Numerical methods

- A numerical method is an approximate computer method for solving a mathematical problem which often has no analytical solution.

Analytical Solutions

- Many problems have well-defined solutions that are obvious once the problem has been defined.
- A set of logical steps that we can follow to calculate an exact outcome.
- In linear algebra, there are a suite of methods that you can use to factorize a matrix, depending on if the properties of your matrix are square, rectangular, contain real or imaginary values, and so on.
- Some problems in applied machine learning are well defined and have an analytical solution.
- For example, the method for transforming a categorical variable into a one hot encoding is simple, repeatable and always the same methodology regardless of the number of integer values in the set.
- Unfortunately, most of the problems that we care about solving in machine learning do not have analytical solutions.

Numerical Solutions

- There are many problems that we are interested in that do not have exact solutions.
- We have to make guesses at solutions and test them to see how good the solution is. This involves framing the problem and using trial and error across a set of candidate solutions.
- In essence, the process of finding a numerical solution can be described as a search

These types of solutions have some interesting properties:

- We often easily can tell a good solution from a bad solution.
- We often don't objectively know what a "*good*" solution looks like; we can only compare the goodness between candidate solutions that we have tested.
- We are often satisfied with an approximate or "*good enough*" solution rather than the single best solution.
- Often the problems that we are trying to solve with numerical solutions are challenging, where any "*good enough*" solution would be useful. It also highlights that there are many solutions to a given problem and even that many of them may be good enough to be usable.
- Most of the problems that we are interested in solving in applied machine learning require a numerical solution.

Analytical vs Numerical Solutions

- An analytical solution involves framing the problem in a well-understood form and calculating the exact solution.
- A numerical solution means making guesses at the solution and testing whether the problem is solved well enough to stop.
- An example is the square root that can be solved both ways.
- We prefer the analytical method in general because it is faster and because the solution is exact. Nevertheless, sometimes we must resort to a numerical method due to limitations of time or hardware capacity.

- A good example is in finding the coefficients in a linear regression equation that can be calculated analytically (e.g. using linear algebra), but can be solved numerically when we cannot fit all the data into the memory of a single computer in order to perform the analytical calculation (e.g. via gradient descent).
- Sometimes, the analytical solution is unknown and all we have to work with is the numerical approach.

Numerical Solutions in Machine Learning

- Applied machine learning is a numerical discipline.
- The core of a given machine learning model is an optimization problem, which is really a search for a set of terms with unknown values needed to fill an equation. Each algorithm has a different "*equation*" and "*terms*", using this terminology loosely.
- The equation is easy to calculate in order to make a prediction for a given set of terms, but we don't know the terms to use in order to get a "*good*" or even "*best*" set of predictions on a given set of data.
- This is the numerical optimization problem that we always seek to solve.
- It's numerical, because we are trying to solve the optimization problem with noisy, incomplete, and error-prone limited samples of observations from our domain. The model is trying hard to interpret the data and create a map between the inputs and the outputs of these observations.

Types of Errors

Numerically computed solutions are subject to certain errors. Mainly there are three types of errors. They are inherent errors, truncation errors and errors due to rounding.
1. Inherent errors or experimental errors arise due to the assumptions made in the mathematical modelling of problem. It can also arise when the data is obtained from certain physical measurements of the parameters of the problem. i.e., errors arising from measurements.
2. Truncation errors are those errors corresponding to the fact that a finite (or infinite) sequence of computational steps necessary to produce an exact result is "truncated" prematurely after a certain number of steps.
3. Round of errors are errors arising from the process of rounding off during computation. These are also called chopping, i.e. discarding all decimals from some decimals on.

Errors in Numerical Procedures

The relationship between the exact, or true, result and the approximation can be formulated as
True value = approximation + error
We get,
E_t = | True value − approximation |
where E_t is used to designate the exact value of the absolute error. The subscript t is included to designate that this is the "absolute true" error.

- A shortcoming of this definition is that it takes no account of the order of magnitude of the value under examination.
- $Absolute\ True\ relative\ error = \left|\frac{true\ error}{true\ value}\right|$
- $Absolute\ percent\ relative\ error\ \varepsilon_T = \left|\frac{true\ error}{true\ value}\right| \times 100$

- However, in machine learning applications, we will obviously not know the true answer beforehand. For these situations, an alternative is to normalize the error using the best available estimate of the true value, that is, to the approximation itself,

$$\varepsilon_a = \frac{approximate\ error}{approximation}\ 100\%$$

- Certain numerical methods use an *iterative approach* to compute answers. In such an approach, a present approximation is made on the basis of a previous approximation. This process is performed repeatedly, or iteratively, to successively compute better and better approximations. For such cases, the error is often estimated as the difference between previous and current approximations. Thus, percent relative error is determined according to
- $\varepsilon_a = \left|\frac{current\ approximation - previous\ approximation}{current\ approximation}\right| \times 100$

- When performing computations, we may not be concerned with the sign of the error, but we are interested in whether the percent absolute value is lower than a prespecified percent tolerance ε.

$$|\varepsilon_a| < \varepsilon$$

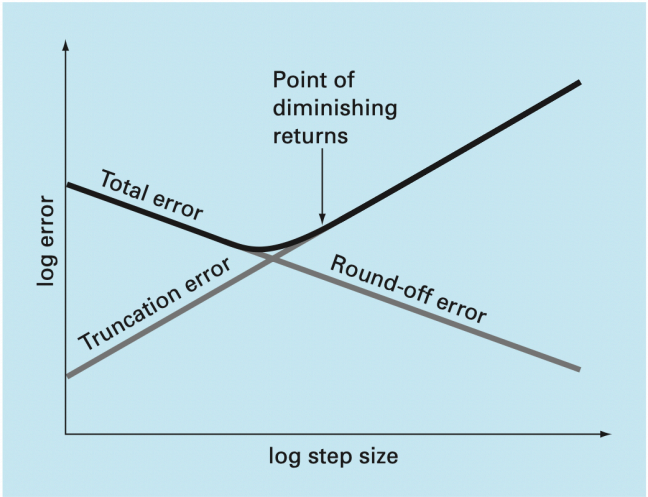
We say that the estimate is correct to *n* decimal digits if:

$$|\text{Error}| \le 10^{-n}$$

We say that the estimate is correct to *n* decimal digits **rounded** if:

$$|\text{Error}| \le \frac{1}{2} \times 10^{-n}$$

- The *total numerical error* is the summation of the truncation and round-off errors. In general, the only way to minimize round-off errors is to increase the number of significant figures of the computer. Further, we have noted that round-off error will *increase* due to subtractive cancellation or due to an increase in the number of computations in an analysis. In contrast, the truncation error can be reduced by decreasing the step size. Because a decrease in step size can lead to subtractive cancellation or to an increase in computations, the truncation errors are *decreased* as the round-off errors are *increased.*



Root Finding Methods

- An important problem in applied mathematics is to "solve $f(x) = 0$" where $f(x)$ is a function of x.
- The values of x that make $f(x) = 0$ are called the *roots* of the equation. They are also called the *zeros* of $f(x)$.
- The root finding methods are divided into two categories: bracketing and open methods.
- The bracketing methods require the limits between which the root lies
- Bisection and False position methods are two known examples of the bracketing methods.
- Open methods require the initial estimation of the solution. Among the open methods, the Newton-Raphson and Secant is most commonly used.
- The most popular method for solving a non-linear equation is the Newton-Raphson method and this method has a high rate of convergence to a solution.

Bisection method

Since the method is based on finding the root between two points, the method falls under the category of bracketing methods.

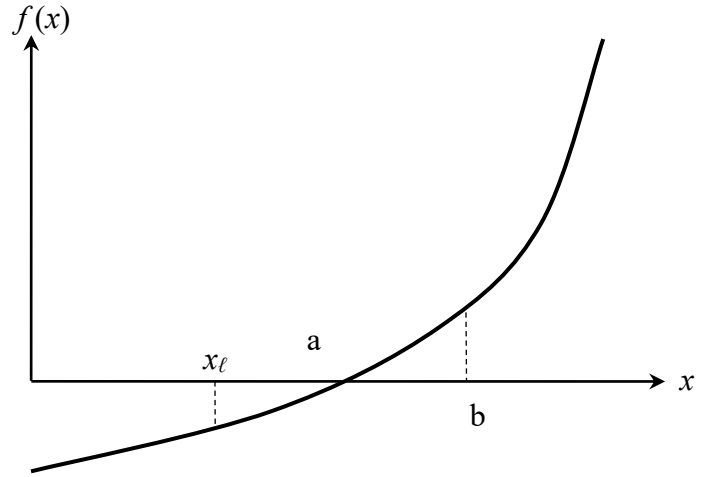


Figure 1 At least one root exists between the two points if the function is real, continuous, and changes sign.

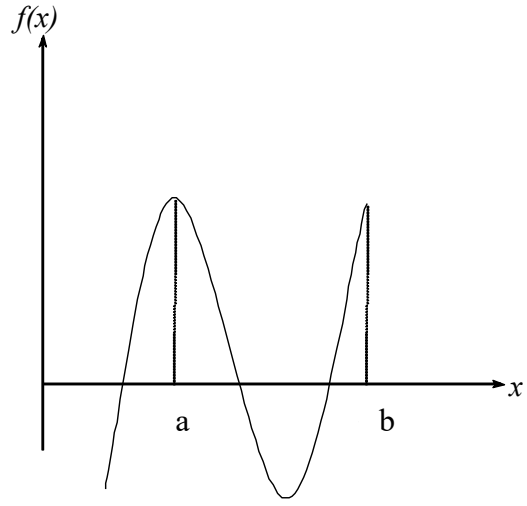


Figure 2 If the function $f(x)$ does not change sign between the two points, roots of the equation $f(x) = 0$ may still exist between the two points.

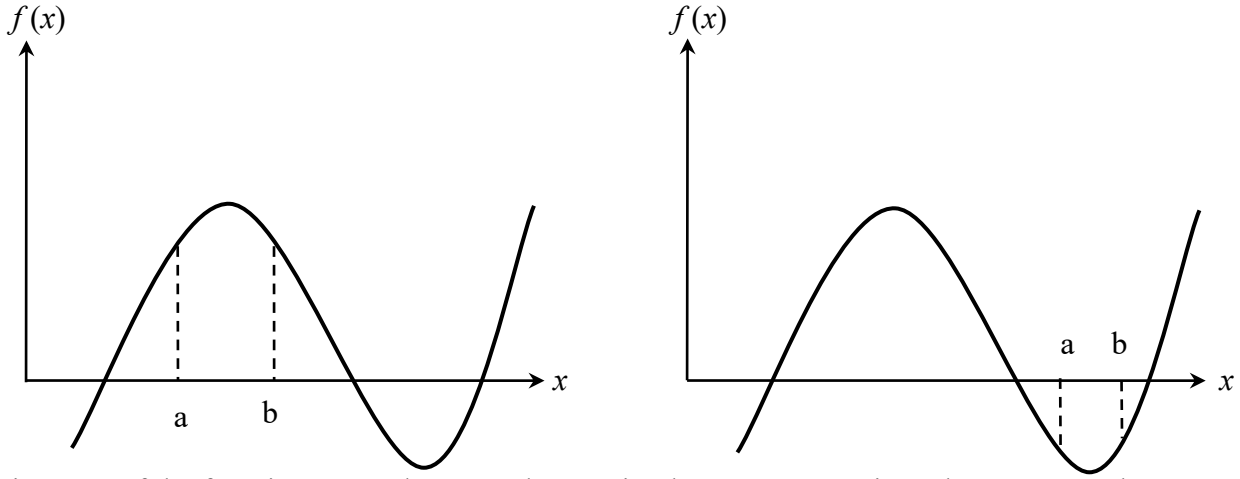


Figure 3 If the function $f(x)$ does not change sign between two points, there may not be any roots for the equation $f(x) = 0$ between the two points.

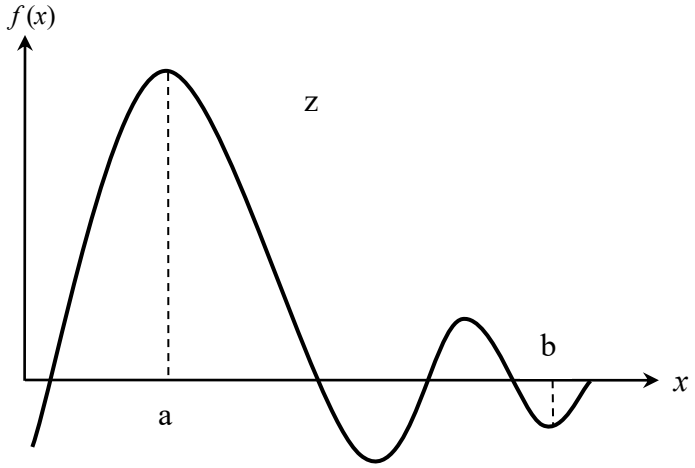


Figure 4 If the function $f(x)$ changes sign between the two points, more than one root for the equation $f(x) = 0$ may exist between the two points.

Since the root is bracketed between two points, a and b, one can find the mid-point, c between a and. This gives us two new intervals

1. a and c, and
2. c and b.

Is the root now between a and c or between c and b? Well, one can find the sign of f(a).f(c), and if f(a).f(c) < 0 then the new bracket is between a and c, otherwise, it is between c and b. So, you can see that you are literally halving the interval. As one repeats this process, the width of the interval becomes smaller and smaller, and you can zero in to the root of the equation $f(x) = 0$.

The first approximation to the root is

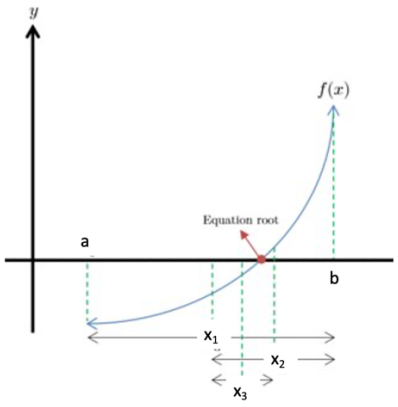
$$x_1 = \frac{(a+b)}{2}.$$

If $f(x_1) = 0$, then x_1 is a root of $f(x) = 0$,
otherwise, the root lies in
(a, x_1) or (x_1, b) according to $f(x_1)$ is (+)ve or (–)ve.

Then we bisect the interval as before and continue the process until the root is found to the desired accuracy.
In the adjoining figure, $f(x_1)$ is (-)ve so that the root lies between b and x_1.
$f(x_1) \cdot f(b) < 0$
The second approximation to the root is

$$x_2 = \frac{(x_1 + b)}{2}.$$

$f(x_2)$ is (+)ve the root lies between
x_1 and x_2.
$f(x_1) \cdot f(x_2) < 0$
Similarly, the third approximation to the root is x_3 and so on.
Then $x_3 = \frac{(x_1+x_2)}{2}$.



Algorithm for the bisection method

The steps to apply the bisection method to find the root of the equation $f(x) = 0$ are

1. Choose a and b as two guesses for the root such that f(a).f(b) < 0 , or in other words, $f(x)$ changes sign between a and b. Let k =1
2. Estimate the root, c, of the equation $f(x) = 0$ as the mid-point between a and b as
   $$c_k = \frac{a+b}{2}$$
3. Now check the following
   a) If f(a).f(c) < 0 , then the root lies between a and c; then b = c.
   b) If f(a).f(c) > 0 , then the root lies between c and b; then a = c
   c) If f(c) = 0; then the root is c. Stop the algorithm if this is true.
4. Find the new estimate of the root
   $$c_k = \frac{a+b}{2}$$
5. Compare the absolute approximate error $|c_k - c_{k-1}|$ or absolute relative approximate error $\frac{|c_k - c_{k-1}|}{|c_k|}$ with the pre-specified relative error tolerance ϵ.
   where
   c_k = estimated root from present iteration
   c_{k-1}= estimated root from previous iteration

   $If\ |c_k - c_{k-1}| < \epsilon\ or\ \frac{|c_k - c_{k-1}|}{|c_k|} < \epsilon$ then exit
   Else
   k = k + 1
   go to Step 3
   Note one should also check whether the number of iterations is more than the maximum

number of iterations allowed.

Advantages of bisection method

1. The bisection method is always convergent. Since the method brackets the root, the method is guaranteed to converge.
2. As iterations are conducted, the interval gets halved. So one can guarantee the error in the solution of the equation.

Drawbacks of bisection method

1. The convergence of the bisection method is slow as it is simply based on halving the interval.
2. If one of the initial guesses is closer to the root, it will take larger number of iterations to reach the root.
3. If a function $f(x)$ is such that it just touches the x-axis (Figure 5) such as $f(x) = x^2 = 0$ it will be unable to find a, and b such that f(a).f(b) < 0

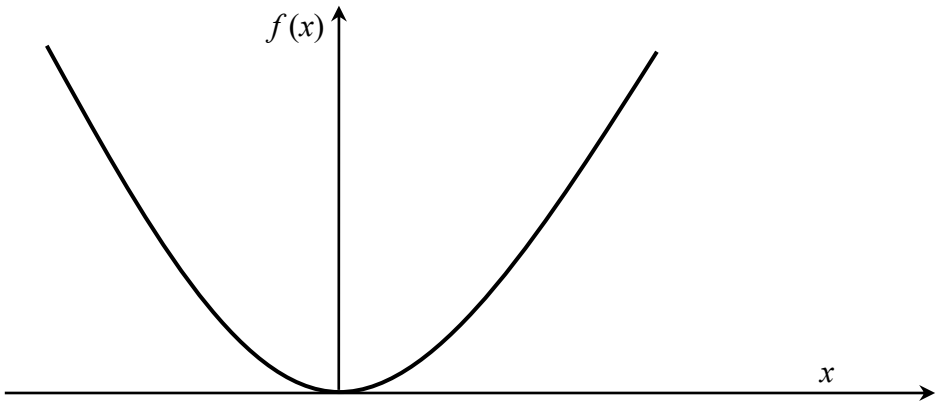


Figure 5 The equation $f(x) = x^2 = 0$ has a single root at $x = 0$ that cannot be bracketed.

4. For functions $f(x)$ where there is a singularity(A singularity in a function is defined as a point where the function becomes infinite.) and it reverses sign at the singularity, the bisection method may converge on the singularity (Figure 6). An example includes

$$f(x) = \frac{1}{x}$$

where a = -2 and b - 3, are valid initial guesses which satisfy f(a).f(b) < 0

However, the function is not continuous and the theorem that a root exists is also not applicable.

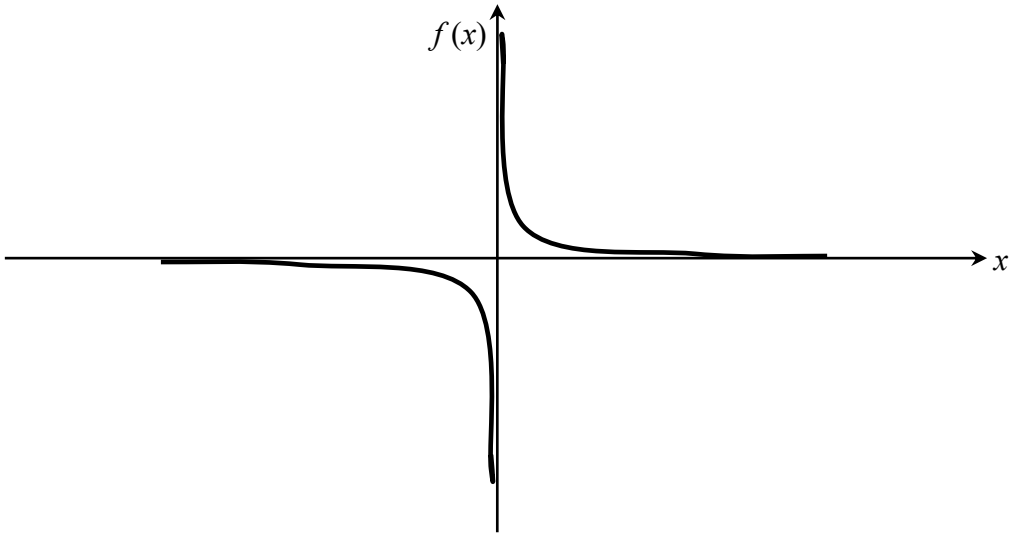
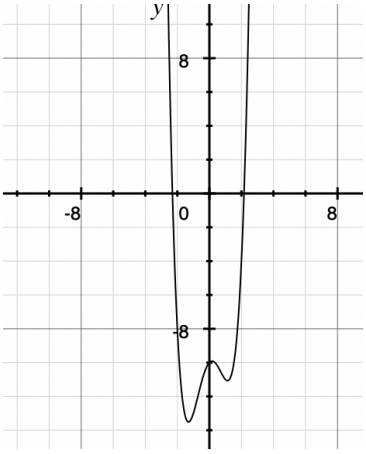
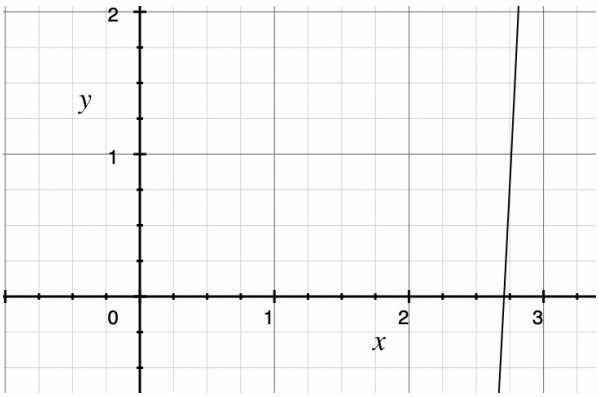


Figure 6 The equation $f(x)=\frac{1}{x}=0$ has no root but changes sign.

Example
$x^4 - 3x^2 + x - 10 = 0$
Find an interval of unit length which contains this root.
Solution
$f(0) = 0 - 0 + 0 - 10 = -10$
$f(1) = 1 - 3 + 1 - 10 = -11$
$f(2) = 16 - 12 + 2 - 10 = -4$
$f(3) = 81 - 27 + 3 - 10 = 47$
An interval of unit length which contains
the root of
$x^4 - 3x^2 + x - 10 = 0$ is (2,3)



Example
Find the real root of the equation $x^3 - 4x - 9 = 0$ by Bisection method correct. Take a = 2.706, b = 2.707, $\epsilon = 0.0001$



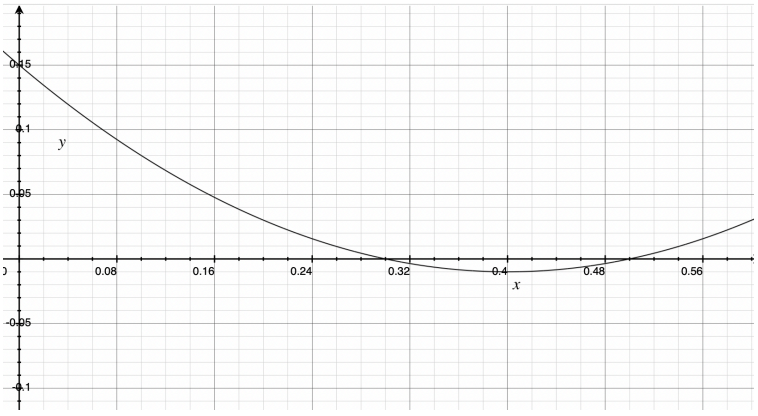
a	f(a)	b	f(b)	c = (a+b)/2	f(c)	$\lvert c_k - c_{k-1} \rvert$
2.706	– 0.009488	2.707	0.008487	2.7065	– 0.0005025	-
2.7065	– 0.0005025	2.707	0.008487	2.70675	0.003992	0.00025
2.7065	– 0.0005025	2.70675	0.003992	2.706625	0.001744	0.000125
2.7065	– 0.0005025	2.706625	0.001744	2.7065625		0.0000625

$\epsilon = 0.0001$, $|c_k - c_{k-1}| = 0.0000625 < \epsilon$
Hence, the root is 2.7065625, correct to three decimal places.

Example
The quadratic (x - 0.3)(x - 0.5) obviously has zeros at 0.3 and 0.5.

a. Why is the interval [0.1, 0.6] not a satisfactory starting interval for bisection?

Ans. Both a and b have same sign
b. What are good starting intervals for each root?
Ans. [0.2,0.4] for 0.3 and [0.4, 0.6] for 0.5
c. If you start with [0,0.491] which root is reached with bisection?
Which root is reached from [0.31, 1.0]?
Ans. 0.3, 0.5



Example
Find the real root of the equation $x^3 - x - 1$ *by Bisection method correct to two decimal places. Take a = 1.25, b = 1.5,*
$\epsilon = 0.01$

a	f(a)	b	f(b)	c = (a+b)/2	f(c)	$\lvert c_k - c_{k-1} \rvert$
1.25	– 0.2969	1.5	0.875	1.375	0.2246	-
1.25	– 0.2969	1.375	0.2246	1.3125	-0.0515	0.0625
1.3125	-0.0515	1.375	0.2246	1.3438	0.0826	0.0313
1.3125	-0.0515	1.3438	0.0826	1.3281	0.0146	0.0157
1.3125	-0.0515	1.3281	0.0146	1.3203		0.0078

False-Position Method or Regula Falsi Method

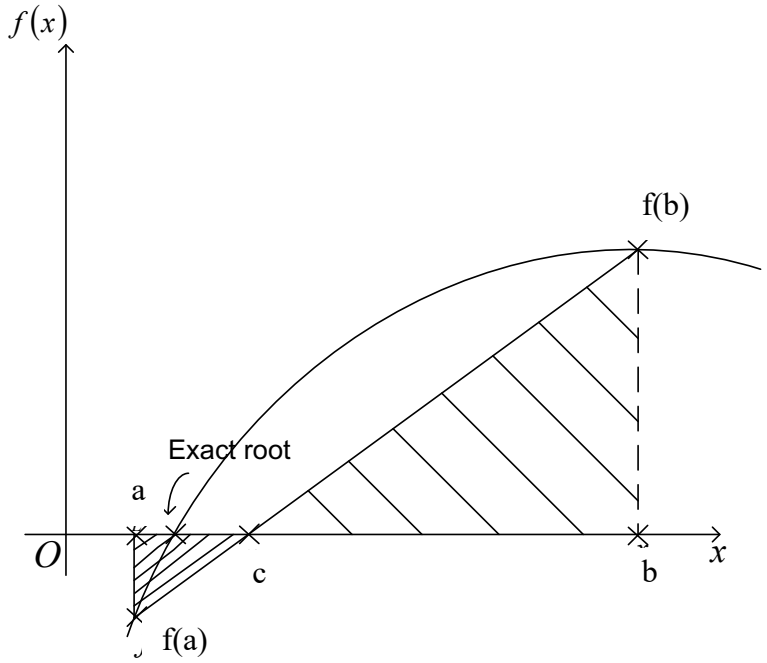


Figure 7 False-Position Method

In the bisection method, we identify proper values of a (lower bound value) and b (upper bound value) for the current bracket.
However, in the example shown in Figure 7, the bisection method may not be efficient because it does not take into consideration that f(a) is much closer to the zero of the function f(x) as compared to f(b). In other words, the next predicted root c would be closer to a (in the example as shown in Figure 1), than the mid-point between a and b. The false-position method takes advantage of this observation mathematically by drawing a straight line from the function value at a to the function value at b, and estimates the root as where it crosses the *x*-axis.

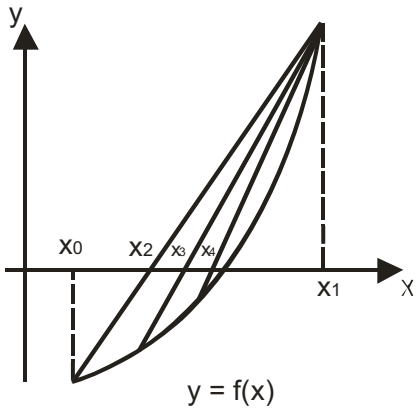


Figure 8 False-Position Method

So, in regular falsi method, to take into consideration the function values at a and b, straight line is drawn, joining (a,f(a))and(b,f(b)).
The point, where it cuts X axis is, the new estimate of the root. Mathematically, estimate of formula for c can be derived as follows:
Equation of straight line joining (a, f (a)) and (b, f (b)) is given by :

$$y - f(a) = \frac{f(b) - f(a)}{b - a}(x - a)$$

Point of intersection of straight line with X – axis being on X-axis is (c , 0). Thus (c , 0) satisfies equation (1). Substituting (c , 0) in place of (x, y) in equation (1) gives

$$0 - f(a) = \frac{f(b) - f(a)}{b - a}(c - a)$$

$$c - a = \frac{-(b - a)f(a)}{f(b) - f(a)}$$

$$c = a - \frac{(b - a)f(a)}{f(b) - f(a)}$$

$$c = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

False-Position Algorithm

The steps to apply the false-position method to find the root of the equation $f(x) = 0$ are as follows.

1. Choose a and b as two guesses for the root such that f(a).f(b) < 0 , or in other words, $f(x)$ changes sign between a and b. Let k =1
2. Estimate the root, c, of the equation $f(x) = 0$ as the mid-point between a and b as
   $$c_k = \frac{af(b) - bf(a)}{f(b) - f(a)}$$
3. Now check the following
   d) If f(a).f(c) < 0 , then the root lies between a and c; then b = c.
   e) If f(a).f(c) > 0 , then the root lies between c and b; then a = c
   f) If f(c) = 0; then the root is c. Stop the algorithm if this is true.
4. Find the new estimate of the root
   $$c_k = \frac{af(b) - bf(a)}{f(b) - f(a)}$$
5. Compare the absolute approximate error $|c_k - c_{k-1}|$ or absolute relative approximate error $\frac{|c_k - c_{k-1}|}{|c_k|}$ with the pre-specified relative error tolerance ϵ.
   where
   c_k = estimated root from present iteration
   c_{k-1}= estimated root from previous iteration

   $If\ |c_k - c_{k-1}| < \epsilon\ or\ \frac{|c_k - c_{k-1}|}{|c_k|} < \epsilon$ then exit
   Else
   K = k + 1
   go to Step 3
   Note one should also check whether the number of iterations is more than the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user about it.

Different Stopping Criterion

The sequence of $c_k's$ approaches the root. That is why, in case of the Bisection section algorithm, iterative process was stopped, when
$|c_k - c_{k-1}| < \epsilon$
It is called x – tolerance criterion (X-TOL). The recent most c k is the estimate of the root. There is another stopping criterion called function tolerance (F–TOL). At root, function value is zero, so if

estimate is quite near the root then function value would be small. Hence, many times, one would like to stop when

$|f(c_k)|<\delta$

where δ is some small positive constant like ϵ.

If slope of the curve is small near the root, curve is almost horizontal, and then function tolerance may not be appropriate stopping criterion, because curve is rising slowly, function values in neighbourhood of the root are going to be small, so even if estimate c is not sufficiently near the root, one may stop.

On the other hand, if it is expected that slope is high near the root or sequence of approximations c_k's may converge to the root, like in case of almost vertical graph, then

function tolerance ensures that estimate is good approximation to the root.

In simple words, for f (x) , if $f'(x)$ is high near the root, FTOL would be better to use as stopping criterion

Many times instead of absolute error $|c_k - c_{k-1}|$, bound on Relative error $\frac{|c_k - c_{k-1}|}{|c_k|} < \epsilon$ is used .This needs to be applied when looking answer correct to certain number of significant digits. For example, the root could be like 1.2749×10^{-12}

So here, if we go for $|c_k - c_{k-1}| < 10^{-5}$we shall get answer zero, but if we go for $\frac{|c_k - c_{k-1}|}{|c_k|} < 10^{-5}$, we would get root correct to 5 significant digits. It root is required to be correct to N significant digits, then one should apply

$$\frac{|c_k - c_{k-1}|}{|c_k|} < 10^{-N}$$

Advantages:

1. It is faster than Bisection Method.
2. It is also simple.
3. It guarantees convergence.
4. Only one function evaluation per iteration is required.

Disadvantages:

1. Not self starting. One needs two initial guesses a and b such that f(a)·f(b) < 0.
2. Though, faster than Bisection Method, still regarded as slow.
3. In rare cases, it may become slower than Bisection Method(Figure 9).

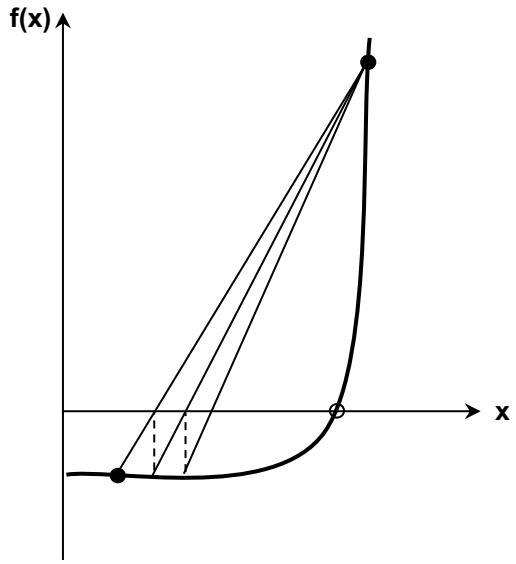


Figure 9. Slow convergence of the false-position method for $f(x) = x^{10} - 1 = 0$

Example

Eg 1. *Find the real root of the equation $x^3 - 4x - 9 = 0$ by* False position method *correct.*

Take a = 2, b = 3, $\epsilon = 0.01$

a	f(a)	b	f(b)	c	f(c)	$\lvert c_k - c_{k-1} \rvert$
2	-9	3	6	2.6	– 1.824	-
2.6	– 1.824	3	6	2.6933	-0.2372	0.0933
2.6933	-0.2372	3	6	2.7049	-0.0289	0.0116
2.7049	-0.0289	3	6	2.7063	-0.0035	0.0014

$\epsilon = 0.01$, $\lvert c_k - c_{k-1} \rvert = 0.0014$
Hence, the root is 2.7063, correct to two decimal places.

Example
Find the real root of the equation $x^3 - x - 1$ *by* False position method *correct to three decimal places.*
$\epsilon = 0.001$

a	f(a)	b	f(b)	c	f(c)	$\lvert c_k - c_{k-1} \rvert$
1	– 1	1.5	0.875	1.2667	-0.2344	-
1.2667	-0.2344	1.5	0.875	1.316	-0.037	0.0493
1.316	-0.037	1.5	0.875	1.3234	-0.0055	0.0074
1.3234	-0.0055	1.5	0.875	1.3245	-0.0008	0.0011
1.3245	-0.0008	1.5	0.875	1.3247	-0.0001	0.0002

$\epsilon = 0.001$ $\lvert c_k - c_{k-1} \rvert = 0.0078 < \epsilon$
The approximate real root is 1.3247

Example

- In bisection and the method of false position, one tests to see that a function changes sign between x = a and x = b. If this is done by seeing if f(a) *f(b) <0, underflow may occur. Is there an alternative way to make the test that avoids this problem?

Newton-Raphson Method

Methods such as the bisection method and the false position method of finding roots of a nonlinear equation f(x) =0 require bracketing of the root by two guesses. Such methods are called *bracketing methods*. These methods are always convergent since they are based on reducing the interval between the two guesses so as to zero in on the root of the equation.

In the Newton-Raphson method, the root is not bracketed. In fact, only one initial guess of the root is needed to get the iterative process started to find the root of an equation. The method hence falls in the category of *open methods*. Convergence in open methods is not guaranteed but if the method does converge, it does so much faster than the bracketing methods.

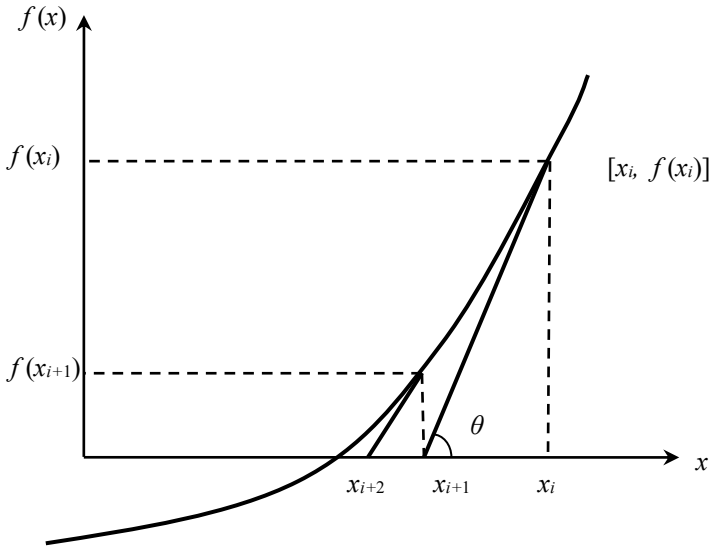


Figure 10 Newton-Raphson method.

The Newton-Raphson method is based on the principle that if the initial guess of the root of f(x) =0 is at x_i, then if one draws the tangent to the curve at $f(x_i)$, the point x_{i+1} where the tangent crosses the x-axis is an improved estimate of the root (Figure 10).

Using the definition of the slope of a function, at $x = x_i$

$$f'(x_i) = \tan\theta = \frac{f(x_i)-0}{x_i - x_{i+1}},$$

which gives

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

This equation is called the Newton-Raphson formula for solving nonlinear equations of the form $f(x)=0$. So starting with an initial guess, x_i, one can find the next guess, x_{i+1}, by using the above equation. One can repeat this process until one finds the root within a desirable tolerance.

Algorithm

The steps of the Newton-Raphson method to find the root of an equation $f(x)=0$ are

1. Let k =1, Evaluate $f'(x)$

2. Use an initial guess of the root, x_i, to estimate the new value of the root, x_{i+1}, as

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

3. Compare the absolute approximate error $|c_k - c_{k-1}|$ or absolute relative approximate error $\frac{|c_k - c_{k-1}|}{|c_k|}$ with the pre-specified relative error tolerance ϵ.

where

c_k = estimated root from present iteration
c_{k-1}= estimated root from previous iteration

If $|c_k - c_{k-1}| < \epsilon$ *or* $\frac{|c_k - c_{k-1}|}{|c_k|} < \epsilon$ then exit

Else
k = k + 1
go to Step 2
Note one should also check whether the number of iterations is more than the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user about it.

Advantages:

1. Only one initial guess is needed.
2. Very rapid convergence. It has quadratic convergence which implies, number of correct figures in the estimate is nearly doubled at each successive step.

Disadvantages:

1. If initial guess is not sufficiently near the root, it may diverge.
2. Function must be differentiable.
3. In case of multiple roots, pace of convergence slows down,
4. There can be many instances, when Newton Raphson may fail to converge,

Example
Find the real root of the equation $x^3 - 4x - 9 = 0$ *by* Newton Raphson method *correct to two decimal places. Take* $x_0 = 2.5$,
$f'(x) = 3x^2 - 4$

x_i	$f(x_i)$	$f'(x_i)$	x_{i+1}	
2.5	-3.375	14.75	2.7288	-
2.7288	0.4046	18.3393	2.7067	0.0221
2.7067	0.004	17.9795	2.7065	0.0002

$\epsilon = 0.001$ $|c_k - c_{k-1}| = 0.0002 < \epsilon$
The approximate real root is 2.7065

Example
Find the real root of the equation $x^3 - x - 1$ *by* Newton Raphson method *correct to three decimal places. Take* $x_0 = 1.5$,
$\epsilon = 0.001$
$f'(x) = 3x^2 - 1$

x_i	f(x_i)	f '(x_i)	x_{i+1}	$\lvert c_k - c_{k-1} \rvert$
1.5	0.875	5.75	1.3478	-
1.3478	0.1007	4.4499	1.3252	0.0226
1.3252	0.0021	4.2646	1.3247	0.0005

Example
This quadratic has two nearly equal roots:
P (x) is $x^2 - 4x + 3.9999$

a. Which root do you get with Newton's method starting at x = 2.1?
b. Repeat part (a) but starting with x = 1.9.

c. What happens with Newton's method starting from
x = 2.0?

Solution
a. Starting from x_0 = 2.1, convergence is to x = 2.0108
b. Starting from x_0 = 1.9, convergence is to x = 1.9892.
c. Starting from x_0 = 2.0 fails, f'(2.0) = zero.

Secant Method

Secant method is an open method. That is, no more interval under consideration needs to bracket the root, though it still requires two initial guesses. Let us denote these initial guesses by x_{-1} and x_0 . The formula for generating the sequence of approximations is

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Firstly, initial choices of x_{-1} and x_0 need not be bracketing the root. So, that is, root need not lie within the endpoints $[x_{-1},x_0]$or$[x_0,x_{-1}]$.That is $f(x_{-1})$and $f(x_0)$can be of same sign. No more, one needs to ensure that $f(x_{-1}).f(x_0)<0$.

Though, in practice one usually chooses x-1, x_0 as ones which bracket the root, there is no such compulsion.

Secondly, in computation of next approximation, older approximation is discarded. Recent most two approximations are used for iteration, irrespective of function values at these end points. So graphically, straight line joining (x_{i-1} , f (x_{i-1})) and (x_i, f (x_i)) is drawn to generate x_{i+1} , next approximation. No more, function values sign are checked to ensure that root lies between (x_{i-1} , f (x_{i-1})) and (x_i , f (x_i)). Because secant is drawn joining$(x_{i-1},f(x_{i-1}))$and$(x_i,f(x_i))$, it is called Secant Method.

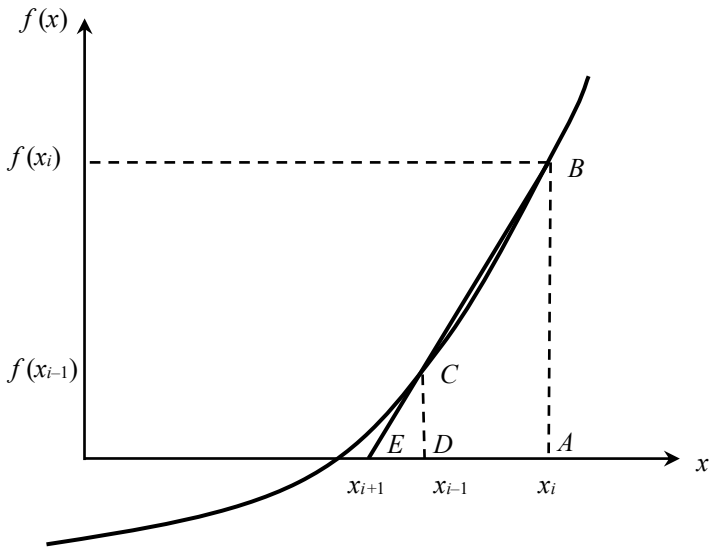


Figure 15 Geometrical representation of the secant method.

Secant Algorithm

The steps of the Secant method to find the root of an equation $f(x)=0$ are

1. Let k = 1, Evaluate f(x_i), f(x_i)
2. Use an initial guess of the root, x_i, to estimate the new value of the root, x_{i+1}, as

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

3. Compare the absolute approximate error $|c_k - c_{k-1}|$ or absolute relative approximate error $\frac{|c_k - c_{k-1}|}{|c_k|}$ with the pre-specified relative error tolerance ϵ.

where

c_k = estimated root from present iteration

c_{k-1}= estimated root from previous iteration

$If\ |c_k - c_{k-1}| < \epsilon\ or\ \frac{|c_k - c_{k-1}|}{|c_k|} < \epsilon$ then exit

Else

K = k + 1

go to Step 2

Note one should also check whether the number of iterations is more than the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user about it.

Advantages:

1. No constraint of end points of interval to converge
2. Very rapid convergence

Disadvantages:

1. If initial guess is not sufficiently near the root, it may diverge.

Difference between the secant and false-position method

In the false position method, the latest estimate of the root replaces whichever of the original values yielded a function value with the same sign as f(x). The root is always bracketed by the bonds and the method will always converge.

For the secant method, x_{i+1} replaces x_i and x_i replaces x_{i-1}. As a result, the two values can sometimes lie in the same side of the root and lead to divergence. But when, the secant method converges, it usually does it at a quicker rate.

Example
Find the real root of the equation $x^3 - 4x - 9 = 0$ *by* Secant Method *correct.*
Take $x_0 = 2$, $x_1 = 3$, $\epsilon = 0.01$

x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$	$\lvert x_k - x_{k-1} \rvert$
2	-9	3	6	2.6	− 1.824	-
3	6	2.6	− 1.824	2.6933	-0.2372	0.0933
2.6	− 1.824	2.6933	-0.2372	2.7072	0.012	0.0139
2.6933	-0.2372	2.7072	0.012	2.7065	0	0.0007

$\epsilon = 0.001, |x_k - x_{k-1}| = 0.0007$
Hence, the root is 2.7065, correct to three decimal places.

Example
Find the real root of the equation $x^3 - x - 1 = 0$ *by* Secant Method *correct. Take* $x_0 = 1$, $x_1 = 1.5$, $\epsilon = 0.001$

x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$	$\lvert x_k - x_{k-1} \rvert$
1	-1	1.5	0.875	1.2667	− 0.2344	-
1.5	0.875	1.2667	− 0.2344	1.316	-0.037	0.0493
1.2667	− 0.2344	1.316	-0.037	1.3252	0.0021	0.0092
1.316	-0.037	1.3252	0.0021	1.3247	0	0.0005

$\epsilon = 0.001, |x_k - x_{k-1}| = 0.0005$
Hence, the root is 1.3247, correct to three decimal places.

Example
This polynomial obviously has roots at x = 2 and at x = 4; one is a double root, the other a triple root:

$$f(x) = (x-2)^3(x-4)^2$$
$$= x^5 - 14x^4 + 76x^3 - 200x^2 + 256x - 128$$

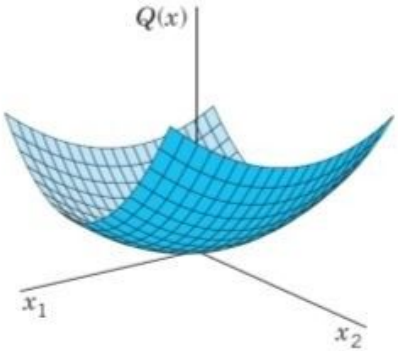
a. Which root can you get with bisection? Which root can't you get?
b. Repeat part (a) with the secant method. Ans. Both roots can be reached with secant method
c. If you begin with the interval [1,5], which root will you get with
(1) bisection, (2) the secant method, (3) false position?
d. Use Newton's method with $x_0 = 3$. Does it converge? To which root?

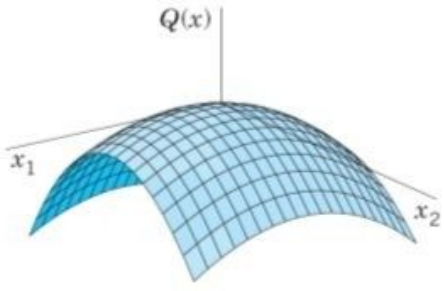
Solution
a. we can get the root 2. Getting Interval for 4 is not possibleb. Both roots can be reached with secant method
c.
(1) Converge to 2, (2) It will converge to 2 (3) Converge to 2
d. Derivate is zero, so the method does not converge. Yes $x_0 = 4.1$, can converge to x = 4

x_i	$f(x_i)$	$f'(x_i)$	x_{i+1}	$\lvert c_k - c_{k-1} \rvert$
4.1	0.0926	1.9845	4.0533	
4.0533	0.0246	0.9594	4.0277	0.0256
4.0277	0.0064	0.4707	4.0141	0.0136
4.0141	0.0016	0.233	4.0071	0.007
4.0071	0.0004	0.1159	4.0036	0.0035

Gradient Descent

- The derivative of a function at a chosen input value describes the rate of change of the function near that input value. The process of finding a derivative is called differentiation.
- Optimization technique called **gradient descent**, which has seen major application in machine learning models.
- Gradient descent is an optimization technique that can find the *minimum* of an **objective function**. It is a greedy technique that finds the optimal solution by taking a step in the direction of the maximum rate of decrease of the function.

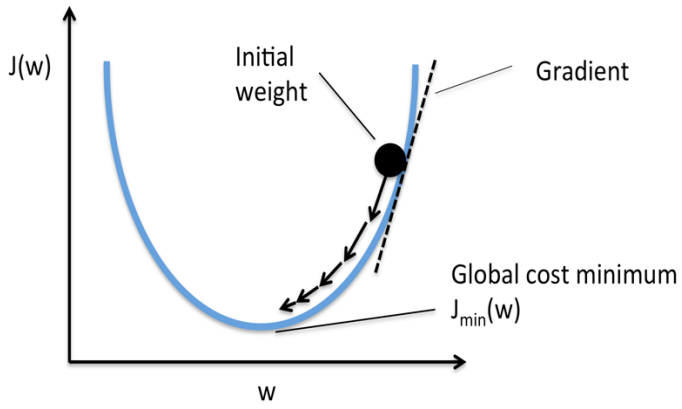




- The basic intuition behind gradient descent can be illustrated by a hypothetical scenario. A blind-folded person is stuck in the mountains and is trying to get down (i.e., trying to find the global minimum). There is heavy fog such that visibility is extremely low. Therefore, the path down the mountain is not visible, so they must use local information to find the minimum.

- They can use the method of gradient descent, which involves looking at the steepness of the hill at their current position, then proceeding in the direction with the steepest descent (i.e., downhill). If they were trying to find the top of the mountain (i.e., the maximum), then they would proceed in the direction of steepest ascent (i.e., uphill).

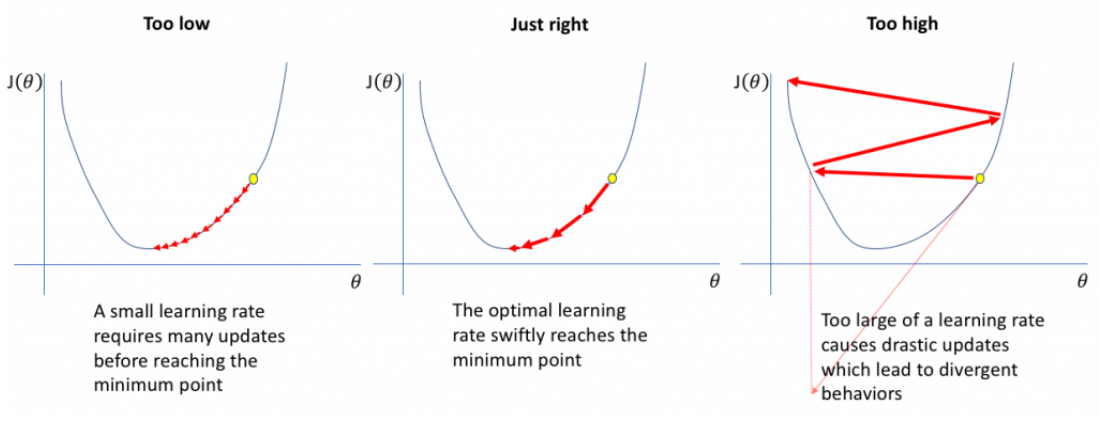
- Using this method, they would eventually find their way down the mountain or possibly get stuck in some hole (i.e., local minimum or saddle point), like a mountain lake. However, assume also that the steepness of the hill is not immediately obvious with simple observation, but rather it requires a sophisticated instrument to measure, which the person happens to have at the moment. It takes quite some time to measure the steepness of the hill with the instrument, thus they should minimize their use of the instrument if they wanted to get down the mountain before sunset. The difficulty then is choosing the frequency at which they should measure the steepness of the hill so not to go off track.
- In this analogy, the person represents the algorithm, and the path taken down the mountain represents the sequence of parameter settings that the algorithm will explore. The steepness of the hill represents the slope of the error surface at that point. The instrument used to measure steepness is differentiation



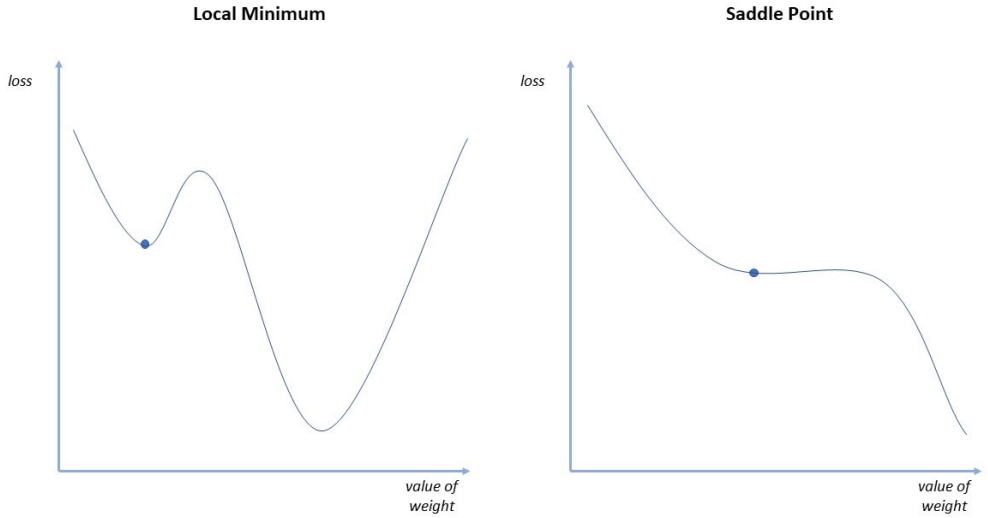
- The direction they choose to travel in aligns with the gradient of the error surface at that point. The amount of time they travel before taking another measurement is the step size.
- A gradient simply measures the change in all weights with regard to the change in error. You can also think of a gradient as the slope of a function. The higher the gradient, the steeper the slope and the faster a model can learn. But if the slope is zero, the model stops learning. In mathematical terms, a gradient is a partial derivative with respect to its inputs

Learning Rate

- Importance of the Learning Rate
- How big the steps are gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights.
- For gradient descent to reach the local minimum we must set the learning rate to an appropriate value, which is neither too low nor too high. This is important because if the steps it takes are too big, it may not reach the local minimum because it bounces back and forth between the convex function of gradient descent. If we set the learning rate to a very small value, gradient descent will eventually reach the local minimum but that may take a while



- **Challenges with gradient descent**
- Local minima and saddle points
- For convex problems, gradient descent can find the global minimum with ease, but as nonconvex problems emerge, gradient descent can struggle to find the global minimum, where the model achieves the best results.
- Recall that when the slope of the cost function is at or close to zero, the model stops learning. A few scenarios beyond the global minimum can also yield this slope, which are local minima and saddle points. Local minima mimic the shape of a global minimum, where the slope of the cost function increases on either side of the current point. However, with saddle points, the negative gradient only exists on one side of the point, reaching a local maximum on one side and a local minimum on the other. Its name inspired by that of a horse's saddle.
- Noisy gradients can help the gradient escape local minimums and saddle points.



Questions

1. The smallest positive root of the equation

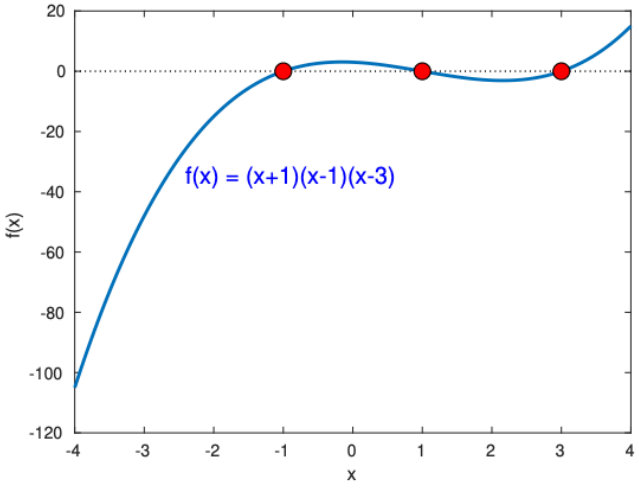
    a. $f(x)=x^4 -3x^2 +x-10=0$
    1. Find an interval of unit length which contains this root.
    b. Find the root using any of the methods

2. Find the root of the following using Bisection Method:
    a. $f(x) = x^3 - x - 1 = 0$
    b. $f(x) = xe^x = 1$
3. Find the root of the following using Newton Raphson Method:
    a. Find an approximation to $\sqrt{5}$ to four decimal places
4. Find the root of using Bisection method, False Position, Newton-Raphson, Secant upto accuracy 0.001.
5. $x^3 - x - 4 = 0$

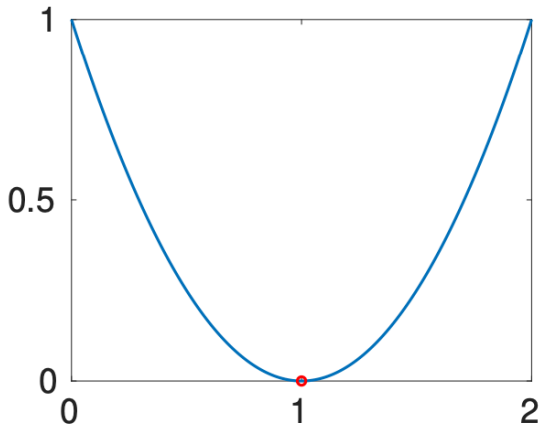
    Ans. 1.7963219
6. $x^4 - x - 10 = 0$

    Ans. 1.8556

7. The function f(x) = (x+ 1)(x−1)(x−3) is pictured in the plot. If the bisection algorithm is applied with initial interval [−4, 4], Which root of f(x) will the bisection method converge to? Which intervals can you take to find the other roots? Find the roots using bisection method.



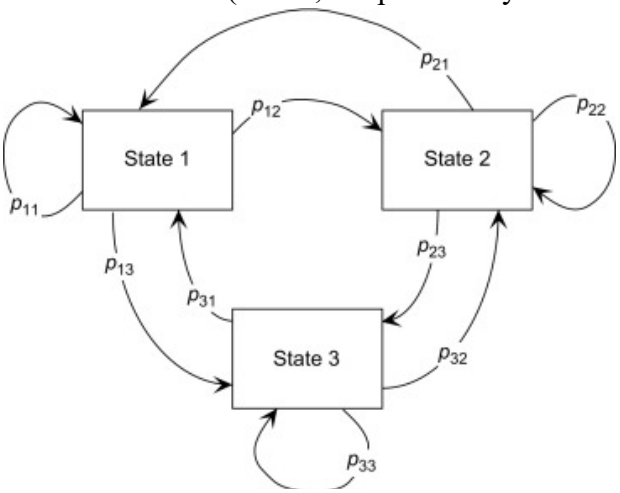
8. Let f(x) = $x^2 - 2x + 1$. Can bisection method be used to approximate the root of the function f(x) pictured.



Eigen Vector Applications

Markov Chains

- Markov chain represents an evolving process consisting of a finite number of *states*.
- At each step or point in time, the process may be in any one of the states; at the next step, the process can remain in its present state or switch to one of the other states.
- The state to which the process moves at the next step and the probability of its doing so depend *only* on the present state and not on the past history of the process.
- These probabilities are called **transition probabilities** and are assumed to be constants (that is, the probability of moving from state *i* to state *j* is always the same).



Stochastic Matrix

A vector with nonnegative entries that add up to 1 is called a **probability vector.** A **stochastic matrix** is a square matrix whose columns are probability vectors. A **Markov chain** is a sequence of probability vectors $x_0, x_1, x_2 \ldots$ together with a stochastic matrix or transition matrix P , such that

$x_1 = \mathrm{P}\, x_0, x_2 = \mathrm{P}\, x_1, x_3 = \mathrm{P}\, x_2, \ldots..$

Thus the Markov chain is described by the first-order difference equation

$x_{k+1} = Px_k$ for k = 0, 1, 2, . . .

From this result it follows that we can compute an arbitrary state vector iteratively once we know x_0 and P. In other words, a Markov chain is completely determined by its transition probabilities and its initial state.

We know that

$x_2 = Px_1 = P(Px_0) = P^2x_0$

In general, $x_k = P^k x_0$ for k = 0, 1, 2, . . .

P_{ij}^k is the probability of moving from state j to state i in k transitions.

Steady State Vector

When a Markov chain of vectors in R^n describes a system or a sequence of experiments, the entries in x_k list, respectively, the probabilities that the system is in each of n possible states, or the probabilities that the outcome of the experiment is one of n possible outcomes. For this reason, x_k is often called a **state vector.**

- If P is a stochastic matrix, then a steady-state vector (or equilibrium vector) for P is a probability vector q such that Pq = q
- It can be shown that every stochastic matrix has a **steady-state vector.**

- If P is an n×n regular stochastic matrix, then P has a unique steady-state vector q. Further, if x_0 is any initial state and

$x_{k+1} = Px_k$ for k = 0,1,2,...,
Then the Markov chain $\{x_k\}$ converges to q as $k \to \infty$.

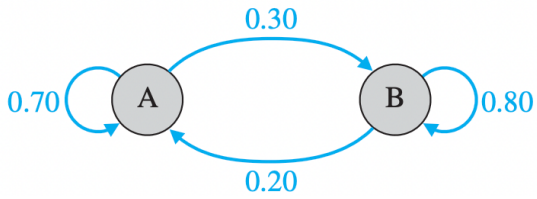
Markov Chain and Eigen Values and Vectors

- If P is the n × n transition matrix of a Markov chain, then 1 is an eigenvalue of P
- Let P be an n × n transition matrix with eigenvalue λ.

$$|\lambda| \leq 1$$

Example 1

- The sample consists of 200 people, each of whom is asked to try two brands of toothpaste over a period of several months. Based on the responses to the survey, the research team compiles the following statistics about toothpaste preferences.
- Of those using Brand A in any month, 70% continue to use it the following month, while 30% switch to Brand B; of those using Brand B in any month, 80% continue to use it the following month, while 20% switch to Brand A.



- In the toothpaste survey, there are just two states—using Brand A and using Brand B—and the transition probabilities are those indicated in Figure. Suppose that, when the survey begins, 120 people are using Brand A and 80 people are using Brand B. How many people will be using each brand 1 month later? 2 months later?
- The number of Brand A users after 1 month will be 70% of those initially using Brand A (those who remain loyal to Brand A) plus 20% of the Brand B users (those who switch from B to A):

0.70(120) + 0.20(80) = 100

- Similarly, the number of Brand B users after 1 month will be a combination of those who switch to Brand B and those who continue to use it: 0.30(120) + 0.80(80) = 100
- We can summarize these two equations in a single matrix equation:

$$\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} 120 \\ 80 \end{bmatrix} = \begin{bmatrix} 100 \\ 100 \end{bmatrix}$$

Let's call the matrix P = $\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix}$ and label the vectors $x_0 = \begin{bmatrix} 120 \\ 80 \end{bmatrix}$ and $x_1 = \begin{bmatrix} 100 \\ 100 \end{bmatrix}$

Thus, we have $x_1 = P\,x_0$

We can think of the columns as being labeled with the present states and the rows as being labeled with the next states:

From:

A B To:

$\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix}$ A
B

The columns of P are probability vectors; any square matrix with this property is called a stochastic matrix.

Let's call the matrix P = $\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix}$ and label the vectors $x_0 = \begin{bmatrix} 120 \\ 80 \end{bmatrix}$ and $x_1 = \begin{bmatrix} 100 \\ 100 \end{bmatrix}$

Thus, we have $x_1 = P\,x_0$

- Extending the notation, let x_k be the vector whose components record the distribution of toothpaste users after k months. To determine the number of users of each brand after 2 months have elapsed, we simply apply the same reasoning, starting with x_1 instead of x_0. We obtain

$x_2 = \mathrm{P}x_1$

- $\begin{bmatrix}0.7 & 0.2\\ 0.3 & 0.8\end{bmatrix}\begin{bmatrix}100\\ 100\end{bmatrix} = \begin{bmatrix}90\\ 110\end{bmatrix}$

from which we see that there are now 90 Brand A users and 110 Brand B users.

- Suppose, in Example, we wanted to keep track of not the actual numbers of toothpaste users but, rather, the relative numbers using each brand. We could convert the data into percentages or fractions by dividing by 200, the total number of users. Thus, we would start with $x_0 = \begin{bmatrix}\frac{120}{200}\\ \frac{80}{200}\end{bmatrix} = \begin{bmatrix}0.60\\ 0.40\end{bmatrix}$

to reflect the fact that, initially, the Brand A–Brand B split is 60%–40%.
Vectors such as these, with nonnegative components that add up to 1, are called **probability vectors.**
What will happen to the distribution of toothpaste users in the long run? Let's work with probability vectors as state vectors. Continuing our calculations (rounding to three decimal places), we find

$$\mathbf{x}_0 = \begin{bmatrix}0.60\\ 0.40\end{bmatrix}, \mathbf{x}_1 = \begin{bmatrix}0.50\\ 0.50\end{bmatrix}, \mathbf{x}_2 = P\mathbf{x}_1 = \begin{bmatrix}0.70 & 0.20\\ 0.30 & 0.80\end{bmatrix}\begin{bmatrix}0.50\\ 0.50\end{bmatrix} = \begin{bmatrix}0.45\\ 0.55\end{bmatrix},$$

$$\mathbf{x}_3 = P\mathbf{x}_2 = \begin{bmatrix}0.70 & 0.20\\ 0.30 & 0.80\end{bmatrix}\begin{bmatrix}0.45\\ 0.55\end{bmatrix} = \begin{bmatrix}0.425\\ 0.575\end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix}0.412\\ 0.588\end{bmatrix}, \mathbf{x}_5 = \begin{bmatrix}0.406\\ 0.594\end{bmatrix},$$

$$\mathbf{x}_6 = \begin{bmatrix}0.403\\ 0.597\end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix}0.402\\ 0.598\end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix}0.401\\ 0.599\end{bmatrix}, \mathbf{x}_9 = \begin{bmatrix}0.400\\ 0.600\end{bmatrix}, \mathbf{x}_{10} = \begin{bmatrix}0.400\\ 0.600\end{bmatrix},$$

Eventually 40% of the toothpaste users in the survey will be using Brand A and 60% will be using Brand B.

$\begin{bmatrix}0.7 & 0.2\\ 0.3 & 0.8\end{bmatrix}\begin{bmatrix}0.40\\ 0.60\end{bmatrix} = \begin{bmatrix}0.40\\ 0.60\end{bmatrix}$

A state vector $\begin{bmatrix}0.40\\ 0.60\end{bmatrix}$ is called a steady state vector since Px = x

Example 2
Find the steady state vector for

$$\mathrm{P} = \begin{bmatrix}0.7 & 0.2\\ 0.3 & 0.8\end{bmatrix}$$

Solution
First, solve the equation P x = x.
Px − x = 0
Px − Ix = 0
(P − I)x = 0 since Ix = x.

$\mathrm{P} - \mathrm{I} = \begin{bmatrix}0.7 & 0.2\\ 0.3 & 0.8\end{bmatrix} - \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} = \begin{bmatrix}-0.3 & 0.2\\ 0.3 & -0.2\end{bmatrix}$

To find all solutions of (P − I)**x** = **0**

$$\begin{bmatrix}-0.3 & 0.2 & 0\\ 0.3 & -0.2 & 0\end{bmatrix} \sim \begin{bmatrix}-0.3 & 0.2 & 0\\ 0 & 0 & 0\end{bmatrix} \sim \begin{bmatrix}1 & -0.2/0.3 & 0\\ 0 & 0 & 0\end{bmatrix}$$

Then $x_1 = {}^{2}/_{3}\, x_2$ and x_2 is free. The general solution is $\begin{bmatrix}2\\3\end{bmatrix}$

The probability vector is given by

$$\begin{bmatrix}{}^{2}/_{5}\\{}^{3}/_{5}\end{bmatrix} = \begin{bmatrix}0.40\\0.60\end{bmatrix}$$

Verify that Pq = q $\begin{bmatrix}0.7 & 0.2\\0.3 & 0.8\end{bmatrix}\begin{bmatrix}0.40\\0.60\end{bmatrix} = \begin{bmatrix}0.40\\0.60\end{bmatrix}$

Example 3

A small remote village receives radio broadcasts from two radio stations, a news station and a music station. Of the listeners who are tuned to the news station, 70% will remain listening to the news after the station break that occurs each half hour, while 30% will switch to the music station at the station break. Of the listeners who are tuned to the music station, 60% will switch to the news station at the station break, while 40% will remain listening to the music. Suppose everyone is listening to the news at 8:15 a.m.

a. Give the stochastic matrix that describes how the radio listeners tend to change stations at each station break. Label the rows and columns.

b. Give the initial state vector.

c. What percentage of the listeners will be listening to the music station at 9:25 a.m. (after the station breaks at 8:30 and 9:00 a.m.)?

d. Find the steady state vector for the Markov chain?

e. At some time late in the day, what fraction of the listeners will be listening to the news?

Solution

a. The stochastics matrix is given by P =

From:
News Music To:

$$\begin{bmatrix}0.7 & 0.6\\0.3 & 0.4\end{bmatrix} \quad \begin{matrix}News\\Music\end{matrix}$$

b. The initial state vector $x_0 = \begin{bmatrix}1\\0\end{bmatrix}\begin{matrix}News\\Music\end{matrix}$

c. $x_1 = \text{P}\, x_0 = \begin{bmatrix}0.7 & 0.6\\0.3 & 0.4\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0.7\\0.3\end{bmatrix}$

$x_2 = \text{P}\, x_1 = \begin{bmatrix}0.7 & 0.6\\0.3 & 0.4\end{bmatrix}\begin{bmatrix}0.7\\0.3\end{bmatrix} = \begin{bmatrix}0.67\\0.33\end{bmatrix}$

33% will be listening to music station

d. P − I $= \begin{bmatrix}0.7 & 0.6\\0.3 & 0.4\end{bmatrix} - \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \begin{bmatrix}-0.3 & 0.6\\0.3 & -0.6\end{bmatrix}$

To find all solutions of (P − I)**x** = **0**

$$\begin{bmatrix}-0.3 & 0.6 & 0\\0.3 & -0.6 & 0\end{bmatrix} \sim \begin{bmatrix}-0.3 & 0.6 & 0\\0 & 0 & 0\end{bmatrix} \sim \begin{bmatrix}1 & -2 & 0\\0 & 0 & 0\end{bmatrix}$$

Then $x_1 = 2x_2$ and x_2 is free. The general solution is $\begin{bmatrix}2\\1\end{bmatrix}$

The probability vector is given by $\begin{bmatrix}{}^{2}/_{3}\\{}^{1}/_{3}\end{bmatrix}$

e. At some time late in the day, ${}^{2}/_{3}$ of the listeners will be listening to the news

Example 4

On any given day, a student is either healthy or ill. Of the students who are healthy today, 95% will be healthy tomorrow. Of the students who are ill today, 55% will still be ill tomorrow.

a. What is the stochastic matrix for this situation?

b. Suppose 20% of the students are ill on Monday. What fraction or percentage of the students are likely to be ill on Tuesday? On Wednesday?

c. If a student is well today, what is the probability that he or she will be well two days from now?

d. Find the steady state vector for the Markov chain?

e. What is the probability that after many days a specific student is ill? Does it matter if that person is ill today?

Solution

Today:

Healthy Ill Tomorrow:

a. The stochastics matrix is given by P = $\begin{bmatrix} 0.95 & 0.45 \\ 0.05 & 0.55 \end{bmatrix}$ $\begin{matrix} Healthy \\ Ill \end{matrix}$

b. The initial state vector $x_0 = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$

$x_1 = P\,x_0 = \begin{bmatrix} 0.95 & 0.45 \\ 0.05 & 0.55 \end{bmatrix}\begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 0.85 \\ 0.15 \end{bmatrix}$

$x_2 = P\,x_1 = \begin{bmatrix} 0.95 & 0.45 \\ 0.05 & 0.55 \end{bmatrix}\begin{bmatrix} 0.85 \\ 0.15 \end{bmatrix} = \begin{bmatrix} 0.875 \\ 0.125 \end{bmatrix}$

15% of the students are likely to be ill on Tuesday and 12.5% on Wednesday.

c. The initial state vector $x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$x_1 = P\,x_0 = \begin{bmatrix} 0.95 & 0.45 \\ 0.05 & 0.55 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix}$

$x_2 = P\,x_1 = \begin{bmatrix} 0.95 & 0.45 \\ 0.05 & 0.55 \end{bmatrix}\begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix} = \begin{bmatrix} 0.925 \\ 0.075 \end{bmatrix}$

If a student is well today, the probability that he or she will be well two days from now = 92.5%

d. $P - I = \begin{bmatrix} 0.95 & 0.45 \\ 0.05 & 0.55 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.05 & 0.45 \\ 0.05 & -0.45 \end{bmatrix}$

To find all solutions of $(P - I)\mathbf{x} = \mathbf{0}$

$$\begin{bmatrix} -0.05 & 0.45 & 0 \\ 0.05 & -0.45 & 0 \end{bmatrix} \sim \begin{bmatrix} -0.05 & 0.45 & 0 \\ 0 & 0 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & -9 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Then $x_1 = 9x_2$ and x_2 is free. The general solution is $\begin{bmatrix} 9 \\ 1 \end{bmatrix}$

The probability vector is given by $\begin{bmatrix} 9/10 \\ 1/10 \end{bmatrix}$

e. The probability that after many days a specific student is ill = $1/10$

No, it does not matter if that person is ill today.

Example 5

A laboratory animal may eat any one of three foods each day. Laboratory records show that if the animal chooses one food on one trial, it will choose the same food on the next trial with a probability of 50%, and it will choose the other foods on the next trial with equal probabilities of 25%.

a. What is the stochastic matrix for this situation?

b. If the animal chooses food #1 on an initial trial, what is the probability that it will choose food #2 on the second trial after the initial trial?
c. Which food will the animal prefer after many trials?

Solution
Let the foods be labelled "1," "2," and "3." Then the animals' behavior is given by the table

From:

	1	2	3	To:
	0.5	0.25	0.25	1
	0.25	0.5	0.25	2
	0.25	0.25	0.5	3

a. The stochastics matrix is given by $P = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}$

b. There are two trials after the initial trial, so we calculate x_2 .The initial state vector $x_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$x_1 = P\,x_0 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.25 \\ 0.25 \end{bmatrix}$

$x_2 = P\,x_1 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}\begin{bmatrix} 0.5 \\ 0.25 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 0.375 \\ 0.3125 \\ 0.3125 \end{bmatrix}$

Thus the probability that the animal will choose food #2 is 0.3125

c. $P - I = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.25 & 0.25 \\ 0.25 & -0.5 & 0.25 \\ 0.25 & 0.25 & -0.5 \end{bmatrix}$

To find all solutions of $(P - I)\mathbf{x} = \mathbf{0}$

$$\begin{bmatrix} -0.5 & 0.25 & 0.25 & 0 \\ 0.25 & -0.5 & 0.25 & 0 \\ 0.25 & 0.25 & -0.5 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Then x_3 is free. The general solution is $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

The probability vector is given by $\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$

Thus in the long run each food will be preferred equally.

Example 6
The Demographic Research Unit of the California State Department of Finance supplied data for the following migration matrix, which describes the movement of the United States population during 1989. In 1989, about 11.7% of the total population lived in California. What percentage of the total population would eventually live in California if the listed migration probabilities were to remain constant over many years?

Solution

From:

	CA	Rest of US	To:
	0.9821	0.0029	CA
	0.0179	0.9971	Rest of US

The stochastics matrix is given by $P = \begin{bmatrix} 0.9821 & 0.0029 \\ 0.0179 & 0.9971 \end{bmatrix}$

$P - I = \begin{bmatrix} 0.9821 & 0.0029 \\ 0.0179 & 0.9971 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.0179 & 0.0029 \\ 0.0179 & -0.0029 \end{bmatrix}$

To find all solutions of $(P - I)\mathbf{x} = \mathbf{0}$

$$\begin{bmatrix} -0.0179 & 0.0029 & 0 \\ 0.0179 & -0.0029 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & -0.162011 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Then x_2 is free. The general solution is $\begin{bmatrix} 0.16211 \\ 1 \end{bmatrix}$

The probability vector is given by $\begin{bmatrix} 0.16211/1.16211 \\ 1/1.16211 \end{bmatrix} = \begin{bmatrix} 0.139423 \\ 0.860577 \end{bmatrix}$

Thus about 13.9% of the total U.S. population would eventually live in California.

Questions

1. What is a transition matrix?
2. For a problem modeled by a Markov matrix, what does a steady-state mean?
3. For a Markov chain, what does the state at a given time step depend on?
4. Build a Markov Chain model for weather predicting in UIUC during summer. We observed that:
   - a sunny day is 60% likely to be followed by another sunny day, 10% likely followed by a rainy day and 30% likely followed by a cloudy day;
   - a rainy day is 40% likely to be followed by another rainy day, 20% likely followed by a sunny day and 40% likely followed by a cloudy day;
   - a cloudy day is 40% likely to be followed by another cloudy day, 30% likely followed by a rainy day and 30% likely followed by a sunny day.

What is the transition matrix? If the weather on day 0 is known to be rainy, what is the weather on day 1? Determine the steady Steady state vector?

Answers

4.

$$M = \begin{bmatrix} 0.6 & 0.2 & 0.3 \\ 0.1 & 0.4 & 0.3 \\ 0.3 & 0.4 & 0.4 \end{bmatrix}$$

QUADRATIC FORMS

A **quadratic form** on R^n is a function Q defined on R^n whose value at a vector **x** in R^n can be computed by an expression of the form $Q(x) = x^T Ax$ where, A is an $n \times n$ symmetric matrix. The matrix A is called the **matrix of the quadratic form**.

Example 1 Let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Compute $x^T Ax$ for the following matrices.

a. $A = \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix}$

b. $A = \begin{bmatrix} 3 & -2 \\ -2 & 7 \end{bmatrix}$

Solution

a. $x^T Ax = [x_1 \quad x_2] \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [x_1 \quad x_2] \begin{bmatrix} 4x_1 \\ 3x_2 \end{bmatrix} = 4x_1^2 + 3x_2^2$

b. $x^T Ax = [x_1 \quad x_2] \begin{bmatrix} 3 & -2 \\ -2 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [x_1 \quad x_2] \begin{bmatrix} 3x_1 - 2x_2 \\ -2x_1 + 7x_2 \end{bmatrix}$

$= x_1(3x_1 - 2x_2) + x_2(-2x_1 + 7x_2)$

$= 3x_1^2 - 2x_1x_2 - 2x_2x_1 + 7x_2^2$

$= 3x_1^2 - 4x_1x_2 + 7x_2^2$

The presence of $-4x_1x_2$ in the quadratic form in Example 1(b) is due to the -2 entries off the diagonal in the matrix A.

In contrast, the quadratic form associated with the diagonal matrix A in Example 1(a) has no x_1x_2 *cross-product* term.

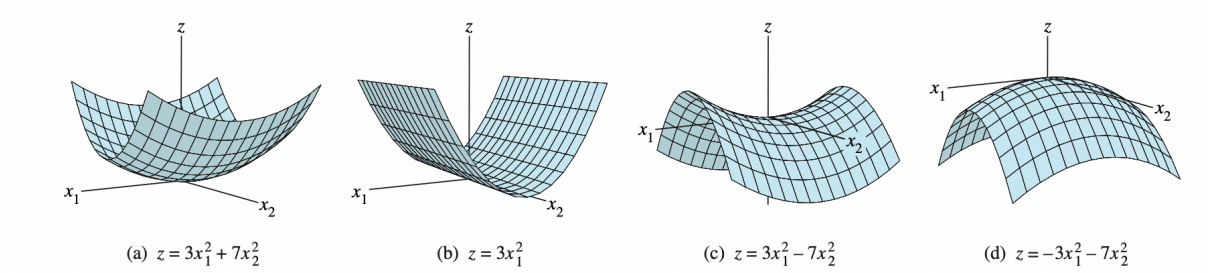


FIGURE 4 Graphs of quadratic forms.

Figure 4 displays the graphs of four quadratic forms. For each point x = (x_1 , x_2) in the domain of a quadratic form Q, a point (x_1, x_2, z) is plotted, where z = Q(x). Notice that except at x = 0, the values of Q(x) are all positive in Fig. 4(a) and all negative in Fig. 4(d). The horizontal cross sections of the graphs are ellipses in Figs. 4(a) and 4(d) and hyperbolas in 4(c).

Types of Quadratic Forms

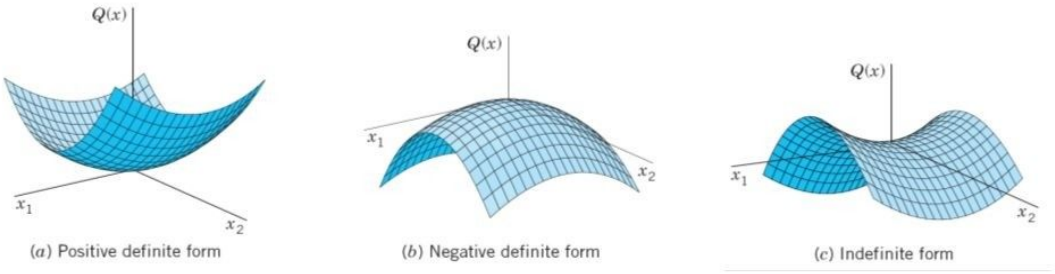
A quadratic form Q is:

a) Positive definite if Q(x) > 0 for all x $\neq$ 0,
b) Negative definite if Q(x) < 0 for all x $\neq$ 0,
c) Indefinite if Q(x) assumes both positive and negative values.

The classification of a quadratic form is often carried over to the matrix of the form.

Thus a **positive definite matrix** A is a *symmetric* matrix for which the quadratic form $x^T Ax$ is positive definite.

Quadratic forms in two variables



Tests for Positive Definiteness

Each of the following tests is a necessary and sufficient condition for the real symmetric matrix A to be *positive definite*:

1. All the eigenvalues of A satisfy $\lambda_i > 0$.
2. All the upper left submatrices A_k have positive determinants.
3. All the pivots (without row exchanges) satisfy $d_k > 0$
4. $x^T Ax > 0$ for all nonzero real vectors x.

ie. For $A = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$

1. Eigen values: $\lambda_1 > 0$, $\lambda_2 > 0$
2. $Submatrices\ determinant$: $a > 0,\ ac - b^2 > 0$
3. $Pivots$: $a > 0,\ \frac{ac-b^2}{a} > 0$
4. $x^T Ax > 0$

Example

Is $A = \begin{bmatrix} 2 & 6 \\ 6 & 20 \end{bmatrix}$ positive definite?

Solution

1. Eigen value 11 - 3 $\sqrt{13} > 0$, 11 + 3 $\sqrt{13} > 0$
2. $Submatrices\ determinant\ a > 0,\ ac - b^2 > 0$;
   (i) 2 > 0, (ii) 40-36 > 0

3. $Pivots\ a > 0,\ \frac{ac-b^2}{a} > 0$;
   (i) 2 > 0, (ii) $\frac{40\text{-}36}{2} > 0$

4. $\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 6 \\ 6 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2x_1 + 6x_2 \\ 6x_1 + 20x_2 \end{bmatrix}$

$= 2x_1^2 + 12x_1x_2 + 20x_2^2$

$= 2(x_1 + 3x_2)^2 + 2x_2^2$

$x^T Ax > 0$

Example

Is $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ positive definite?

Solution

1. Eigen value $2 - \sqrt{2} > 0$, $2 > 0$, $2 + \sqrt{2} > 0$
2. $Submatrices\ determinant$
   (i) 2 (ii) 4-1 = 3
   (iii) 2(4 - 1) + 1(-2 + 0) + 0 = 4
3. $Pivots$ 2 > 0, 3/2 > 0, 4/2 > 0
4. $[x_1 \quad x_2 \quad x_3]\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

$= 2x_1^2 + 2x_2^2 + 2x_3^2 - 2x_1x_2 - 2x_2x_3$

$x^T Ax > 0$

Singular Value Decomposition

The SVD is closely associated with the eigenvalue-eigenvector factorization QDQ^T of a positive definite matrix. The eigenvalues are in the diagonal matrix D.
The eigenvector matrix Q *is orthogonal* ($Q^TQ = I$) because eigenvectors of a symmetric matrix can be chosen to be orthonormal. For most matrices that is not true, and for rectangular matrices it is ridiculous (eigenvalues undefined). But now we allow the Q on the left and the Q^T on the right to be *any two orthogonal matrices* U and V^T—not necessarily transposes of each other. Then every matrix will split into $A = U\Sigma V^T$

The diagonal (but rectangular) matrix Σ has eigenvalues from A^TA. Those positive entries (also called sigma) will be $\sigma_1,...,\sigma_r$. They are the *singular values* of A. They fill the first r places on the main diagonal of Σ—when A has rank r. The rest of Σ is zero.
With rectangular matrices, the key is almost always to consider A^TA and AA^T.

Singular Value Decomposition: Any m by n matrix A can be factored into
$A = U\Sigma V^T$ = (orthogonal)(diagonal)(orthogonal).
The columns of U (m by m) are eigenvectors of AA^T, and the columns of V (n by n) are eigenvectors of A^TA. The r singular values on the diagonal of Σ (m by n) are the square roots of the nonzero eigenvalues of both AA^T and A^TA.

For positive definite matrices, Σ is D and $U\Sigma V^T$ is identical to QDQ^T. For other symmetric matrices, any negative eigenvalues in D become positive in Σ.

U and V give orthonormal bases for all four fundamental subspaces:
first r columns of U : column space of A
last m−r columns of U: left nullspace of A
first r columns of V : row space of A
last n−r columns of V: nullspace of A

The SVD chooses those bases in an extremely special way. They are more than just orthonormal. *When A multiplies a column* v_j *of V , it produces* σ_j *times a column of U* . That comes directly from $AV = U\Sigma$, looked at a column at a time.
Eigenvectors of AA^T and A^TA must go into the columns of U and V :
$AA^T = (U\Sigma V^T)(V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$ and, similarly, $A^TA = V\Sigma^T\Sigma V^T$. (1)
U *must be the eigenvector matrix for* AA^T. The eigenvalue matrix in the middle is $\Sigma\Sigma^T$—which is m by m with $\sigma_1^2,..., \sigma_r^2$ on the diagonal.
From the $A^TA = V\Sigma^T\Sigma V^T$, the V *matrix must be the eigenvector matrix for* A^TA. The diagonal matrix $\Sigma^T\Sigma$ has the same $\sigma_1^2,..., \sigma_r^2$, but it is n by n.

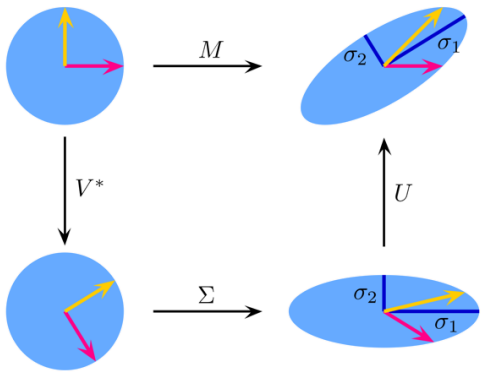
The decomposition of A involves an m×n "diagonal" matr ix Σ of the form

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$$

m − r rows

n − r columns

where D is an r×r diagonal matrix for some r not exceeding the smaller of m and n.
(If r equals m or n or both, some or all of the zero matrices do not appear.)



$$M = U \cdot \Sigma \cdot V^*$$

Illustration of the singular value decomposition $\mathbf{U\Sigma V}^{\mathrm{T}}$ of a real 2×2 matrix **M**.
Top: The action of **M**, indicated by its effect on the unit disc *D* and the two canonical unit vectors e_1 and e_2.
Left: The action of $\mathbf{V}^{\mathrm{T}}$, a rotation, on *D*, e_1, and e_2.
Bottom: The action of **Σ**, a scaling by the singular values σ_1 horizontally and σ_2 vertically.
Right: The action of **U**, another rotation.

The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler. $A = U\Sigma V^T$
Where A is the real n×m matrix that we wish to decompose, U is an m×m matrix, Σ represented by the upper case Greek letter sigma is an m×n diagonal matrix, and V^T is the V transpose of an n × n matrix.

U and V are orthogonal matrices with orthonormal eigenvectors chosen from AA^{T} and $A^{\mathrm{T}}A$
The columns of the U matrix are called the left-singular vectors of A, and the columns of V are called the right-singular vectors of A.
The diagonal values in the Σ matrix are known as the singular values of the original matrix A which are equal to the root of the positive eigenvalues of AA^T or $A^T A$ (both matrices have the same positive eigenvalues)
So we have A = (orthogonal matrix)(diagonal matrix)(orthogonal matrix)$^{\mathrm{T}}$
We can arrange eigenvectors in different orders to produce *U* and *V*. To standardize the solution, we order the eigenvectors such that vectors with higher eigenvalues come before those with smaller values. SVD is not unique.

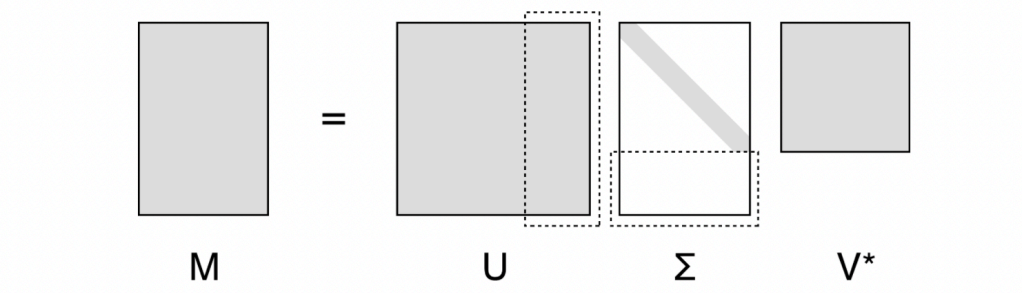
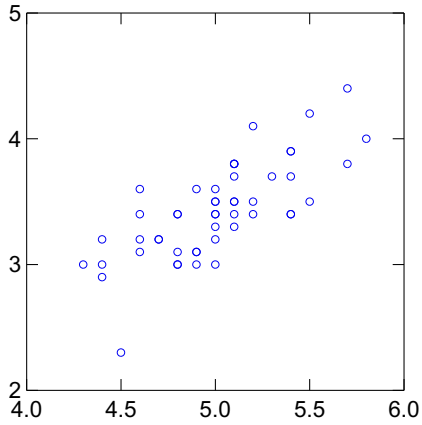
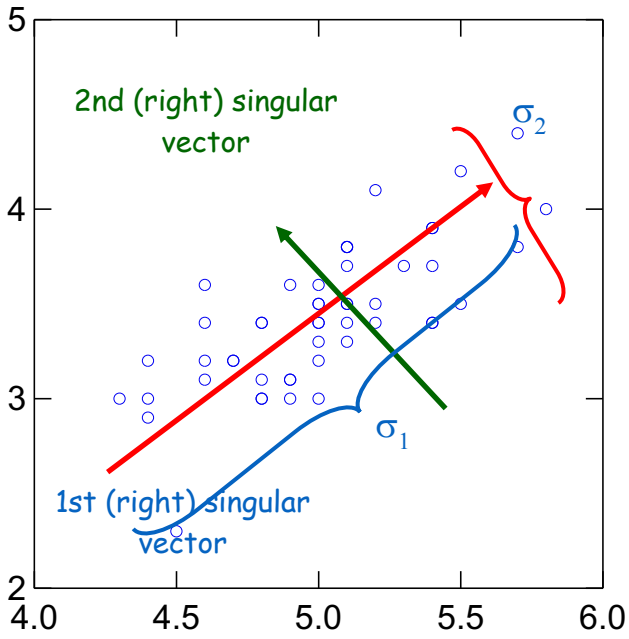


Figure 7: The canonical diagram of the SVD decomposition of a matrix M. The columns of U are the orthonormal left singular vectors; Σ is a diagonal matrix of singular values; and the rows of $V^\top$ are the orthonormal right singular vectors. The dashed areas are padding.



Let the blue circles represent m data points .
Then, the SVD of the m-by-2 matrix of the data will return …



1st (right) singular vector: direction of maximal variance,

2nd (right) singular vector: direction of maximal variance, after removing the projection of the data along the first singular vector.

σ_1: measures how much of the data variance is explained by the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

Example

Construct a singular value decomposition of A

$$A = \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 4 & 8 \\ 11 & 7 \\ 14 & -2 \end{bmatrix} \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix} = \begin{bmatrix} 80 & 100 & 40 \\ 100 & 170 & 140 \\ 40 & 140 & 200 \end{bmatrix}$$

The eigenvalues of A^TA are $\lambda_1 = 360$, $\lambda_2 = 90$, and $\lambda_3 = 0$. Corresponding unit eigen vectors are, respectively,

$$v_1 = \begin{bmatrix} 1/3 \\ 2/3 \\ 2/3 \end{bmatrix}, v_2 = \begin{bmatrix} -2/3 \\ -1/3 \\ 2/3 \end{bmatrix}, v_3 = \begin{bmatrix} 2/3 \\ -2/3 \\ 1/3 \end{bmatrix}$$

The maximum value of $\|A\mathbf{x}\|^2$ is 360, attained when $\mathbf{x}$ is the unit vector $\mathbf{v}_1$. The vector $A\mathbf{v}_1$ is a point on the ellipse in Figure 1 farthest from the origin.

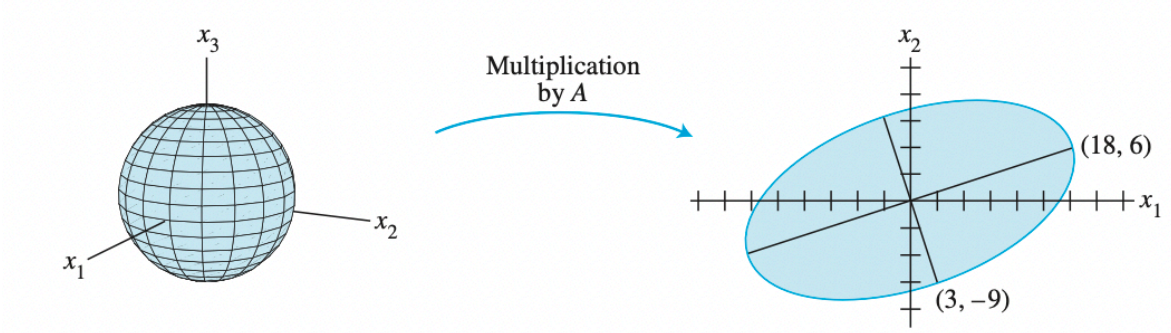


FIGURE 1 A transformation from $\mathbb{R}^3$ to $\mathbb{R}^2$.

$$A\,v_1 = \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix} \begin{bmatrix} 1/3 \\ 2/3 \\ 2/3 \end{bmatrix} = \begin{bmatrix} 18 \\ 6 \end{bmatrix}$$

For $\|\mathbf{x}\| = 1$, the maximum value of $\|A\mathbf{x}\|$ is $\|A\mathbf{v}_1\| = \sqrt{360} = 6\sqrt{10}$

The effect of A on the unit sphere in R3 is related to the quadratic form $\mathbf{x}^T(A^TA)\mathbf{x}$. In fact, the entire geometric behavior of the transformation $\mathbf{x} \mapsto A\mathbf{x}$ is captured by this quadratic form

Since the eigenvalues of A^TA are 360, 90, and 0, the singular values of A are

$\sigma_1 = \sqrt{360} = 6\sqrt{10}$, $\sigma_2 = \sqrt{90} = 3\sqrt{10}$, $\sigma_3 = 0$

$$A\,v_2 = \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix} \begin{bmatrix} -2/3 \\ -1/3 \\ 2/3 \end{bmatrix} = \begin{bmatrix} 3 \\ -9 \end{bmatrix}$$

This point is on the minor axis of the ellipse in Fig. 1, just as $A\mathbf{v}_1$ is on the major axis. (See Fig. 2.) The first two singular values of A are the lengths of the major and minor semiaxes of the ellipse.

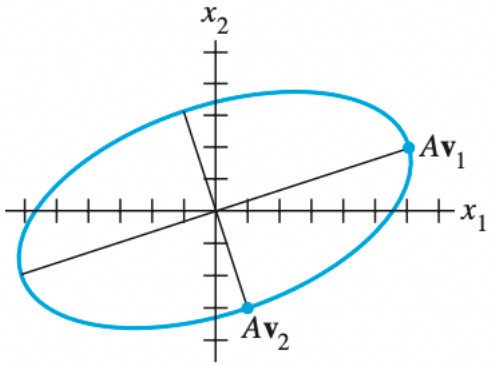


FIGURE 2

$A\mathbf{v}_1$ and $A\mathbf{v}_2$ are orthogonal to each other

Arrange the eigenvalues of A^TA in decreasing order: 360, 90, and 0. The corresponding unit eigenvectors, $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$, are the right singular vectors of A. Using construct V

$$V = [v_1 \quad v_2 \quad v_3] = \begin{bmatrix} 1/3 & -2/3 & 2/3 \\ 2/3 & -1/3 & -2/3 \\ 2/3 & 2/3 & 1/3 \end{bmatrix}$$

The nonzero singular values are the diagonal entries of D. The matrix Σ is the same size as A, with D in its upper-left corner and with 0's elsewhere.

$$D = \begin{bmatrix} 6\sqrt{10} & 0 \\ 0 & 3\sqrt{10} \end{bmatrix}, \Sigma = [D \quad 0] = \begin{bmatrix} 6\sqrt{10} & 0 & 0 \\ 0 & 3\sqrt{10} & 0 \end{bmatrix}$$

Construct U. When A has rank r, the first r columns of U are the normalized vectors obtained from $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$. A has two nonzero singular values, so rank A = 2. $\|A\mathbf{v}_1\| = \sigma_1$ and $\|A\mathbf{v}_2\| = \sigma_2$. Thus

$$u_1 = \frac{1}{\sigma_1}Av_1 = \frac{1}{6\sqrt{10}}\begin{bmatrix} 18 \\ 6 \end{bmatrix} = \begin{bmatrix} 3/\sqrt{10} \\ 1/\sqrt{10} \end{bmatrix}$$

$$u_2 = \frac{1}{\sigma_2}Av_2 = \frac{1}{3\sqrt{10}}\begin{bmatrix} 3 \\ -9 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{10} \\ -3/\sqrt{10} \end{bmatrix}$$

Note that $\{\mathbf{u}_1, \mathbf{u}_2\}$ is already a basis. Thus no additional vectors are needed for U , and U = [**u**1 **u**2]. If additional vectors are needed then we use GramSchmidt Orthogonalization Process.

The singular value decomposition of A is

$$A = \begin{bmatrix} 3/\sqrt{10} & 1/\sqrt{10} \\ 1/\sqrt{10} & -3/\sqrt{10} \end{bmatrix} \begin{bmatrix} 6\sqrt{10} & 0 & 0 \\ 0 & 3\sqrt{10} & 0 \end{bmatrix} \begin{bmatrix} 1/3 & 2/3 & 2/3 \\ -2/3 & -1/3 & -2/3 \\ 2/3 & -2/3 & 1/3 \end{bmatrix}$$

Application of SVD

Finding Structure in Movie Ratings & Consumers

Consider three viewers (Ali, Beatrix, Chandra) rating four different movies (Star Wars, Blade Runner, Amelie, Delicatessen). Their ratings are values between 0 (worst) and 5 (best) and encoded in a data matrix $A \in R^{4\times3}$ as shown in Figure 4.10. Each row represents a movie and each column a user. Thus, the column vectors of movie ratings, one for each viewer, are x_{Ali}, $x_{Beatrix}$, $x_{Chandra}$

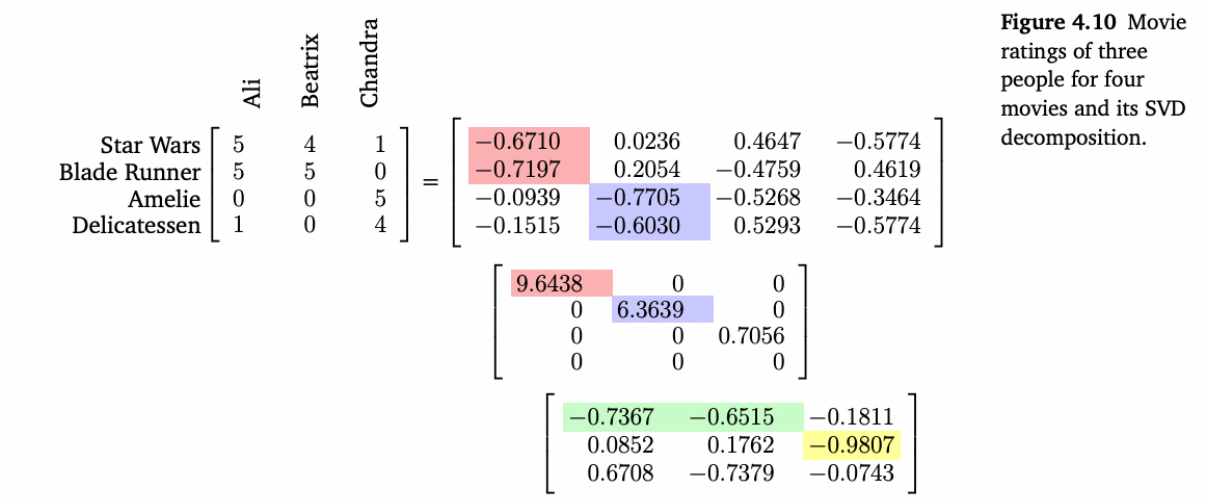


Figure 4.10 Movie ratings of three people for four movies and its SVD decomposition.

- Factoring A using the SVD offers us a way to capture the relationships of how people rate movies, and especially if there is a structure linking which people like which movies.
- Any viewer's specific movie preferences can be expressed as a linear combination of the v_j . Similarly, any movie's likeability can be expressed as a linear combination of the u_i . Therefore, a vector in the domain of the SVD can be interpreted as a viewer in the "space" of stereotypical viewers, and a vector in the codomain of the SVD These two "spaces" correspondingly as a movie in the "space" of stereotypical movies
- The first left-singular vector u_1 has large absolute values for the two science fiction movies and a large first singular value (red shading) Thus, this groups a type of users with a specific set of movies (science fiction theme). Similarly, the first right-singular v_1 shows large absolute values for Ali and Beatrix, who give high ratings to science fiction movies (green shading).
- This suggests that v_1 reflects that Ali and Beatrix are sciencefiction lover. Movies of this genre will be recommended

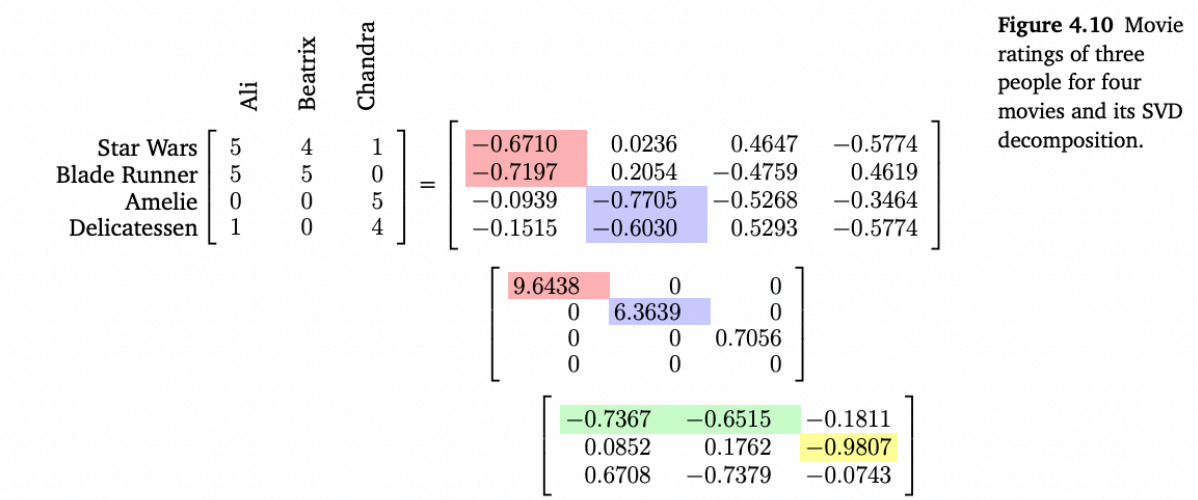


Figure 4.10 Movie ratings of three people for four movies and its SVD decomposition.

- Similarly, u_2, seems to capture a French art house film theme, and v_2 indicates that Chandra is close to an idealized lover of such movies. An idealized science fiction lover is a purist and only loves science fiction movies, so a science fiction lover v_1 gives a rating of zero to everything but science fiction themed—this logic is implied by the diagonal substructure for the singular value matrix Σ. A specific movie is therefore represented by how it decomposes (linearly) into its stereotypical movies. Likewise, a person would be represented by how they decompose (via linear combination) into movie themes.

Latent semantic indexing

One of first uses was in the field of information retrieval. The method that uses SVD is called latent semantic indexing (LSI) or latent semantic analysis.

In LSI, a matrix is constructed of documents and words. When the SVD is done on this matrix, it creates a set of singular values. The singular values represent concepts or topics contained in the documents. This was developed to allow more efficient searching of documents.

A simple search that looks only for the existence of words may have problems if the words are misspelled. Another problem with a simple search is that synonyms may be used, and looking for the existence of a word wouldn't tell you if a synonym was used to construct the document. If a concept is derived from thousands of similar documents, both of the synonyms will map to the same concept.

Condition Number

Most numerical calculations involving an equation A**x** = **b** are as reliable as possible when the SVD of A is used. The two orthogonal matrices U and V do not affect lengths of vectors or angles between vectors. Any possible instabilities in numerical calculations are identified in Σ. If the singular values of A are extremely large or small, roundoff errors are almost inevitable, but an error analysis is aided by knowing the entries in Σ and V .

If A is an invertible n×n matrix, then the ratio $\frac{\sigma_{max}}{\sigma_{min}}$ of the largest and smallest singular values gives the **condition number** of A.

The condition number affects the sensitivity of a solution of A**x** = **b** to changes (or errors) in the entries of A. (Actually, a "condition number" of A can be computed in several ways, but the definition given here is widely used for studying A**x** = **b**.)
Consider the following system of equations

$$(1 + 10^{-9})x + y = 1$$
$$x + (1 - 10^{-9})y = 1$$

Let $Let\ A = \begin{bmatrix} 1 + 10^{-9} & 1 \\ 1 & 1 - 10^{-9} \end{bmatrix}$

It is easily shown that the singular vectors are approximately $\begin{bmatrix} \sqrt{1/2} \\ \sqrt{1/2} \end{bmatrix}$ with singular value 2

and $\begin{bmatrix} \sqrt{1/2} \\ -\sqrt{1/2} \end{bmatrix}$ with singular value 10^{-9} .

Hence, the condition number is 2/ 10^{-9} = 2 · 10^{9}, so this matrix is very ill-conditioned. In solving the linear equation *Ax* = [1,1], note that the singular values of A^{-1} are 10^{9}, and 1/2, and the condition number is again 2 · 10^{9} .

Truncated SVD

Suppose that your data are encoded in an *m* × *n* matrix A . Compute the SVD decomposition A = $U\Sigma V^T$. Choose a value *k* that is substantially less than *m* or *n*. Now let Σ_k be the k×k diagonal matrix with the first *k* singular values. Let V_k^T be the n×kmatrix whose first *k* rows are the same as V^T and the rest 0. Let U_k be the m×k matrix whose first *k* columns are the same as U and the rest 0.
Let A ≈ A_k= $U_k\ \Sigma_k V_k^T$.
A_k closely approximates A, particularly if the singular values that have been dropped are small compared to those that have been retained.
It is not difficult to see that the matrix A_k is of rank-k, and therefore it is viewed as a low-rank approximation of A .
Almost all forms of matrix factorization, including singular value decomposition, are low-rank approximations of the original matrix. Truncated singular value decomposition can retain a surprisingly large level of accuracy using values of k that are much smaller than min {m,n}. This is because only a very small proportion of the singular values are large in real world matrices. In such cases, A_k becomes an excellent approximation of A by retaining the few singular vectors that are large.
A useful property of truncated singular value decomposition is that it is also possible to create a lower dimensional representation of the data by changing the basis toV_k, so that each n-dimensional data point is now represented in only k dimensions. In other words, we change the axes so that the basis vectors correspond to the columns of V_k. This transformation is achieved by post-multiplying the data matrix A with $V_k$to obtain the m × k matrix P_k.
By post-multiplying A ≈ $U_k\Sigma_k V_k^T$ with V_k and using $V_k^T\ V_k = I_k$, we obtain the following:

P_k = A $V_k = U_k \Sigma_k$
Each row of P_k contains a reduced k-dimensional representation of the corresponding row in A. Therefore, we can obtain a reduced representation of the data either by post-multiplying the data matrix with the matrix containing the dominant right singular vectors (i.e., using A V_k), or we can simply scale the dominant left singular vectors with the singular values (i.e., using $U_k \Sigma_k$. Both these types of methods are used in real applications, depending on whether m or n is larger.
The reduction in dimensionality can be very significant in some domains such as images and text.
The matrix can be approximately reconstructed if the SVD is available.
How did we know how many singular values to keep? There are a number of heuristics for the number of singular values to keep. You typically want to keep 90% of the energy expressed in the matrix. To calculate the total energy, you add up all the squared singular values. You can then add squared singular values until you reach 90% of the total. Another heuristic to use when you have tens of thousands of singular values is to keep the first 2,000 or 3,000. This is a little less elegant than the energy method, but it's easier to implement in practice. It's less elegant because you can't guarantee that 3,000 values contain 90% of the energy in any dataset.

Example of Truncated SVD

We provide an example of truncated SVD with the use of a toy text collection, which has 6 documents and 6 words. The (i,j)th in the data matrix D is the frequency of word j in document i. The 6 × 6 data matrix D is defined over the following vocabulary:
lion, tiger, cheetah, jaguar, porsche, ferrari

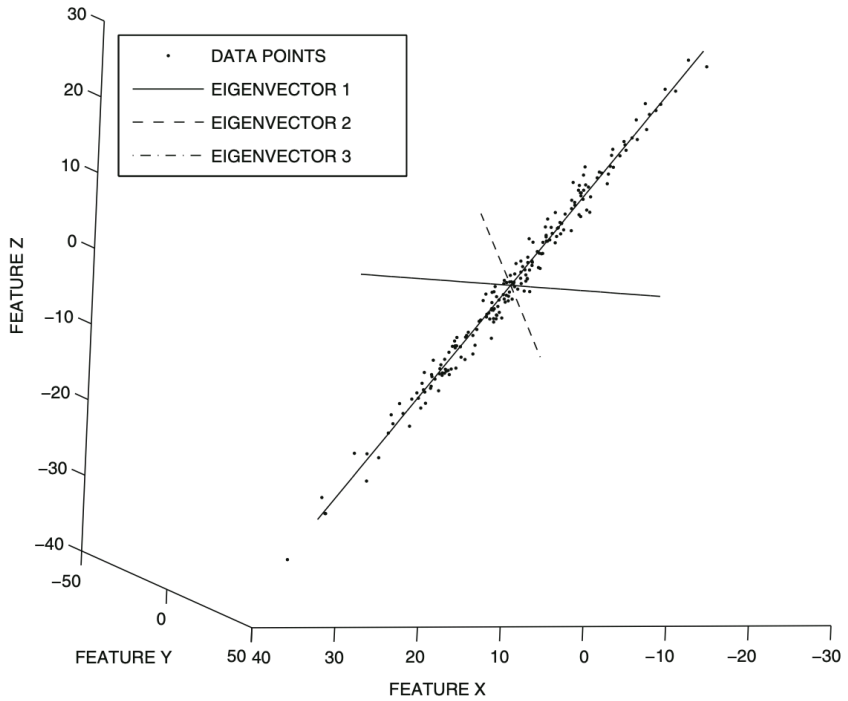


Figure 7.2: Most of energy of the data is retained in the projection along the one or t
largest eigenvectors of the 3 × 3 matrix $D^T D$

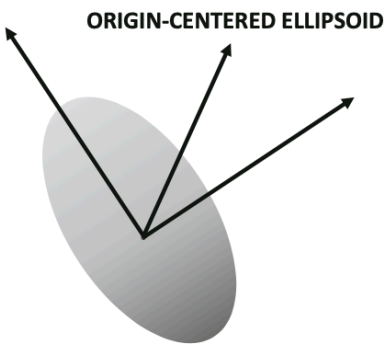


Figure 7.3: SVD models the data to be distributed in an ellipsoid centered at the origin

The frequencies of the words in each document of the data matrix D are illustrated below:

$$D = \begin{pmatrix} & \text{lion} & \text{tiger} & \text{cheetah} & \text{jaguar} & \text{porsche} & \text{ferrari} \\ \text{Document-1} & 2 & 2 & 1 & 2 & 0 & 0 \\ \text{Document-2} & 2 & 3 & 3 & 3 & 0 & 0 \\ \text{Document-3} & 1 & 1 & 1 & 1 & 0 & 0 \\ \text{Document-4} & 2 & 2 & 2 & 3 & 1 & 1 \\ \text{Document-5} & 0 & 0 & 0 & 1 & 1 & 1 \\ \text{Document-6} & 0 & 0 & 0 & 2 & 1 & 2 \end{pmatrix}$$

Note that this matrix represents topics related to both cars and cats. The first three documents are primarily related to cats, the fourth is related to both, and the last two are primarily related to cars. The word "jaguar" is ambiguous because it could correspond to either a car or a cat. We perform an SVD of rank-2 to capture the two latent components in the collection, which is as follows:

D $\approx U_2 \Sigma_2 V_2^T$

$$\approx \begin{pmatrix} -0.41 & 0.17 \\ -0.65 & 0.31 \\ -0.23 & 0.13 \\ -0.56 & -0.20 \\ -0.10 & -0.46 \\ -0.19 & -0.78 \end{pmatrix} \begin{pmatrix} 8.4 & 0 \\ 0 & 3.3 \end{pmatrix} \begin{pmatrix} -0.41 & -0.49 & -0.44 & -0.61 & -0.10 & -0.12 \\ 0.21 & 0.31 & 0.26 & -0.37 & -0.44 & -0.68 \end{pmatrix}$$

$$= \begin{pmatrix} 1.55 & 1.87 & 1.67 & 1.91 & 0.10 & 0.04 \\ 2.46 & 2.98 & 2.66 & 2.95 & 0.10 & -0.03 \\ 0.89 & 1.08 & 0.96 & 1.04 & 0.01 & -0.04 \\ 1.81 & 2.11 & 1.91 & 3.14 & 0.77 & 1.03 \\ 0.02 & -0.05 & -0.02 & 1.06 & 0.74 & 1.11 \\ 0.10 & -0.02 & 0.04 & 1.89 & 1.28 & 1.92 \end{pmatrix}$$

The reconstructed matrix is a very good approximation of the original data matrix D. One can also obtain a 2-dimensional embedding of each row of D as D $V_2 = U_2 \Sigma_2$:

D $V_2 = U_2 \Sigma_2 \approx$

$$\begin{pmatrix} -3.46 & 0.57 \\ -5.44 & 1.03 \\ -1.95 & 0.41 \\ -4.74 & -0.66 \\ -0.83 & -1.49 \\ -1.57 & -2.54 \end{pmatrix}$$

It is clear that the reduced representations of the first three rows are quite similar, which is not surprising. After all the corresponding documents belong to similar topics. At the same time, the reduced representations of the last two rows are also similar. The fourth row seems to be somewhat different because it contains a combination of two topics. Therefore, the latent components seem to capture the hidden "concepts" in the data matrix. In this case, these hidden concepts correspond to cats and cars.

Moore-Penrose Pseudoinverse of A

The Moore-Penrose pseudoinverse can be used for solving systems of linear equations, and for providing a solution to the problem of linear regression. SVD can be used to efficiently compute the Moore-Penrose pseudoinverse.

$$\mathrm{A} \approx U_k \Sigma_k V_k^T$$

- This factorization of A is called Truncated Singular Value Decomposition
- Since the diagonal entries in Σ are nonzero, we can form the following matrix, called the **pseudoinverse** (also, the **Moore–Penrose inverse**) of A:
- $A^+ = V_k \Sigma^{-1} U_k^T$

The effective rank

- The rank of a matrix is the number of independent rows, and the number of independent columns. That can be hard to decide in computations!
- In exact arithmetic, counting the pivots is correct. Real arithmetic can be misleading—but discarding small pivots is not the answer.
- Rank of matrix *A* can be computed by using Gaussian elimination to reduce *A* to row-echelon form and then counting the number of nonzero rows. This works very well when the coefficients of *A* are known exactly and the Gaussian elimination algorithm can be carried out precisely. However, if there is any noise in the coefficients of *A* or any roundoff error in the execution of Gaussian elimination, then Gaussian elimination does not give any useful information about the rank. For example, it does not work to discard any rows that are "close to" 0; Gaussian elimination normalizes the first coefficient in each row to 1, so no nonzero row is close to 0.
- We set a small threshold, compute the singular values of *A* and conjecture that singular values below the threshold are actually zero and therefore the rank is equal to the number of singular values above the threshold. This conjecture is reasonable in two senses. First, it often does give the true rank of *A*; it is indeed often the case that the singular values that are computed as small are actually zero, and they appear as nonzero only due to roundoff error. Second, even if the small singular values are actually nonzero, the rank *r* computed by this procedure is "nearly" right, in the sense that *A* is very close to a matrix D_R of rank *r*; namely, the r^{th} SVD reconstruction of *A*.
- Choosing a suitable threshold is a process of some delicacy; the right choice depends the size of *A* and on how *A* was constructed.
- Consider the following:
- $\begin{bmatrix} \varepsilon & 2\varepsilon \\ 1 & 2 \end{bmatrix}, \begin{bmatrix} \varepsilon & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} \varepsilon & 1 \\ \varepsilon & 1+\varepsilon \end{bmatrix}$
- The first has rank 1, although roundoff error will probably produce a second pivot. Both pivots will be small; how many do we ignore? The second has one small pivot, but we cannot pretend that its row is insignificant. The third has two pivots and its rank is 2, but its "effective rank" ought to be 1.
- We go to a more stable measure of rank. The first step is to use $A^{\mathrm{T}}A$ or AA^{T}, which are symmetric but share the same rank as *A*. Their eigenvalues—the singular values squared—are *not* misleading. Based on the accuracy of the data, we decide on a

tolerance like 10^{-6} and count the singular values above it—that is the effective rank. The examples above have effective rank 1 (when ε is very small).

Image Processing Using SVD

Lossy Compression

A *compression* algorithm Φ is one that takes as input a body of data *D* and constructs a representation $E = \Phi(D)$ with the following two constraints.

- The computer memory needed to record *E* is much less than the natural encoding of *D*.

There is a decompression algorithm Ψ to reconstruct *D* from *E* .

- If $\Psi(E)$ is exactly equal to *D* then Φ is a *lossless* compression algorithm, If $\Psi(E)$ is approximately equal to *D*, then Φ is a *lossy* compression algorithm.

- Whether a lossless compression algorithm is needed or a lossy compression algorithm will suffice depends on the application. For compressing text or computer programs, we need lossless compression; reconstituting a text or program with each character code off by 1 bit is not acceptable. For pictures, sound, or video, however, lossy compression may merely result in a loss of quality but still be usable; if every pixel in an image is within 1% of its true gray level, a human viewer will barely notice the difference.
- The compressed encoding here is to record the nonzero elements of U_k , Σ_k andV_k^T. U' has km nonzero elements, Σ_k has k, andV_k^T has kn, for a total of $k(m+n+1)$; if k is small, then this is substantially smaller than the mn nonzero elements in A.

Figure 7.10. Using the SVD for lossy image compression. Original gray-scale image (upper left) and reconstructions with two singular values (upper right), five singular values (lower left), and 30 singular values (lower right).

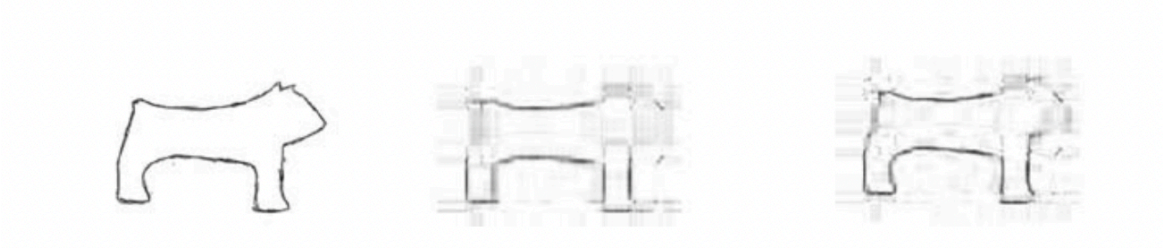
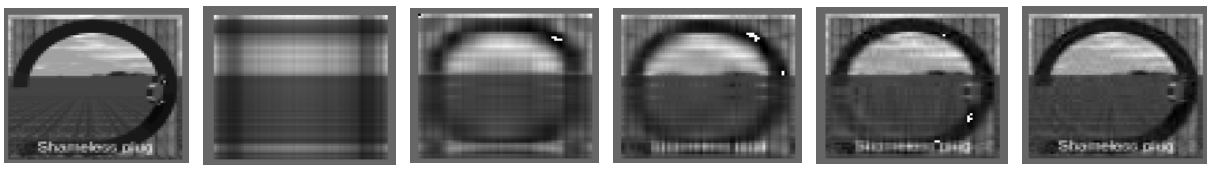


Figure 7.11. Using the SVD for lossy image compression. Original line drawing of a bear (left) and reconstructions with 4 singular values (middle), and 10 singular values (right).

Image Processing

- Suppose a satellite takes a picture, and wants to send it to Earth. The picture may contain 1000 by 1000 "pixels"—a million little squares, each with a definite color. We can code the colors, and send back 1,000,000 numbers. It is better to find the *essential* information inside the 1000 by 1000 matrix, and send only that.
- Suppose we know the SVD. The key is in the singular values (in Σ). Typically, some σ's are significant and others are extremely small. If we keep 20 and throw away 980, then we send only the corresponding 20 columns of U and V. The other 980 columns are multiplied in $U\Sigma V^{\mathrm{T}}$ by the small σ 's that are being ignored.
- *We can do the matrix multiplication as columns times rows*:
- $A = U\Sigma V^T = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \ldots\ldots + u_r\sigma_r v_r^T$

- Any matrix is the sum of r matrices of rank 1. If only 20 terms are kept, we send 20 times 2000 numbers instead of a million (25 to 1 compression).
- The pictures are really striking, as more and more singular values are included. At first you see nothing, and suddenly you recognize everything. The cost is in computing the SVD—this has become much more efficient, but it is expensive for a big matrix.



Orig

r = 1,3,5,10,16 (no perceivable

Principal Component Analysis

Consider for a moment the mass of data in figure 13.1. If I asked you to draw a line covering the data points, what's the longest possible line you could draw? I've drawn a few choices. Line B is the longest of these three lines. In PCA, we rotate the axes of the data. The rotation is determined by the data itself. The first axis is rotated to cover the largest variation in the data: line B in figure 13.1. The largest variation is the data telling us what's most important. It is the **first principal component.**

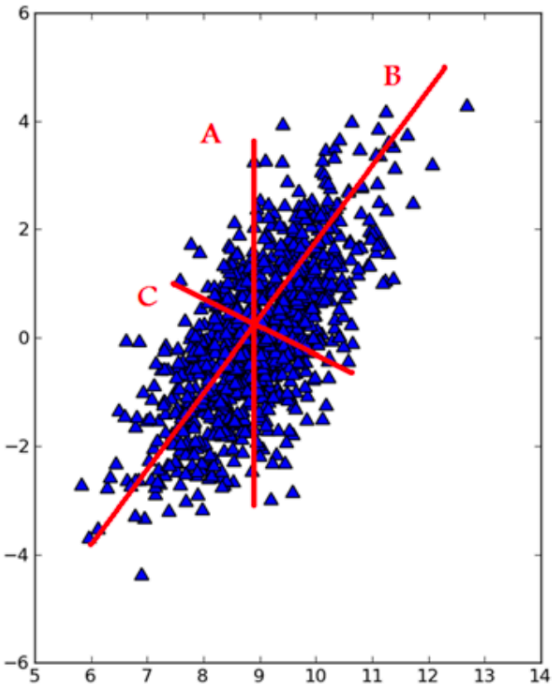


Figure 13.1 Three choices for lines that span the entire dataset. Line B is the longest and accounts for the most variability in the dataset.

After choosing the axis covering the most variability, we choose the next axis, which has the second most variability, provided it's perpendicular to the first axis. The real term used is orthogonal. On this two-dimensional plot, perpendicular and orthogonal are the same. In figure 13.1, line C would be our second axis. With PCA, we're rotating the axes so that they're lined up with the most important directions from the data's perspective.
We can get the values of the principal components by taking the covariance matrix of the data- set and doing eigenvalue analysis on the covariance matrix.
Once we have the eigenvectors of the covariance matrix, we can take the top N eigenvectors. The top N eigenvectors will give us the true structure of the N most important features. We can then multiply the data by the top N eigenvectors to transform our data into the new space.

Principal component analysis can be applied to any data that consist of lists of measurements made on a collection of objects or individuals
The data represented in matrix form is called the matrix of observation.
Let X_1 denote the **observation vector**
The set of observation vectors can be visualized as a two-dimensional *scatter plot*.

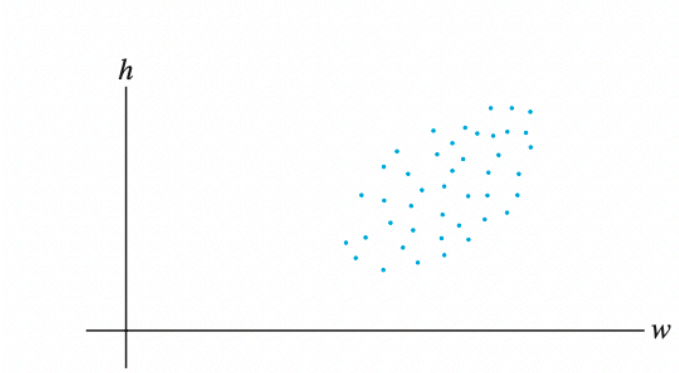


FIGURE 1 A scatter plot of observation vectors $\mathbf{X}_1, \ldots, \mathbf{X}_N$.

To prepare for principal component analysis, let $[X_1 \cdots X_n]$ be a p × n matrix of observations. The sample mean, M, of the observation vectors $X_1 \cdots X_n$ is given by

$$M = \frac{1}{n}(X_1 + \cdots \ldots + X_n)$$

For the data in Fig. 1, the sample mean is the point in the "center" of the scatter plot. For k = 1, . . . , n , let

$$\hat{X}_k = X_k - M$$

The columns of the p×n matrix $B = [\hat{X}_1 \hat{X}_2 \ldots\ldots \hat{X}_n]$ have a zero sample mean, and B is said to be in **mean-deviation form**. When the sample mean is subtracted from the data in Fig. 1, the resulting scatter plot has the form in Fig. 3.

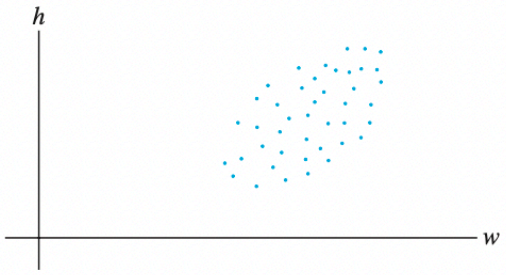


FIGURE 1 A scatter plot of observation vectors $\mathbf{X}_1, \ldots, \mathbf{X}_N$.

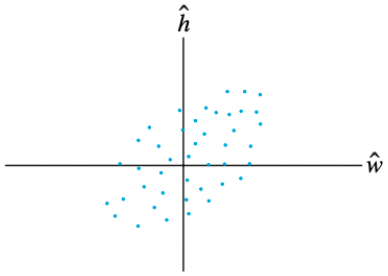


FIGURE 3

The (**sample**) **covariance matrix** is the p×p matrix S defined by

$$S = \frac{1}{n-1} B\, B^T$$

Since any matrix of the form BB^T is positive semidefinite, so is S.

To discuss the entries in S = $[s_{ij}]$, let **X** represent a vector that varies over the set of observation vectors and denote the coordinates of **X** by $x_1, \ldots, x_p$. Then x_1, for example, is a scalar that varies over the set of first coordinates of $\mathbf{X}_1, \ldots, \mathbf{X}_N$. For j =1,...,p, the diagonal entry s_{ij} in S is called the **variance** of x_j.

The variance of x_j measures the spread of the values of x_j .

The **total variance** of the data is the sum of the variances on the diagonal of S. In general, the sum of the diagonal entries of a square matrix S is called the **trace** of the matrix, written tr(S). Thus

{total variance} = tr(S)

The entry s_{ij} in S for $i \neq j$ is called the **covariance** of x_i and x_j . Analysis of the multivariate data in $\mathbf{X}_1,...,\mathbf{X}_N$ is greatly simplified when most or all of the variables $x_1,...,x_p$ are uncorrelated, that is, when the covariance matrix of $\mathbf{X}_1,...,\mathbf{X}_N$ is diagonal or nearly diagonal. For simplicity, assume that the matrix $[X_1 \cdots X_N]$ is already in mean-deviation form. The goal of principal component analysis is to find an orthogonal p×p matrix

Let D be a diagonal matrix with the eigenvalues $\lambda_1, \ldots, \lambda_p$ of S on the diagonal, arranged so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, and let P be an orthogonal matrix whose columns are the corresponding unit eigenvectors $u_1,...,u_p$.

Then $S = PDP^T$ and $P^TSP = D$.

The covariance matrix S can be represented in the form

$S = PDP^T$

Principal Components

The unit eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_p$ of the covariance matrix S are called the **principal components** of the data (in the matrix of observations). The **first principal component** is the eigenvector corresponding to the largest eigenvalue of S, the **second principal component** is the eigenvector corresponding to the second largest eigenvalue, and so on.

The first principal component $\mathbf{u}_1$ determines the new variable y_1 in the following way. Let c1,...,cp be the entries in $\mathbf{u}_1$. Since $\mathbf{u}^T1$ is the first row of P^T , the equation $\mathbf{Y} = P^T \mathbf{X}$ shows that

$y_1 = u_1^T X = c_1 x_1 + c_2 x_2 + \ldots\ldots\ldots c_p x_p$

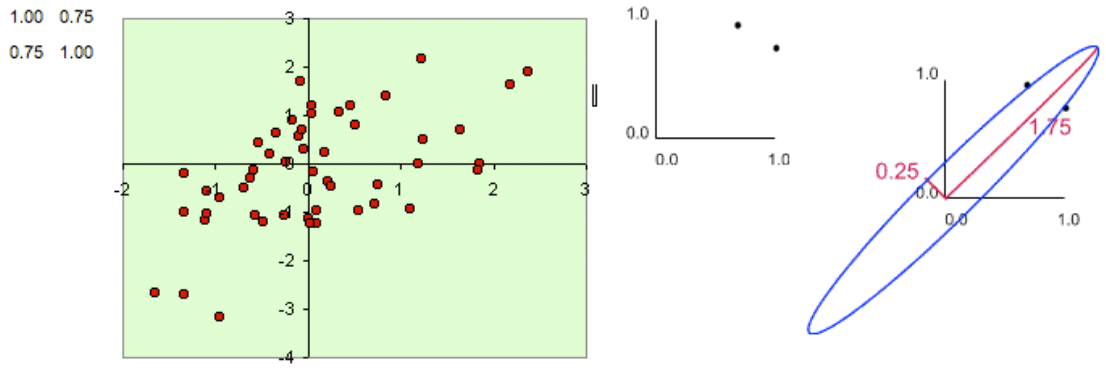
Thus y_1 is a linear combination of the original variables $x_1 , \ldots , x_p$, using the entries in the eigenvector $\mathbf{u}_1$ as weights. In a similar fashion, $\mathbf{u}_2$ determines the variable y_2, and so on.

Covariance Matrix

- **What do the covariances that we have as entries of the matrix tell us about the correlations between the observations?**
- It's actually the sign of the covariance that matters :
  - if positive then : the two observations increase or decrease together (correlated)
  - if negative then : One increases when the other decreases (Inversely correlated)
  - If zero: No relation between the observations
- Now, that we know that the covariance matrix is not more than a table that summaries the correlations between all the possible pairs of observations.

Consider a covariance matrix, **A**, i.e., $A = 1/n \; S \; S^T$ for some S

$$\mathbf{A} = \begin{bmatrix} 1 & .75 \\ .75 & 1 \end{bmatrix} \Rightarrow \lambda_1 = 1.75, \lambda_2 = 0.25$$



Error ellipse with the major axis as the larger eigenvalue and the minor axis as the smaller eigenvalue

Physical interpretation

First principal component is the direction of greatest variability (covariance) in the data

Second is the next orthogonal (uncorrelated) direction of greatest variability

So first remove all the variability along the first component, and then find the next direction of greatest variability

And so on …

Thus each eigenvectors provides the directions of data variances in decreasing order of eigenvalues

Application of PCA

Image Processing

- The first three photographs of Railroad Valley, Nevada, shown in **Linear Algebra and Its Applications by** David Lay, can be viewed as *one* image of the region, with *three spectral components*, because simultaneous measurements of the region were made at three separate wavelengths. Each photograph gives different information about the same physical region. For instance, the first pixel in the upper-left corner of each photograph corresponds to the same place on the ground (about 30 meters by 30 meters). To each pixel there corresponds an observation vector in R^3 that lists the signal intensities for that pixel in the three spectral bands.

- Typically, the image is 2000×2000 pixels, so there are 4 million pixels in the image. The data for the image form a matrix with 3 rows and 4 million columns (with columns arranged in any convenient order). In this case, the "multidimensional" character of the data refers to the three *spectral* dimensions rather than the two *spatial* dimensions that naturally belong to any photograph. The data can be visualized as a cluster of 4 million points in R^3, perhaps as in Fig. 2.

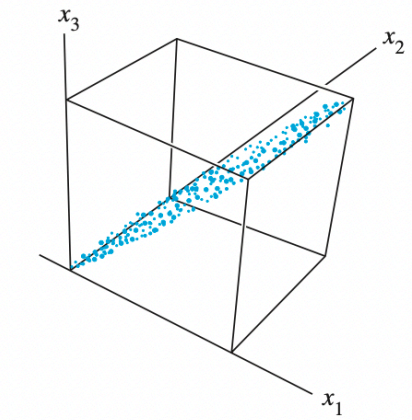


FIGURE 2
A scatter plot of spectral data for a satellite image.

Example

- Three measurements are made on each of four individuals in a random sample from a population. The observation vectors are

$$\mathbf{X}_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad \mathbf{X}_2 = \begin{bmatrix} 4 \\ 2 \\ 13 \end{bmatrix}, \quad \mathbf{X}_3 = \begin{bmatrix} 7 \\ 8 \\ 1 \end{bmatrix}, \quad \mathbf{X}_4 = \begin{bmatrix} 8 \\ 4 \\ 5 \end{bmatrix}$$

Compute the sample mean and the covariance matrix.
Solution

The sample mean is

$$\mathbf{M} = \frac{1}{4}\left(\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 4 \\ 2 \\ 13 \end{bmatrix} + \begin{bmatrix} 7 \\ 8 \\ 1 \end{bmatrix} + \begin{bmatrix} 8 \\ 4 \\ 5 \end{bmatrix} \right) = \frac{1}{4}\begin{bmatrix} 20 \\ 16 \\ 20 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 5 \end{bmatrix}$$

The initial data for the multispectral image of Railroad Valley consisted of 4 million vectors. The associated covariance matrix is

Subtract the sample mean from $\mathbf{X}_1, \ldots, \mathbf{X}_4$ to obtain

$$\hat{\mathbf{X}}_1 = \begin{bmatrix} -4 \\ -2 \\ -4 \end{bmatrix}, \quad \hat{\mathbf{X}}_2 = \begin{bmatrix} -1 \\ -2 \\ 8 \end{bmatrix}, \quad \hat{\mathbf{X}}_3 = \begin{bmatrix} 2 \\ 4 \\ -4 \end{bmatrix}, \quad \hat{\mathbf{X}}_4 = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix}$$

and

$$B = \begin{bmatrix} -4 & -1 & 2 & 3 \\ -2 & -2 & 4 & 0 \\ -4 & 8 & -4 & 0 \end{bmatrix}$$

The sample covariance matrix is

$$S = \frac{1}{3}\begin{bmatrix} -4 & -1 & 2 & 3 \\ -2 & -2 & 4 & 0 \\ -4 & 8 & -4 & 0 \end{bmatrix}\begin{bmatrix} -4 & -2 & -4 \\ -1 & -2 & 8 \\ 2 & 4 & -4 \\ 3 & 0 & 0 \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 30 & 18 & 0 \\ 18 & 24 & -24 \\ 0 & -24 & 96 \end{bmatrix} = \begin{bmatrix} 10 & 6 & 0 \\ 6 & 8 & -8 \\ 0 & -8 & 32 \end{bmatrix}$$

Example
The initial data for the multispectral image of Railroad Valley consisted of 4 million vectors. The associated covariance matrix is

$$S = \begin{bmatrix} 2382.78 & 2611.84 & 2136.20 \\ 2611.84 & 3106.47 & 2553.90 \\ 2136.20 & 2553.90 & 2650.71 \end{bmatrix}$$

Find the principal components of the data, and list the new variable determined by the first principal component.

Solution The eigenvalues of S and the associated principal components (the unit eigenvectors) are

$$\lambda_1 = 7614.23 \qquad \lambda_2 = 427.63 \qquad \lambda_3 = 98.10$$

$$\mathbf{u}_1 = \begin{bmatrix} .5417 \\ .6295 \\ .5570 \end{bmatrix} \qquad \mathbf{u}_2 = \begin{bmatrix} -.4894 \\ -.3026 \\ .8179 \end{bmatrix} \qquad \mathbf{u}_3 = \begin{bmatrix} .6834 \\ -.7157 \\ .1441 \end{bmatrix}$$

Using two decimal places for simplicity, the variable for the first principal component is

$$y_1 = .54x_1 + .63x_2 + .56x_3$$

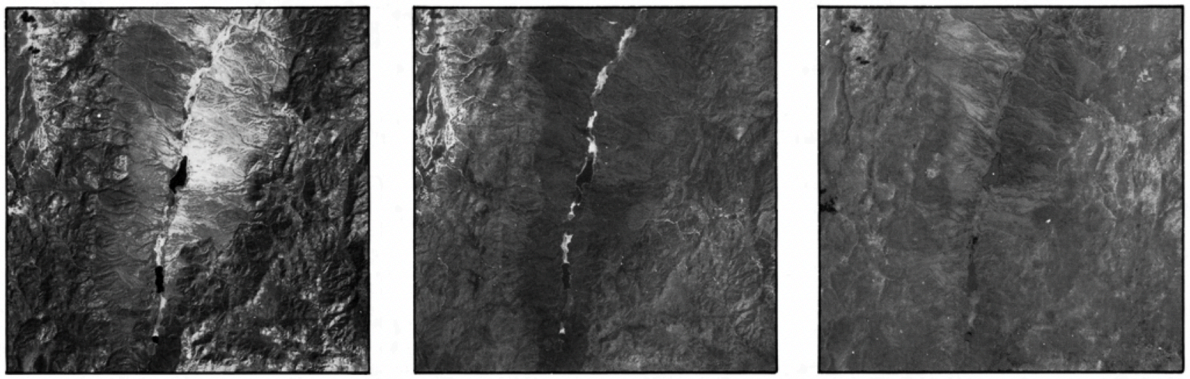
This equation was used to create photograph (d) in the chapter introduction.

- The variables x_1, x_2, x_3 are the signal intensities in the three spectral bands. The values of x_1, converted to a grey scale between black and white, produced photograph(a). Similarly, the values of x_2 and x_3 produced photographs (b) and (c), respectively. At each pixel in photograph (d), the grey scale value is computed from y_1, a weighted linear combination of x_1, x_2, x_3. In this sense, photograph (d) "displays" the first principal component of the data.
- The covariance matrix for the transformed data, using variables y_1, y_2 and y_3

$$D = \begin{bmatrix} 7614.23 & 0 & 0 \\ 0 & 427.63 & 0 \\ 0 & 0 & 98.10 \end{bmatrix}$$

- Although D is obviously simpler than the original covariance matrix S, the merit of constructing the new variables is not yet apparent. However, the variances of the variables y_1, y_2, y_3 appear on the diagonal of D, and obviously the first variance in D is much larger than the other two. As we shall see, this fact will permit us to view the data as essentially one dimensional rather than three-dimensional.

(a) Spectral band 1: Visible blue. (b) Spectral band 4: Near infrared. (c) Spectral band 7: Mid infrared.



(d) Principal component 1: 93.5%. (e) Principal component 2: 5.3%. (f) Principal component 3: 1.2%.

Dimensionality Reduction

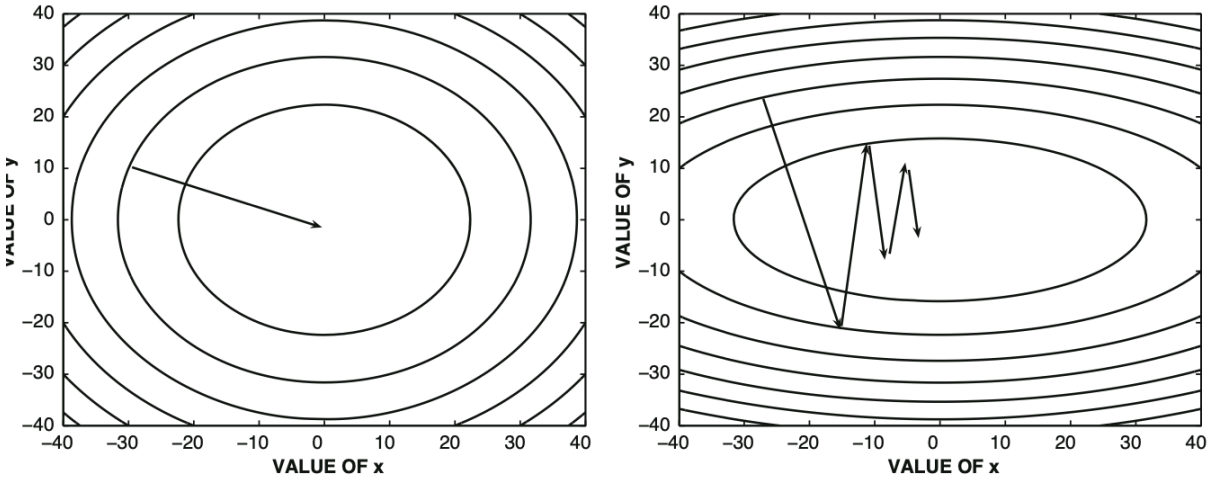
- Principal component analysis is potentially valuable for applications in which most of the variation, or dynamic range, in the data is due to variations in *only a few* of the new variables, $y_1, \ldots, y_p$.
- It can be shown that an orthogonal change of variables, $\mathbf{X} = P\,\mathbf{Y}$, does not change the total variance of the data. (Roughly speaking, this is true because left-multiplication by P does not change the lengths of vectors or the angles between them.
- This means that if $S = PDP^T$, then
- $\left\{ \begin{matrix} Total\ Variance \\ of\ x_1, \ldots . x_p \end{matrix} \right\} = \left\{ \begin{matrix} Total\ Variance \\ of\ y_1, \ldots . y_p \end{matrix} \right\} = tr(D) = \lambda_1 + \ \ldots . + \lambda_p$
- The variance of y_j is λ_j , and the quotient λ_j / tr(S) measures the fraction of the total variance that is "explained" or "captured" by y_j .

Feature Preprocessing and Whitening in Machine Learning

Principal component analysis is used for feature preprocessing in machine learning, by first reducing the dimensionality of the data and then normalizing the newly transformed features, so that the variance along each transformed direction is the same. Let V_k be the $d \times k$ matrix containing the top-k eigenvectors found by principal component analysis. Then, the first step is to transform the mean-centered data matrix D to the k-dimensional representation U_k as follows:

$$U_k = DV_k$$

- The next step is to divide each column of U_k by its standard deviation. As a result, the original data distribution becomes roughly spherical in shape. This type of approach is referred to as whitening.
- Standard deviation is how much variation the data has with respect to the mean
- This type of data distribution works much more effectively with gradient-descent algorithms. This is because widely varying variances along different directions also lead to loss functions in which different directions have different levels of curvature. Examples of two loss functions with different levels of curvature are illustrated in Figure 5.2. A loss function like Figure 5.2(a) tends to be more easily optimized with gradient descent algorithms.



(a) Loss function is circular bowl $L = x^2 + y^2$

(b) Loss function is elliptical bowl $L = x^2 + 4y^2$

Figure 5.2: The effect of the shape of the loss function on steepest-gradient descent

- Normalizing the data to have unit variance in all directions tends to reduce obvious forms of ill-conditioning in the loss function. As a result, gradient-descent tends to become much faster. Furthermore, the normalization of the data in this way sometimes prevents some subsets of features from having undue influence on the final results.

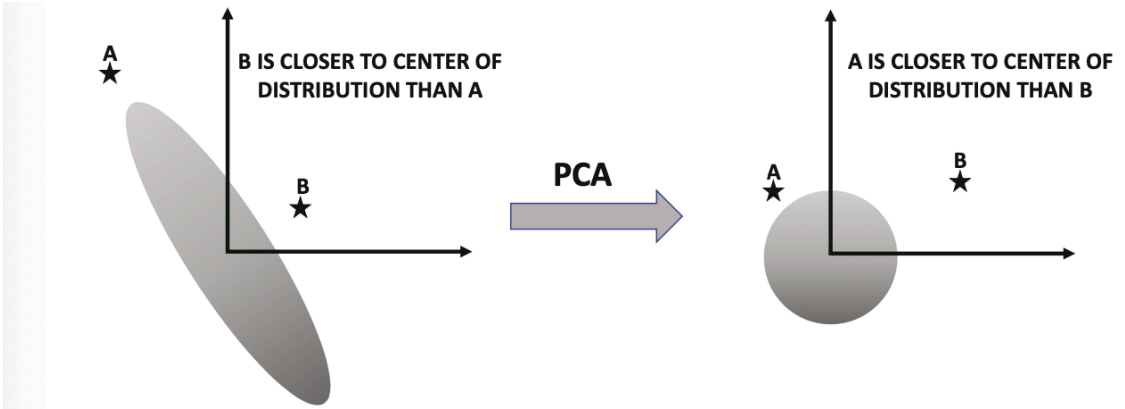


Figure 7.8: An example of the whitening of an ellipsoidal data distribution by principal component analysis and its use in outlier detection

This type of preprocessing is also used in unsupervised applications like outlier detection. In fact, whitening is arguably more important in unsupervised applications because one does not have labels to provide guidance about the relative importance of different directions in the data. An example of the whitening of an ellipsoidal data distribution is illustrated in Figure 7.8. The resulting data distribution has a spherical shape.

Comparison of SVD and PCA

The mean-centering of PCA helps in improving the accuracy of the approximation. In order to understand this point, we have shown an example of a 3-dimensional data set that is not originally mean-centered in Figure 7.6.

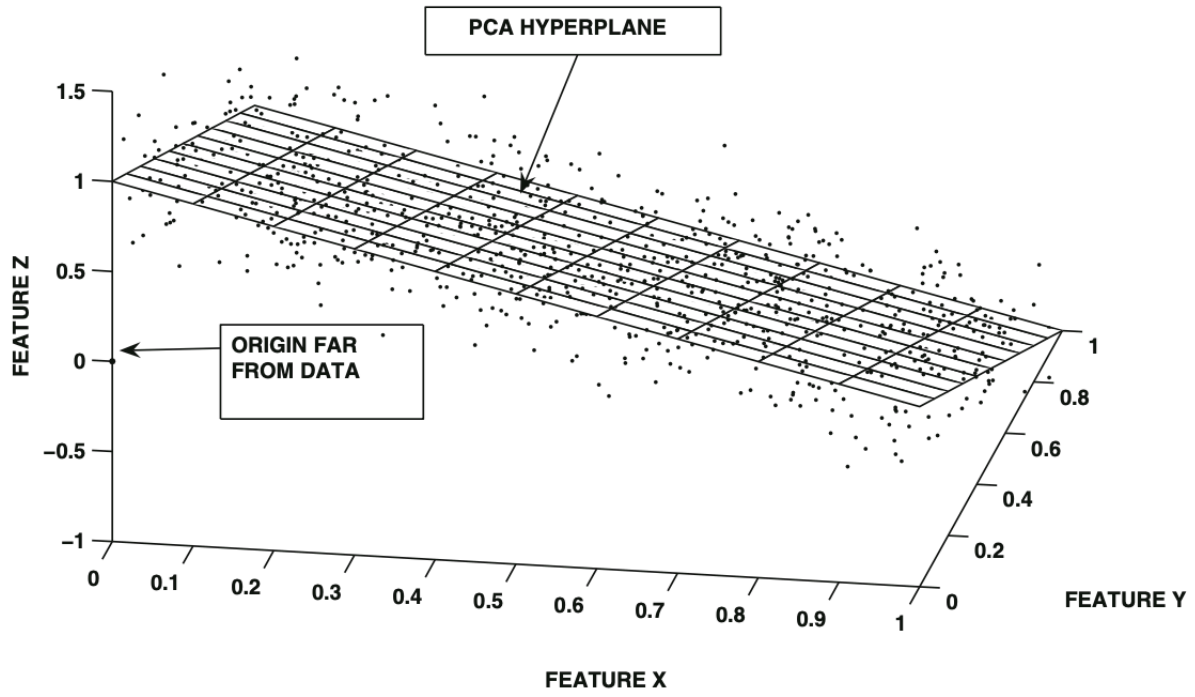


Figure 7.6: PCA for data that is not originally mean-centered

Most of the data is distributed near a plane far away from the origin (before preprocessing or mean-centering).

In this case, a 2-dimensional hyperplane can approximate the data quite well, where the mean-centering process ensures that the PCA hyperplane passed through the mean of the original data set. This is not the case for SVD, which will struggle to approximate the data without using all the three dimensions. It can be explicitly shown that the accuracy of PCA is at least as good as that of SVD for the same number of eigenvectors.

Questions

1. Define:
    a. Quadratic Form
    b. Positive definite matrix
    c. Singular values
    d. Covariance matrix
    e. Principal Components
    f. First Principal Component
    g. Second Principal Components
2. Mark each statement True or False. Justify each answer.
    a. The matrix of a quadratic form is a symmetric matrix.
    b. A quadratic form has no cross-product terms if and only if the matrix of the quadratic form is a diagonal matrix.
    c. A Positive definite Quadratic Form Q satisfies Q(x) > 0 for all x $\neq$ 0
    d. If the eigenvalues of a symmetric matrix A are all positive, then the quadratic form x^TAx is positive definite.
3. Compute the quadratic form x^TAx, when $A = \begin{bmatrix} 5 & 1/3 \\ 1/3 & 1 \end{bmatrix}$
    a. $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $b.\begin{bmatrix} 6 \\ 1 \end{bmatrix}$ $c.\begin{bmatrix} 1 \\ 3 \end{bmatrix}$
4. Find the singular values of the matrices

$(i)\begin{bmatrix} 1 & 0 \\ 0 & -3 \end{bmatrix}, (ii)\begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}, (iii)\begin{bmatrix} \sqrt{6} & -1 \\ 0 & \sqrt{6} \end{bmatrix}$

5. Explain SVD in brief.
6. Explain how SVD can be used for Image processing
7. How can SVD be used for finding the rank of a matrix.
8. The SVD of a matrix was given as:

$$U = \begin{bmatrix} -0.4346 & -0.3010 & 0.7745 & 0.3326 & -0.1000 \\ -0.1933 & -0.3934 & 0.1103 & -0.8886 & -0.0777 \\ 0.5484 & 0.5071 & 0.6045 & -0.2605 & -0.0944 \\ 0.6715 & -0.6841 & 0.0061 & 0.1770 & -0.2231 \\ 0.1488 & -0.1720 & 0.1502 & -0.0217 & 0.9619 \end{bmatrix}$$

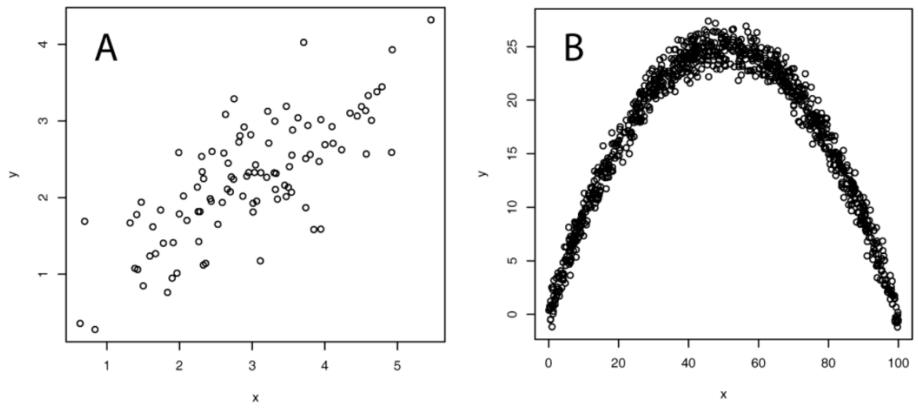
$$\Sigma = \begin{bmatrix} 5.72 & 0 & 0 \\ 0 & 2.89 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.2321 & -0.9483 & 0.2166 \\ -0.2770 & 0.1490 & 0.9493 \\ 0.9324 & 0.2803 & 0.2281 \end{bmatrix}$$

(a) Which columns form a basis for the null space of A? For the column space of A? For the row space of A?
(b) What are the singular values?
(c) What is the rank of A?
(d) What is the condition number of A? Is A ill-conditioned?

9. Can the SVD be used to determine if a matrix A is invertible? How?

10. To what type of matrices is SVD applicable? Why?
11. What are the left singular vectors of a matrix ? What are the right singular vectors of a matrix ?
12. What is the condition number of a matrix and how can it be expressed in terms of singular values?
13. How can one compute the rank of a matrix by knowing its singular values?
14. Given the SVD decomposition of the matrix : A = $U\Sigma V^T$, how can the exact pseudo-inverse A^+be written?
15. For a matrix A with SVD decomposition A = $U\Sigma V^T$, what are the columns of U and how can we find them? What are the columns of V and how can we find them? What are the entries of Σ and how can we find them?
16. What are the dimensions of U, Σ, and V in the full SVD of an m×n matrix?
17. How do you use the SVD to compute a low-rank approximation of a matrix?
18. Explain any one application of PCA
19. Is rotation necessary in PCA? If yes, Why? What will happen if you don't rotate the components?
20. What will happen when eigenvalues are roughly equal?
21. Consider the data shown in Panel A, above. Sketch (on the panel) a pair of eigenvectors that you would expect to obtain from a Principal Components Analysis (PCA) of the data. Indicate which eigenvector you would expect to have the larger corresponding eigenvalue. What do the numerical values of the eigenvalues tell you about the data? Do you think it is appropriate to use PCA to reduce the dimensionality of the dataset shown in panel B? Why or why not?



22.

SOLUTION:

8. (a) The matrix A is 5 × 3, so the null space and row space are subspaces of R^3 . The column space (and null space of A^T) are in R^3 . The dimension of the column space (which is also the dimension of the row space) is 2.

Now we can answer the questions:

• The last column of V is a basis for the 1 - dimensional null space.

• The column space is spanned by the first two columns of U.

• The row space is spanned by the first two columns of V .

(b) $\sigma_1 = 5.72\,,\ \sigma_2 = 8.61$

(c) The rank is the dimension of the column space, which (given the SVD), is the number of non-zero singular values. In this case, the answer is 2.

(d) Condition number = $\frac{\sigma_{max}}{\sigma_{min}} = \frac{8.61}{5.72} = 1.5$

A is not ill-conditioned as the condition number is not very large.

9.The SVD can be used to determine whether a matrix is invertible, and can provide a formula for the inverse. The matrix A is invertible if it is square and all singular values are positive (not zero). Then the formula for the inverse is much the same as the formula for the pseudoinverse.
19. Yes, rotation (orthogonal) is necessary to account the maximum variance of the training set. If we don't rotate the components, the effect of PCA will diminish and we'll have to select more number of components to explain variance in the training set.
20. If all eigenvectors are same then PCA won't be able to select the principal components because in that case, all principal components are equal.

Complexity

As an engineer or computer scientist, it is important not only to be able to solve problems, but also to know which problems one can expect to solve efficiently.
We will explore the complexity of various problems, which is a measure of how efficiently they can be solved.

Algorithm

An algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. An algorithm is thus a sequence of computational steps that transform the input into the output.

An instance of a problem consists of the input (satisfying whatever constraints are imposed in the problem statement) needed to compute a solution to the problem.

An algorithm is said to be correct if, for every input instance, it halts with the correct output.

An incorrect algorithm might not halt at all on some input instances, or it might halt with an incorrect answer.

An algorithm can be specified in English, as a computer program, or even as a hardware design. The only requirement is that the specification must provide a precise description of the computational procedure to be followed.

There are some problems, however, for which no efficient solution is known .

Bounds of an Algorithm

O(N)	Big O	Upper Bound
$\theta(n)$	Theta	Tight Bound
$\Omega(n)$	Omega	Lower Bound

Upper bound of the Algorithm Upper bound of the algorithm is the pessimistic view of the algorithm. It can be used to indicate worst case analysis of the algorithm and often expressed as a function. It can be viewed as a proof that the given problem can be solved using at most ‘n’ operations, even in the worst case.

Lower Bound Theory Lower bound is for problems and finding it is difficult compared to upper bound of the algorithm. Lower bound is the smallest number of operations necessary to solve a problem over all inputs of size n. In short, it is “At least this much work to be done”

Lower bound is an indication of how hard the algorithm is! It is done for problems and not algorithms. In other words, it is obtained the “best’ algorithms that is required to solve the given problem.

Tight Bound defines the average case of an algorithm's time complexity, the Theta notation defines when the set of functions lies in both O(expression) and Omega(expression), then Theta notation is used.

Example

Upper Bound

If it's adding two matrices containing n elements each, then it's O(n). If it's adding two n x n matrices, then it's $O(n^2)$,since each matrix has n^2 elements.

Polynomial Time

An algorithm is polynomial-time if there exists a constant r such that the running time on an input of size n is $O(n^r)$. The set of all decision problems which have polynomial-time solutions is called P.

Polynomial time is the shortest class of running times that is invariant across the vast majority of reasonable, mainstream models of computation.

Complexity class P

The ***complexity class*** **P** as the set of concrete decision problems that are polynomial-time solvable.

A deterministic algorithm is (essentially) one that always computes the correct answer

E.g., Searching, Sorting, Matrix Multiplication

Decision problems

We consider decision problems, whose answer is YES or NO.

E.g., "Does the given network have a flow of value at least k?"

For such problems, we split all instances into two categories:

YES-instances (whose correct answer is YES) and

NO-instances (whose correct answer is NO). We put any ill-formed instances into the NO category.

Hamiltonian cycle

The problem of finding a hamiltonian cycle in an undirected graph has been studied for over a hundred years.

Formally, a ***hamiltonian cycle*** of an undirected graph G =(V, E) is a simple cycle that contains each vertex in V . A graph that contains a hamiltonian cycle is said to be ***hamiltonian***; otherwise, it is ***nonhamiltonian***. The name honors W. R. Hamilton, who described a mathematical game on the dodecahedron in which one player sticks five pins in any five consecutive vertices and the other player must complete the path to form a cycle containing all the vertices.

The dodecahedron is hamiltonian, and Figure 34.2(a) shows one hamiltonian cycle. Not all graphs are hamiltonian, however. For example, Figure 34.2(b) shows a bipartite graph with an odd number of vertices.

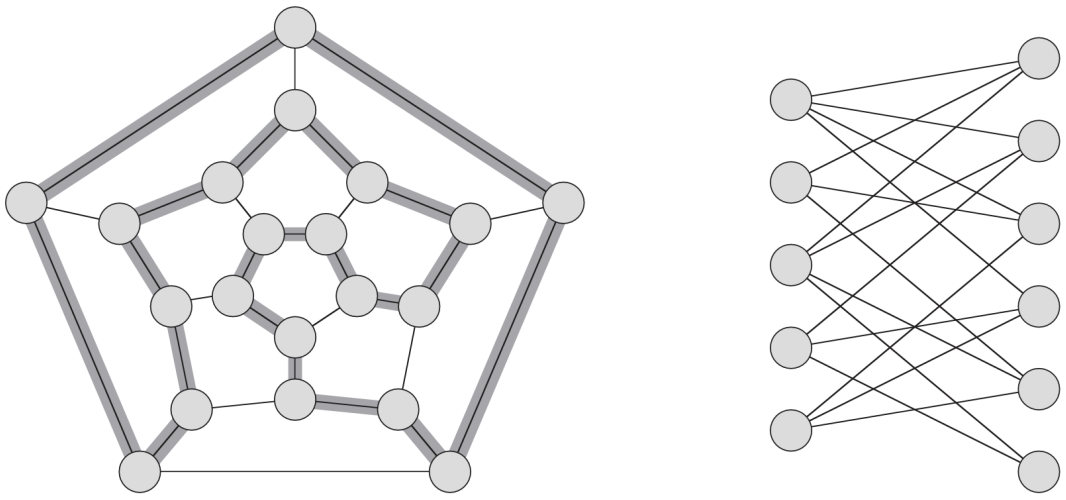


Figure 34.2 **(a)** A graph representing the vertices, edges, and faces of a dodecahedron, with a hamiltonian cycle shown by shaded edges. **(b)** A bipartite graph with an odd number of vertices. Any such graph is nonhamiltonian.

We can define the ***hamiltonian-cycle problem***, "Does a graph G have a hamiltonian cycle?"

Given a problem instance {G}, one possible decision algorithm lists all permutations of the vertices of G and then checks each permutation to see if it is a hamiltonian path.

What is the running time of this algorithm? If we use the "reasonable" encoding of a graph as its adjacency matrix, the number m of vertices in the graph is $\Omega\left(\sqrt{n}\right)$, where n is the length of the encoding of G. There are m! possible permutations of the vertices, and therefore the running time is $\Omega\left(2^{\sqrt{n}}\right)$, which is not $O(n^k)$ for any constant k.

Thus, this naive algorithm does not run in polynomial time.

The Concept of NP

NP stands for "nondeterministic polynomial time. Intuitively, it means that a solution to any search problem can be found and verified in polynomial time by a special (and quite

unrealistic) sort of algorithm, called a nondeterministic algorithm. Such an algorithm has the power of guessing correctly at every step.

Incidentally, the original definition of NP (and its most common usage to this day) was not as a class of search problems, but as a class of decision problems: algorithmic questions that can be answered by yes or no. Example: “Is there a truth assignment that satisfies this Boolean formula?”

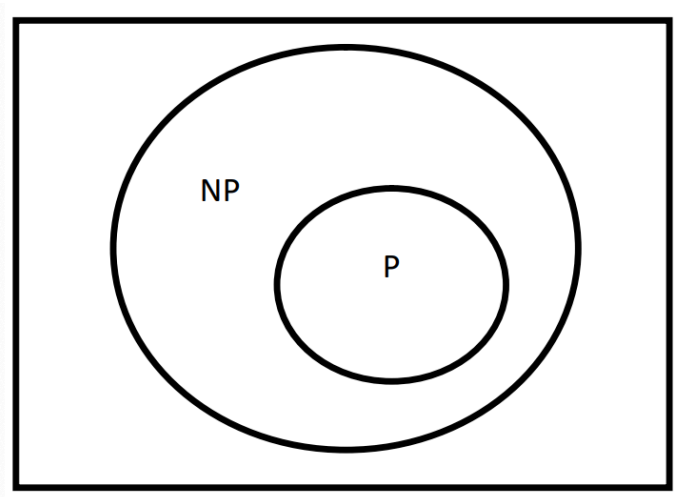
Thus NP can also be thought of as the class of problems “whose solutions can be verified in polynomial time”

Example

Classroom Scheduling

Is P = NP?

Clearly P $\subseteq NP$



If an algorithm can be solved in polynomial time then can we verify the solution in polynomial time?

P$\neq NP$

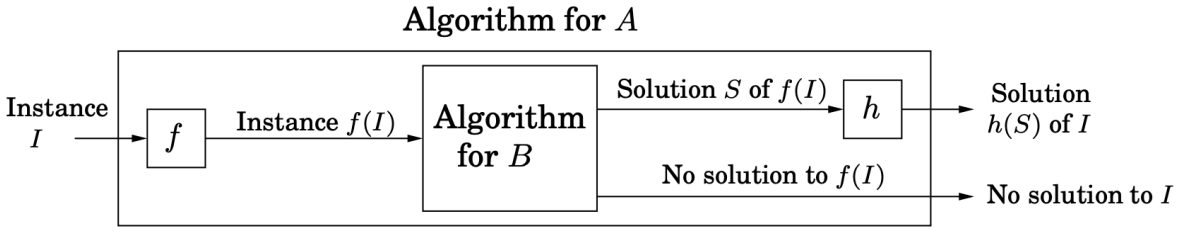
Solving problems is harder than checking solutions

Reducibility

A reduction from search problem A to search problem B is a polynomial-time algorithm f that transforms any instance I of A into an instance f(I) of B, together with another polynomial-time algorithm h that maps any solution S of f(I) back into a solution h(S) of I; see the following diagram. If f(I) has no solution, then neither does I. These two translation procedures f and h imply that any algorithm for B can be converted into an algorithm for A by bracketing it between f and h.

Example: lcm(m, n) = (m * n) / gcd(m, n),

Least common multiple (LCM) of (m,n) problem is reduced to Greatest common divisor (GCD)of (m, n) problem



Definition of NP-Complete

A problem is NP-complete if the problem is both

Problem is in NP (i.e. its solution can be verified in polynomial time)

A lot of times you can solve a problem by reducing it to a different problem. One can reduce Problem B to Problem A if, given a solution to Problem A, one can easily construct a solution to Problem B. (In this case, "easily" means "in polynomial time.“).

In other words, the problem can be reduced to another problem and the solution can be verified

Interesting facts of NP-complete problems

First, although no efficient algorithm for an NP-complete problem has ever been found, nobody has ever proven that an efficient algorithm for one cannot exist. In other words, no one knows whether or not efficient algorithms exist for NP-complete problems.

Second, the set of NP-complete problems has the remarkable property that if an efficient algorithm exists for any one of them, then efficient algorithms exist for all of them. This relationship among the NP-complete problems makes the lack of efficient solutions all the more tantalizing.

Third, several NP-complete problems are similar, but not identical, to problems for which we do know of efficient algorithms. Computer scientists are intrigued by how a small change to the problem statement can cause a big change to the efficiency of the best known algorithm.

Example of NP complete

- Hamiltonian cycle
- SAT

- Travelling Salesman Problem (TSP)
- Vertex Cover Problem
- Set Cover Problem

SAT

Boolean combinational circuits are built from boolean combinational elements that are interconnected by wires.

A ***boolean combinational element*** is any circuit element that has a constant number of boolean inputs and outputs and that performs a well-defined function.

Boolean values are drawn from the set {0;1}, where 0 represents FALSE and 1 represents TRUE.

The boolean combinational elements that we use in the circuit-satisfiability problem compute simple boolean functions, and they are known as ***logic gates***.

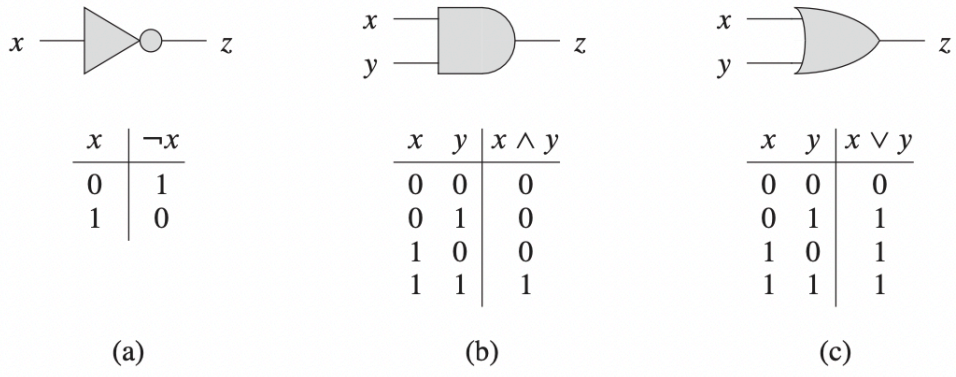
The three basic logic gates that we use in the circuit-satisfiability problem:

the ***NOT gate*** (or ***inverter***),

the ***AND gate***, and

the ***OR gate***

The NOT gate takes a single binary ***input*** x, whose value is either 0 or 1, and produces a binary ***output*** whose value is opposite that of the input value. Each of the other two gates takes two binary inputs x and y and produces a single binary output z.



x	$\neg x$
0	1
1	0

(a)

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

(b)

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

(c)

Figure 34.7 Three basic logic gates, with binary inputs and outputs. Under each gate is the truth table that describes the gate's operation. **(a)** The NOT gate. **(b)** The AND gate. **(c)** The OR gate.

We can describe the operation of each gate, and of any boolean combinational element, by a ***truth table***, shown under each gate in Figure 34.7. A truth table gives the outputs of the combinational element for each possible setting of the inputs. F

example, the truth table for the OR gate tells us that when the inputs are $x = 0$ and $y = 1$, the output value is $z = 1$. We use the symbols $\neg$ to denote the NOT function, $\wedge$ to denote the AND function, and $\vee$ to denote the OR function. Thus, for example, $0 \vee 1 = 1$.

We can generalize AND and OR gates to take more than two inputs. An AND gate's output is 1 if all of its inputs are 1, and its output is 0 otherwise. An OR gate's output is 1 if any of its inputs are 1, and its output is 0 otherwise.

A ***boolean combinational circuit*** consists of one or more boolean combinational elements interconnected by ***wires***. A wire can connect the output of one element to the input of another, thereby providing the output value of the first element as an input value of the second.

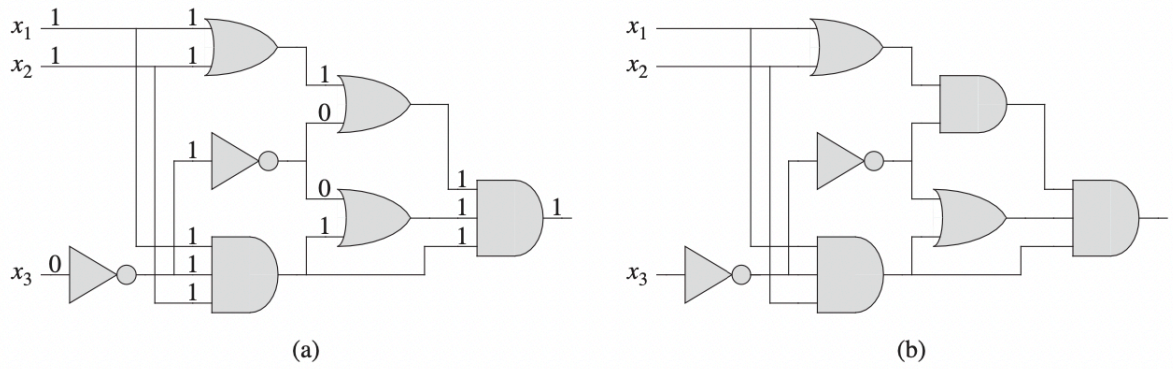


Figure 34.8 Two instances of the circuit-satisfiability problem. **(a)** The assignment $\langle x_1 = 1, x_2 = 1, x_3 = 0 \rangle$ to the inputs of this circuit causes the output of the circuit to be 1. The circuit is therefore satisfiable. **(b)** No assignment to the inputs of this circuit can cause the output of the circuit to be 1. The circuit is therefore unsatisfiable.

A ***truth assignment*** for a boolean combinational circuit is a set of boolean input values. We say that a one-output boolean combinational circuit is ***satisfiable*** if it has a ***satisfying assignment***: a truth assignment that causes the output of the circuit to be 1.

The ***circuit-satisfiability problem*** is, "Given a boolean combinational circuit composed of AND, OR, and NOT gates, is it satisfiable?"

The circuit-satisfiability problem arises in the area of computer-aided hardware optimization. If a subcircuit always produces 0, that subcircuit is unnecessary; the designer can replace it by a simpler subcircuit that omits all logic gates and provides the constant 0 value as its output. You can see why we would like to have a polynomial-time algorithm for this problem.

SAT was the first algorithm which was proved to be NP-complete

Reducing HAMILTON PATH to SAT

The problem HAMILTON PATH is defined as follows:

HAMILTON PATH:

INSTANCE: A graph G.

QUESTION: Is there a path in G that visits every vertex exactly once?

To show that SAT is at least as hard as HAMILTON PATH we must establish a reduction R from HAMILTON PATH to SAT.

If such a reduction exists, we can say that Hamiltonian problem is NP-complete.

Traveling-salesman problem

In the *traveling-salesman problem(TSP)*, which is closely related to the hamiltonian-cycle problem, a salesman must visit n cities. Modeling the problem as a complete graph with n vertices, we can say that the salesman wishes to make a *tour*, or hamiltonian cycle, visiting each city exactly once and finishing at the city he starts from. The salesman incurs a nonnegative integer cost c(i,j) to travel from city i to city j , and the salesman wishes to make the tour whose total cost is minimum, where the total cost is the sum of the individual costs along the edges of the tour.

Proof that Traveling-salesman problem is NP-Complete

We first show that TSP belongs to NP. Given an instance of the problem, we use as a certificate the sequence of n vertices in the tour. The verification algorithm checks that this sequence contains each vertex exactly once, sums up the edge costs, and checks whether the sum is at most k. This process can certainly be done in polynomial time.

The problem can be reduced to Hamiltonian cycle. Hence, we can say that Traveling-salesman problem is NP-complete, as Hamiltonian cycle is NP-Complete

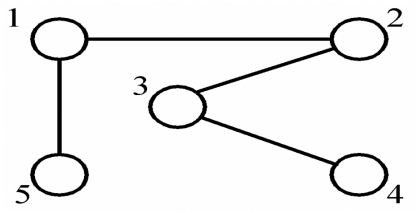
Vertex Cover

Vertex cover is an important problem. It is formally stated as follows: Given a graph G=(V, E), S is the node cover if $S \subseteq V$ and for every edge $(u, v) \in E$, either $u \in S$ or $v \in S$ or both.

The ***size*** of a vertex cover is the number of vertices in it.

Example 1: Consider the graph shown in Fig. 1.

The vertex cover finds minimal vertices that cover the entire graph. Some of the possible solutions node covers of this problem are {1, 3} and {5, 2, 4}.



Proof that Vertex cover problem is NP-Complete

Vertex cover is a NP complete problem as the solution involves guessing a subset of vertices, count them, and show that each edge is covered. This is simple if the number of vertices is smaller. But if the number of vertices becomes large, the number of possibilities becomes larger. Therefore, the algorithm becomes exponential algorithm.

The Vertex Cover problem can be reduced to Hamiltonian problem and we can also verify the solution if we have a solution of the problem.

Set Cover Problem

The set cover decision problem is to determine if F has a cover T containing no more than c sets.

Example 2: Consider the following sets :

$F = \{(a_1, a_3), (a_2, a_4), (a_2, a_3), (a_4), (a_1, a_3, a_4)\}$

s_1 s_2 s_3 s_4 s_5

Find set cover of the above problem?

Like vertex cover, set cover finds the minimum number of sets that covers all the elements. Some of the possible solutions are

$T = \{s_1, s_3, s_4\}$ and $T = \{s_1, s_2\}$.

Proof that Vertex cover problem is NP-Complete:

Set cover is a NP complete problem as the solution involves guessing a subset of vertices, count them, and show that universal set is well covered. This is simple if the number of elements is smaller. But if the number of elements becomes large, the number of possible set covers becomes larger. Therefore, the algorithm for set cover becomes exponential algorithm.

The Set Cover problem can be reduced to Hamiltonian problem and we can also verify the solution if we have a solution of the problem.

Therefore, set cover problem is NP complete.

NP-hard

What does NP-hard mean?

> A lot of times you can solve a problem by reducing it to a different problem.
>
> One can reduce Problem B to Problem A if, given a solution to Problem A, one can easily construct a solution to Problem B. (In this case, "easily" means "in polynomial time.“).

A problem is NP-hard if all problems in NP are polynomial time reducible to it.

In other word, the problem can be reduced to a different problem but we have no know way to verify if the solution is correct

Example of NP Hard

PCA?

Proof of P = NP

Difference between NP hard and NP complete problem

NP hard	**NP complete**
NP-Hard problems(say X) can be solved if and only if there is a NP-Complete problem(say Y) that can be reducible into X in polynomial time.	NP-Complete problems can be solved by a non-deterministic Algorithm/Turing Machine in polynomial time.
To solve this problem, it do not have to be in NP .	To solve this problem, it must be both NP and NP-hard problems.
Do not have to be a Decision problem.	It is exclusively a Decision problem.
Example: Halting problem, etc.	Example: Determine whether a graph has a Hamiltonian cycle, Determine whether a Boolean formula is satisfiable or not, Circuit-satisfiability problem, etc.

Questions

1.a. How are P and NP problems related?
2.Compare NP-hard and NP-completeness?
3.Explain the class of P and NP with example?
4. Differentiate between NP- complete and NP-hard problems?
5. Explain the satisifiability problem
6. Explain Reduction
7. Mention any two decision problems which are NP-Complete.
8. Prove that Vertex Cover problem is NP-Complete

Bibliography

Machine Learning in Action Peter Harrington
Charu Aggarwal
Linear Algebra and Its Applications David Lay
Linear Algebra and its Applications ,Gilbert Strang
Linear Algebra and Probability for Computer Science Applications – Ernest Davis
Introduction to Algorithms
Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford Stein
Algorithms
S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani