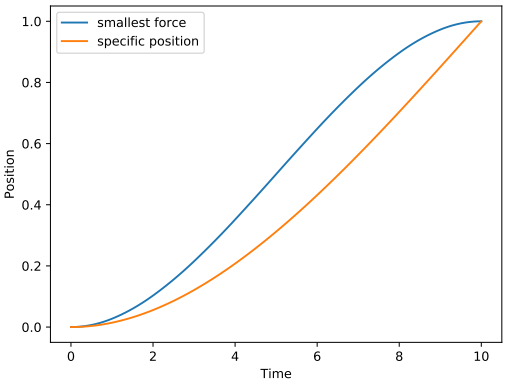
$$\begin{array}{ll} \text{minimize} & \|f\|^2 \\ \text{subject to} & \begin{bmatrix} 19/2 & 17/2 & \cdots & 1/2 \end{bmatrix} f = 1 \end{array}$$

use KKT equation to solve this problem, and plot it with the result in example, we have

The Force sequence

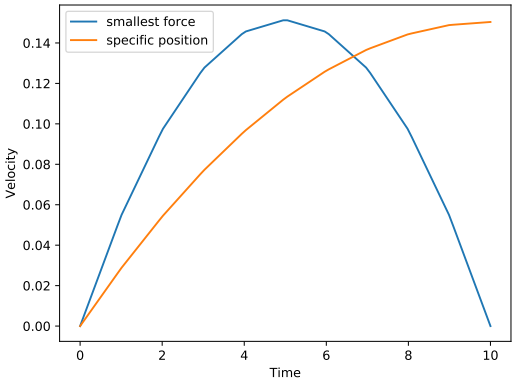


The result position



and the velocity is

The result velocity



We can find that in the example there are negative force which will imply the velocity to decrese to zero after the velocity increased, but in this problem as we don't consider about the final velocity, therefore the velocity is increasing with time, and the norm of example is 0.11, the norm of forces of this problem is 0.05 is more smaller.

16.11 Least distance problem. A variation on the least norm problem (16.2) is the least distance problem,

$$\begin{array}{ll} \text{minimize} & \|x-a\|^2 \\ \text{subject to} & Cx = d \end{array}$$

where the n-vector x is to be determined, the n-vector a is given, the $p \times n$ matrix C is given, and the p-vector d is given. Show that the solution of this problem is

$$\hat{x} = a - C^{\dagger}(Ca - d),$$

assuming the rows of C are linearly independent. *Hint.* You can argue directly from the KKT equations for the least distance problem, or solve for variable $y = x - a$ instead of x.

Solution:

The KKT euqations is

$$\begin{bmatrix} 2I & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} 2a \\ d \end{bmatrix},$$

the top block equation is

$$2\hat{x} + C^T \hat{z} = 2a,$$

from which we can get

$$\hat{x} = a - (1/2)C^T \hat{z},$$

substitute for $\hat{x}$ in the below block equation $C\hat{x} = d$, we have

$$Ca - (1/2)CC^T \hat{z} = d,$$

as the rows of C are linearly independent, implies CC^T is invertible, therefore

$$\hat{z} = 2(CC^T)^{-1}(Ca - d),$$

substitute this in the expression for $\hat{x}$, we have

$$\hat{x} = a - C^T(CC^T)^{-1}(Ca - d),$$

as $C^T(CC^T)^{-1}$ is the pseudo-inverse of C, we can form the expression as

$$\hat{x} = a - C^{\dagger}(Ca - d).$$

16.12 Least norm polynomial interpolation. (Continuation of exercise 8.7.) Find the polynomial of degree 4 that satisfies the interpolation conditions given in exercise 8.7, and minimize the sum of the squares of its coefficients. Plot it, to verify that if satisfies the interpolation conditions.

Solution:

This problem can be expressed as a least norm problem,

$$\begin{array}{ll} \text{minimize} & \|c\|^2 \\ \text{subject to} & Ac = b, \end{array}$$

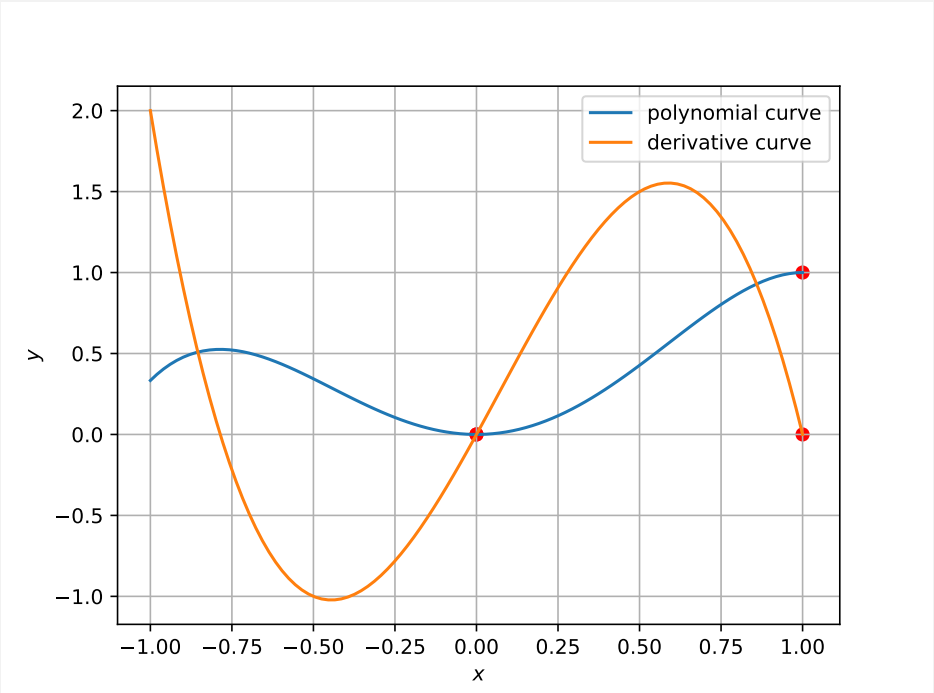
here c is a 5-vector obtained from coefficients, and A is a 4×5 matrix, b is a 4-vector defined in the solution of exercise 8.7. When the rows of A are linearly independent, the solution is

$$\hat{c} = A^{\dagger} b.$$

Solve with specified values of A and b given in exercise 8.7, we have

c_1	c_2	c_3	c_4	c_5
0	0	1.83	0.33	−1.17

The plot of the polynomial curve and its derivative is



16.13 Steganography via least norm. In steganography, a secret message is embedded in an image in such a way that the image looks the same, but an accomplice can decode the message. In this exercise we explore a simple

approach to steganography that relies on constrained least squares. The secret message is given by a k-vector s with entries that all either $+1$ or -1 (*i.e.*, it is a Boolean vector). The original image is given by the n-vector x, where n is usually much than k. We send (or publish or transimit) the modified message $x + z$, where z is an n-vector of modifications. We would like z to be small, so that the original image x and the modified one $x + z$ look (almost) the same. Our accomplice decodes the message s by multiplying the modified image by a $k \times n$ matrix D, which yields the k-vector $y = D(x + z)$. The message is then decoded as $\hat{s} = \mathbf{sign}(y)$. (We write $\hat{s}$ to show that it is an estimate, and might not be the same as the original.) The matrix D must have linearly independent rows, but otherwise is arbitrary.

1. *Encoding via least norm.* Let α to be a positive constant. We choose z to minimize $\|z\|^2$ subject to $D(x + z) = \alpha s$. (This guarantees that the decoded message is correct, *i.e.*, $\hat{s} = s$.) Give a formula for z in terms of $D^\dagger$, α, and x.

2. *Complexity.* What is the complexity of encoding a secret message in an image? (You can assume that $D^\dagger$ is already computed and saved.) What is the complexity of decoding the secret message? About how long would each if these take with a computer capable of carrying out 1 Gflop./s, for $k = 128$ and $n = 512^2 = 262144$ (a 512×512 image)?

3. *Try it out.* Choose an image x, with entries between 0 (black) and 1 (white), and a secret message s with k small compared to n, for example, $k = 128$ for a 512×512 image. (This corresponds to 16 bytes, which can encode 16 characters, *i.e.*, letters, numbers, or punctuation marks.) Choose the entries of D randomly, and compute $D^\dagger$. The modified image $x + z$ may have entries outside the range $[0, 1]$. We replace any negative values in the modified image with zero, and any values greater than one with one. Adjust α until the original and modified images look the same, but the secret message is still decode correctly. (If α is too small, the clipping of the modified image values, ot the round-off errors that occur in the computations, can lead to decoding error, *i.e.*, $\hat{s} \neq s$. If α is too large, the modification will be visually apparent.) Once you've chosen α, send several different secret messages embedded in several different original images.

Solution:

(a) The problem here can be expressed as

$$\begin{aligned} \text{minimize} \quad & \|z\|^2 \\ \text{subject to} \quad & Dz = \alpha s - Dx \end{aligned}$$

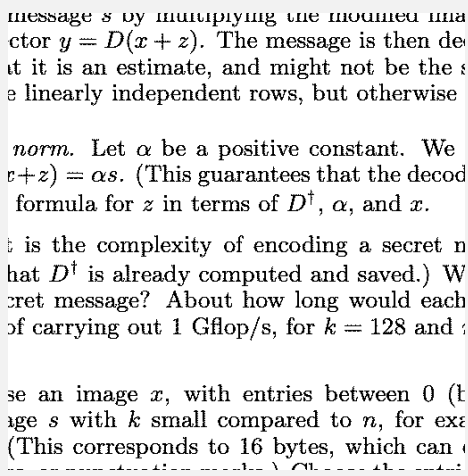
As the rows of D are linearly independent, we have

$$\hat{z} = D^{\dagger}(\alpha s - Dx).$$

(b) The complexity of Dx is $2kn$ flops, and k flops for αs, then k flops for the subtraction $\alpha s - Dx$, at last, we multiply $D^{\dagger}$ and $\alpha s - Dx$, which need $2kn$ flops, thus, total complexity is $4kn + 2k$, as $k << n$, we can write the complexity as $4kn$. It will take 0.13 second to solve the specific case.

(c) I use a clipping of these question to be the image x, and with the message *"I love you"*, padding the space at the tail of the string and change it to a 16-length string, and encode each characters with 8 bytes, we can get 128-vector represents the string. I choose $\alpha = 100, 1000, 10000$. For $\alpha = 100$, get the message **"fnêQRc¢Å¸MBdb!"** which was wrong, and with $\alpha = 1000$ and 10000, we can get *"I love you"* correctly, the image are listed below:

Original Image

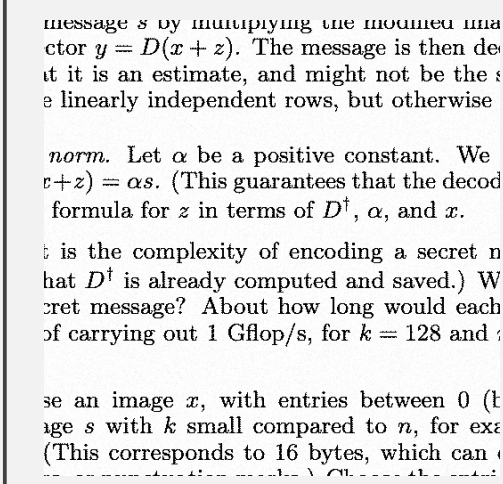

message s by multiplying the modified ima
ctor $y = D(x + z)$. The message is then de
ut it is an estimate, and might not be the
e linearly independent rows, but otherwise

norm. Let α be a positive constant. We
$+z) = \alpha s$. (This guarantees that the decod
formula for z in terms of $D^{\dagger}$, α, and x.

is the complexity of encoding a secret n
hat $D^{\dagger}$ is already computed and saved.) W
cret message? About how long would each
of carrying out 1 Gflop/s, for $k = 128$ and

se an image x, with entries between 0 (
age s with k small compared to n, for exa
(This corresponds to 16 bytes, which can

Steganography with $\alpha = 10^2$

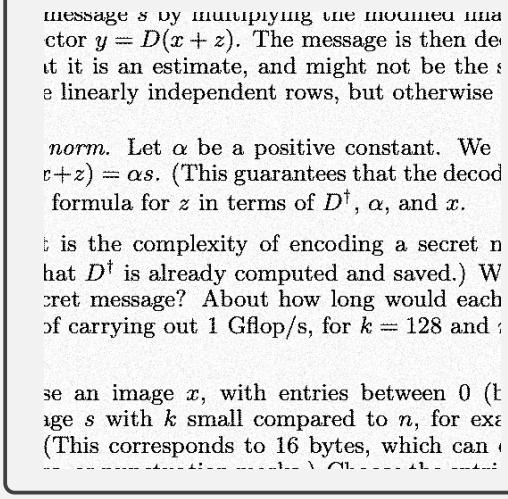

message s by multiplying the modified ima
ctor $y = D(x + z)$. The message is then de
ut it is an estimate, and might not be the
e linearly independent rows, but otherwise

norm. Let α be a positive constant. We
$+z) = \alpha s$. (This guarantees that the decod
formula for z in terms of $D^{\dagger}$, α, and x.

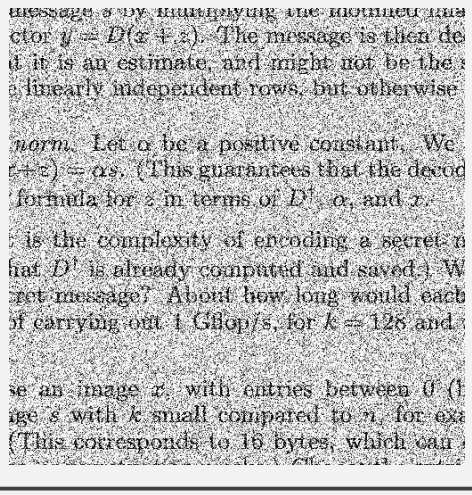
is the complexity of encoding a secret n
hat $D^{\dagger}$ is already computed and saved.) W
cret message? About how long would each
of carrying out 1 Gflop/s, for $k = 128$ and

se an image x, with entries between 0 (
age s with k small compared to n, for exa
(This corresponds to 16 bytes, which can

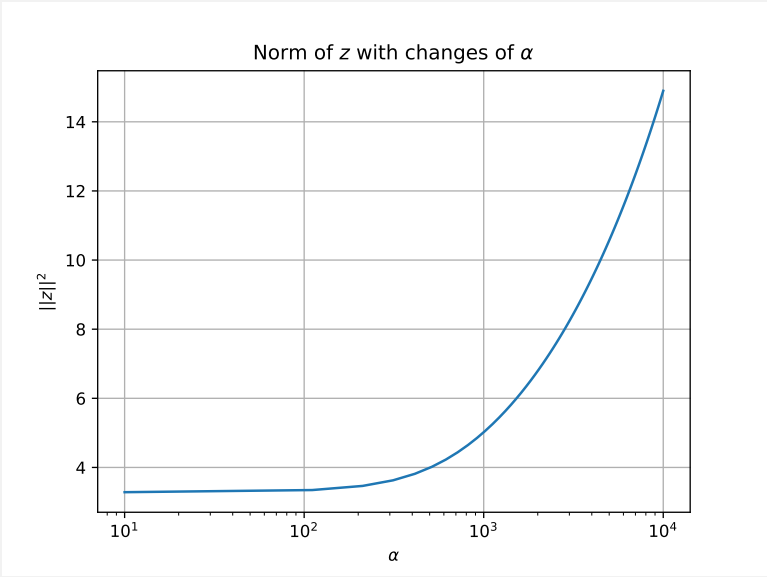
Steganography with $\alpha = 10^3$



Steganography with $\alpha = 10^4$



With which we know even when α get greater, we can get our message from the steganography, but the modification is visually apparent, the change of the $\|z\|$ with α is plot as below:



16.14 Invertibility of matrix in sparse constrained least squares formulation. Show that the $(m+n+p)\times(m+n+p)$ coefficient matrix appearing in equation (16.11) is invertible if and only if the KKT matrix is invertible, *i.e.*, the conditions (16.5) hold.

Solution:

The equation (16.11) is

$$\begin{bmatrix} 0 & A^T & C^T \\ A & -(1/2)I & 0 \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} 0 \\ b \\ d \end{bmatrix}.$$

The conditions (16.5) is

C has linearly independent rows, and $\begin{bmatrix} A \\ C \end{bmatrix}$ has linearly independent columns.

First, we suppose the conditions holds but the coefficient matrix is not invertible, means there is a nonzero vector $(\bar{x}, \bar{y}, \bar{z})$ with

$$\begin{bmatrix} 0 & A^T & C^T \\ A & -(1/2)I & 0 \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

From the center block equation $A\bar{x} - (1/2)\bar{y} = 0$, we have

$$\bar{y} = 2\bar{x},$$

substitute this to the top block equation $A^T\bar{y} + C^T\bar{z} = 0$, we have

$$2A^TA\bar{x} + C^T\bar{z} = 0,$$

multiply it on the left by $\bar{x}^T$ to get

$$2\|A\bar{x}\|^2 + \bar{x}^TC^T\bar{z} = 0,$$

from the below block equation $C\bar{x} = 0$, which is same with $\bar{x}^TC^T = 0$, the euqation above become to $\|A\bar{x}\|^2 = 0$, *i.e.*, $A\bar{x} = 0$, with $C\bar{x} = 0$, we have

$$\begin{bmatrix} A \\ C \end{bmatrix} \bar{x} = 0,$$

since the matrix on the left has linearly independent columns, we conclude that $\bar{x} = 0$. Substitute $\bar{x} = 0$ to the expression for $\bar{y} = 2A\bar{x}$, we have $\bar{y} = 0$, substitute $\bar{y} = 0$ to the top block euqation we have $C^T\bar{z} = 0$, since C^T has linearly independent columns (as assumption), conclude that $\bar{z} = 0$, thus we have

$$(\bar{x}, \bar{y}, \bar{z}) = 0,$$

which is a contradiction. Thus if condition (16.5) holds, the

coefficient matrix of (16.11) is invertible.

The converse is also right, we assume that the coefficient matrix of (16.11) is invertible. First we suppose that the rows of C are linearly dependent, means there is a nonzero vector $\bar{z}$ with

$$C^T \bar{z} = 0,$$

we have

$$\begin{bmatrix} 0 & A^T & C^T \\ A & -(1/2)I & 0 \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \bar{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

means that the coefficient matrix is not invertible, which is contradiction.

Second we suppose that the stacked matrix in conditions (16.5) has linearly dependent columns, means there is a nonzero vector $\bar{x}$ with

$$\begin{bmatrix} A \\ C \end{bmatrix} \bar{x} = 0,$$

implies that

$$\begin{bmatrix} 0 & A^T & C^T \\ A & -(1/2)I & 0 \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{x} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

means that the coefficient matrix is not invertible, which is contradiction, thus, when the coefficient matrix of (16.11) is invertible, the condition 16.5 holds.

Conclude, the coefficient matrix of (16.11) is invertible, if and only if condition 16.5 holds.

16.15 Approximating each column of a matrix as a linear combination of the others. Suppose A is an $m \times n$ matrix with linearly independent columns $a_1, \ldots, a_n$. For each i the problem of finding the linear combination of $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ that is closest to a_i. There are n standard least squares problems, which can be solved by using the methods of chapter 12. In this exercise we explore a simple formula that allows us to solve these n least squares problem all at once. Let $G = A^T A$ denote the Gram matrix, and $H = G^{-1}$ its inverse, with columns $h_1, \ldots, h_n$.

(a) Explain why minimizing $\|Ax^{(i)}\|^2$ subject to $x_i^{(i)} = -1$ solves the problem of finding the linear combination of $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ that closest

to a_i. There are n constrained least squares problems.

(b) Solve the KKT equations for these constrained least squares problems,

$$\begin{bmatrix} 2A^TA & e_i \\ e_i^T & 0 \end{bmatrix} \begin{bmatrix} x^{(i)}, z_i \end{bmatrix} = \begin{bmatrix} 0, -1 \end{bmatrix},$$

to conclude that $x^{(i)} = -(1/H_{ii})h_i$. In words: $x^{(i)}$ is the ith column of $(A^TA)^{-1}$, scaled so its ith entries is -1.

(c) Each of the n original least squares problems has $n-1$ variables, so the complexity is $n(2m(n-1)^2)$ flops, which we can approximate as $2mn^3$ flops. Compare this to the complexity of a method based on the result of part (b): First find the QR factorization of A; then compute H.

(d) Let d_i denote the distance between a_i and the linear combination of the other columns that is closest to it. Show that $d_i = 1/\sqrt{H_{ii}}$.

Remark. When the matrix A is a data matrix, with A_{ij} the value of the jth feature on the ith example, the problem addressed here is the problem of predicting each of the feature from the others. The numbers d_i tells us how well each feature can be predicted from the others.

Solution:

(a) The problem to find the combination of $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ that is closest to a_i can be expressed as to find $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n$ that minimize

$$\|c_1a_1 + \cdots + c_{i-1}a_{i-1} + c_{i+1}a_{i+1} + \cdots + c_na_n - a_i\|^2.$$

Let $x^{(i)} = (c_1, \ldots, c_{i-1}, -1, c_{i+1}, \ldots, c_n)$, the expression above can be formed as

$$\begin{array}{ll} \text{minimize} & \left\|Ax^{(i)}\right\|^2 \\ \text{subject to} & x_i^{(i)} = -1 \end{array}$$

(b) With $x_i^{(i)} = e_i^T x^{(i)}$, we have the KKT equaitons

$$\begin{bmatrix} 2A^TA & e_i \\ e_i^T & 0 \end{bmatrix} \begin{bmatrix} x^{(i)}, z_i \end{bmatrix} = \begin{bmatrix} 0, -1 \end{bmatrix},$$

from the upper block equation $2A^TAx^{(i)} + e_iz_i = 0$, we have

$$x^{(i)} = (1/2)(A^TA)^{-1}e_iz_i = -(1/2)He_iz_i.$$

Substitute this for $x^{(i)}$ in the below block equation $e_i^T x^{(i)} = -1$, we have

$$(1/2)e_i^T H e_i z_i = -1,$$

with

$$e_i^T H e_i = H_{ii}$$

is a scalar, we have

$$z_i = 2/H_{ii},$$

substitute this for z_i in the expression for $x^{(i)}$, we have

$$x^{(i)} = -(1/H_{ii})He_i = -(1/H_{ii})h_i.$$

(c) The complexity to QR factorize A with size $m \times n$ is $2mn^2$ flops, use back-substition to calculate R^{-1} need n^2 flops, as $(A^T A)^{-1} = R^{-1}R^{-T}$, the multiplication of two matrix with size $n \times m$ and $m \times n$ needs $2mn^2$ flops, totally $4mn^2 + n^2$ flops, which has order $4mn^2$, is smaller than the original method with complexity $2mn^3$ flops.

(d) We have

$$d_i^2 = \left\| Ax^{(i)} \right\|^2 = (1/H_{ii}^2) \left\| Ah_i \right\|^2,$$

as we have $\|x\|^2 = x^T x$ for any vector x, therefore

$$\begin{aligned} \|Ah_i\|^2 &= (Ah_i)^T Ah_i \\ &= h_i^T A^T AHe_i \\ &= h_i^T A^T A(A^T A)^{-1} e_i \\ &= h_i^T e_i \\ &= H_{ii}, \end{aligned}$$

thus, we have

$$d_i^2 = (1/H_{ii}^2) \left\| Ah_i \right\| = 1/H_{ii},$$

i.e.,

$$d_i = 1/\sqrt{H_{ii}}.$$

17 Constrained least squares applications

17.1 A variation on the portfolio optimization formulation. Consider the following variation on the linear constrained least squares problem (17.2):

$$\begin{array}{ll} \text{minimize} & \|Rw\|^2 \\ \text{subject to} & \begin{bmatrix} \mathbf{1}^T \\ \mu^T \end{bmatrix} w = \begin{bmatrix} 1 \\ \rho \end{bmatrix}, \end{array}$$

with variable w. (The difference is that here we drop the term $\rho\mathbf{1}$ that appears inside the norm square objective in (17.2).) Show that this problem is equivalent to (17.2). This means w is a solution of (17.12) if and only if it is a solution of (17.2).

Hint. You can argue directly by expanding the objective in (17.2) or via the KKT systems of the two problems.

Solution:

Expand the objective of (17.2) $\left\|Rw - \rho\mathbf{1}^T\right\|^2$ as

$$\left\|Rw - \rho\mathbf{1}^T\right\|^2 = \|Rw\|^2 - 2\rho\mathbf{1}^T Rw + \rho^2\mathbf{1}^T\mathbf{1},$$

Suppose w is a solution of (17.2), with the constrained condition we have

$$\mu^T w = \rho,$$

as

$$\mu = R^T\mathbf{1}/T,$$

we can reform $\mu^T w = \rho$, as

$$\mathbf{1}^T Rw = T\rho,$$

therefore the second term of the objective above is

$$-2\rho\mathbf{1}^T Rw = -2\rho(T\rho) = -2T\rho^2,$$

the third term of the objective above is

$$\rho^2\mathbf{1}^T\mathbf{1} = T\rho^2,$$

therefore objective is

$$\|Rw\|^2 - T\rho^2.$$

As $T\rho^2$ is constant for give data, which meas find w to minimize $\|Rw\|^2 - T\rho^2$ is same to find w ti minimize $\|Rw\|^2$, and can expressed like (17.12).

17.2 A more conventional formulation of the portfolio optimization problem. In this problem we derive an equivalent formulation of the portfilio optimization problem (17.2) that appears more frequently in the literature than our version. (equivalent means that the tow problem always have the same solution.) This formulation is based on the *return covatiance matrix*, which we define below. (See also exercixe 10.16.)

The means of the columns of the asset return R are the entries of the vector μ. The *de-meaned returns matrix* is given by $\tilde{R} = R - \mathbf{1}\mu^T$. (The columns of the matrix $\tilde{R} = R - \mathbf{1}\mu^T$ are the de-meaned return time series for the asset.) The return covariance matrix, traditionally denoted Σ, is its Gram matrix $\Sigma = (1/T)\tilde{R}^T\tilde{R}$.

(a) Show that $\sigma_i = \sqrt{\Sigma_{ij}}$ is the standard deviation (risk) of asset i return. (The symbol σ_i is a traditional one for the standard deviation of asset i.)

(b) Show that the correlation coefficient between asset i and asset j returns is given by $\rho_{ij} = \Sigma_{ij}/(\sigma_i\sigma_j)$. (Assuming neither asset has constant return; if either one does, they are uncorrelated.)

(c) *Portfilio optimization using the return covariance matrix.* Show that the following problem is equivalent to our portfilio optimization problem (17.2):

$$\begin{array}{ll} \text{minimize} & w^T\Sigma w \\ \text{subject to} & \begin{bmatrix} \mathbf{1}^T \\ \mu^T \end{bmatrix} w = \begin{bmatrix} 1 \\ \rho \end{bmatrix}, \end{array}$$

with variable w. This is the form of the portfilio optimization problem that you will find in the literature. *Hint.* Show that the objective is the same as $\left\|\tilde{R}w\right\|^2$, and that this is the same as $\|Rw - \rho\mathbf{1}\|^2$ for any feasible w.

Solution:

Let $r_1,\ldots,r_n$ denote the columns of R, we have

$$mu = (\mathbf{avg}(r_1),\ldots,\mathbf{avg}(r_n)),$$

then $\tilde{r}_i = r_i - \mathbf{1}^T\mathbf{avg}(r_i)$ is ith column of $\tilde{R}$, here $\tilde{r}_i$ denote the de-meaned vector of assert i among T time periods, therefore we can expression Σ as

$$\Sigma = (1/T)\begin{bmatrix} \tilde{r}_1^T\tilde{r}_1 & \ldots & \tilde{r}_1^T\tilde{r}_n \\ \vdots & \ddots & \vdots \\ \tilde{r}_n^T\tilde{r}_1 & \ldots & \tilde{r}_n^T\tilde{r}_n \end{bmatrix}. \tag{17.1}$$

(a) According to the formular (17.1), we have

$$\Sigma_{ii} = \frac{\tilde{r}_i^T\tilde{r}_i}{T} = \mathbf{std}(r_i)^2 = \sigma_i^2,$$

thus,

$$\sigma_i = \sqrt{\Sigma_{ii}}.$$

(b) According to the formula (17.1), we have

$$\Sigma_{ij} = \frac{\tilde{r}_i^T\tilde{r}_i}{T} = \rho_{ij}\sigma_i\sigma_j,$$

thus

$$\rho_{ij} = \Sigma_{ij}/(\sigma_i\sigma_j).$$

(c) For $w^T\sigma w$, we have

$$w^T\Sigma w = w^T\tilde{R}^T\tilde{R}w = \left\|\tilde{R}w\right\|^2,$$

and with

$$\tilde{R}w = (R - \mathbf{1}u^T)w = Rw - \mathbf{1}u^Tw = Rw - \mathbf{1}\rho = Rw - \rho\mathbf{1},$$

which we use the fact that

$$\rho = u^Tw.$$

Thus,

$$w^T\Sigma w = \|Rw - \rho\mathbf{1}\|^2.$$

17.3 A simple portfilio optimization problem.

(a) Find an analytical solution for the portfilio optimization problem with $n = 2$ assets. You can assume that $\mu_1 \neq \mu_2$, *i.e.*, the two assets

have different mean returns. *Hint.* The optimal weights depend only on μ and ρ, and not (directly) on the return matrix R.

(b) Find the conditions under which the optimal portfilio takes long positions in both assets, a short position in one and a long position in the other, or a short position in both assets. You can assume that $\mu_1 < \mu_2$, *i.e.*, asset 2 has the higher return. *Hint.* Your answer should depend on whether $\rho < \mu_1$, $\mu_1 < \rho < \mu_2$, or $\mu_2 < \rho$, *i.e.*, how the required return compares to the tow asset returns.

Solution:

(a) When $n = 2$, *i.e.*, there are two assets, the constraine matrix become 2×2 square matrix, when $\mu_1 \neq \mu_2$, the constrain matrix is invertible, means there is only one feasible w which can be sovle by

$$w = \begin{bmatrix} \mathbf{1}^T \\ \mu^T \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ \rho \end{bmatrix},$$

with

$$\begin{bmatrix} \mathbf{1}^T \\ \mu^T \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 1 \\ \mu_1 & \mu_2 \end{bmatrix}^{-1} = \frac{1}{\mu_2 - \mu_1} \begin{bmatrix} \mu_2 & -1 \\ -\mu_1 & 1 \end{bmatrix},$$

we have

$$w = \frac{1}{\mu_2 - \mu_1} \begin{bmatrix} \mu_2 & -1 \\ -\mu_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \rho \end{bmatrix} = \frac{1}{\mu_2 - \mu_1} \begin{bmatrix} \mu_2 - \rho \\ \rho - \mu_1 \end{bmatrix},$$

i.e.,

$$w_1 = \frac{\mu_2 - \rho}{\mu_2 - \mu_1}, \qquad w_2 = \frac{\rho - \mu_1}{\mu_2 - \mu_1}.$$

(b) According the results we get in part (a),

- If optimal portfilio takes long position in both assets, *i.e.*, $w_1 > 0$ and $w_2 > 0$, implies

$$\mu_1 < \rho < \mu_2.$$

- If optimal portfilio takes long position in one asset and short for another, *i.e.*, $w_1 w_2 < 0$, implies

$$\rho < \mu_1, \text{ or } , \rho > \mu_2.$$

- If optimal portfilio takes short position in both side, *i.e.*, $w_1 < 0$ and $w_2 < 0$, implies

$$\mu_2 < \rho < \mu_1,$$

which make confliction with our assumption $\mu_2 > \mu_1$, *i.e.*, there is no such a ρ.

17.4 Index tracking. Index tracking is a variation on the portfilio optimization problem described in 17.1. As in that problem we choose a portfilio allocation weight vector w that satisfies $\mathbf{1}^T w = 1$. This weight vector gives a portfilio return time series Rw, which is a T-vector. In index tracking, the goal is for this return time series to track (or follow) as closely as possible a given target return time series r^{tar}. We choose w to minimize the RMS deviation between the target return time series r^{tar} and the portfilio return time series r. (Typically the target return is the return of an index, like the Dow Jones Industrial Average or the Russell 3000.) Formulate the index tracking problem as a linearly constrained least squares problem, analogous to (17.2). Give an explicit solution, analogous to (17.3).

Solution:

The problem of index tracking can be expressed as a linearly constrained least squares problem:

$$\begin{array}{ll} \text{minimize} & \|Rw - r^{\mathrm{tar}}\|^2 \\ \text{subject to} & \mathbf{1}^T w = 1 \end{array}$$

which has solution

$$\begin{bmatrix} w \\ z \end{bmatrix} = \begin{bmatrix} 2R^T R & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2R^T r^{\mathrm{tar}} \\ 1 \end{bmatrix},$$

where z is the Lagrange multiplier for the equality constraint.

17.5 Portfilio optimization with market neutral constraint. In the portfilio optimization problem (17.2) the portfilio return time series is the T-vector Rw. Let r^{mkt} denote the T-vector that gives the return of the whole market over the time periods $t = 1, \ldots, T$. (This is the return associated with the total value of the market, *i.e.*, the sum over the assets of asset share price times number of outstanding shares.) A portfilio is said to be *market neutral* if Rw and r^{mkt} are uncorrelated.

Explain how to formulated the portfilio optimization problem, with

the additional constraint of market neutrality, as a constrained least squares problem. Give an explicit solution, analogous to (17.3).

Solution:

As $r = Rw$ and r^{mkt} are uncorrelated, we have

$$(r - \mathbf{avg}(r)\mathbf{1})^T (r^{\mathrm{mkt}} - \mathbf{avg}(r^{\mathrm{mkt}})\mathbf{1}) = 0,$$

and the left term can be expanded as

$$\begin{aligned}(r - \mathbf{avg}(r)\mathbf{1})^T \left(r^{\mathrm{mkt}} - \mathbf{avg}(r^{\mathrm{mkt}})\mathbf{1}\right) &= r^T r^{\mathrm{mkt}} - \mathbf{avg}(r^{\mathrm{mkt}}) r^T \mathbf{1} - \mathbf{avg}(r)\mathbf{1}^T r^{\mathrm{mkt}} + \mathbf{avg}(r)\mathbf{avg}(r^{\mathrm{mkt}})\mathbf{1}^T\mathbf{1} \\ &= r^T r^{\mathrm{mkt}} - T\mathbf{avg}(r)\mathbf{avg}(r^{\mathrm{mkt}}) \\ &= (r^{mkt})^T Rw - \rho \mathbf{1}^T r^{\mathrm{mkt}}\end{aligned}$$

where we use

$$\mathbf{avg}(r^{\mathrm{mkt}}) r^T \mathbf{1} = \mathbf{avg}(r)\mathbf{1}^T r^{\mathrm{mkt}} = \mathbf{avg}(r)\mathbf{avg}(r^{\mathrm{mkt}})\mathbf{1}^T\mathbf{1} = T\mathbf{avg}(r)\mathbf{avg}(r^{\mathrm{mkt}}).$$

The problem above can be expressed as a linearly constrained least squares problem:

$$\begin{array}{ll} \text{minimize} & \|Rw - \rho\mathbf{1}\|^2 \\ \text{subject to} & \begin{bmatrix} 1^T \\ \mu^T \\ (r^{mkt})^T R \end{bmatrix} w = \begin{bmatrix} 1 \\ \rho \\ \rho \mathbf{1}^T r^{\mathrm{mkt}} \end{bmatrix}, \end{array}$$

which has solution

$$\begin{bmatrix} w \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 2R^T R & 1 & \mu & R^T r^{\mathrm{mkt}} \\ 1^T & 0 & 0 & 0 \\ \mu^T & 0 & 0 & 0 \\ (r^{mkt})^T R & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2\rho T\mu \\ 1 \\ \rho \\ \rho \mathbf{1}^T r^{\mathrm{mkt}} \end{bmatrix},$$

where z_1, z_2, and z_3 are Lagrange multipliers for the equality constraints.

17.6 State feedback control of the longitudinal motions of a Boeing 747 aircraft. In this exercise we consider the control of the longitudinal motions of a Boeing 747 aircraft in steady level flight, at an altitude of 40000 ft, and speed 744 ft/s, which is 528 MPH or 460 knots, around Mach 0.8 at that altitude. (Longitudianl means that we consider climb rate and speed, but not turning or rolling motions.) For modest deviations

from these steady state or *trim* conditions, the dynamics is given by the linear dynamical system $x_{t+1} = Ax_t + Bu_t$, with

$$A = \begin{bmatrix} 0.99 & 0.03 & -0.02 & -0.32 \\ 0.01 & 0.47 & 4.70 & 0.00 \\ 0.02 & -0.06 & 0.40 & 0.00 \\ 0.01 & -0.04 & 0.72 & 0.99 \end{bmatrix}, \qquad B = \begin{bmatrix} 0.01 & 0.99 \\ -3.44 & 1.66 \\ -0.83 & 0.44 \\ -0.47 & 0.25 \end{bmatrix},$$

with time unit one second. The state 4-vector x_t consists of deviations from the trim conditions of the following quantities.

- $(x_t)_1$ is the velocity along the airplane body axis, in ft/s, with forward motion positive.
- $(x_t)_2$ is the velocity perpendicular to the body axis, in ft/s, with positive down.
- $(x_t)_3$ is the angle of the body axis above borizontal, in units of 0.01 radian ($0.57°$).
- $(x_t)_4$ is the derivative of the angle of the body axis, called the *pitch rate*, in units of 0.01 radian/s ($0.57°$/s).

The input 2-vector u_t (which we can control) consists of deviations from the trim conditions of the following quantities.

- $(u_t)_1$ is the elevator (control surface) angle, in units of 0.01 radian.
- $(u_t)_2$ is the engine thrust, in units of 10000 lbs.

You do not need to know these details; we mention them only so you know what the entries of x_t and u_t mean.

(a) *Open loop trajectory.* Simulate the motion of the Boeing 747 with initial condition $x_1 = e_4$, in open-loop (*i.e.*, with $u_t = 0$). Plot the state variables over the time interval $t = 1, \ldots, 120$ (two minutes). The oscillation you will see in the open-loop simulation is well known to polits, and called the *phugoid mode*.

(b) *Linear quadratic control.* Solve the linear quadratic control problem with $C = I$, $\rho = 100$, and $T = 100$, with initial state $x_1 = e_4$, and desired terminal state $x^{\text{des}} = 0$. Plot the state and input variables over $t = 1, \ldots, 120$. (For $t = 100, \ldots, 120$, the state and input variables are zero.)

(c) Find the 2×4 state feedback gain K obtained by solving the linear quadratic control problem with $C = I$, $\rho = 100$, $T = 100$, as described in

17.2.3. Verify that it is almost the same as the one obtained with $T = 50$.

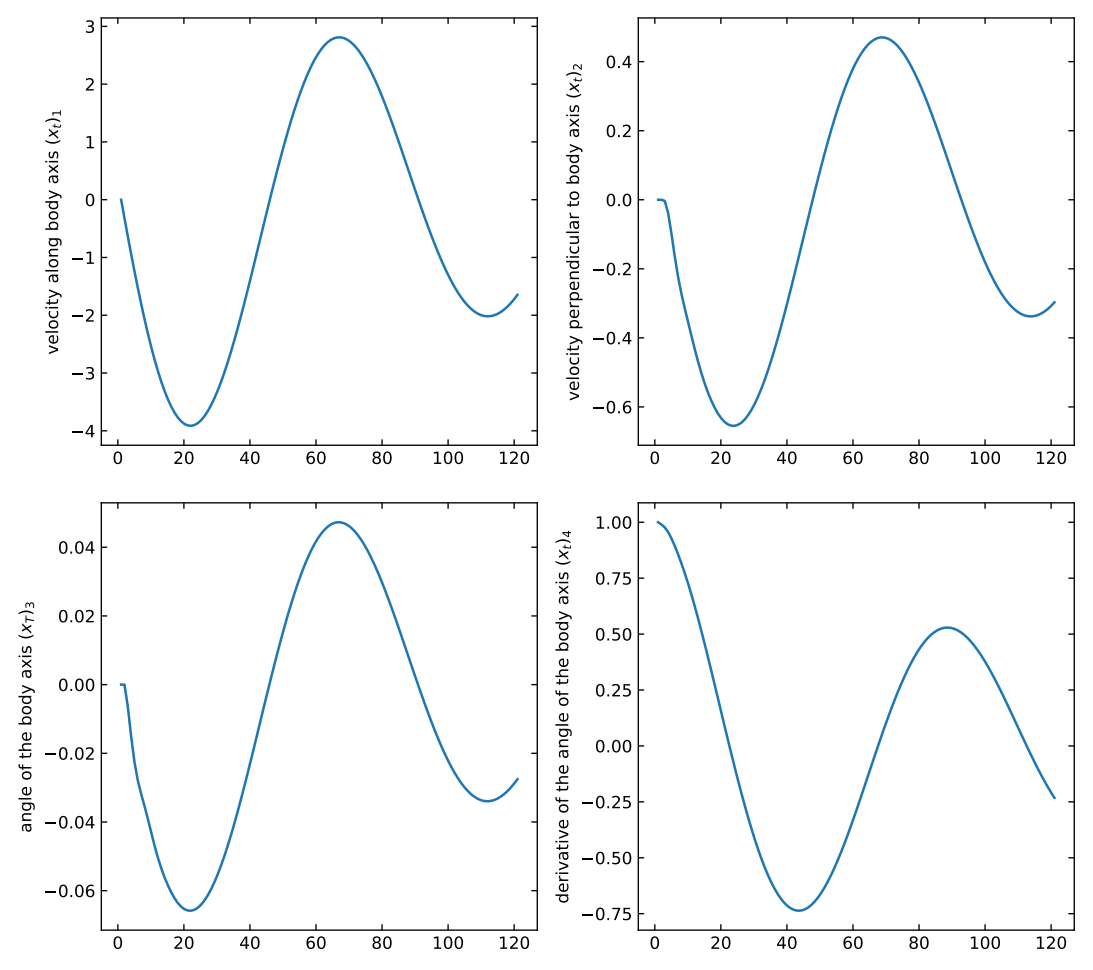
(d) Simulate the motion of Boeing 747 with initial condition $x_1 = e_4$, under state feedback control (*i.e.*, with $u_t = Kx_t$). Plot the state and input variables over the time interval $t = 1, \ldots, 120$.

Solution:

(a) Use the code in listing 6 to get the simulation states of the motion of Boeing 747 in open-loop, and plot them as show below.

Listing 6: Code to simulate with open-loop

```
def simulation_open_loop(A, x0, T):
    '''
    Simulation for a linear dynamical system without control, ie,
    open-loop, in time series t = 1, ..., T.
        x(t+1) = Ax(t)
    --------------------------------
    Parameters:
        A: State change matrix
        x0: initialization state, array like.
        T: float, time for simulation
    '''
    n = len(x0)
    states = np.zeros((T, n))
    states[0] = x0
    for i in range(1, T):
        states[i] = A @ states[i-1].T
    return states
```



(b) The objective has the form $J = J_{\text{output}} + \rho J_{\text{input}}$, where

$$J_{\text{output}} = \sum_{i=1}^{T} \|y_i\|^2 = \sum_{i=1}^{T} \|C_i x_i\|^2 = \sum_{i=1}^{T} \|x\|^2$$
$$J_{\text{input}} = \sum_{i=1}^{T-1} \|u_i\|^2 .$$

The linear quadratic control problem is

$$\begin{array}{ll} \text{minimize} & J_{\text{putput}} + \rho J_{\text{input}} \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t, \qquad t = 1, \ldots, T-1 \\ & x_1 = e_4, \qquad x_T = 0, \end{array}$$

Let

$$z = (x_1, \ldots, x_T, u_1, \ldots, u_{T-1}),$$

the objective $J_{\text{output}} + \rho J_{\text{input}}$ can be expressed as

$$J = \|\tilde{A}z\|^2,$$

where

$$\tilde{A} = \left[\begin{array}{ccc|ccc} I & & & & & \\ & \ddots & & & & \\ & & I & & & \\ \hline & & & \sqrt{\rho}I & & \\ & & & & \ddots & \\ & & & & & \sqrt{\rho}I \end{array}\right],$$

and the constraints can be formed as

$$\tilde{C}z = \tilde{d},$$

where

$$\tilde{C} = \left[\begin{array}{ccccc|ccc} A & -I & & & & B & & \\ & \ddots & \ddots & & & & \ddots & \\ & & & A & -I & & & B \\ \hline I & & & & & & & \\ & & & & I & & & \end{array}\right], \qquad \tilde{d} = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hline e_4 \\ 0 \end{array}\right],$$

therefore the linear quadratic control problem is the constrained least squares problem:

$$\begin{aligned} \text{minimize} \quad & \|\tilde{A}z\|^2 \\ \text{subject to} \quad & \tilde{C}z = d \end{aligned}$$

Code in listing 7 show the function to solve linearly constrained least squares problem,

Listing 7: Code to solve linearly constrained least squares problem

```
def least_square_constrained(A, b, C, d):
    """
    Solve the linear constrained least squares problem:
        minimize    ||Ax - b||^2
        subject to  Cx = d
    let x' = (x, z), z is the Langrange multipliers of constraits equalities
        ,
    problem above is same to solve
        | 2A^TA C^T | | x |   | 2A^Tb |
        |           | |   | = |       |
```

```
        | C     0   | | z |   | d     |
    ---------------------------------------
    Parameters:
    A : matrix
    b : array-like
    C : matrix
    d : array-like
    """
    _, n = A.shape
    p, _ = C.shape

    M = np.block([
        [2 * A.T @ A, C.T],
        [C, zeros((p,p))]
    ])
    N = np.hstack([2 * A.T @ b, d])

    X = pinv(M) @ N
    x, z = X[:n], X[n:]
    return x, z
```

codes in listing 8 and 9 shows how to expand original matrix to get the stack matrix, and solve the quadratic control problem to get the input vectors.

Listing 8: Code to get stacked matrix

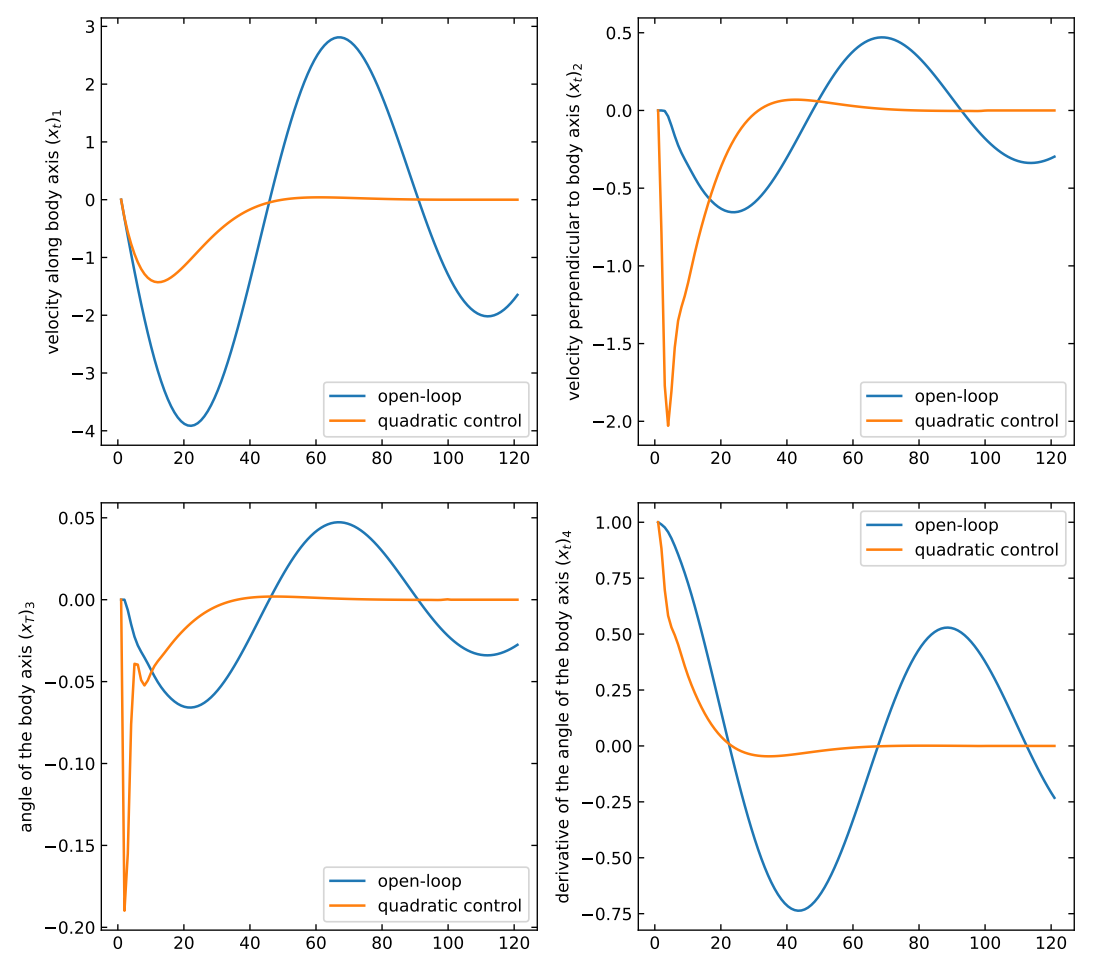
```
def create_diag_matrix(A, T, yshift=None):
    '''
    Create a diagnal matrix with A, as
        A ... ...
        ... A ...
        ... ... A
    ------------------------
    A: matrix
    T: repeat time, int
    '''
    n, m = A.shape
    if not yshift:
        yshift = m

    ans = zeros( (n*T, yshift*(T-1)+m) )
    for i in range(T):
```

```
        ans[i*n:i*n+n, i*yshift:i*yshift+m] = A
    return ans
```

Listing 9: Code to solve the quadratic control problem

```
def quadratic_control(A, B, C, x0, rho=100, T=100):
    n, m, q = A.shape[0], B.shape[1], C.shape[0]
    big_A = np.block([
        [create_diag_matrix(C, T), zeros((T*q, (T-1)*m))],
        [zeros(((T-1)*m, T*n)), sqrt(rho) * np.eye((T-1)*m)]
    ])

    big_C = np.block([
        [ create_diag_matrix(np.hstack([A, -np.eye(n)]), T-1, n),
            create_diag_matrix(B, T-1) ],
        [ np.hstack( [np.eye(n), zeros((n, n*(T-1)))] ), np.zeros((n,m*(T-1)
            )) ],
        [ np.hstack( [zeros((n, n*(T-1))), np.eye(n)] ), np.zeros((n,m*(T-1)
            )) ]
    ])

    d = np.hstack([ zeros(n*(T-1)), x0, zeros(n)])

    X, z = least_square_constrained(big_A, zeros(len(big_A)), big_C, d)
    x, u = X[:n*T].reshape(-1, n), X[n*T:].reshape(-1, m)
    return x, u
```

Solve this and get the input $\hat{u}_1,\ldots,\hat{u}_{T-1}$, simulate and plot as below:



(c) As argued in the book, we know with $x^{\text{des}} = 0$, we have $u_1 = Kx^{\text{init}}$, we choose $x^{\text{init}} = e_i$, means the ith column of K is u_1 we obtained from the solution in part (b). As code showed in listing 10,

Listing 10: Code for estimate K

```
def estimate_K(A, B, C, rho=100, T=100):
    n, m = A.shape[0], B.shape[1]
    K = zeros( (n,m) )
    for (i, x0) in enumerate(np.eye(n)):
        _, u = quadratic_control(A, B, C, x0, rho=rho, T=T)
        K[i] = u[0]
    return K.T
```

implement it with $T = 100$, we have

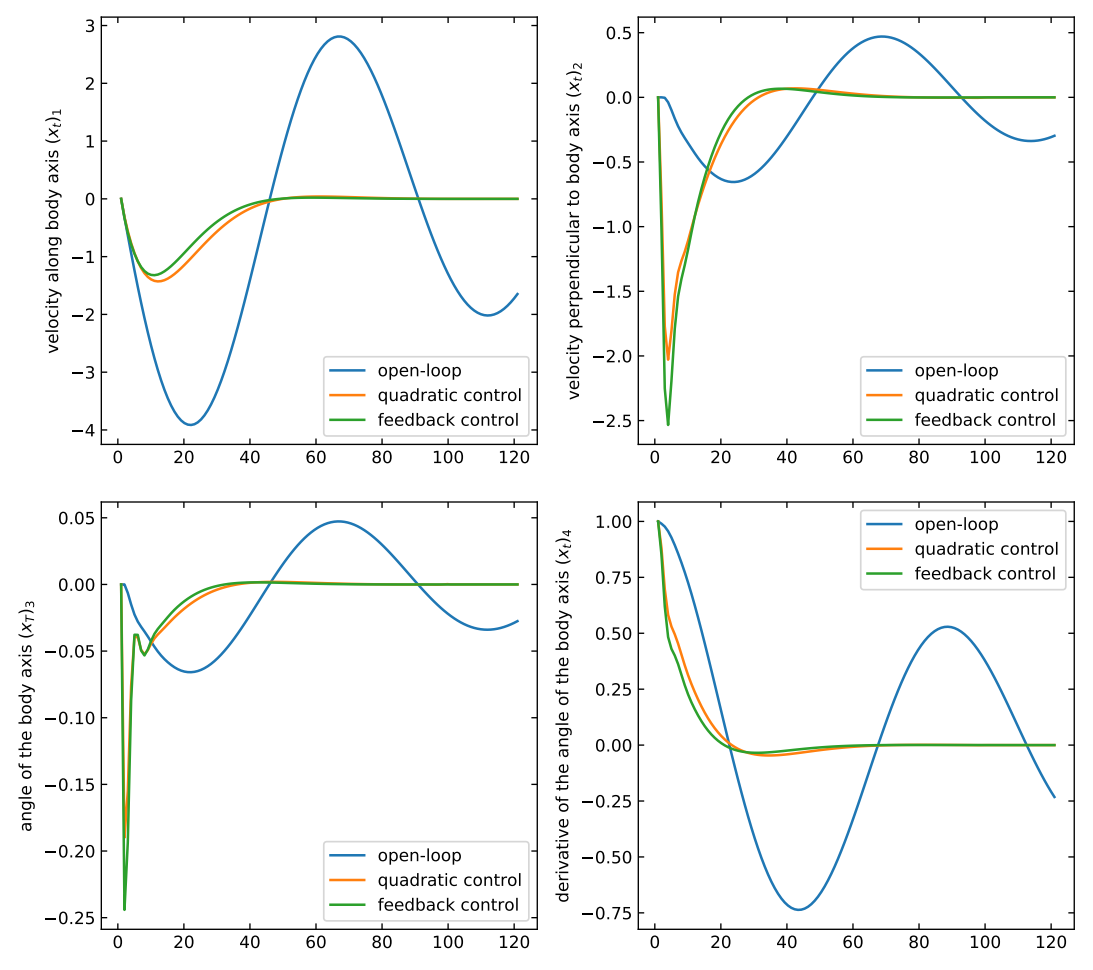
$$K = \begin{bmatrix} -0.0318 & -0.0180 & 0.3071 & 0.2321 \\ -0.0590 & -0.0030 & -0.1185 & 0.0063 \end{bmatrix},$$

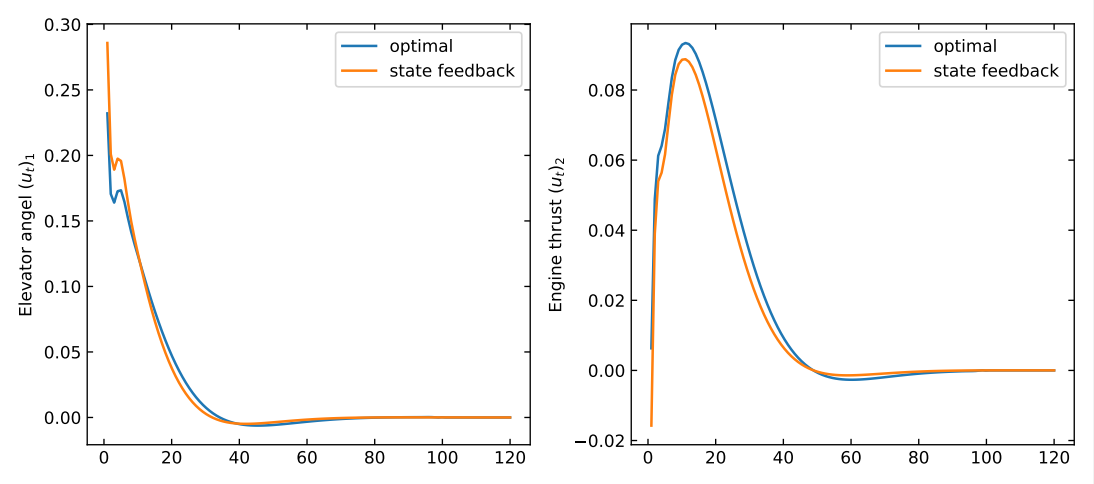
here we round the value to 4 digitals. And for $T = 50$, we have

$$K = \begin{bmatrix} -0.0318 & -0.0180 & 0.3072 & 0.2325 \\ -0.0592 & -0.0030 & -0.1186 & 0.0061 \end{bmatrix},$$

we can find that they are quite similar.

(d) We use K computed in part (c) with $T = 20$ (will make different), and use the feedback gain matrix to simulate, with plots below





we can find out that the feedback gain curve is similar to the quadratic control curve, in fact, if we choose K computed by $T = 100$, the two curve will become same.

17.7 Bio-mass estimation. A bio-reactor is used to grow three different bacteria. We let x_t be the 3-vector of the bio-masses of the three bacteria, at time period (say, hour) t, for $t = 1,\ldots,T$. We believe that they each grow, independently, with growth rates given by the 3-vector r (which has positive entries). This means that $(x_{t+1})_i \approx (1+r_i)(x_t)_i$, for $i = 1,2,3$. (These equations are approximate; the real rate is not constant.) At every time sample we measure the total bio-mass in the reactor, *i.e.*, we have measurements $y_t \approx \mathbf{1}^T x_t$, for $t = 1,\ldots,T$. (The measurements are not exactly equal to the total mass; there are small measurement errors.) We do not know the bio-masses $x_1,\ldots,x_T$, but wish to estimate them based on the measurements $y_1,\ldots,y_T$.

Set this up as a linear quadratic state estimation problem as in 17.3. Identity the matrices A_t, B_t, and C_t. Explain what effect the parameter λ has on the estimated bio-mass trajectory $\hat{x}_1,\ldots,\hat{x}_T$.

Solution:

It is easy to identify that

$$A_t = \mathbf{diag}(r) + I, \qquad B_t = I, \qquad C_t = \mathbf{1}.$$

let w_t to be a 3-vector denote the growth residual, which means

$$x_{t+1} = A_t x_t + B_t w_t.$$

In order to change the estimate problem to constrained least squares problem, we choose two objectives, the primary objectives is the sum of squares of the norm of the measurement residuals,

$$J_{\text{meas}} = \|C_1 x_1 - y_1\|^2 + \cdots + \|C_T x_t - y_t\|^2 .$$

the secondary objectives is the sum of squares of the norm of the grow residuals,

$$J_{\text{grow}} = \|w_1\|^2 + \cdots + \|w_{T-1}\|^2,$$

therefore we can get estimate bio-mass by solve the constrained least squares problem,

$$\begin{array}{ll} \text{minimize} & J_{\text{meas}} + \lambda J_{\text{grow}} \\ \text{subject to} & x_{t+1} = A_t x_t + B_t w_t, \quad t = 1, \ldots, T-1 \end{array}$$

here λ is a positive parameter.

18 Nonlinear least squares

18.1 Lambert W-function. The *Lambert W-function*, denoted $W : [0, \infty) \to \mathrm{R}$, is defined as $W(u) = x$, where x is the unique number $x \geq 0$ for which $xe^x = u$. (The notation just means that we restrict the argument x to be nonnegative.) The lambert function arise in a variety of applications, and it named after the mathematician Johann Heinrich Lambert. There is no analytical formula for $W(u)$; it must be computed numerically. In this exercise you will develop a solver to computer $W(u)$, given a nonnegative number u, using the Levenberg-Marquardt algorithm, by minimizing $f(x)^2$ over x, where $f(x) = xe^x - u$.

(a) Give the Levenberg-Marquardt update (18.13) for f.

(b) Implement the Levenberg-Marquardt algorithm for minimizing $f(x)^2$. You can start with $x^{(1)} = 1$ and $\lambda^{(1)} = 1$ (but it should work with other initializations). You can stop the algorithm when $|f(x)|$ is small,

say, less than 10^{-6}.

Solution:

(a) The update in Levenberg-Marquardt algorithm for f is

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{\lambda^{(k)} + (f'(x^{(k)}))^2} f(x^{(k)}),$$

as

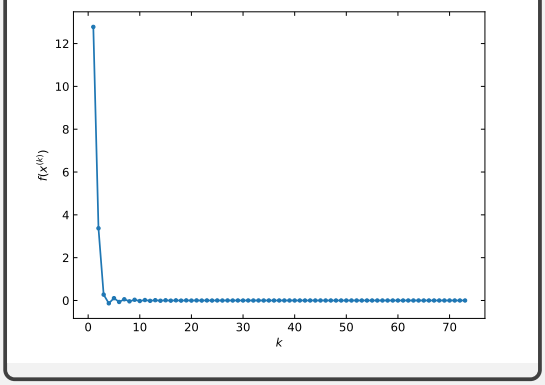
$$f'(x) = e^x + xe^x,$$

we have

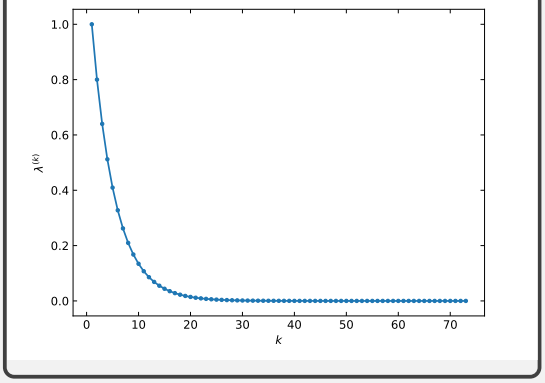
$$x^{(k+1)} = x^{(k)} - \frac{e^{x^{(k)}} + x^{(k)}e^{x^{(k)}}}{\lambda^{(k)} + (e^{x^{(k)}} + x^{(k)}e^{x^{(k)}})^2} x^{(k)}e^{x^{(k)}}$$

(b) At first we consider $u = 2$, we have $x \approx \hat{x} = 0.852606$ (around to 6 digitals), and the values $f(x^{(k)})$ and $\lambda^{(k)}$ versus iteration number k are show as below:

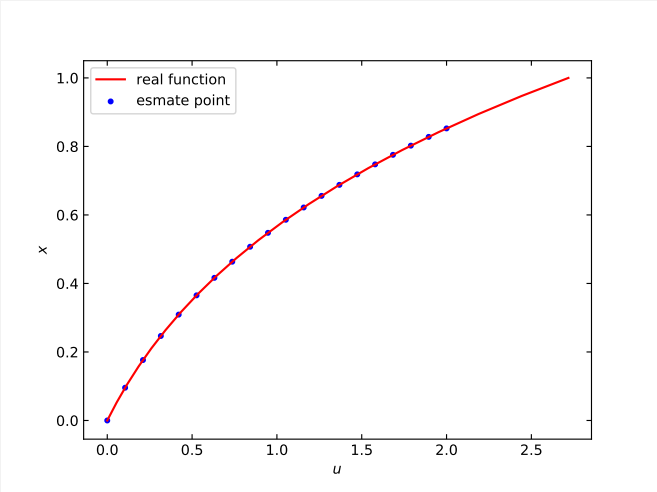
$f(x^{(k)})$ versus iteration k for $u = 2$



$\lambda^{(k)}$ versus iteration k for $u = 2$



Now we choose 20 number of u between $[0,2]$, and plot it versus corresponded x, we have



18.2 Internal rate of return. Let the n-vector c denote a cash flow over n time periods, with positive entries meaning cash received and negative meaning payments. We assume that its NPV (net present value, see page 22) with interest rate $r \geq 0$ is given by

$$N(r) = \sum_{i=1}^{n} \frac{c_i}{(1+r)^{i-1}}.$$

The *internal rate of return* (IRR) of the cash flow is defined as the smallest positive value of r for which $N(r) = 0$. The Levenberg-Marquardt algorithm can be used to compute the IRR of a given cash flow sequence, by minimizing $N(r)^2$ over r.

(a) Work out a specific formula for the Levenberg-Marquardt update for r, *i.e.*, (18.13).

(b) Implement the Levenberg-Marquardt algorithm to find the IRR of the cash flow sequence

$$c = (-\mathbf{1}_3,\ \ 0.3\,\mathbf{1}_5,\ \ 0.6\,\mathbf{1}_6),$$

where the subscripts give the dimensions. (This corresponds to three periods in which you make investments, which pay off at one rate for 5 periods, and a higher rate for the next 6 periods.) You can initialize with $r^{(0)} = 0$, and stop when $N(r^{(k)})^2$ is small. Plot $N(r^{(k)})^2$ versus k.

Solution:

(a) As we know,

$$N'(r) = \sum_{i=1}^{n} \frac{(1-i)c_i}{(1+r)^i},$$

According to formula (18.13):

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{\lambda^{(k)} + (f'(x^{(k)}))^2} f(x^{(k)}),$$

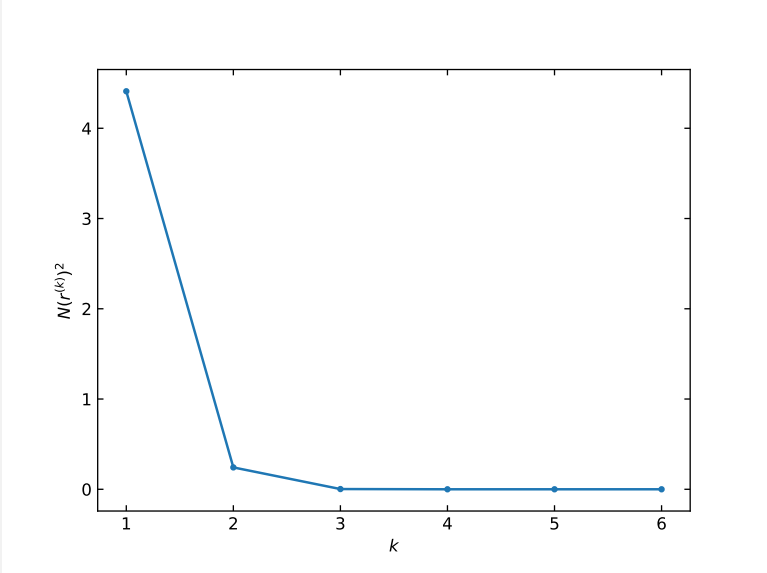
we have

$$r^{(k+1)} = r^{(k)} - \frac{\sum_{i=1}^{n} \frac{(1-i)c_i}{(1+r^{(k)})^i}}{\lambda^{(k)} + \left(\sum_{i=1}^{n} \frac{(1-i)c_i}{(1+r^{(k)})^i}\right)^2} \sum_{i=1}^{n} \frac{c_i}{(1+r^{(k)})^{i-1}}.$$

(b) Implement it with

$$c = (-\mathbf{1}_3,\ 0.3\,\mathbf{1}_5,\ 0.6\,\mathbf{1}_6),$$

we have $\hat{r} \approx 0.072458$, and the plot of $N(r^{(k)})^2$ versus k is showed below:



18.3 A common form for the residual. In many nonlinear least squares problems the residual function $f : \mathrm{R}^n \to \mathrm{R}^m$ has the specific form

$$f_i(x) = \phi_i(a_i^T x - b_i), \qquad i = 1, \ldots, m,$$

where a_i is an n-vector, b_i is a scalar, and $\phi_i : \mathrm{R} \to \mathrm{R}$ is a scalar-valued function of a scalar. In other words, $f_i(x)$ is a scalar function of an

affine function of x. In this case the objective of the nonlinear least squares problem has the form

$$\|f(x)\|^2 = \sum_{i=1}^{m} \left(\phi_i(a_i^T x - b_i)\right)^2.$$

We define the $m \times n$ matrix A to have rows $a_1^T, \ldots, a_m^T$, and the m-vector b to have entries $b_1, \ldots, b_m$. Note that if the function ϕ_i are the identity function, *i.e.*, $\phi_i(u) = u$ for all u, then the objective becomes $\|Ax - b\|^2$, and in this case the nonlinear least squares problem reduces to the linear least quares problem. Show that the derivative matrix $Df(x)$ has the form

$$Df(x) = \mathbf{diag}(d)A,$$

where $d_i = \phi_i'(r_i)$ for $i = 1, \ldots, m$, with $r = Ax - b$.

Remark. This means that in each iteration of the Gauss-Newton method we solve a weighted least squares problem (and in the Levenberg-Marquardt algorithm, a regularized weighted least squares problem); see exercise 12.4. The weights change in each iteration.

Solution:

We know

$$Df(x)_{ij} = \frac{\partial f_i}{\partial x_j}(x) = \frac{\partial \phi_i}{\partial x_j}(a_i^T x - b_i) = \frac{\partial \phi_i}{\partial r_i}\frac{\partial (a_i^T x - b_j)}{\partial x_j} = \phi_i'(r_i)A_{ij} = d_i A_{ij},$$

therefore the ith row of $Df(x)$ is $d_i a_i^T$, which means

$$Df(x) = \begin{bmatrix} d_1 a_1^T \\ \vdots \\ d_m a_m^T \end{bmatrix} = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_m \end{bmatrix} \begin{bmatrix} a_1^T \\ \vdots \\ a_m^T \end{bmatrix} = \mathbf{diag}(d)A$$

18.4 Fitting an exponential to data. Use the Levenberg-Marquardt algorithm to fit an exponential function of the form $\hat{f}(x;\theta) = \theta_1 e^{\theta_2 x}$ to the data

$$0,\ 1,\ \ldots,\ 5 \qquad 5.2,\ 4.5,\ 2.7,\ 2.5,\ 2.1,\ 1.9.$$

(The first list gives $x^{(i)}$; the second list gives $y^{(i)}$.) Plot your model $\hat{f}(x;\hat{\theta})$ versus x, along with the data points.

Solution:

The problem can be formed as a nonlinear least squares problem,

$$\text{minimize} \quad \sum_{i=1}^{6} (\hat{f}(x^{(i)};\theta) - y^{(i)})^2,$$

let $f_i(\theta) = \hat{f}(x^{(i)};\theta) - y^{(i)}$ be the ith component of f, we can for this problem as

$$\text{minimize} \quad \|f(\theta)\|^2 .$$

The gradient of $f_i(theta)$ is

$$\nabla f_i(\theta) = \begin{bmatrix} e^{\theta_2 x^{(i)}} \\ \theta_1 x^{(i)} e^{\theta_2 x^{(i)}} \end{bmatrix},$$

we can get the derivative matrix of f is

$$Df(\theta) = \begin{bmatrix} \nabla f_1(\theta)^T \\ \vdots \\ \nabla f_6(\theta)^T \end{bmatrix},$$

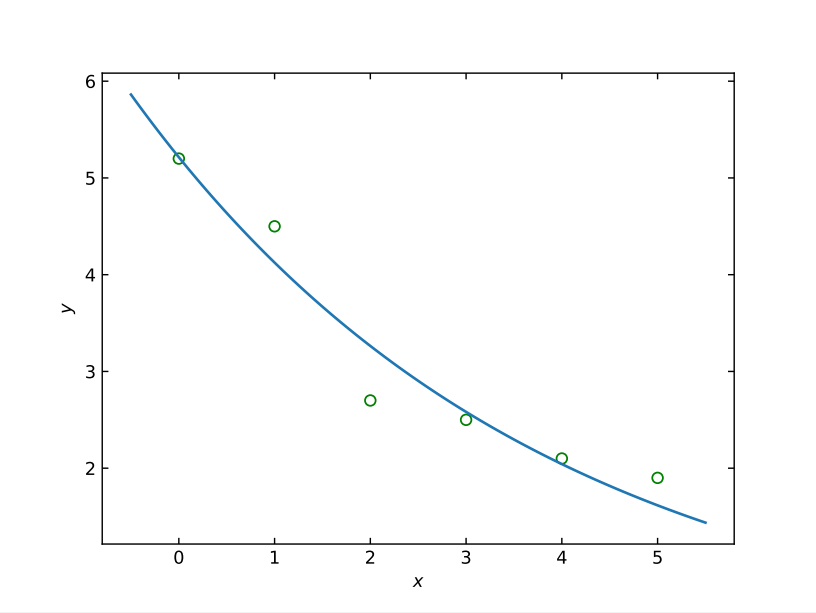
the update of θ is

$$\theta^{(k+1)} = \theta^{(k)} - \left(Df(\theta^{(k)})^T Df(\theta^{(k)}) + \lambda^{(k)} I \right)^{-1} Df(\theta^{(k)})^T f(\theta^{(k)}).$$

We have

$$\hat{\theta}_1 \approx 5.214, \qquad \hat{\theta}_2 \approx -0.234,$$

and the plot with below:



The code is showed as list 11,

Listing 11: code for exercise 18.4

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exerises 18.4
# Author: Utoppia
# Date  : 24 May 2020

import numpy as np
from numpy.linalg import inv, pinv, norm
import matplotlib.pyplot as plt
import math

def f(x, y):
    def wrapper(theta):
        return theta[0] * np.exp(theta[1]*x) - y
    return wrapper

def f_diff(x):
    def wrapper(theta):
        ans = np.zeros((len(x), len(theta)))
        for i in range(len(x)):
            ans[i] = [math.exp(theta[1]*x[i]), x[i]*theta[0]*math.exp(theta[1]*x[
                i])]
```

```
        return ans
    return wrapper


def lambert_marquardt(x_init, f, f_diff, lbd=1, err=10**(-6), nMax = 100):
    """
    Lambert-Marquardt algorithm to solve nonlinear least squares problem,
    for 1 dimension variable
    -------------------
    Parameters:
        x_init : Float. Initialization of the variable with 1 dimension
        f : calable function
        f_diff: calable function, the differential coefficient of f
        lbd: Float. Trust parameter
    """
    x, lbds = [x_init], [lbd]
    counter = 0
    while norm(f(x[counter])) > err and counter < nMax: # iteration
        D = f_diff(x[counter])
        F = f(x[counter]).reshape(-1, 1)

        x_net = x[counter] - inv(D.T.dot(D) + lbds[counter]*np.identity(2)).dot(D
            .T.dot(F)).reshape(1,-1)[0] # update
        if norm(f(x_net)) < norm(f(x[counter])): # judge
            x.append(x_net)
            lbds.append(0.8 * lbds[counter])
        else:
            x.append(x[counter])
            lbds.append(lbds[counter] * 2)

        counter += 1
        print("iteration {}: {}, error: {}".format(counter, x[counter], norm(f(x[
            counter]))))
    return x, lbds, counter

def solve(x, y):
    thetas, lbds, cnt = lambert_marquardt([1,1], f(x, y), f_diff(x))
    theta = thetas[-1]

    fig = plt.figure()
    plt.scatter(x, y, color='g', marker='o', facecolors='none')
```

```
    x = np.linspace(-0.5, 5.5, 100)
    y = theta[0]*np.exp(theta[1]*x)
    plt.plot(x, y)

    plt.xlabel('$x$')
    plt.ylabel('$y$')
    fig.savefig('18-4.pdf')

    print(thetas[-1], lbds[-1])
    plt.show()

def main():
    x = np.linspace(0, 5, 6)
    y = np.array([5.2, 4.5, 2.7, 2.5, 2.1, 1.9])

    solve(x, y)

if __name__ == '__main__':
    main()
```

18.5 Mechanical equilibrium. A mass m, at position given by the 2-vector x, is subject to three forces acting on it. The first force F^{grav} is gravity, which has value $F^{\text{grav}} = -mg(0,1)$, where $g = 9.8$ is the acceleration of gravity. (This force points straight down, with a force that does not depend on the position x.) The mass is attached to two cables, whose other ends are anchored at (2-vector) locations a_1 and a_2. The force F_i apllied to the mass by cable i is given by

$$F_i = T_i(a_i - x)/\|a_i - x\|,$$

where T_i is the cable tension. (This means that each cable applies a force on the mass that points from the mass to the cable anchor, with magnitude given by the tension T_i.) The cable tensions are given by

$$T_i = k\frac{\max\{\|a_i - x\| - L_i, 0\}}{L_i},$$

where k is a position constant, and L_i is the natural or unloaded length of cable i (also positive). In words: The tension is proportional to the fractional stretch of the cable, above its natural length. (The

max appearing in this formula means the tension is not a differentiable function of the position, when $\|a_i - x\| = L_i$, but we will simply ignore this.) The mass is in equilibrium at position x if the three forces acting on it sum to zero,

$$F^{\mathrm{grav}} + F_1 + F_2 = 0.$$

We refer to the left-hand side as the residual force. It is a function of mass position x, and we write it as $f(x)$.

Compute an equilibrium position for

$$a_1 = (3,2), \qquad a_2 = (-1,1), \qquad L_1 = 3, \qquad L_2 = 2, \qquad m = 1, \qquad k = 100,$$

by applying the Levenberg-Marquardt algorithm to the residual force $f(x)$. Use $x^{(1)} = (0,0)$ as starting point. (Note that it is important to start at a point where $T_1 > 0$ and $T_2 > 0$, because otherwise the derivative matrix $Df(x^{(1)})$ is zero, and the Levenberg-Marquardt update gives $x^{(2)} = x^{(1)}$.) Plot the components of the mass position and the residual force versus iterations.

Solution:

At first we calculate $DF_i(x)$, for $F_i(x)$ we can formed it as two component as

$$F_i(x) = \begin{bmatrix} T_i((a_i)_1 - x_1)/\|a_i - x\| \\ T_i((a_i)_2 - x_2)/\|a_i - x\| \end{bmatrix},$$

Assume $\|a_i - x\|^2 > L_i$, we have

$$T_i = k\frac{\|a_i - x\| - L_i}{L_i},$$

therefore

$$F_i(x) = \begin{bmatrix} k((a_i)_1 - x_1)(1/L_i - L_i/\|a_i - x\|) \\ k((a_i)_2 - x_2)(1/L_i - L_i/\|a_i - x\|) \end{bmatrix},$$

let $f_i^{(j)}(x)$ denote the jth component of $F_i(x)$ for $j = 1,2$, *i.e.*,

$$f_i^{(1)}(x) = k((a_i)_1 - x_1)(1/L_i - 1/\|a_i - x\|), \quad f_i^{(2)}(x) = k((a_i)_2 - x_2)(1/L_i - 1/\|a_i - x\|),$$

we have

$$DF_i(x) = \begin{bmatrix} \nabla f_i^{(1)}(x)^T \\ \nabla f_i^{(2)}(x)^T \end{bmatrix}.$$

As

$$\frac{\partial f_i^{(1)}}{\partial x_1}(x) = -k\left(\frac{1}{L_i} - \frac{1}{\|a_i - x\|} + \frac{((a_i)_1 - x_1)^2}{\|a - x\|^3}\right),$$

and

$$\frac{\partial f_i^{(1)}}{\partial x_2}(x) = -k\left(\frac{((a_i)_1 - x_1)((a_i)_2 - x_2)}{\|a_i - x\|^3}\right),$$

and

$$\frac{\partial f_i^{(2)}}{\partial x_1}(x) = -k\left(\frac{((a_i)_1 - x_1)((a_i)_2 - x_2)}{\|a_i - x\|^3}\right),$$

and

$$\frac{\partial f_i^{(2)}}{\partial x_2}(x) = -k\left(\frac{1}{L_i} - \frac{1}{\|a_i - x\|} + \frac{((a_i)_2 - x_2)^2}{\|a - x\|^3}\right),$$

thus, we have

$$DF_i(x) = \begin{bmatrix} -k\left(\frac{1}{L_i} - \frac{1}{\|a_i - x\|} + \frac{((a_i)_1 - x_1)^2}{\|a - x\|^3}\right) & -k\left(\frac{((a_i)_1 - x_1)((a_i)_2 - x_2)}{\|a_i - x\|^3}\right) \\ -k\left(\frac{((a_i)_1 - x_1)((a_i)_2 - x_2)}{\|a_i - x\|^3}\right) & -k\left(\frac{1}{L_i} - \frac{1}{\|a_i - x\|} + \frac{((a_i)_2 - x_2)^2}{\|a - x\|^3}\right) \end{bmatrix}.$$

As $DF^{\text{grav}} = 0$, we have

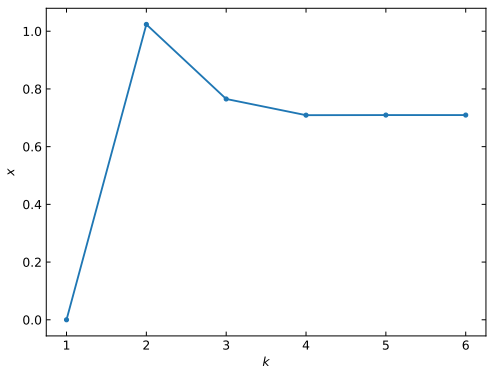
$$DF(x) = DF^{\text{grav}}(x) + DF_1(x) + DF_2(x) = DF_1(x) + DF_2(x).$$

with $DF(x)$, we can express the update formula as

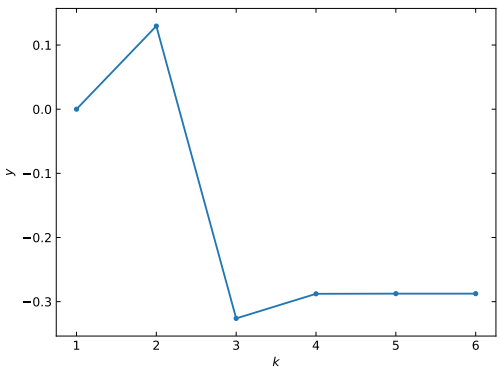
$$x^{(k+1)} = x^{(k)} - \left(DF(x^{(k)})^T DF(x^{(k)}) + \lambda^{(k)} I\right)^{-1} DF(x^{(k)})^T F(x^{(k)}).$$

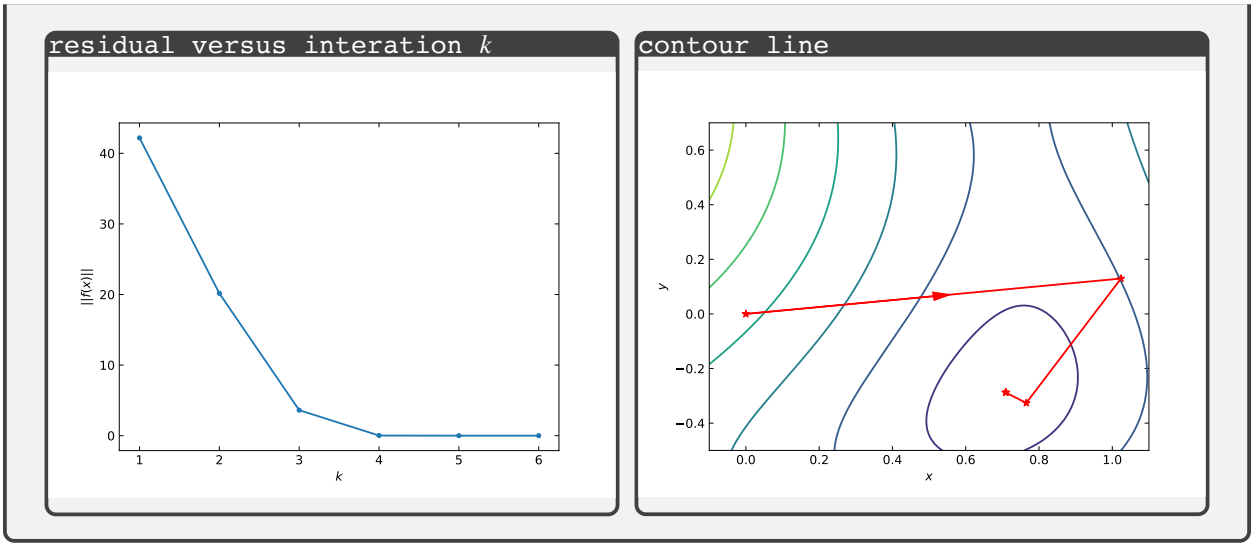
Implement with the specific values, we have

position of x versus iteration k



position of y versus iteration k





18.6 Fitting a simple neural network model. A neural network is a widely used model of the form $\hat{y} = \hat{f}(x;\theta)$, where the n-vector x is the feature vector and the p-vector θ is the maodel parameter. In a neural network model, the function $\hat{f}$ is *not* an affine function of the parameter vector θ. In this exercise we consider a very simple nerual network, with two layers, three internal nodes, and two points (*i.e.*, $n = 2$). This model has $p = 13$ parameters, and is given by

$$\begin{aligned}\hat{f}(x;\theta) \quad &= \quad \theta_1\phi(\theta_2 x_1 + \theta_3 x_2 + \theta_4) + \theta_5\phi(\theta_6 x_1 + \theta_7 x_2 + \theta_8) \\ &\quad + \theta_9\phi(\theta_{10} x_1 + \theta_{11} x_2 + \theta_{12}) + \theta_{13}\end{aligned}$$

where $\phi : \mathrm{R} \to \mathrm{R}$ is the sigmoid function define in (18.16). This function is shown as a *signal flow graph* in figure 18.3. In this graph each edge from an input to an internal node, or from an internal node to the output node, corresponds to multiplication by one of the parameters. At each node (shown as small filled circles) the incoming values and the constant offset are added together, then passed through the sigmoid function, to become the outgoing edge value.

Fitting such a model to a data set consisting of the n-vectors $x^{(1)},\ldots,x^{(N)}$ and the associated scalar outcomes $y^{(1)},\ldots,y^{(N)}$ by minimizing the sum of the squares of the residuals is a nonlinear least squares problem with objective (18.4).

(a) Derive an expression for $\nabla_\theta \hat{f}(x;\theta)$. Your expression can use ϕ and ϕ', the sigmoid function and its derivative. (You do not need to express

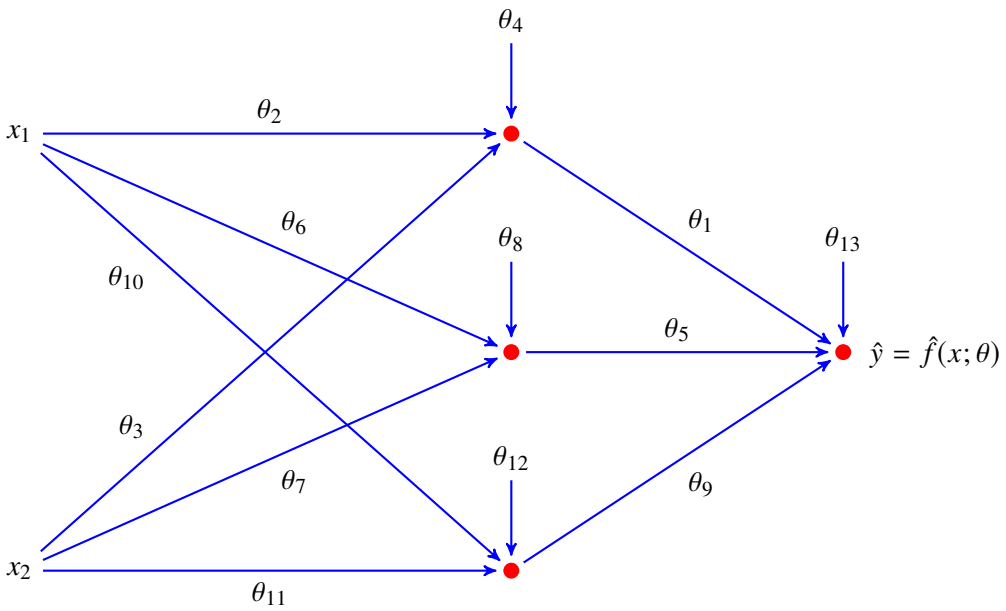


Figure 18.2: Signal flow graph of a simple neural network.

these in terms of exponentials.)

(b) Derive an expression for the derivative matrix $Dr(\theta)$, where $r : \mathrm{R}^p \to \mathrm{R}^N$ is the vector of model fitting residuals,

$$r(\theta)_i = \hat{f}(x^{(i)};\theta) - y^{(i)}, \qquad i = 1,\ldots,N.$$

Your expression can use the gradient found in part (a).

(c) Try fitting this neural network to the function $g(x_1, x_2) = x_1 x_2$. First generate $N = 200$ random points $x^{(i)}$ and take $y^{(i)} = (x^{(i)})_1 (x^{(i)})_2$ for $i = 1,\ldots,200$. Use the Levenberg-Marquardt algorithm to try to minimize

$$f(\theta) = \|r(\theta)\|^2 + \gamma \|\theta\|^2$$

with $\gamma = 10^{-5}$. Plot the value of f and the norm of its gradient versus iteration. Report the RMS fitting error achieved by the neural network model. Experiment with choosing different starting points to see the effect on the final model found.

(d) Fit the same data set with a (linear) regression model $f^{\mathrm{lin}}(x;\beta,v) = x^T\beta + v$ and report the RMS fitting error achieved. (You can add regularization in your fitting but it won't improve the results.) Compare the RMS fitting error with the neural network model RMS fitting error from part (c).

Remark. Nerual networks used in practive employ many more regressors, layers, and internal models. Specialized methods and software are used to minimize the fitting objective, and evalute the required gradients and derivatives.

Solution:

(a) For convenient, denote $r_1 = \theta_2 x_1 + \theta_3 x_2 + \theta_4$, $r_2 = \theta_6 x_1 + \theta_7 x_2 + \theta_8$, $r_3 = \theta_{10} x_1 + \theta_{11} x_2 + \theta_{12}$, for θ_1, we have

$$\frac{\partial \hat{f}}{\partial \theta_1}(x;\theta) = \phi(r_1),$$

for θ_2, we have

$$\frac{\partial \hat{f}}{\partial \theta_2}(x;\theta) = \theta_1 x_1 \phi'(r_1),$$

for θ_3, we have

$$\frac{\partial \hat{f}}{\partial \theta_3}(x;\theta) = \theta_1 x_2 \phi'(r_1),$$

for θ_4, we have

$$\frac{\partial \hat{f}}{\partial \theta_4}(x;\theta) = \theta_1 \phi'(r_1),$$

for θ_5, we have

$$\frac{\partial \hat{f}}{\partial \theta_5}(x;\theta) = \phi(r_2),$$

for θ_6, we have

$$\frac{\partial \hat{f}}{\partial \theta_6}(x;\theta) = \theta_5 x_1 \phi'(r_2),$$

for θ_7, we have

$$\frac{\partial \hat{f}}{\partial \theta_7}(x;\theta) = \theta_5 x_2 \phi'(r_2),$$

for θ_8, we have

$$\frac{\partial \hat{f}}{\partial \theta_8}(x;\theta) = \theta_5 \phi'(r_2),$$

for θ_9, we have

$$\frac{\partial \hat{f}}{\partial \theta_9}(x;\theta) = \phi(r_3),$$

for θ_{10}, we have

$$\frac{\partial \hat{f}}{\partial \theta_{10}}(x;\theta) = \theta_9 x_1 \phi'(r_3),$$

for θ_{11}, we have

$$\frac{\partial \hat{f}}{\partial \theta_{11}}(x;\theta) = \theta_9 x_2 \phi'(r_3),$$

for θ_{12}, we have

$$\frac{\partial \hat{f}}{\partial \theta_{12}}(x;\theta) = \theta_9 \phi'(r_3),$$

for θ_{13}, we have

$$\frac{\partial \hat{f}}{\partial \theta_{13}}(x;\theta) = 1.$$

Therefore

$$\nabla_\theta \hat{f}(x;\theta) = \begin{bmatrix} \phi(r_1) \\ \theta_1 x_1 \phi'(r_1) \\ \theta_1 x_2 \phi'(r_1) \\ \theta_1 \phi'(r_1) \\ \phi(r_2) \\ \theta_5 x_1 \phi'(r_2) \\ \theta_5 x_2 \phi'(r_2) \\ \theta_5 \phi'(r_2) \\ \phi(r_3) \\ \theta_9 x_1 \phi'(r_3) \\ \theta_9 x_2 \phi'(r_3) \\ \theta_9 \phi'(r_3) \\ 1 \end{bmatrix} = \begin{bmatrix} \phi(\theta_2 x_1 + \theta_3 x_2 + \theta_4) \\ \theta_1 x_1 \phi'(\theta_2 x_1 + \theta_3 x_2 + \theta_4) \\ \theta_1 x_2 \phi'(\theta_2 x_1 + \theta_3 x_2 + \theta_4) \\ \theta_1 \phi'(\theta_2 x_1 + \theta_3 x_2 + \theta_4) \\ \phi(\theta_6 x_1 + \theta_7 x_2 + \theta_8) \\ \theta_5 x_1 \phi'(\theta_6 x_1 + \theta_7 x_2 + \theta_8) \\ \theta_5 x_2 \phi'(\theta_6 x_1 + \theta_7 x_2 + \theta_8) \\ \theta_5 \phi'(\theta_6 x_1 + \theta_7 x_2 + \theta_8) \\ \phi(\theta_{10} x_1 + \theta_{11} x_2 + \theta_{12}) \\ \theta_9 x_1 \phi'(\theta_{10} x_1 + \theta_{11} x_2 + \theta_{12}) \\ \theta_9 x_2 \phi'(\theta_{10} x_1 + \theta_{11} x_2 + \theta_{12}) \\ \theta_9 \phi'(\theta_{10} x_1 + \theta_{11} x_2 + \theta_{12}) \\ 1 \end{bmatrix}$$

Note, for sigmod function

$$\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

we have

$$\phi'(x) = 1 - \phi(x)^2.$$

(b) The ith row of $Dr(\theta)$ is the transpose of the gradient of $r(\theta)_i$, and

$$\nabla r(\theta)_i = \nabla_\theta \hat{f}(x^{(i)};\theta),$$

thus

$$Dr(\theta) = \begin{bmatrix} \nabla_\theta \hat{f}(x^{(1)};\theta)^T \\ \vdots \\ \nabla_\theta \hat{f}(x^{(p)};\theta)^T \end{bmatrix}$$

(c) As our objective is

$$f(\theta) = \|r(\theta)\|^2 + \gamma \|\theta\|^2,$$

we can reformed it as

$$f(\theta) = \left\| \begin{bmatrix} r(\theta)_1 \\ \vdots \\ r(\theta)_N \\ \sqrt{\gamma}\theta_1 \\ \vdots \\ \sqrt{\gamma}\theta_{13} \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} \hat{f}(x^{(1)};\theta) - y^{(1)} \\ \vdots \\ \hat{f}(x^{(N)};\theta) - y^{(N)} \\ \sqrt{\gamma}\theta_1 \\ \vdots \\ \sqrt{\gamma}\theta_{13} \end{bmatrix} \right\|^2$$

and we have

$$Df(\theta) = \begin{bmatrix} D_\theta r(\theta) \\ \sqrt{\gamma} I_{13} \end{bmatrix}$$

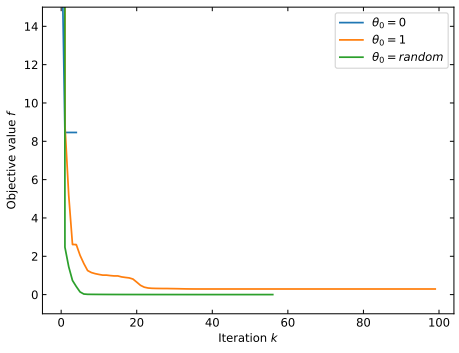
we can give the formula of the update in Levenberg-Marquardt algorithm as

$$\theta^{(k+1)} = \theta^{(k)} - \left(Df(\theta^{(k)})^T Df(\theta^{(k)}) + \lambda^{(k)} I \right)^{-1} Df(\theta^{(k)})^T f(\theta^{(k)})$$

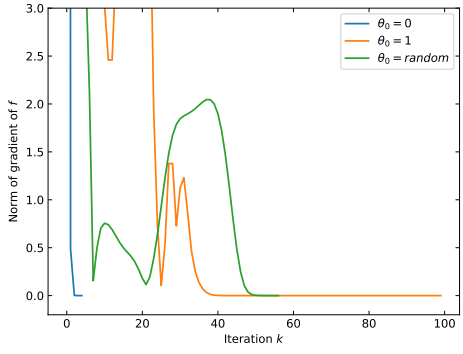
Generate $x^{(i)}$ random in interval $[0,1]$ and compute $y^{(i)}$, we implement this algorithm by three types of begining of θ, one for $\theta_0 = 0$, and second for $\theta_0 = \mathbf{1}$, last for ramdom θ_0. The objective f and norm of gradient of f versus iteration are showed below, from table below, we can conclude that it is better to choose a random start θ_0 to get better result.

θ_0	Objective f	$\|\nabla f\|$	RMS
0	8.4605	1.57×10^{-8}	0.2057
1	0.2927	3.2×10^{-5}	0.0382
random	0.0010	6.03×10^{-7}	0.0016

f versus k



Norm of gradient of f versus k



(d) Fit this data to linear model we have coeficients $\beta = (0.4790.506)$

and $v = -0.243$, and its residual RMS is 0.0732, larger than nerual network model except we use $\theta_0 = 0$.

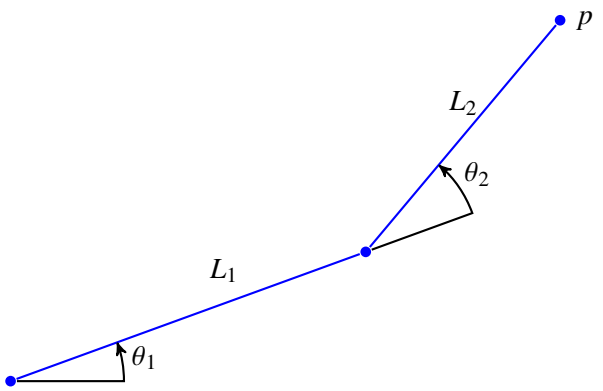


Figure 18.3: Two-link robot manipulator in a plane.

18.7 Robot manipulator. Figure 18.3 shows a two-link robot manipulator in a plane. The robot manipulator endpoint is at the position

$$p = L_1 \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} + L_2 \begin{bmatrix} \cos(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) \end{bmatrix},$$

where L_1 and L_2 are the lengths of the first and second links, θ_1 is the first joint angle, and θ_2 is second joint angle. We will assume that $L_2 < L_1$, *i.e.*, the second link is shorter than the first. We are given a desired endpoint position p^{des}, and seek joint angles $\theta = (\theta_1, \theta_2)$ for which $p = p^{\text{des}}$.

This problem can be solved analytically. We find θ_2 using the equation

$$\begin{aligned} \left\|p^{\text{des}}\right\|^2 &= (L_1 + L_2\cos\theta_2)^2 + (L_2\sin\theta_2)^2 \\ &= L_1^2 + L_2^2 + 2L_1L_2\cos\theta_2. \end{aligned}$$

When $L_1 - L_2 \leq \left\|p^{\text{des}}\right\| < L_1 + L_2$, there are two choices of θ_2 (one positive and one negative). For each solution θ_2 we can find θ_1 using

$$\begin{aligned} p^{\text{des}} &= L_1 \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} + L_2 \begin{bmatrix} \cos(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) \end{bmatrix} \\ &= \begin{bmatrix} L_1 + L_2\cos\theta_2 & -L_2\sin\theta_2 \\ L_2\sin\theta_2 & L_1 + L_2\cos\theta_2 \end{bmatrix} \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix}. \end{aligned}$$

In this exercise you will use the Levenberg-Marquardt algorithm to find joint angles, by minimizing $\left\|p - p^{\text{des}}\right\|$.

(a) Identify the function $f(\theta)$ in the nonlinear least squares problem, and give its derivative $Df(\theta)$.

(b) Implement the Levenberg-Marquardt algorithm to solve the nonlinear least squares problem. Try your implementation on a robot with $L_1 = 2, L_2 = 1$, and the desired endpoints

$$(1.0, 0.5), \quad (-2.0, 1.0) \quad (-0.2, 3.1).$$

For each endpoint, plot the cost function $\left\|f(\theta^{(k)})\right\|^2$ versus iteration number k.

Note that the norm of the least endpoint exceeds $L_1 + L_2 = 3$, so there are no joint angles for which $p = p^{\text{des}}$. Explain the angles your algorithm finds in this case.

Solution:

(a) The funtion can be formed as two components

$$f(\theta) = \begin{bmatrix} f_1(\theta) \\ f_2(\theta) \end{bmatrix} = \begin{bmatrix} L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) - (y^{\text{des}})_1 \\ L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) - (y^{\text{des}})_2 \end{bmatrix}.$$

And with

$$\nabla f_1(\theta) = \begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) \\ -L_2 \sin(\theta_1 + \theta_2) \end{bmatrix}, \qquad \nabla f_2(\theta) = \begin{bmatrix} L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ L_2 \cos(\theta_1 + \theta_2) \end{bmatrix},$$

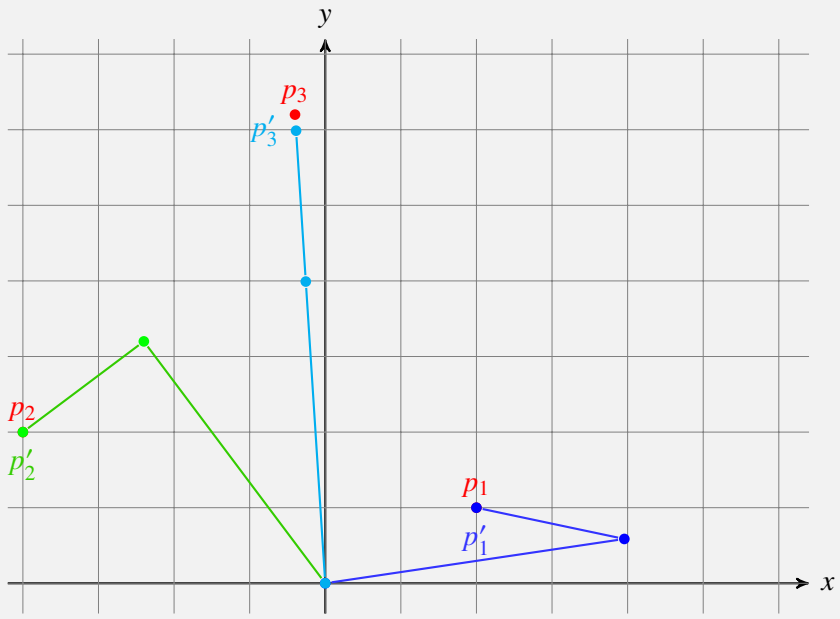
we have

$$Df(\theta) = \begin{bmatrix} \nabla f_1(\theta)^T \\ \nabla f_2(\theta)^2 \end{bmatrix} = \begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix}.$$

(b) The result of the angles are

$$(8.43°, 159.64°), \qquad (126.87°, 90.00°), \qquad (93.69°, 0.00°),$$

we plot the desired point the approximate point as below:



The angles here means the closest endpoint we can find to the specifical endpoints. Codes are listed in Listing 12.

Listing 12: Code for exercise 18.7

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exerises 18.7
# Author: Utoppia
# Date  : 25 May 2020

import numpy as np
from math import cos, sin
from utils import levenberg_marquadt

def f(p_des, l):
    def wrapper(theta):
        return np.array([
            l[0]*cos(theta[0]) + l[1]*cos(theta[0]+theta[1]) - p_des[0],
            l[0]*sin(theta[0]) + l[1]*sin(theta[0]+theta[1]) - p_des[1]
        ])
    return wrapper

def Df(p_des, l):
    def wrapper(theta):
```

```
        return np.array([
            [-l[0]*sin(theta[0])-l[1]*sin(theta[0]+theta[1]), -l[1]*sin(theta[0]+
                theta[1])],
            [l[0]*cos(theta[0])+l[1]*cos(theta[0]+theta[1]), l[1]*cos(theta[0]+
                theta[1])]
        ])
    return wrapper

p_des_list = np.array([
    [1.0, 0.5],
    [-2.0, 1.0],
    [-0.2, 3.1]
])
l = [2, 1]

for p_des in p_des_list:
    F = f(p_des, l)
    DF = Df(p_des, l)

    theta, history = levenberg_marquadt([0,0], F, DF)
    print(theta*180/np.pi)
```

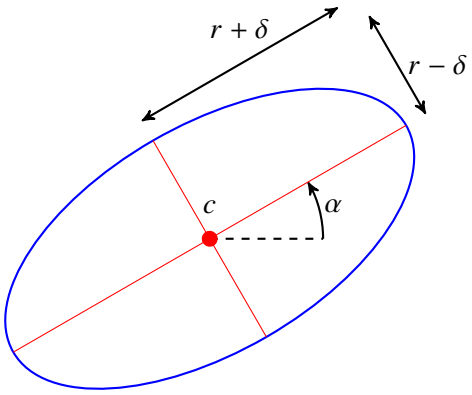


Figure 18.4: Ellipse with center (c_1, c_2), and radii $r+\delta$ and $r-\delta$. The largest semi-axis makes an angle α with respect to horizontal.

18.8 Fitting an ellipse to points in a plane. An ellipse in a plane can be described as the set of points

$$\hat{f}(t;\theta) = \begin{bmatrix} c_1 + r\cos(\alpha+t) + \delta\cos(\alpha-t) \\ c_2 + r\sin(\alpha+t) + \delta\sin(\alpha-t) \end{bmatrix},$$

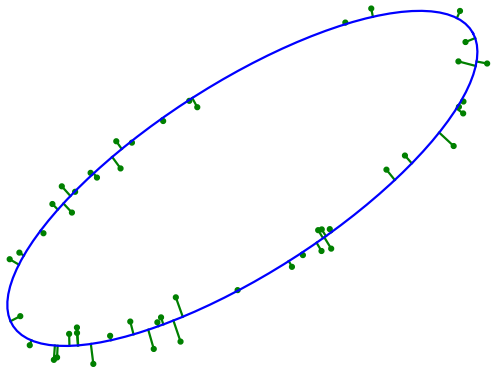


Figure 18.5: Ellipse fit to 50 points in a plane.

where t ranges from 0 to 2π. The vector $\theta = (c_1, c_2, r, \delta, \alpha)$ constains five parameters, with geometrical meanings illustrated in figure 18.4. We consider the problem of fitting an ellipse to N points $x^{(1)}, \ldots, x^{(N)}$ in a plane, as shown in figure 18.5. The circles show the N points. The short lines connect each point to the nearest point on the ellipse. We will fit the ellipse by minimizing the sum of the squared distances of the N points to the ellipse.

(a) The squared distance of a data point $x^{(i)}$ to the ellipse is the minimum of $\left\|\hat{f}(t^{(i)}; \theta) - x^{(i)}\right\|^2$ over the scalar $t^{(i)}$. Minimizing the sum of the squared distances of the data points $x^{(1)}, \ldots, x^{(N)}$ to the ellipse is therefore equivalent to minimizing

$$\sum_{i=1}^{N} \left\|\hat{f}(t^{(i)}; \theta) - x^{(i)}\right\|^2$$

over $t^{(1)}, \ldots, t^{(N)}$ and θ. Formulate this as a nonlinear least squares problem. Give expression for the derivatives of the residuals.

(b) Use the Levenberg-Marquardt algprithm to fit an ellipse to the 10 points:

$$(0.5, 1.5), \quad (-0.3, 0.6), \quad (1.0, 1.8), \quad (-0.4, 0.2), \quad (0.2, 1.3)$$

$$(0.7, 0.1), \quad (2.3, 0.8), \quad (1.4, 0.5), \quad (0.0, 0.2), \quad (2.4, 1.7).$$

To select a starting point, you can choose parameters θ that describe a circle with radius one and center at average of the data points, and initial values of $t^{(i)}$ that minimize the objective function for the those initial value of θ.

Solution:

(a) Consider the ith term of the objective $\left\|\hat{f}(t^{(i)};\theta) - x^{(i)}\right\|^2$, which can be formed as

$$\left\|\hat{f}(t^{(i)};\theta) - x^{(i)}\right\|^2 = \left((\hat{f}(t^{(i)};\theta))_1 - (x^{(i)})_1\right)^2 + \left((\hat{f}(t^{(i)};\theta))_2 - (x^{(i)})_2\right)^2,$$

the subscripts denote the entries of the point, for convenient, let $t = (t_1, \ldots, t_N,)$ and $x = (t, \theta)$, and

$$f_i(x) = (\hat{f}(t^{(i)};\theta))_1 - (x^{(i)})_1 = c_1 + r\cos(\alpha + t^{(i)}) + \delta\cos(\alpha - t^{(i)}) - (x^{(i)})_1$$
$$g_i(x) = (\hat{f}(t^{(i)};\theta))_2 - (x^{(i)})_2 = c_2 + r\sin(\alpha + t^{(i)}) + \delta\sin(\alpha - t^{(i)}) - (x^{(i)})_2$$

therefore the ith term of objective can be expressed as

$$\left\|\hat{f}(t^{(i)};\theta) - x^{(i)}\right\|^2 = \|f_i(x)\|^2 + \|g_i(x)\|^2.$$

Let $F : \mathrm{R}^{N+4} \to \mathrm{R}^N$ denote $F(x) = (f_1(x), \ldots, f_N(x))$, and $G : \mathrm{R}^{N+4} \to \mathrm{R}^N$ denote $G(x) = (g_1(x), \ldots, g_N(x))$, therefore the objective can be form as

$$\begin{aligned}\sum_{i=1}^{N}\left\|\hat{f}(t^{(i)};\theta) - x^{(i)}\right\|^2 &= \sum_{i=1}^{N}\|f_i(x)\|^2 + \sum_{i=1}^{N}\|g_i(x)\|^2 \\ &= \|F(x)\|^2 + \|G(x)\|^2 \\ &= \left\|\begin{bmatrix}F(x)\\G(x)\end{bmatrix}\right\|^2 \\ &= \|f(x)\|^2,\end{aligned}$$

where $f(x) = (F(x), F(x))$ is $f : \mathrm{R}^{N+4} \to \mathrm{R}^{2N}$ function. The gradient of the objective is

$$\nabla f(x) = 2Df(x)^T f(x).$$

as

$$Df(x) = \begin{bmatrix}DF(x)\\DG(x)\end{bmatrix},$$

we have

$$\nabla f(x) = 2DF(x)^T F(x) + 2DG(x)^T G(x).$$

Expand $DF(x)$, we have

$$DF(x) = \begin{bmatrix}DF_t(x) & DF_\theta(x)\end{bmatrix},$$

and

$$DF_t(x) = \begin{bmatrix} \nabla_t f_1(x)^T \\ \vdots \\ \nabla_t f_N(x)^T \end{bmatrix} = \begin{bmatrix} \left(-r\sin(\alpha + t^{(1)}) + \delta\sin(\alpha - t^{(1)})\right) e_1^T \\ \vdots \\ \left(-r\sin(\alpha + t^{(N)}) + \delta\sin(\alpha - t^{(N)})\right) e_N^T \end{bmatrix},$$

in which we use the fact

$$\frac{\partial f_i}{\partial t^{(i)}}(x) = -r\sin(\alpha + t^{(1)}) + \delta\sin(\alpha - t^{(1)}), \qquad \frac{\partial f_i}{\partial t^{(j)}}(x) = 0, \quad \text{if} \quad j \neq i,$$

which implies

$$\nabla_t f_i(x) = \left(-r\sin(\alpha + t^{(1)}) + \delta\sin(\alpha - t^{(1)})\right) e_i.$$

Let N-vector b with ith entry has value $b_i = -r\sin(\alpha + t^{(1)}) + \delta\sin(\alpha - t^{(1)})$, the expression of $DF_t(x)$ can be formed as

$$DF_t(x) = \mathbf{diag}(b).$$

For $DF_\theta(x)$, we have

$$DF_\theta(x) = \begin{bmatrix} \nabla_\theta f_1(x)^T \\ \vdots \\ \nabla_\theta f_N(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cos(\alpha + t^{(1)}) & \cos(\alpha - t^{(1)}) & -r\sin(\alpha + t^{(1)}) - \delta\sin(\alpha - t^{(1)}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & \cos(\alpha + t^{(N)}) & \cos(\alpha - t^{(N)}) & -r\sin(\alpha + t^{(N)}) - \delta\sin(\alpha - t^{(N)}) \end{bmatrix}$$

With same method, we have

$$\nabla_t g_i(x) = \left(r\cos(\alpha + t^{(i)}) - \delta\cos(\alpha - t^{(i)})\right) e_i^T,$$

let N-vector d with ith entry has value $d_i = r\cos(\alpha + t^{(i)}) - \delta\cos(\alpha - t^{(i)})$, we have

$$DG_t(x) = \mathbf{diag}(d).$$

And for $DG_\theta(x)$, we have

$$DG_\theta(x) = \begin{bmatrix} \nabla_\theta g_1(x)^T \\ \vdots \\ \nabla_\theta g_N(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 & \sin(\alpha + t^{(1)}) & \sin(\alpha - t^{(1)}) & r\cos(\alpha + t^{(1)}) + \delta\cos(\alpha - t^{(1)}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \sin(\alpha + t^{(N)}) & \sin(\alpha - t^{(N)}) & r\cos(\alpha + t^{(N)}) + \delta\cos(\alpha - t^{(N)}) \end{bmatrix}$$

Until now, we have expression $DF_t(x)$, $DF_\theta(x)$, $DG_t(x)$ and $DG_\theta(x)$, with

$$Df(x) = \begin{bmatrix} DF_t(x) & DF_\theta(x) \\ DG_t(x) & DG_\theta(x) \end{bmatrix} = \begin{bmatrix} \mathbf{diag}(b) & DF_\theta(x) \\ \mathbf{diag}(d) & DG_\theta(x) \end{bmatrix},$$

we can give the expression of $Df(x)$. And for $\nabla f(x)$, we have

$$\nabla f(x) = 2DF(x)^T F(x) + 2DG(x)^T G(x) = 2\begin{bmatrix} \mathbf{diag}(b)F(x) + \mathbf{diag}(d)G(x) \\ DF_\theta(x)^T F(x) + DG_\theta(x)^T G(x) \end{bmatrix}.$$

(b) With part (a), we know this problem can be changed to a nonlinear least squares problem, and we can use Levenberg-Marquardt algorithm to solve it, with update equation

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\right)^{-1} Df(x^{(k)}) f(x^{(k)}).$$

Now let consider how to choose the start point, we choose a circle with radius one with center at average of the data point, *i.e.*,

$$c_1 = 0.78, \qquad c_2 = 0.87, \qquad r = 1, \qquad \delta = 0, \qquad \alpha = 0.$$

the starting value of t is a nonlinear least squares problem, too, which can be form as

$$\text{minimize} \qquad \|\hat{f}(t)\|^2 = \left\|f(t;\theta^{(1)})\right\|^2,$$

which can be sovle by Levenberg-Marquardt algorithm, with update equation

$$t_0^{(k+1)} = t_0^{(k)} - \left(D\hat{f}(t_0^{(k)})^T D\hat{f}(t_0^{(k)}) + \lambda^{(k)} I\right)^{-1} D\hat{f}(t_0^{(k)}) \hat{f}(t_0^{(k)}),$$

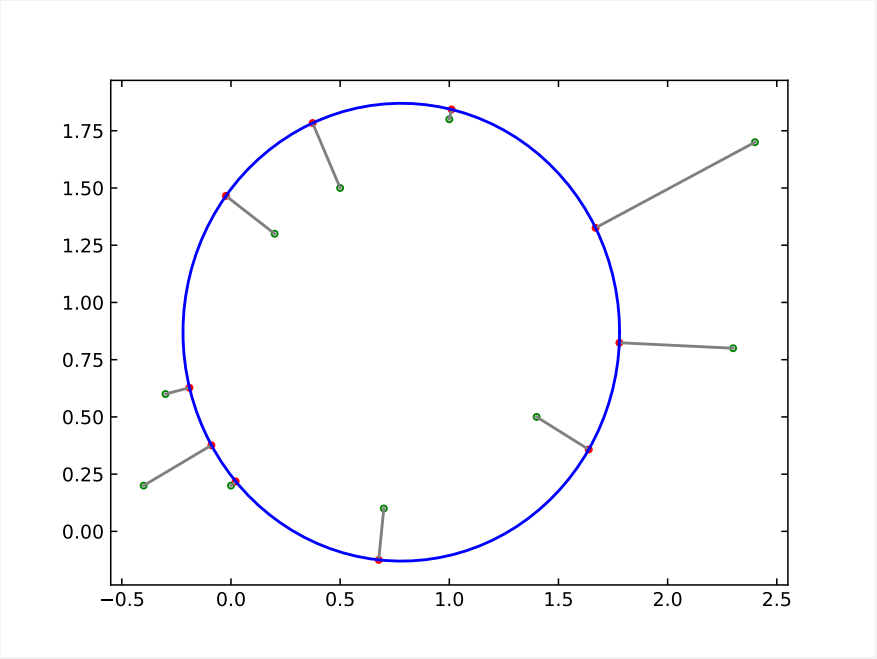
easy to know that

$$D\hat{f}(t) = \begin{bmatrix} DF_t(t;\theta^{(1)}) \\ DG_t(t;\theta^{(1)}) \end{bmatrix}.$$

Implement Levenberg-Marquardt algorithm, following

$$t^{(1)} = (1.98902065,\ -2.89661399,\ 1.33850734,\ -2.62518876,\ 2.50363165,$$
$$-1.674321,\ -0.04602012,\ -0.53804421,\ -2.43191153,\ 0.4734753)$$

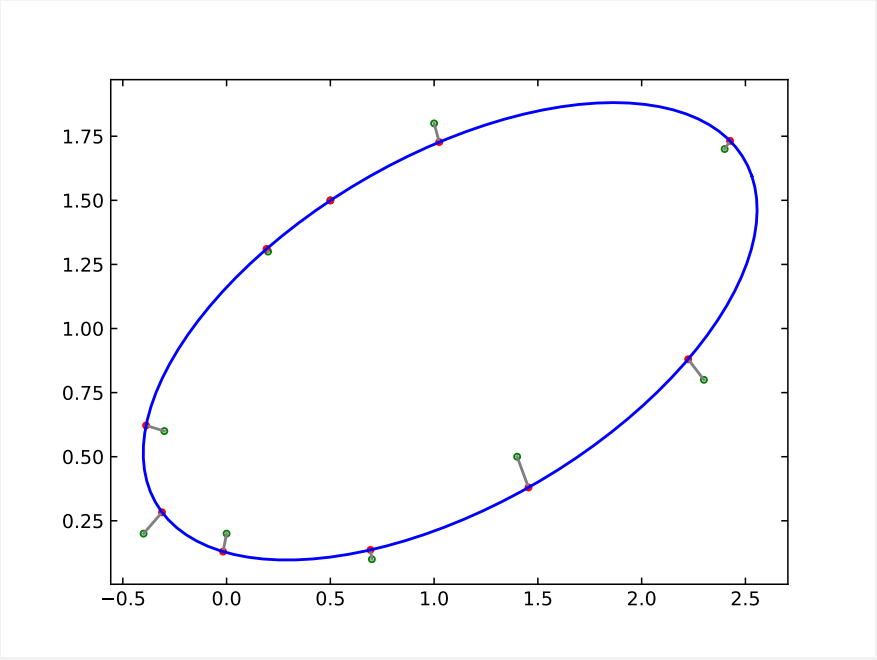
The initialization parameters forms the approximate as



Now we implement it with Levenberg-Marquardt algorithm again, we get

$$c_1 = 1.0773305,\ \ c_2 = 0.98909013,\ \ r = 1.14209934,\ \ \delta = 0.43272539,\ \ \alpha = 0.39490442.$$

The approximate ellipse is showed below:



Here we consider another question, how to get the figure 18.5, we

consider real ellipse with

$$c_1 = 1, \quad c_2 = 1, \quad r = 1, \quad \delta = 0.5, \quad \alpha = 30^\circ,$$

we generate 50 data set of t by random choose from 0 to 2π, then we use t to get data point, add small ramdom residuals to each of data point. Then use the methods described above to solve it.

19 Constrained nonlinear least squares

19.1 Projection on a curve. We consider a constrained nonlinear least squares problem with three variables $x = (x_1, x_2, x_3)$ and two equations:

$$\begin{array}{rl} \text{minimize} & (x_1 - 1)^2 + (x_2 - 1)^2 + (x_3 - 1)^2 \\ \text{subject to} & x_1^2 + 0.5x_2^2 + x_3^2 - 1 = 0 \\ & 0.8x_1^2 + 2.5x_2^2 + x_3^2 + 2x_1x_3 - x_1 - x_2 - x_3 - 1 = 0. \end{array}$$

The solution is the point closest to $(1, 1, 1)$ on the nonlinear curve defined by the two equations.

(a) Solve the problem using the augmented Lagrangian method. You can start the algorithm at $x^{(1)} = 0$, $z^{(1)} = 0$, $\mu^{(1)} = 1$, and start each run of the Levenberg-Marquardt method with $\lambda^{(1)} = 1$. Stop the augmented Lagrangian method when the feasibility residual $\|g(x^{(k)})\|$ and the optimality condition residual

$$\left\|2Df(x^{(k)})^T f(x^{(k)}) + Dg(x^{(k)})^T z^{(k)}\right\|$$

are less than 10^{-5}. Make a plot of the two residuals and of the penalty parameter μ versus the cumulative number of Levenberg-Marquardt iterations.

(b) Solve the problem using the penalty method, start at $x^{(1)} = 0$ and $\mu^{(1)} = 1$, and with the same stopping condition. Compare the convergence and the value of the penalty parameter with the results for the augmented Lagrangian method in part (a).

Solution:

(a) First we should compute the derivative matrix of f and g,

$$Df(x) = \nabla f(x)^T = \begin{bmatrix} 2x_1 - 2 & 2x_2 - 2 & 2x_3 - 2 \end{bmatrix},$$

$$Dg(x) = \begin{bmatrix} 2x_1 & x_2 & 2x_3 \\ 1.6x_1 + 2x_3 - 1 & 5x_2 - 1 & 2x_3 + 2x_1 - 1 \end{bmatrix},$$

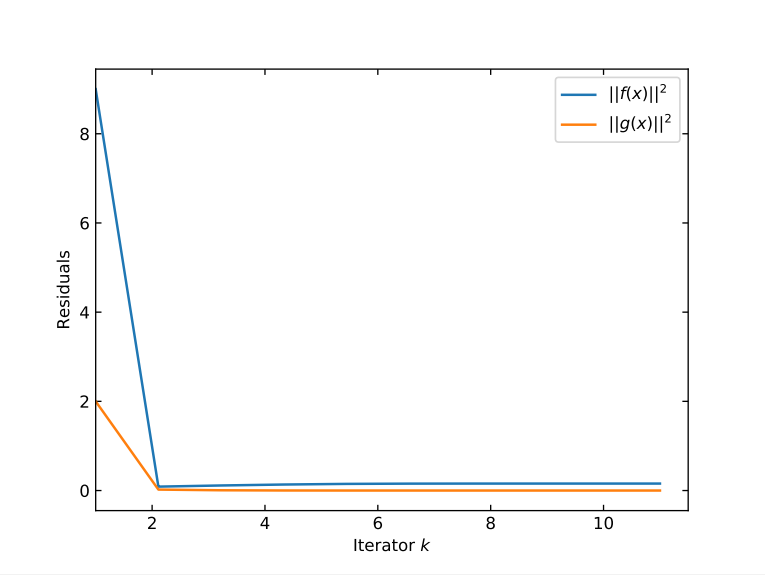
Using the augmented Lagrangian algorithm, it takes about 10 iteration to get the approximate point

$$\hat{x} = (0.56770243,\ 0.83277701,\ 0.57529005),$$

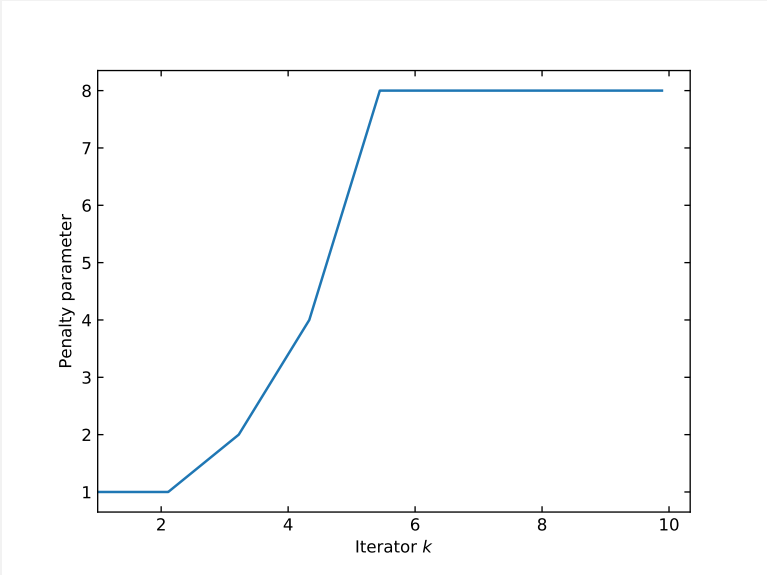
the objective value

$$\|f(\hat{x})\|^2 \approx 0.156201.$$

The two residuls versus iteration are ploted as below



and the penalty parameter versus iteration are



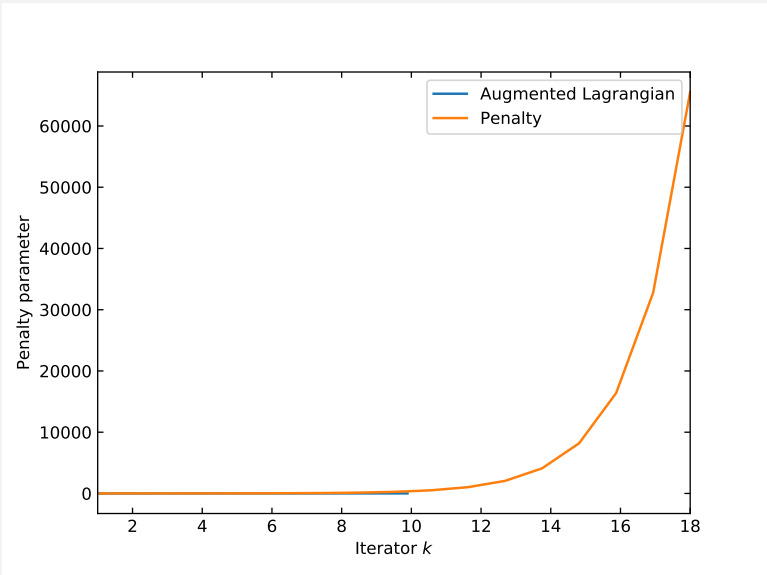
(b) Use penalty method, it takes about 18 iteration to get the approximate point

$$\hat{x}' = (0.56770351,\ \ 0.83277631,\ \ 0.57529108),$$

the objective value

$$\|f(\hat{x}')\|^2 \approx 0.156201,$$

which has the similar result with the augmented Lagrangian algorithm but take more iteration and the penalty parameter changes to be large which can be figure out in the following plot:



Listing 13 show the code for part (b), where function *augmented_lagrangian* is defined in listing 14.

Listing 13: Code for exercise 19.1

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exercise 19.1
# Author: Utoppia
# Date  : 31 May 2020

from numpy import array, linspace
from numpy.linalg import norm
from utils import augmented_lagrangian, penalty
import matplotlib.pyplot as plt

f = lambda x: array([(x[0]-1)**2 + (x[1]-1)**2 + (x[2]-1)**2 ])
Df = lambda x: array([
    [2*x[0]-2, 2*x[1]-2, 2*x[2]-2]
])

g = lambda x: array([ x[0]**2 + 0.5*x[1]**2 + x[2]**2-1, 0.8*x[0]**2 + 2.5*x
    [1]**2 + x[2]**2 + 2*x[0]*x[2] - x[0]-x[1]-x[2]-1 ])
Dg = lambda x: array([
    [2*x[0], x[1], 2*x[2]],
    [1.6*x[0] + 2*x[2]-1, 5*x[1]-1, 2*x[2]+2*x[0]-1]
])

x0 = array([0,0,0])
z0 = array([0,0])

x, z, status, history = augmented_lagrangian(x0, z0, f, Df, g, Dg, tol
    =10**(-5), ttol=10**(-5))
iterator = len(history['residual1'])
print('Using Augmented Lagrangian algorithm')
print('x is', x)
print('minimum objective is', norm(f(x))**2 )
print('Iterator is', iterator )


# Plot
fig, ax = plt.subplots()
```

```
k = linspace(1, iterator+1, iterator)
ax.plot(k, history['residual1'], label='$||f(x)||^2$')
ax.plot(k, history['residual2'], label='$||g(x)||^2$')
ax.set_xlim(1)
ax.set_xlabel('Iterator $k$')
ax.set_ylabel('Residuals')
ax.legend()
fig.savefig('19-1-1.pdf')

fig, ax = plt.subplots()
ax.plot(k[:-1], history['penalty'], label='Augmented Lagrangian')
ax.set_xlabel('Iterator $k$')
ax.set_ylabel('Penalty parameter')
ax.set_xlim(left=1)
fig.savefig('19-1-2.pdf')

x1, _, history = penalty(x0, f, Df, g, Dg, tol=10**(-5))
iterator = len(history['penalty'])
print(50*'-')
print('Using penalty method')
print('x is', x1)
print('Minimum objectice is', norm(f(x))**2)
print('Iterator is', iterator + 1)

k = linspace(1, iterator+1, iterator)
ax.plot(k, history['penalty'], label='Penalty')
ax.set_xlim(right=iterator+1)
ax.legend()
fig.savefig('19-1-3.pdf')
#plt.show()
```

Listing 14: utils.py - Function for 19

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Section 19 Util Functions
# Author: Utoppia
# Date  : 31 May 2020

from numpy.linalg import norm, inv, pinv
from numpy import eye, sqrt, vstack, hstack
```

```
def levenberg_marquadt(x0, F, DF, lamda0=1, nMax=100, tol=10**(-6)):
    """
    Levenberg-Marquadt algorithm to solve nonlinear least squres problem
        with
    objective F(x) and derivative matrix DF(x).
         minimize ||F(x)||^2 + `\lambda'*||x-x_k||^2
    ----------------------------------------------
    Parameters:
        x0: initialization of x, array-like
        F: objevtice function, callable
        DF: derivative matrix function, callable
        lamda0: float
        tol: if gradient of f smaller than tol, quit the iteration
    Return:
        x: array like
        history: dict object
    """
    n = len(x0)
    x, lamda = x0, lamda0

    objectives = [0]
    residuals = [0]
    gradients = [0]

    for k in range(nMax):

        fk = F(x)
        Dfk = DF(x)
        objectives.append(norm(fk)**2)
        gradients.append(norm(2*Dfk.T @ fk))

        if norm( 2 * Dfk.T @ fk) < tol:
            break
        xk = x - inv( Dfk.T @ Dfk + lamda * eye(n) ) @ Dfk.T @ fk

        if norm(F(xk)) < norm(fk):
            x = xk
            lamda = 0.8 * lamda
        else:
            lamda = 2 * lamda

    return x, { 'objectives': objectives, 'gradients': gradients }
```

```
def penalty(x0, F, DF, G, DG, mu0=1, nMax=100, tol=10**(-6)):
    '''
    Panalty algorithm to solve the constrained nonlinear least squares
        problem :
        minimize    || F(x) ||^2
        subject to  G(x) = 0
    ----------------------------------------
    Parameters:
        x0: initial feature vector
        F: function
        DF: derivative function of F
        G: constrained function
        DG: derivative function of G
    '''

    mu, x = mu0, x0
    status = 0
    u = []
    for k in range(0, nMax):
        f = lambda x: hstack([ F(x), sqrt(mu) * G(x) ])
        Df = lambda x: vstack([ DF(x), sqrt(mu) * DG(x) ])
        x, _ = levenberg_marquadt(x, f, Df)
        u.append(mu)
        if norm(G(x)) < tol:
            status = 1
            break
        mu = 2 * mu
    return x, mu, {'penalty': u}

def augmented_lagrangian(x0, z0, F, DF, G, DG, mu0=1, nMax=100, tol=10**(-6)
    , ttol=10**(-6)):
    '''
    Augmented Lagrangian algorithm to solve the constrained nonlinear least
        squares problem :
        minimize    || F(x) ||^2
        subject to  G(x) = 0
    changed to
        minimize || F(x) ||^2 + mu * || G(x) + z / (2mu)||^2
    with update of z:
        z = z + 2 * u * G(x)
```

```
    ----------------------------------------
    Parameters:
        x0: initial feature vector
        z0: Optimal Lagrange multipliers
        F: function
        DF: derivative function of F
        G: constrained function
        DG: derivative function of G
    '''
    mu, x, z = mu0, x0, z0
    status = 0
    residual_y = [norm(F(x))**2]
    residual_g = [norm(G(x))**2]
    us = []
    for k in range(nMax):
        f = lambda x: hstack([ F(x), sqrt(mu) * (G(x)+(z/(2*mu))) ])
        Df = lambda x: vstack([ DF(x), sqrt(mu) * DG(x) ])
        xk, _ = levenberg_marquadt(x, f, Df)
        residual_y.append(norm(F(xk))**2)
        residual_g.append(norm(G(xk))**2)
        us.append(mu)
        if norm(G(xk)) < tol: # if terminal condition 1 holds
            x = xk
            status = 1
            break

        if norm(2*DF(xk).T @ F(xk) + DG(xk).T @ z) < ttol: # if terminal
            condition 2 holds
            x = xk
            status = 2
            break

        z = z + 2 * mu * G(xk) # Update z
        if norm(G(xk)) < 0.25 * norm(G(x)): # Update mu
            mu = mu
        else :
            mu = 2 * mu
        x = xk

    return x, z, status, {'residual1': residual_y, 'residual2': residual_g,
        'penalty': us}
```

19.2 Portfilio optimization with downside risk. In standard portfilio optimization (as described in 17.1) we choose the weight vector w to achieve a given target mean return, and to minimize deviations from the target return value (*i.e.*, the risk). This leads to the constrained linear least squares problem (17.2). One criticism of this formulation is that it treats portfilio returns that exceed our target value the same as returns that fall short of our target value, whereas in fact we should be dilighted to have a return that exceeds our target value. To address this deficiency in the formulation, researchers have defined the *downside risk* of a portfilio return time series T-vector r, which is sensitive only to portfilio returns that fall short of our target value ρ^{tar}. The downside risk of a portfilio return time series (T-vector) r is given by

$$D = \frac{1}{T}\sum_{t=1}^{T}\left(\max\{\rho^{\mathrm{tar}} - r_t, 0\}\right)^2.$$

The quantity $\max\{\rho^{\mathrm{tar}} - r_t, 0\}$ is the shortfall, *i.e.*, the amount by which the return in period t falls short of the target; it is zero when the return exceeds the target value. The downside risk is the mean square value of the return shortfall.

(a) Formulate the portfilio optimization problem, using downside risk in place of the usual risk, as a constrained nonlinear least squares problem. Be sure to explain what the functions f and g are.

(b) Since the function g is affine (if you formulate the problem correctly), you can use the Levenberg-Marquardt algorithm, modified to handle linear equality constraints, to approximately solve the problem. (See page 420.) Find an expression for $Df(x^{(k)})$. You can ignore the fact that the function f is not differentiable at some points.

(c) Implement the Levenberg-Marquardt algorithm to find weights that minimize downside risk for a given target annualizaed return. A very reasonable starting point is the solution of the standard portfilio optimization problem with the same target return. Check your implementation with some simulated or real return data (avariable online). Compare the weights, and the risk and downside risk, for the minimum risk and the minimum downside risk portfilio.

Solution:

(a) We know the return time series r can be expressed as

$$r = Rw,$$

where R is the given $T \times n$ matrix, therefor

$$r_t = e_t^T Rw,$$

let

$$f_i(w) = \max\{\rho^{\text{tar}} - r_t, 0\} = \max\{\rho^{\text{tar}} - e_i^T Rw, 0\}, \quad i = 1, \ldots, T,$$

The downside risk can be formed as

$$D = \frac{1}{T} \|f(w)\|^2 .$$

The constraints are

$$\mathbf{1}^T w = 1, \qquad \mu^T w = \rho^{\text{tar}},$$

where $\mu = R^T \mathbf{1}/T$ is the n-vector of the average asset returns described in page 360, define constraint function g as

$$g(w) = \begin{bmatrix} \mathbf{1}^T \\ \mu^T \end{bmatrix} w - \begin{bmatrix} 1 \\ \rho^{\text{tar}} \end{bmatrix},$$

therefore this problem can be considered as to find the minimizer of the constrained nonlinear least squares problem:

$$\begin{array}{ll} \text{minimize} & \|f(w)\|^2 \\ \text{subject to} & g(w) = 0 \end{array}$$

(b) For $i = 1, \ldots, T$, the gradient of f_i at point $x^{(k)}$ can be computed by

$$\nabla f_i(x^{(k)}) = \begin{cases} R^T e_i, & \text{if } \rho^{\text{tar}} > \mathbf{1}^T R x^{(k)} \\ 0, & \text{otherwise} \end{cases},$$

therefore the derivative matrix of f can be computed by

$$Df(x^{(k)}) = \begin{bmatrix} \nabla f_i(x^{(k)})^T \\ \vdots \\ \nabla f_T(x^{(k)})^T \end{bmatrix}.$$

19.3 Boolean least squares. The *Boolean least squares problem* is a special case of the constrained nonlinear least squares problem (19.1), with the form

$$\begin{array}{ll} \text{minimize} & \|Ax-b\|^2 \\ \text{subject to} & x_i^2 = 1, i = 1, \ldots, n, \end{array}$$

where the n-vector x is the variable to be chosen, and the $m \times n$ matrix A and the m-vector b are the (given) problem data. The constraints require that each entry of x is either -1 or $+1$, *i.e.*, x is a Boolean vector. Since each entry can take one of two values, there are 2^n feasible values for vector x. The Boolean least squares problem arises in many applications.

One simple method for solving the Boolean least squares problem, sometimes called the *brute force method*, is to evalute the objective function $\|Ax-b\|^2$ for each of the 2^n possible values, and choose one that has the least value. This method is not practical for n larger than 30 or so. There are many heuristic methods that are much faster to carry out than the brute force method, and approximately solve it, *i.e.*, find an x for which the objective is small, if not the samllest possible value over all 2^n feasible values of x. One such heuristic is the augmented Lagrangian algorithm 19.2.

(a) Work out the details of the update step in the Levenberg-Marquardt algorithm used in each iteration of the augmented Lagrangian algorithm, for the Boolean least squares problem.

(b) Implement the augmented Lagrangian algorithm for the Boolean least squares problem. You can choose the starting point $x^{(1)}$ as the minimizer of $\|Ax-b\|^2$. At each iteration, you can obtain a feasible point $\tilde{x}^{(k)}$ by rounding the entries of $x^{(k)}$ to the values ± 1, *i.e.*, $\tilde{x}^{(k)} = \mathbf{sign}(x^{(k)})$. You should evaluate and plot the objective value of these feasible points, *i.e.*, $\left\|A\tilde{x}^{(k)}-b\right\|^2$. Your implementation can return the best rounded value found during the iterations. Try your method on some small problems, with $n = m = 10$ (say), for which you can find the actual solution by the brute force method. Try it on much larger problems, with $n = m = 50$ (say), for which the brute force method is not practical.

Solution:

(a) Here we have $f(x) = Ax - b$, therefore

$$Df(x) = A,$$

and $g_i(x) = x_i^2 - 1$ for $i = 1, \ldots, n$, therefore

$$Dg(x) = 2\mathbf{diag}(x),$$

where I is an $n \times n$ unit matrix. In the kth iteration of augmented Lagrangian algorithm, we use the Levenberg-Marquardt algorithm to find the minimizer of

$$\|f(x)\|^2 + \mu^{(k)} \left\| g(x) + z^{(k)}/(2\mu^{(k)}) \right\|^2,$$

which is same to solve the nonlinear least squares problem

$$\left\| \begin{bmatrix} f(x) \\ \sqrt{\mu^{(k)}} g(x) + z^{(k)}/(2\sqrt{\mu^{(k)}}) \end{bmatrix} \right\|^2,$$

let

$$\phi(x) = \begin{bmatrix} f(x) \\ \sqrt{\mu^{(k)}} g(x) + z^{(k)}/(2\sqrt{\mu^{(k)}}) \end{bmatrix},$$

the problem can be formed as

$$\text{minimize} \quad \|\phi(x)\|^2,$$

use Levenberg-Marquardt algorithm to solve this problem, we have the update equation:

$$x^{(m+1)} = x^{(m)} - \left(D\phi(x^{(m)})^T D\phi(x^{(m)}) + \lambda^{(m)} I \right)^{-1} D\phi(x^{(m)})^T \phi(x^{(m)}),$$

we have

$$D\phi(x) = \begin{bmatrix} Df(x) \\ \sqrt{u^{(k)}} Dg(x) \end{bmatrix} = \begin{bmatrix} A \\ 2\sqrt{u^{(k)}} \mathbf{diag}(x) \end{bmatrix},$$

therefore

$$D\phi(x)^T D\phi(x) = \begin{bmatrix} A^T & 2\sqrt{u^{(k)}} \mathbf{diag}(x) \end{bmatrix} \begin{bmatrix} A \\ 2\sqrt{u^{(k)}} \mathbf{diag}(x) \end{bmatrix} = A^T A + 4u^{(k)} \mathbf{diag}(x^2),$$

and

$$
\begin{aligned}
D\phi(x)^T\phi(x) &= \begin{bmatrix} A^T & 2\sqrt{u^{(k)}}\mathbf{diag}(x) \end{bmatrix} \begin{bmatrix} f(x) \\ \sqrt{\mu^{(k)}}g(x) + z^{(k)}/(2\sqrt{\mu^{(k)}}) \end{bmatrix} \\
&= A^T f(x) + 2\mu^{(k)}\mathbf{diag}(x)g(x) + \mathbf{diag}(x)z^{(k)} \\
&= A^T Ax + A^T b + 2\mu^{(k)}\mathbf{diag}(x)g(x) + \mathbf{diag}(x)z^{(k)},
\end{aligned}
$$

thus the update equation can be expressed as

$$
\begin{aligned}
x^{(m+1)} &= x^{(m)} \\
&\quad - \left(A^T A + 4u^{(k)}\mathbf{diag}(x) + \lambda^{(m)} I\right)^{-1} \left(A^T Ax^{(m)} + A^T b + 2\mu^{(k)}\mathbf{diag}(x)g(x) + \mathbf{diag}(x)z^{(k)}\right).
\end{aligned}
$$

(b) The code are listed in listing 15.

Listing 15: Code for exercise 19.3

```
#!/usr/local/bin/python3
# -*- coding:utf-8 -*-
# Python 3.6+
# Code for Exercise 19.3
# Author: Utoppia
# Date  : 31 May 2020
from numpy import array, random, eye, ones, hstack, vstack, diag, zeros
from numpy.linalg import norm, pinv
from utils import levenberg_marquadt, augmented_lagrangian
from math import sqrt

def sign(x):
    return array([1 if item > 0 else -1 for item in x])

def brute(A, b, n):
    def split(x):
        ret = []
        for _ in range(n):
            ret.append(x%2)
            x = x//2
        return ret
    res, x = 10**10, None
    for i in range(2**n):
        q = split(i)
        xk = sign(q)
```

```
        res_k = norm(A @ xk - b)**2
        if res_k < res:
            res = res_k
            x = xk
    return x, res

def solver(A, b):
    f = lambda x: A @ x - b
    Df = lambda x: A
    g = lambda x: x**2 - 1
    Dg = lambda x: 2 * diag(x)
    x0 = pinv(A) @ b # x0 to be the minimize of ||Ax-b||^2
    x0 = sign(x0)
    z0 = ones(n)
    x, z, _, history = augmented_lagrangian(x0, z0, f, Df, g, Dg)
    return sign(x), norm(A @ sign(x) - b)**2


for _ in range(100):
    n, m = 10, 10
    A = random.random((m,m))
    b = random.random(m)

    #x, res = brute(A, b, n)
    #print('Brute force')
    #print(x, res)
    x, res = solver(A, b)
    print('Augmented Lagrangian')
    print(x, res)
```