**Tutorial for Optimization Algorithms**

最佳化演算化的基礎介紹

Yu-Chia Huang

revised by Jian-Jiun Ding

黃昱佳 著 丁建均老師 修訂

國立台灣大學電信工程學研究所

Graduate Institute of Communication Engineering

National Taiwan University

December, 2021

**Content**

**Notation 2**

**Chapter 1 Convex Optimization 2**

1-1 Affine and convex sets................................................................................................................2

1-2 Operations that preserve convexity.............................................................................................4

1-3 Convex functions........................................................................................................................5

1-4 Convex optimization problems...................................................................................................8

1-5 Duality.......................................................................................................................................11

**Chapter 2 Optimization Algorithms 17**

2-1 Descent methods.......................................................................................................................19

2-2 Gradient descent method...........................................................................................................20

2-3 Newton’s method......................................................................................................................24

**Chapter 3 Inverse Problems 28**

3-1 Background...............................................................................................................................28

3-2 Gradient Projection for Sparse Reconstruction (GPSR)...........................................................29

3-3 minimization problems........................................................................................................32

**Chapter 4 Maximum Likelihood Estimation 40**

**Chapter 5 Conclusion and Further Reading 42**

**References 43**

**Notation**

: the set of real numbers.

: the set of nonnegative real numbers.

: the set of positive real numbers.

: the set of symmetric matrices.

: the set of symmetric positive semidefinite (PSD) matrices.

(semidefinite means that **xH** **x** ≥ 0 for all real and nonzero n-entry **x**)

: the set of symmetric positive definite matrices.

(definite means that **xH** **x** > 0 for all real and nonzero n-entry **x**)

denotes an -valued function on some subset of .

denotes the effective domain of function If , then

**dom** *f* = {*x* ∈ | *f*(*x*) is not ±∞}.

**The curled inequalities** : They are generalizations of inequalities. They represent component-wise inequality between two vectors or two matrices. For example, **x** **y** means that *x*[*n*] > *y*[*n*] for all *n*. It sometimes means the matrix inequality between two symmetric matrices. For example, if a symmetric matrix satisfies , then , , is positive semidefinite.

1. **Convex Optimization**
   1. **Affine and Convex Sets**

**1-1-1 Affine sets**

Before introducing convex sets, we first take a look at affine sets.

**[Definition 1.1] (Affine set):**

We say that a set is affine if for any two distinct points on *A*, all the points on the line passing through the two points also lies in the set , i.e., for two distinct points and for any ,

(1)

For example, any plane in is an affine set. The affine set is a generalization of a convex set.

**[Definition 1.2] (Affine combination):**

Let . With an affine combination of these points is defined as

(2)

**(Property 1.1):**

A set is affine if and only if it contains every affine combination of the points on the set.

**[Definition 1.3] (Affine hull):**

The set of all affine combinations of points in some set is called the affine hull of and is denoted as **aff** , i.e.,

(3)

Alternatively speaking, an affine hull for one set is the smallest affine set that contains it.

**1-1-2 Affine dimension and relative interior**

**[Definition 1.4] (Affine dimension):**

The affine dimension of a set is defined as the dimension of its affine hull.For example, consider a unit circle in . The affine hull of , is the whole space. (Note that a vector in can be expressed as (*a*, *b*) where *a* and *b* are real. We can set

.

Then, *rx*1 + (1−*r*)*x*2 = (*a*, *b*). Therefore, the affine dimension of this unit circle is 2.

**[Definition 1.5] (Interior):**

The interior of a set is defined as

,

where a point is called an interior point of if there exists an for which is a subset of .

**[Definition 1.6] (Relative interior):**

We define the relative interior of a set , denoted as its interior relative to:

,

where is the ball of radius and center in the *L*2-norm.

For example, consider a circle in the -plane in , which is

(4)

1. Its affine hull is the -plane, .
2. The interior of is (Note that since *x*3 = 0 should be satisfied, if , then there is no *ε* such that for all that satisfy .)
3. The relative interior of is (Note that, in this case, and *x* = (0, 0, 0)).

**1-1-3 Convex sets**

**[Definition 1.7] (Convex set):**

A set is convex if the line segment of any two points in lies in , i.e., for any and any , we have

(5)

Apparently, any affine set is also convex.

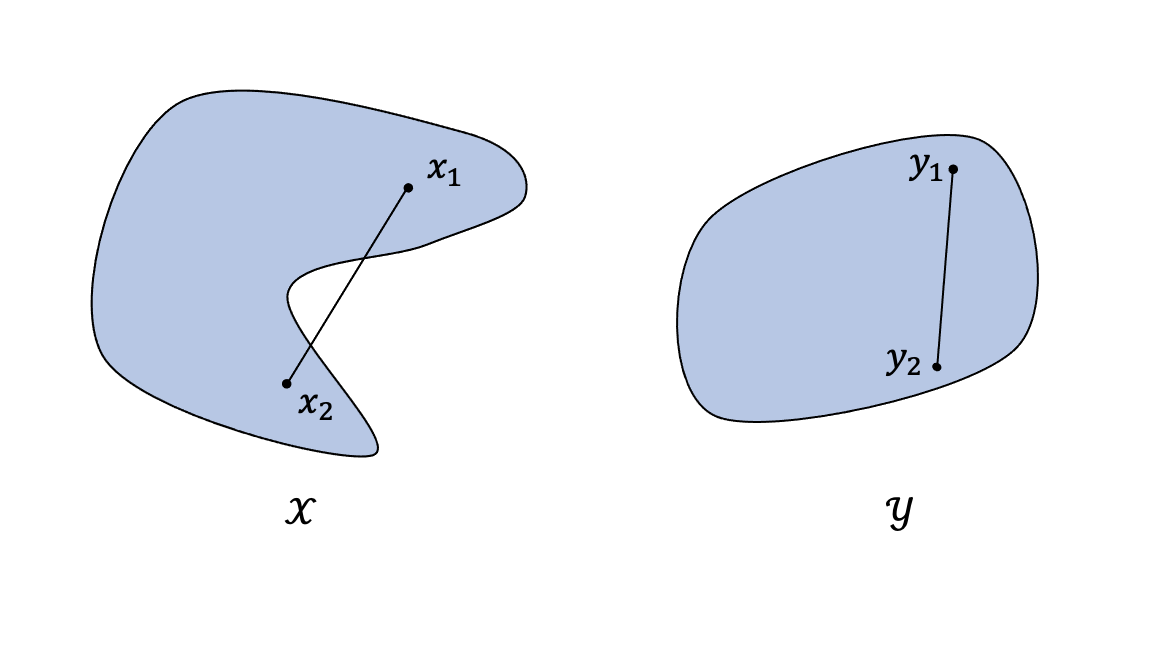


Figure 1.1 Nonconvex set : there are some convex combination of is not in

and convex set .

**[Definition 1.8] (Convex Combination):**

Let . With and , a convex combination of these points is defined as

(6)

Note that, compared to affine combination, the constraint that is required.

**(Property 1.2):**

A set is convex if and only if it contains every convex combination of its points.

**[Definition 1.9] (Convex hull):**

The set of all convex combinations of points in some set is called the convex hull of and is denoted as **conv**, i.e.

(7)

Alternatively speaking, an affine hull for one set is the smallest convex set that contains it.

**1-2 Operations that preserve convexity**

In this section, we discuss about some operators that can preserve the convexity of sets.

**1-2-1 Intersection**

**(Property 1.3):**

Intersection preserves convexity. Suppose are convex sets, then is convex.

**1-2-2 Affine Functions**

**[Definition 1.10] (Affine Function)**

A function is affine if it is a sum of a linear function and a constant. That is, it has the form , where and .

**(Property 1.4):**

Suppose that is a convex set and is an affine function. Then the image of under , i.e., , is also convex.

(Proof):

Let be two distinct points in and be in the images of under (*x*) = *Ax*+*b*.

Since is convex, for , Thus, .

**1-3 Convex functions**

**1-3-1 Definitions**

**[Definition 1.11] (Convex Function):**

A function is convex if **dom** is a convex set:

. (8)

for all **dom** and for all .

The inequality (8) means that the line segment between and , as known as the chord from to , lies above the graph of . Figure 1.2 shows an example of the convex function.

**[Definition 1.12] (Epigraph)**

The epigraph of a function is defined as **epi** , which is a subset of (*x* ∈ and *t* has the dimension of 1). Observe the epigraph of convex functions, we can find out that a function is convex if and only if its epigraph is a convex set (another definition of convex functions).

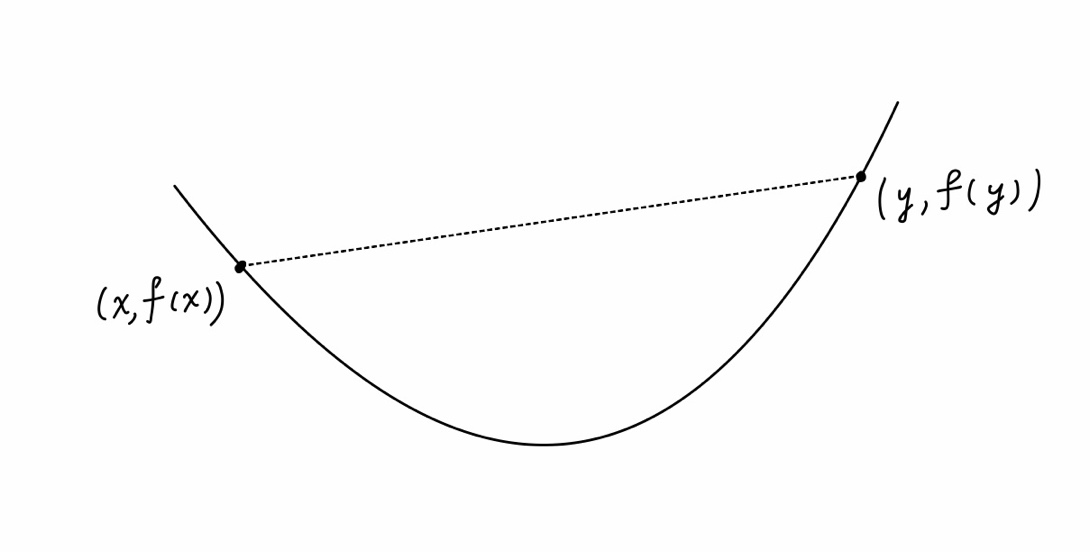


Figure 1.2 Graph of a convex function.

**[Definition 1.13] (Strictly Convex):**

A function is strictly convex if strict inequality holds in (8) (≤ is replaced by <), that is for all **dom** and for all ,

. (9)

**[Definition 1.14] (Concave Function and Strictly Concave):**

A function is concave if −*f* is convex, and *f* is strictly concave if −*f* is strictly convex.

**[Example 1.1]**

Some convex function examples are as follows:

1. Exponential functions: is convex on , for any .
2. on is convex.
3. The -norm (*p* ≥ 1) (see Definition 1.16) is convex.

Some nonconvex (and neither concave) function examples are as follows:

1. on is nonconvex.
2. Periodic function, for example, sine function on is nonconvex.

**[Definition 1.15] (Norm):**

We call a function a norm if for any , we have the following four properties:

1. is nonnegative, i.e., .
2. is definite, i.e., only if .
3. is homogenous, i.e.,
4. satisfies triangular inequality, i.e.,

**[Definition 1.16] (-norm):**

Let The -norm for a vector is defined as

(10)

The -norm is convex for all . However, the-norm with is nonconvex. In fact, when , the constraint (iv) in Definition 1.15 may not be satisfied. For example, = 4 but Therefore, (10) is a norm only when

**1-3-2 First-Order Condition**

**[Theorem 1.1]: First-Order Condition for Convex**

Assume that is differentiable, that is, its gradient exists at every point in . Then is convex if and only if (i) is convex and (ii) for all

. (11)

The inequality (11) shows that from a local analysis (function value and derivative at a point), we can derive the global information (a global under-estimation of it). For example, in inequality (11), if , then for all , , which means that is a global minimum of the function .

**(Corollary 1.1): First-Order Condition for Concave**

Similarly, is concave if and only if is convex and for all

. (12)

**1-3-3 Second-Order Condition**

**[Theorem 1.2]: Second-Order Condition for Convex**

Suppose that a function is twice differentiable, that is, its Hessian exists at each point in . Then is convex if and only if is convex and its Hessian is positive semidefinite, i.e.

for all .

**(Corollary 1.2): Second-Order Condition for Concave**

Similarly, is concave if and only if is convex and for all .

Take quadratic functions as an example. Consider a quadratic function , with, given by

(13)

with , and . Since

,

*f* is convex if and only if . Specially, if *n* = 1, then *f* is convex if and only *P* ≥ 0.

**1-4 Convex Optimization Problems**

In this section, several examples of convex optimization problems are described. We use the notation

minimize

subject to , (14)

, .

to describe the problem of finding that can minimize among all that satisfy the conditions , , and , . We call the *optimization variables*, and the function the *objective function*. The functions are called the *inequality constraint functions*, the corresponding inequalities are *inequality constraints*, and means the number of inequality constraints. The functions are called the *equality constraint functions*, the corresponding equalities are *equality constraints*, and means the number of equality constraints. We say that the problem (14) is *unconstrained* if there are no constraints, i.e.,

The set

is called the *domain* of the problem. A point is called *feasible* if for all and for all . If there exists some that is feasible, then the problem is called feasible; otherwise, the problem is called infeasible if there is no feasible point in . The set of all feasible points is called the *feasible set*.

The *optimal value* is denoted as , which is defined as

where inf means the infimum (i.e., the greatest lower bound). Correspondingly, we call the *optimal point*, if is a feasible point, and . The set of all optimal points is the *optimal set*

We say the problem is *solvable* or *achievable* if there exists at least one point in the optimal set .

**1-4-1 Convex Optimization Problems in the Standard Form:**

**[Definition 1.17] Standard Form of the Convex Minimization Problem**

minimize

subject to , (15)

, .

The objective function and the inequality functions must be convex. The equality constraint functions , as the form shows, must be affine (see Definition 1.10). Also, the feasible set of a convex optimization problem is convex.

**[Definition 1.18] Standard Form of the Concave Maximization Problem**

maximize

subject to , (16)

, .

with objective function being concave. The constraint functions (including inequalities and equalities) have the same forms as standard form of convex optimization problems. Since we know that from the definition of the concave function (Definition 1.14), a function is concave if is convex, we can solve this problem (16) with minimizing subject to the same constraints.

**[Example 1.2]: Reformulate a Problem to the Standard Form**

Consider an optimization problem,

maximize

subject to (17)

,

we can reformulate it to the standard form of convex optimization problem:

minimize

subject to (18)

Note that is a concave function for and is a convex function.

**1-4-2 Linear Optimization Problems**

When the objective and constraint functions are all affine (see Definition 1.10), the convex optimization problem in (15) is called a *linear program* (LP). Generally, a linear optimization problem has the following form:

maximize

subject to , (19)

where variable , the coefficient vector , and the constant . In the constraint functions, we summarize linear inequalities in one matrix inequality and the linear equalities in one matrix equality, where and .

**1-4-3 Quadratic Optimization Problems**

When the objective function is (convex) quadratic, and the constraint functions are all affine, the convex optimization problem (15) is called a *quadratic program* (QP) and it has the form,

minimize

subject to (20)

,

where , , and .

If the objective function and the inequality constraint function in the convex optimization problem (15) are both quadratic,

minimize

subject to (21)

,

where , and . Then the convex optimization problem is called the quadratic constrained quadratic program (QCQP). If in the inequality constraints of QCQP problem (21), then it becomes an QP problem. Thus, we know that the QCQP problem is a more general one than the QP problem.

**1-5 Duality**

**1-5-1 The Lagrangian and Dual Function**

Consider an optimization problem in the standard form (14),

minimize

subject to , (22)

, .

with variable . We assume that the domain of the problem (22) is nonempty, and denoted the minimal value of *f*0(*x*) as . The problem is unnecessary to be convex.

**[Definition 1.20] The Lagrange Duality**

The idea of Lagrange duality is to take the constraints in (22) into account by augmenting the objective function with a weighted sum of the constraint functions. Thus, the associated Lagrangian is defined as

(23)

with . In the Lagrangian, we refer as the Lagrange multipliers associated with the th inequality constraints and refer as the parameter of the th equality constraints. and are called the dual variables associated with the original problem in (22).

**[Definition 1.20] The Lagrange Dual Function**

The Lagrange dual function (also called the dual function) is defined as the minimum value of the Lagrangian over *x*. It can be formulated as:

(24)

Since the dual function is the pointwise infimum of a family of affine function of , it is concave, even in the condition that the problem (22) is not convex.

**(Property 1.5): Lower Bounds on the Optimal Value**

Assuming that is a feasible point for the problem in (22), that is and , and the Lagrange multiplier *λ* satisfies . Then, we have

and therefore

Thus,

(25)

The inequality (25) holds for all in the feasible set of (22). Thus, for any and any , we have

, (26)

where denotes the optimal value. The dual function produces the lower bound of *f*0(*x*) on all feasible points of problem (22), which is also the optimal value .

**[Example 1.3] Example for Lagrangian and the Dual Function**

For example, consider a linear program

minimize

subject to (27)

with variable Since is equivalent to , the inequality constraint functions are and . The associated Lagrangian is

Then the dual function is

Note that, when ≠ 0, then does not have a finite lower bound.

**1-5-2 The Lagrange Dual Problem**

From the discussion in last section, we know that forand any , the dual function yields a lower bound on the optimal value of of the optimization problem (22).

**[Theorem 1.3]: Lagrange Dual Problem**

Determining *x* to minimize *f*0(*x*) in (22) is equivalent to solve the *Lagrange dual problem* as follows:

maximize

subject to . (28)

With no required constraint on variable . This problem is called the *Lagrange dual problem* associated to the original problem (22), which we sometimes call the *primal problem*. We describe any pair with and to be dual feasible. That means the pair is feasible for the dual problem (28). We describe a pair as dual optimal if it is optimal for this problem.

The optimization problem (28) is a convex optimization problem since we know that the objective function being maximized (i.e., is concave from previous discussion, and the inequality constraint function is convex.

**[Example 1.4]** Suppose that the object function is:

. (29)

If we want to minimize *f*0(*x*) subject to the constraint of   
  , (30)  
then we can convert the constraint into  and generate the associated Lagrangian as   
 , (31)

   
The extreme points of *L*(*x*, *λ*) satisfy   
 , (32)  
  .   
Therefore, we can set   
 .   
Then   
 .   
Moreover, at the extreme point,   
 . (33)  
Note that *λ* = 5/2 satisfies the inequality in (28). Therefore,   
 . (34)  
In other words, the point that can minimizes  and satisfies  is (*x*1, *x*2) = (2/5, 1/5). At this point, *f*0(*x*) = 1/5.

**1-5-3 Weak and Strong Duality**

**[Definition 1.21]: Weak duality**

We denote the optimal value of the dual problem as , which is the greatest lower bound on (the optimal value of the primal problem (22) that we can obtain from the dual function (28)). That is, an inequality

(35)

holds for any optimization problem in the form (22). And we call this *weak duality*. This is sometimes used to find a bound on the optimal value of a problem that is difficult to solve, while the duality problem is always convex and can often be solved efficiently.

**[Definition 1.22]: Optimal Duality Gap**

We call the difference the optimal duality gap of the primal problem (22), since it shows the gap between the optimal value of the primal problem and the best lower bound that can be obtained from solving the dual problem in (28). Since the weak duality always holds, the optimal gap is always nonnegative.

**[Definition 1.23]: Strong Duality**

If the equality holds in (35), that is

(36)

holds, then we say the *strong duality* holds, i.e., the optimal duality gap is 0. This property means that the greatest lower bound obtained by the Lagrange dual function is tight. It also means that we can get the optimal value by solving dual problem.

**[Definition 1.24]: Slater’s Condition**

Consider a convex optimization problem, just like problem (15)

minimize

subject to , (37)

,

with the inequality constraint functions convex. The problem is said to satisfy *Slater’s condition* if there exist an (Definition 1.6) such that

**[Theorem 1.4]: Slater’s Theorem**

If the problem is convex and it satisfies Slater’s condition, then the strong duality holds. The proof can be seen in subsection 5.3.2 in [1]. It should be noted that this is not the only condition that guarantees strong duality.

**1-5-4 Optimality Conditions**

Optimality conditions are derived by assuming that we are at an optimum point, and then study the behavior of the functions at that point. [2]

**[Theorem 1.5]: Complementary Slackness and Strong Duality**

Suppose that the strong duality holds in problem (22). Let be the primal optimal point and be the dual optimal point. That is,

The first equality follows the strong duality. The second equality comes from the definition of the duality function. The third inequality is from that the infimum is the lower bound for every point , include the primal optimal point . The final line comes from and . We can conclude that the two inequalities hold with equality.

Therefore, we have a conclusion:

Since each term in this sum is nonpositive (), we have

(38)

This condition is called complementary slackness; it holds when strong duality holds.

**[Definition 1.25]: Karush-Kuhn-Tucker (KKT) Conditions**

Consider an optimization problem as the form in (22), which is not necessarily convex. We assume that the all functionsin the problem are differentiable. Suppose that any pair of primal and dual optimal points with zero duality gap. Since minimizes the Lagrangian then its gradient must vanish at , that is

(39)

Thus, we conclude that it must satisfy the following conditions,

(40)

The first and second lines are the constraints in the primal problem (22). The third condition comes from the constraint in the dual problem. The forth line is the complementary slackness.

If the optimization problem is convex, i.e., the form in (37), the *Karush-Kuhn-Tucker (KKT) conditions* as follows are sufficient for a point to be optimal. That is, if are any points that satisfy KKT conditions,

(41)

Note that, in Example 1.4, (32) is analogous to (40). Therefore, Example 1.4 is essentially to apply the KKT condition to solve the optimization problem.

**[Example 1.5]** Consider the equality constrained quadratic program

minimize

subject to , (42)

where and . The Lagrangian is

(43)

where the Lagrange multiplier vector Then KKT condition is

, . (44)

We can write the optimal conditions as

(45)

Solving these equations of variables gives the optimal primal and dual variables for (42).

1. **Optimization Algorithms**

This chapter is about the way to solve optimization problems. We first consider solving the unconstrained optimization problem

(46)

where is convex and twice continuously differentiable. Assume that the problem is solvable, i.e., the optimal point exists.

**Initial Point and Sublevel Set:** We require a suitable starting point which lies in (domain of function . In addition, the sublevel set

must be closed. It is clear that, since we are minimizing the function , we should only consider the point that has the function value lower than the initial value.

In order to analyze the convergence of optimization algorithms, we first see some definitions related to the objective .

**[Definition 2.1]: (Strong Convexity)**

A twice differential function *f* is said to be -strongly convex if there is an such that

, (47)

for all , where is an identity matrix. The symbol means that is positive semidefinite. We know that, from Taylor’s theorem, for all , there exists some such that

(48)

By the assumption of strong convexity (47), the last term is at least , then

(49)

for all

We then show that the inequality (49) can be used to bound , where denotes the optimal value of problem (41). That is, to analyze how far is one point from optimal value. By setting

we find that minimizes the righthand side of (49). Thus,

The inequality holds for any , so

(50)

This shows that if the gradient of a point is small, then it is nearly optimal.

We can also derive a bound on , the distance between any and optimal point

(51)

(Proof):

Apply (49) with ,

We use the Cauchy-Schwarz inequality in the second inequality. Since is the optimal value, we have

then we get inequality (51).

**[Definition 2.2]: (Smoothness):**

We say a function is -smooth for some if

, for all (52)

This also means an upper bound on Hessian . Analogous to (49), smoothness also implies for all

(53)

Analogous to the process to derive (50), after minimizing each side of (53) over , we have

(54)

In these two conditions, we can bound the optimal value, and is also benefit to the further discussion of convergence.

**[Theorem 2.1]: Condition Number**

Suppose that a function is -strongly convex and -smooth, i.e., The ratio is thus an upper bound on the condition number of the Hessian that is, the ratio of its largest eigenvalue to its smallest eigenvalue. The condition number is the ratio of the largest eigenvalue to the smallest eigenvalue of a matrix.

We know that the eigenvalues of the Hessian characterize the steepness of a point. The condition number of the Hessian, being the ratio of the largest eigenvalue to the smallest, is the ratio of the steepest ridge's steepness to the shallowest ridge's steepness. See the example of the gradient descent method to discuss how the condition number influences the convergence rate.

**2-1 Descent Methods**

All methods mentioned in this section is to minimize the sequence , , where

(55)

and the step-size . means the search direction. The “descent” means that

(56)

except when is optimal. That is, we should set to satisfy

(57)

**{Algorithm 2.1} (General Descent Method)**

**(Initial)**: A starting point (may be any point in the domain of the function )

**repeat**

1. Determine a descent direction
2. (Line search) Choose the step size .
3. (Update) .

**until** the stopping criterion is satisfied.

We call the second step as line search since we choose a step size along the line are two common ways to search for an appropriate step size.

**[Definition 2.3] Exact Line Search**

Exact line search is simply to choose a step size that minimizes along the ray ,

), (58)

which is also an optimization problem. It is used when minimizing (58) takes lower cost (in time or space) than computing the search direction takes.

**{Algorithm 2.2} Backtracking Line Search**

Instead of finding the exact step size, we approximate the step size of each iteration. Its process is as follows:

**given** a descent direction ,

**while**

This method is called “backtracking” because we start the searching from the unit step and then reduce it by the factor . Since we have mentioned that , then by using Taylor’s first order approximation, we have for small enough ,

(59)

which means that backtracking line search eventually terminates.

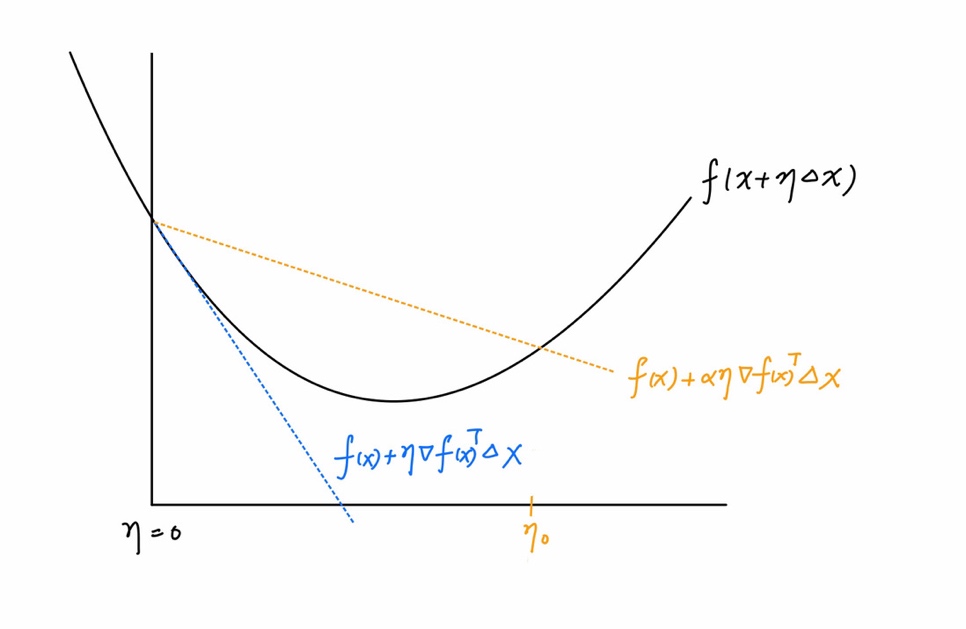


Figure 2.1 Graph of backtracking line search

As in Fig. 2.1, is the step size that satisfies

It follows that we might get a step size or

**2-2 Gradient Descent Method**

The gradient descent method is the simplest strategy to minimize a differential function on . The idea is to make a small step in the direction that minimizes the local first order Taylor approximation of . Therefore, we set the descent direction as the negative gradient .

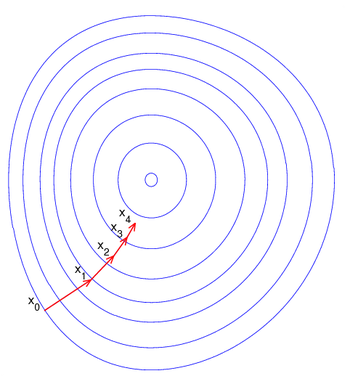


Figure 2.3 Illustration of the gradient descent method with several iterations referred to [2].

**{Algorithm 2.3} (Gradient Descent Method)**

(**Initial**) a starting point (may be any point in the domain of the function )

**repeat**

1. (i.e., is equal to the inverse of the gradient of *f*(*x*))
2. (Line search) Choose a step size .
3. (Update) .

**until** stopping criterion is satisfied.

The stopping criterion is usually of the form

**2-2-1 Convergence of Gradient Descent with Fixed Step Size**

We then discuss the convergence of gradient descent without the line search step. Suppose that the objective function is -smooth and -strongly convex for some , i.e., Given the starting point the gradient descent with satisfies the following inequality for

(60)

where denotes the minimizer. The proof of (60) and its related reading is in Section 3.4 of [3]. We know that . Therefore, the inequality implies the convergence of gradient descent (to the optimal point We can conclude that the convergence rate is at least linear in the log scale.

**2-2-2 Convergence with Line Search**

In this section, we use the notation to describe the current point and the point in the next step of the gradient descend method. We assume that for all Define the function   by that is, a function of the step length in the negative gradient direction. From the inequality in (53) and set *y* − *x* = , we have

(61)

Then, we discuss the case where **exact line search** (see Definition 2.3) is used, then we minimize over both sides of the inequality (61). We denote the step size as . The righthand side is minimized by (by taking partial derivative over , and the minimum value is . Therefore, we get

(62)

Subtracting both side by optimal value ,

Applying the inequality in (50), by the assumption of strong convexity, , we get

(63)

Recursively applying this inequality, we have

(64)

where is the initial point, and . We get the result similar to the one we get with fixed step size (45). We can conclude that the convergence rate is at least linear (in the log scale).

**backtracking line search**

We then consider the case where **backtracking line search** (see Algorithm 2.2). Now we will show that the backtracking termination condition with

(65)

is satisfied whenever Note that if ,

Therefore, with the bound in (61), we have the termination as

We have the last inequality due to the setting of is in the range of Thus, the line search terminates when (since we start from or with the value , where is the one we mention in descent method. The second condition is because is -smooth, , with the bound , , thus the worst case for the termination is the bound

In the first case, we have

(66)

and in the second case, we have

(67)

Combine (66) and (67), we get

Then, analogous to the process to derive (63) in exact line search, we have

(68)

Therefore, we have the conclusion of convergence

(69)

where we know that . Same as above, the convergence rate is at least linear (in the log scale).

**[Example 2.1] An Example of the Gradient Descend**

We take a quadratic problem in as example. The objective function is

(70)

where It is clear that the optimal point should be and the optimal value should be

To discuss the convergence of minimizing (70) by the gradient descend method, first, we determine the condition number. The Hessian is constant, , the eigenvalue is 1 and . The condition number is

Thus, the strictest choice of strongly convex constants and smooth constant is

and .

We apply the gradient method with exact line search, from the starting point It can be shown that

and the corresponding function value is

The convergence is linear (in the log scale), that is the error is exactly a geometric series, reduced by the factor at each iteration.

If , then we find the solution in one iteration; if is not far away from 1 (say, between 2 and ½), the convergence is rapid. However, if or (condition number is far from 1), the convergence is very slow (e.g., in the case where = 10, from Figure 2.3, the route is zigzag).

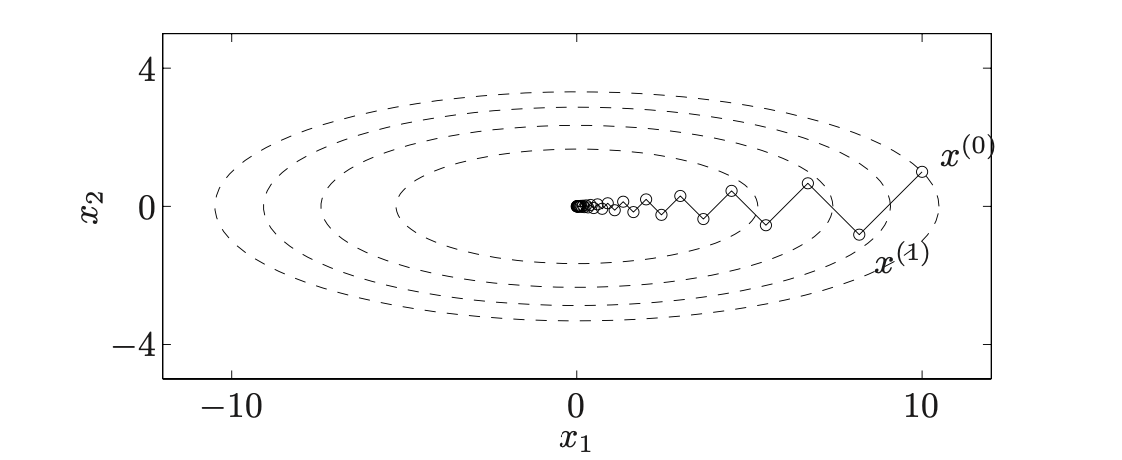


Figure 2.3 Contour lines of function . The figure shows the iteration steps of gradient descent with exact line search

**2-3 Newton’s Method**

Before introducing Newton’s method, let us take a look at the 2nd order Taylor approximation   of at

, (71)

which is a convex quadratic function of . It is minimized when by taking the partial derivative over and setting to zero. We call the vector the *Newton step* , for .

(72)

Thus, the Newton step is the vector that we should add to point to minimize the second-order approximation.

Also due to the convexity of the function, implies that

(73)

unless

**2-3-1 Steepest Descent**

**[Definition 2.4] (Normalized Steepest Descent Direction):**

Let be any norm on . A *normalized* steepest descent direction with respect to the norm is defined as

(74)

which gives the largest decrease with unit norm (distance) in the linear approximation of , i.e.,

(75)

**[Definition 2.5] (Steepest Descent Direction):**

The steepest descent step is the unnormalized form of It is defined as

(76)

where is the associated dual norm of the norm , which is defined as

In the above equation, sup means the supremum (i.e., the least upper bound).

Let us go back to the steepest descent. For the Euclidean norm (-norm, the dual norm is Euclidean norm, too), the steepest descent (same as the gradient descent direction vector). We can simply observe that the normalized steepest descent direction of it is . Thus, the steepest descent direction is

**(Property 2.1) Affine Invariance**

The Newton step is independent of linear (or affine) changes if coordinates. Assume that is nonsingular ( exists), and define Then

where . Thus, from (72), the Newton step for at is

where is the Newton step for at . Therefore, the relationship of and and the Newton steps and are the same,

(77)

**2-3-2 The Newton Decrement**

The *Newton decrement* at is defined by

(78)

Note that it is similar to (73). We often use this quantity as the stopping criterion (see Algorithm 2.4). The following equation shows how the New decrement relates to the quantity where is the 2nd order approximation of at (see equation (71)):

The above equation is from the facts that, if (71) is satisfied, then and that = . Because , where is the optimal value, is an estimator of which means how far is the value from the optimal value.

In backtracking line search, we have

. (79)

(see Algorithm 2.2), and can be used as the directional derivative of at

Therefore, the direction we aim for is established.

**2-3-3 Newton’s Method**

**{Algorithm 2.4} (Newton’s Method)**

**repeat**

Compute the Newton step and decrement:

Stopping criterion. If , then **quit**.

Line search. Choose step size by backtracking line search.

Update.

**2-3-4 Convergence Analysis**

Assume that the function is -convex and the Hessian is Lipschitz continuous on with constant , i.e.,

(80)

measures how will can be approximated by a quadratic function. For example, if the objective function itself is quadratic, then the constant can be taken 0. Then, there exists some constants , such that

(81)

See the proof and further discussion in subsection 9.5.3 of [1].

**2-3-5 Compare to Gradient Descent Method**

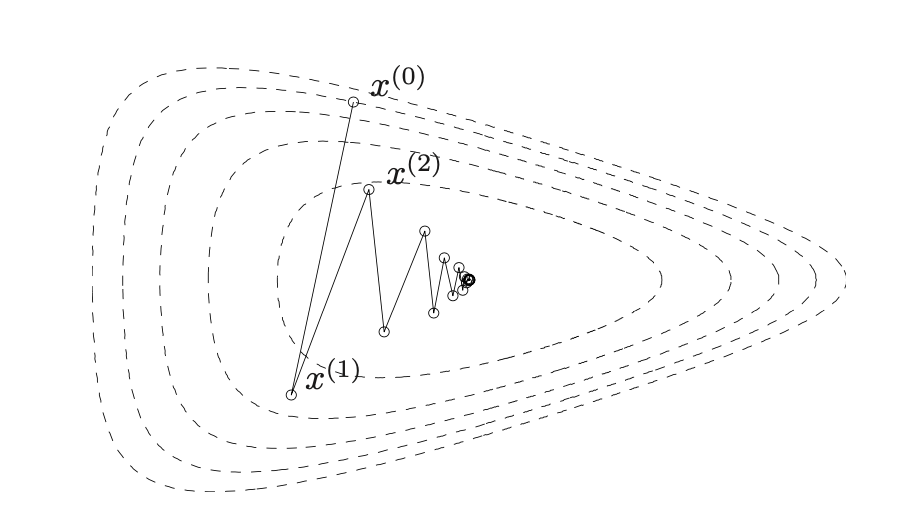


Figure 2.4 Iterations of the gradient descent method to minimize (82) [1]

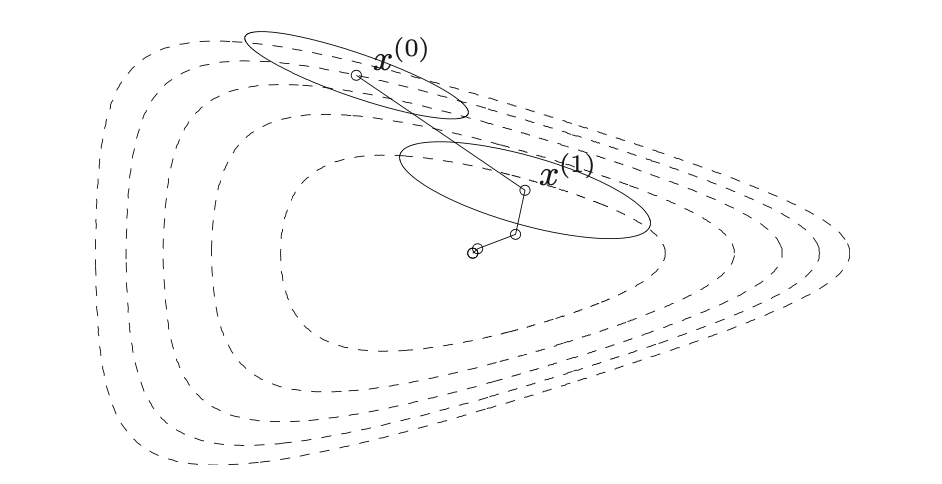


Figure 2.5 Iterations of Newton’s method to minimize (82) [1]. Note that the number of iterations is less when using Newton’s method.

We have introduced two important methods used in unconstrained convex optimization problems, including the gradient descent method and Newton’s method. It is obvious that Newton’s method is the level-up version of gradient descent. We can see that gradient descent in every iteration only confirm that the function value will decrease in next step. In contrast, the Newton’s method not only fulfill this property, but also go to the steepest direction. In subsection 2-3-2, we see that Newton’s method is a very good method when approaching optimal point. Moreover, we know that (in Property 2-1), the Newton’s method is affine invariant and insensitive to the choice of coordinates. This is also the property that the simple gradient descent method does not have.

However, there are also some disadvantages of Newton’s method. The main one is the large cost of forming and storing the Hessian at each point, which requires solving a set of linear equations.

**[Example 2.2]: A Nonquadratic Problem in**

We consider a function

(82)

We minimize this function by applying the gradient descent method with backtracking line search ( and Newton’s method with backtracking line search (. The result is in Figs. 2.4 and 2.5.

1. **Inverse Problems**

**3-1 Background**

We would like to estimate from the observations

(83)

where is unknown but the prior on it is Laplacian, i.e., with parameters and ), is an matrix (known) with , and is a Gaussian white noise of variance . There are many optimization problems that can be formulated as (83) or

, (84)

which is similar to (83) but without the noise term. Since (i.e., the size of the output is smaller than that of the input) in most of the cases, our goal is to overcome the singular nature of the matrix.

Let us look at some frequently used optimization forms to solve y in (83). First, consider a convex unconstrained problem

, (85)

where is a nonnegative parameter. We use the term to make the difference of *y* and *Ax* as small as possible and use the term to limit the magnitudes of the entries of *x*. Two convex constrained problems related to (85) are

subject to (86)

and

subject to (87)

where and are nonnegative parameters. Eq. (86) is a quadratic program (QP) whereas (87) is a quadratic constrained linear program (QCLP). It can be shown that, from the Lagrangian (see subsection 1-5-1), the solution of problem (86) is also the minimizer of (85) for some Similarly, the solution of (87) is either or the minimizer of (85). These claims can be proved by using Theorem 27.4 in [4]

A popular problem, basis pursuit, is (87) with i.e.,

subject to (88)

which is a linear program (LP). We minimize the objective function to get a sparse solution Note that we describe a vector sparse if there contain many zero elements in the vector.

To further get the sparse solution, we may go on to solve the problem form

subject to , (89)

where  and . We know that, if , then the objective function is convex. To minimize this kind of problem is equal to minimize that is the -norm, i.e., . However, for the objective function is no longer convex [5]. Thus, we call this kind of problem the minimization problem. We will discuss more about it in Section 3-3.

This problem often occurs in compressed sensing [6], image deblurring, or image reconstruction.

**3-2 Gradient Projection for Sparse Reconstruction (GPSR)**

In this section, we solve the convex unconstrained problem in (85)

, (90)

where is a nonnegative parameter.

**3-2-1 Formulation**

First, reformulate our problem in (90) as a quadratic problem (QP). We introduce vectors and then make the substitution.

, (91)

where and , the negative part for all . The former one is the positive part of and the latter one is the negative part of for denotes the positive-part operator. Therefore, rewrite (90) as

subject to

(92)

where with length . We can further formulate (92) as

subject to (93)

where , and . This is actually a standard bound-constrained quadratic program (BCQP). We can simply take the gradient of it by matrix calculus, . It is noticeable that the dimension of (93) is twice that of the problem in (90).

**3-2-2 Gradient Projection Algorithms**

In this subsection, we discuss the gradient projection (GP) method for solving problem (93). Generally, for each iteration, we move to the next point as following steps. First, we choose the step size and set

(94)

where the operator keeps the positive component of the vector , otherwise, set to 0, which obeys the constraint of problem (93).

Then we choose another scalar and make a convex combination

(95)

describing the relationship between the points of this iteration and next step. We have two sub-methods with the different choices of and

1. **Basic gradient projection (the GPSR-basic algorithm)**

This is basically a gradient descent method with backtracking line search projecting on the required domain. In this method, we search from along the negative gradient , project onto the nonnegative orthant and perform a backtracking line search until there is enough decrease in . We define the descent direction vector as

(96)

Based on this, we choose the initial guess of as

,

which is

(97)

by taking .

To prevent the value of from getting too small or too high, we confine it to the interval where The complete algorithm is Algorithm 3.1.

**{Algorithm 3.1}: (GPSR-basic Algorithm):**

1. (initialization): Given initial and . Also choose the upper and lower bounds of *η*, i.e., *η*min and *η*max.

**repeat**

1. Compute step size from (97), and replace by
2. (Backtracking line search, like we mention in algorithm 2.2)
3. **while**
5. (Update):
6. If it reaches the convergence test, then **quit**.
7. **Barzilai-Borwein Gradient Projection (the GPSR-BB algorithm)**

With the approximation of Hessian, the second order derivative, the Barzilai-Borwein is introduced. The Hessian is set to be for iteration . The value of is chosen so that

**{Algorithm 3.2}: (GPSR-BB Algorithm):**

1. (initialization): Given initial initialize .

**repeat**

1. (Compute):
2. (line search): Find the scalar that minimizes on the interval .
3. (Update ):
4. (Update : Compute a scalar . Then
5. If it reaches the convergence test, then **quit**.

In the step 3 of Algorithm 3.2, since is quadratic, we have a closed form

(98)

by taking

1. **Termination**

Since we care about the “sparse” property of the extracting vector , we may use the following criterion. First, we define two sets in each iteration step ,

and terminate if

(99)

where tolA is some tolerance threshold. In fact, (99) is equivalent to the ratio that *zi*(*t* −1) ≠ 0 but *zi*(*t*) = 0 (or reversely, *zi*(*t* −1) = 0 but *zi*(*t*) ≠ 0). The algorithms using this criterion terminates when the places of nonzero number in vector does not have many changes in recent steps.

**3-3 minimization problems**

In Section 3-1, we mention the problem (89)

subject to (100)

where is a matrix. We do not consider the presence of noise in this problem setting. As we mentioned in Section 3-1, if , then the objective function is convex, minimizing is equal to minimize -norm. If , then the objective function is nonconvex, and minimize is equal to minimize , but this is not a norm (see Definition 1.16). Therefore, we call this problem minimization problem, instead of -norm minimization problem.

We use an example to show the nonconvexity for . If , consider on . From the definition, the inequality must hold for and . But if we take , and then we have

Therefore, is not a convex function. Other values of is similar to this case. We can also observe the convexity in Figs. 3.1 and 3.2, which were drawn by using the software of GeoGebra.

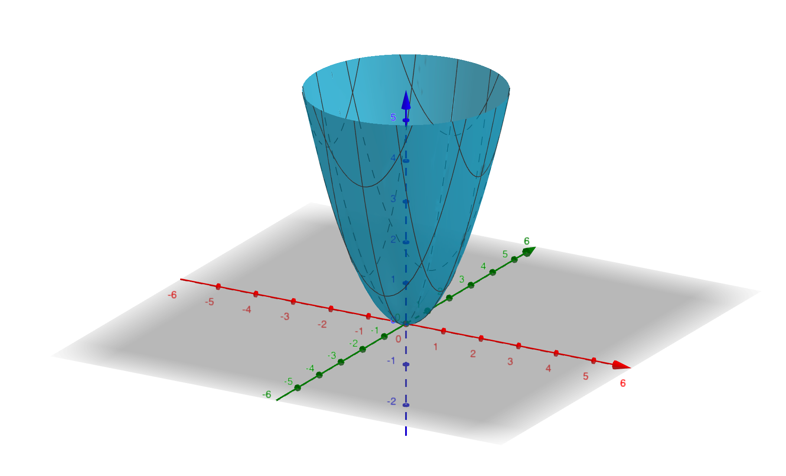
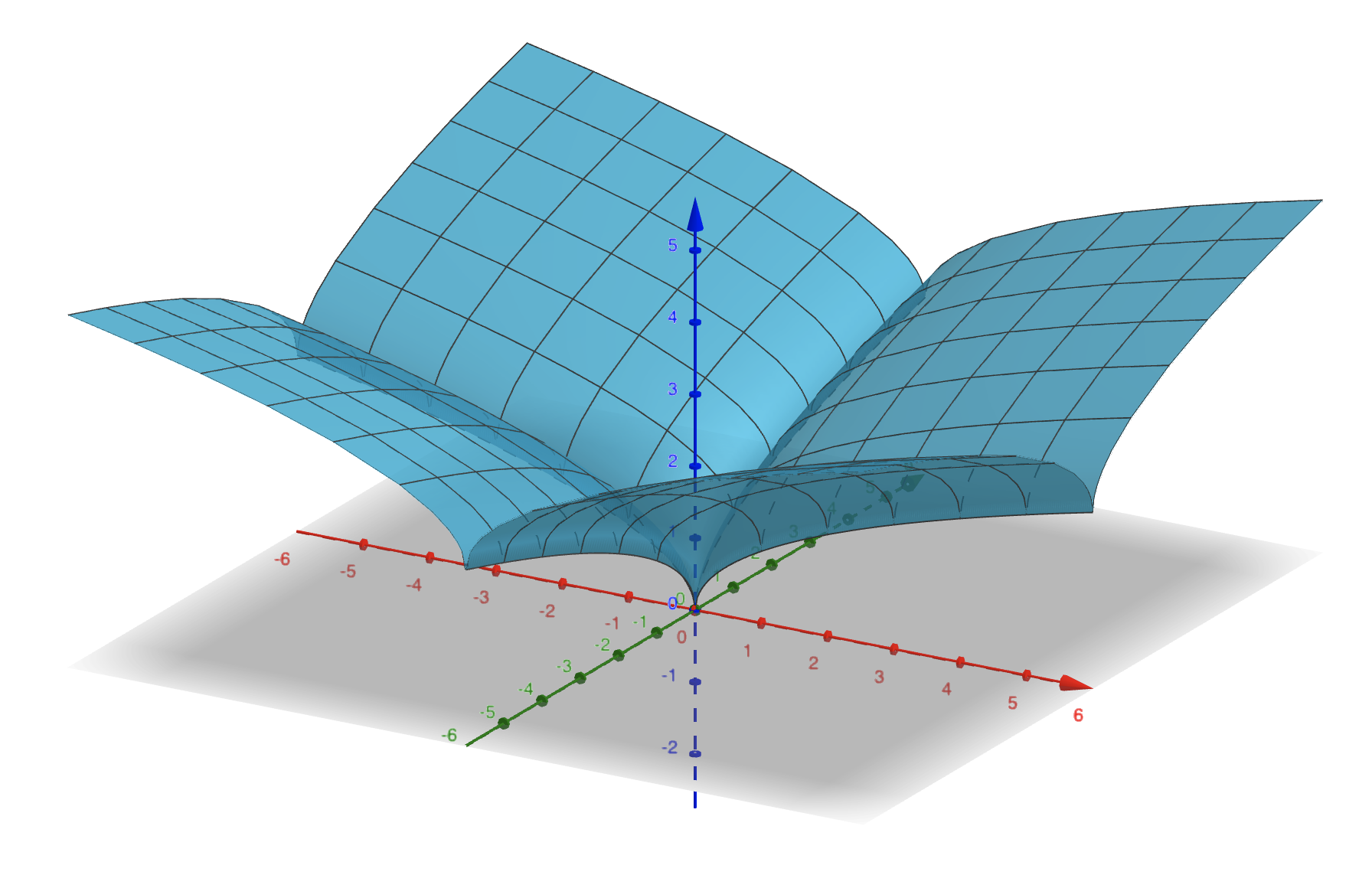


Figure 3.1 with Figure 3.2 with

For different conditions, different algorithms can be applied to solve the optimization problem in (100), as in Table 3.1.

Table 3.1 The methods to solve the optimization problem in (73) for different *p*.

|  |  |
| --- | --- |
| Conditions of | Method |
|  | Pseudo-inverse method |
|  | IRLS algorithm |
|  | SL0 method |
|  | Big-M simplex method |
|  | Normal convex optimization method, e.g., descent method |

**3-3-1 Pseudo-Inverse Method**

When , then we have the optimization problem

subject to (101)

To solve this, we have the optimal point

(102)

To prove that, suppose some point such that . So we have and

,

which means that is perpendicular to . Therefore,

the equality holds in the last inequality only when . We conclude that the pseudo-inverse produces a point that has the least -norm.

**3-3-2 Iteratively Reweighted Least Squares (IRLS)**

When , one can apply the IRLS method as follows to find the optimal solution for the problem in (71). This method was proposed by Rao and Kreutz-Delgado in [7]. What we mention here is referred to the summary of Lyu *et al.* [5]. IRLS is essentially composed of a series of the weighted optimization problem as

, subject to , (103)

where our weights in Step are

(104)

The sequence is composed of positive number to avoid division by zeros (since we have and ). It is obvious that the objective function converges to in (100) when Eq. (103) was proposed by Mourad and Reilly in [8] to replace the objective function with a convex function (Eq. (103) is a quadratic function and hence a convex function).

Let be a matrix to have the weights on its diagonal, i.e., then the problem in (103) at the th iteration can be formulated to

subject to . (105)

Let , then we have

subject to (106)

which has exactly the same form as we have in (101) , thus the optimal solution is

Since we know that is a diagonal matrix, , then

Therefore, the optimal point for th iteration is

(107)

**{Algorithm 3.3}: Iteratively Reweighted Least Squares Algorithm**

(**Initialization**): Initialize such that for example, the minimization solution in subsection 3-3-1. Set a sequence such that For example, we can set

**repeat**

Compute for . Then compute

If the stopping criterion is satisfied, then **quit**.

We may use for some small number as the stopping criterion. Or we may try the stopping criterion mentioned in Section 3-1. One can also refer to the implementation in “optimization/code/lp\_norm/IRLS” in the files we offer.

**3-3-3 Smoothed Method** **for**

In this section, we introduce a method similar to IRLS in subsection 3-3-2, but with some smoothed function (like Gaussian model) to approximate minimization. First, plug into the problem in (100), we have

subject to , (108)

where is equal to the number of nonzero terms in the vector . Then, we solve (108) by the **smoothed (SL0) method.**

The **smoothed (SL0) method** was proposed by Mohimani *et al*. [9]. We use the Gaussian model to approximate . First, we define

, (109)

then we have

(110)

where If we set

(111)

we have

(112)

Thus, we can take as an approximation of :

. (113)

Then, we conclude that minimizing is equal to maximizing . We know that the gradient of is

(114)

The value of specifies a trade-off between accuracy and smoothness of the approximation. From (110), we know that the smaller the is, the better approximation we get. However, by contrast, to make the model smooth, should be large. The concept is for small values of , there are a lot of local maxima in But as the value of grows, we have smoother and smoother.

Thus, in this algorithm, we first have relatively large value of and decrease the value gradually. For each we use a steepest ascent method algorithm for maximizing (similar to the descent method in Sections 2-1 and 2-2).

**Algorithm 3.4 (Smoothed Method)**

(**Initialization**): Initialize such that for example, the minimization solution in subsection 3-3-1. Choose a suitable decreasing sequence .

**repeat**

(Update.) .

Maximize subject to .

(Initialization):

**repeat**

Compute ascent step

(Update.) , where step size is small enough.

(Projection.) (from (102)).

If is small enough, then **quit,**

where

(Update.) .

If reach stopping criterion, then **quit**.

One can also refer to the implementation in “optimization/code/l0\_norm/SL0MatlabCodeV2” in the files we offer.

**3-3-4 Big-M Simplex Method for *p* = 1**

The big-*M* simplex method is suitable for the case where , which means that it is a basis pursuit problem:

subject to (115)

First, we reformulate this to a standard [10] linear program (LP), which is suitable for the simplex method.

subject to (116)

where denotes the positive part of while is the negative part of .

The process of the Big-M simplex method is summarized as follows. It is suitable for any linear program:

* + - 1. Modify the constraints so that RHS (right hand side) of each constraint is nonnegative.
      2. Make each constraint be the standard form (for each constraint with , add a slack variable ; for each constraint with , subtract an excess variable .
      3. For each constraint with or , add an artificial variable .
      4. For a minimization problem, add to the objective function where is a significantly large number. Similarly, add for the maximization problem,
      5. Write the problem as Tableau. Solve the transformed problem by the simplex method.
      6. Within each iteration of simplex method, exactly one variable goes from nonbasic (or free) to basic, where a variable is called basic if it corresponds to a pivot column. The variable that goes from nonbasic to basic is called the entering variable. It is chosen to meet the goal (say, to minimize or maximize the objective function) and it is also the best one among all the meetable variables. The variable that goes from basic to nonbasic is called the leaving variable. We choose this to preserve nonnegative of the current basic variables. See the example as in Example 3.1.

**[Example 3-1] Using the Simplex Method to Solve a Linear Programming Problem** [10]

Consider the linear programming problem as follows:

minimize

subject to

(117)

Then, we reformulation, add slack and excess variables, and preserve the nonnegative of all variables.

Row 0: ,

Row 1: ,

Row 2: ,

Row 3: . (118)

Moreover, *x*1, *x*2, *s*1, *e*2 ≥ 0 should be satisfied. Then, artificial variables *a*2 and *a*3 are created to the constraint equations originally with or (i.e., Row 2 and Row 3).

Row 0: ,

Row 1: ,

Row 2: ,

Row 3: . (119)

where *x*1, *x*2, *s*1, *e*2 ≥ 0. In the optimal solution, all artificial variables must be set equal to zero in the end. In a minimization LP problem, a term is added to the objective function (i.e., Row 0) for each artificial variable .

Row 0: ,

Row 1: ,

Row 2: ,

Row 3: . (120)

Then, we construct the initial tableau as follows:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Row |  |  |  |  |  |  |  | rhs |
| 0 | 1 | -2 | -3 |  |  | -M | -M | 0 |
| 1 |  | 0.5 | 0.25 | 1 |  |  |  | 4 |
| 2 |  | 1 | 3 |  | -1 | 1 |  | 20 |
| 3 |  | 1 | 1 |  |  |  | 1 | 10 |

where rhs means the right-handed side. Then, we eliminate the coefficients of artificial variables *a*2 and *a*3 in Row 0. Then, choose the variable with largest coefficient(s) as basic variable (pivot) (in this case, the pivot is *x*2). Then, we determine Ratio = RHS / pivot column and obtain:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 | 2M-2 | 4M-3 |  | -M |  |  | 30M |  | Row0M(Row2)M(Row3) |
| 1 |  | 0.5 | 0.25 | 1 |  |  |  | 4 | 16 |  |
| 2 |  | 1 | 3 |  | -1 | 1 |  | 20 | 6.67 |  |
| 3 |  | 1 | 1 |  |  |  | 1 | 10 | 10 |  |

Then, choose the row with the least ratio (i.e., Row 2) and normalize its coefficient of the pivot to 1:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 | 2M-2 | 4M-3 |  | -M |  |  | 30M |  |  |
| 1 |  | 0.5 | 0.25 | 1 |  |  |  | 4 |  |  |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 |  | Row2 divided by 3 |
| 3 |  | 1 | 1 |  |  |  | 1 | 10 |  |  |

Then, we eliminate of the coefficients of the pivot (i.e., *x*2) in other rows.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 | (2M-3)/3 |  |  | (M-3)/3 | (3-4M)/3 |  | (60+10M)/3 |  | Row0-(4M-3)(Row2) |
| 1 |  | 0.5 | 0.25 | 1 |  |  |  | 4 |  |  |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 |  |  |
| 3 |  | 1 | 1 |  |  |  | 1 | 10 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 | (2M-3)/3 |  |  | (M-3)/3 | (3-4M)/3 |  | (60+10M)/3 |  |  |
| 1 |  | 5/12 |  | 1 | 1/12 | -1/12 |  | 7/3 |  | Row1-0.25(Row2) |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 |  |  |
| 3 |  | 1 | 1 |  |  |  | 1 | 10 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 | (2M-3)/3 |  |  | (M-3)/3 | (3-4M)/3 |  | (60+10M)/3 |  |  |
| 1 |  | 5/12 |  | 1 | 1/12 | -1/12 |  | 7/3 |  |  |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 |  |  |
| 3 |  | 2/3 |  |  | 1/3 | -1/3 | 1 | 10/3 |  | Row3-Row2 |

Again, we choose the pivot. Since *x*2 has been chosen as the pivot in the previous iteration, we choose *x*1 as the pivot in this iteration. The ratio is recalculated according to Ratio = RHS / pivot column.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 | (2M-3)/3 |  |  | (M-3)/3 | (3-4M)/3 |  | (60+10M)/3 |  |  |
| 1 |  | 5/12 |  | 1 | 1/12 | -1/12 |  | 7/3 | 28/5 |  |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 | 20 |  |
| 3 |  | 2/3 |  |  | 1/3 | -1/3 | 1 | 10/3 | 5 |  |

Then, the row with the least ratio is Row 3. We normalize its coefficient of the pivot to 1 as follows:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 | (2M-3)/3 |  |  | (M-3)/3 | (3-4M)/3 |  | (60+10M)/3 |  |  |
| 1 |  | 5/12 |  | 1 | 1/12 | -1/12 |  | 7/3 |  |  |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 |  |  |
| 3 |  | 1 |  |  | 1/2 | -1/2 | 3/2 | 5 |  | Row3 divided by 2/3 |

Then, we eliminate of the coefficients of the pivot (i.e., *x*1) in other rows.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 |  |  |  | -1/2 | (1-2M)/2 | (3-2M)/2 | 25 |  | Row0-(2M-3)(Row3)/3 |
| 1 |  | 5/12 |  | 1 | 1/12 | -1/12 |  | 7/3 |  |  |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 |  |  |
| 3 |  | 1 |  |  | 1/2 | -1/2 | 3/2 | 5 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 |  |  |  | -1/2 | (1-2M)/2 | (3-2M)/2 | 25 |  |  |
| 1 |  |  |  | 1 | -1/8 | 1/8 | -5/8 | 1/4 |  | Row1-(5/12)(Row3) |
| 2 |  | 1/3 | 1 |  | -1/3 | 1/3 |  | 20/3 |  |  |
| 3 |  | 1 |  |  | 1/2 | -1/2 | 3/2 | 5 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | RHS | Ratio | operation |
| 0 | 1 |  |  |  | -1/2 | (1-2M)/2 | (3-2M)/2 | 25 |  |  |
| 1 |  |  |  | 1 | -1/8 | 1/8 | -5/8 | 1/4 |  |  |
| 2 |  |  | 1 |  | -1/2 | 1/2 | -0.5 | 5 |  | Row2-(1/3)(Row3) |
| 3 |  | 1 |  |  | 1/2 | -1/2 | 3/2 | 5 |  |  |

Then, according to Rows 0, 2, 3 of the last tableau, after setting *a*2 = *a*3 = *e*2 = 0, we get the optimal value 25, with the optimal point , .

Back to the problem in (116), we can set and then rewrite it as

min

subject to , (121)

then do the process like we do in Example 3.1. The implementation can be acquired from “optimization/code/l1norm\_Big\_M” in the files we offer.

**Efficiency**

It can be shown that the complexity of the worst case is pretty high, but this worst case rarely happens. Further proof can be found in [11].

1. **Maximum Likelihood Estimator (MLE)**

In maximum likelihood estimation, our goal is to maximize the probability of observing the data from the joint probability distribution given in a specific probability distribution and parameters.

**[Definition 4.1] Likelihood Function**

A probability distribution of a random sample with parameter is denoted as

(122)

If the sample is comprised of n samples , we have

(123)

This resulted conditional probability distribution is referred to the likelihood function

(124)

or

(125)

for (123).

If we take all data as independent and identically distributed (i.i.d.), then we can further write (123) as

(126)

**[Definition 4.2] Maximum Likelihood Estimator (MLE)**

We say that is a maximum likelihood estimator of if

, (127)

where . The logarithm of the likelihood function

, , (128)

is often more convenient to use. Since the logarithm is a strictly increasing function, the point maximizing is the same as the point maximizing . If , as in (126), then we have

(129)

If the probability density function is differentiable, then the maximum likelihood estimator can be solved from

. (130)

**Relationship to Machine Learning**

We can frame the problem of fitting a machine learning model as the problem of probability density estimation. More specifically, choosing the model and associated parameters is referred to as a modeling hypothesis , and the problem involves finding that best explains the data

Therefore, after finding that can maximize the likelihood function

(131)

we solve an optimization problem to fit the machine learning model.

**[Example 4.1]** [12]

Let be one or zero if, respectively, the outcome of a Bernoulli experiment is success or failure. Let , , denotes the probability of success. Then the probability mass function (pmf) of is

If the Bernoulli experiment is performed n times and is a sample on , then the likelihood function is

(132)

Then, we take the log of it

The partial derivative of is:

(133)

We obtain

(134)

by setting the partial derivative to . Then we get the MLE is the average of successes in the *n* trials.

1. **Conclusion and Further Reading**

In this tutorial, we introduce the fundamental of optimization. We discuss most of the methods in the case of convex optimization problems. However, with some modifications, like properly choosing the step size in the descent method, we can deal with other nonconvex problems.

In Chapter 1, we only focused on the concepts that is needed in latter chapters, e.g., some specific properties of convex functions. For more details in convex optimization (the structure, concepts, ......), please refer to [13], which is a website with a basic tutorial.

In Chapter 2, we mentioned some basic optimization methods. They are not the most powerful ones, but understanding their concepts can help you get into other complicated algorithms. We just discussed a little about the convergence analysis of the optimization algorithms. Thus, for more detailed discussion, please refer to [1], [3] and [13]. In Chapter 3, we focus on a special optimization problem, the inverse problem. For more comparison between different methods, please refer to [5].

As you can see in Chapter 4, what we do in machine learning (or further, deep learning) is the optimization problem, too. To learn more, one can see the discussion in [13].

# References

|  |  |
| --- | --- |
| [1] | S. Boyd and L. Vandenberghe, Convex Optimization, New York, 2004. |
| [2] | " Gradient descent - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Gradient\_descent. |
| [3] | S. Bubeck, Convex optimization: Algorithms and complexity, 2014. |
| [4] | R. T. Rockafellar, Convex Analysis, Princeton University Press, 1970. |
| [5] | Q. e. a. Lyu, "A comparison of typical lp minimization algorithms," *Neurocomputing,* vol. 119, pp. 413-424, 2013. |
| [6] | D. L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory,* vol. 52, no. 4, pp. 1289-1306, 2006. |
| [7] | B. D. Rao and K. Kreutz-Delgado, "An Affine Scaling Methodology for Best Basis Selection," *IEEE Transactions on Signal Processing,* vol. 47, no. 1, pp. 187-200, 1999. |
| [8] | N. Mourad and J. Reilly, "MInimizing nonconvex functions for sparse vector reconstruction," *IEEE Transactions on Signal Processing,* vol. 58, no. 7, pp. 3485-3496, 2010. |
| [9] | G. H. Mohimani, M. Barbaie-Zadeh and C. Jutten, "Fast Sparse Representation Based on Smoothed L0 Norm," in *International Conference on Independet Component Analysis and Signal Separation*, Berlin, 2007. |
| [10] | "The Big M Method," [Online]. Available: http://www.columbia.edu/~cs2035/courses/ ieor3608.F05/david-bigM.pdf. |
| [11] | R. J. Vanderbei, Linear Programming: Foundations and Extensions 2E, 2001. |
| [12] | R. V. Hogg, J. W. McKean and A. T. Craig, Introduction to Mathematical Statistics 7E, Pearson, 2013. |
| [13] | J. Brownlee, "A Gentle Introduction to Maximum Likelihood Estimation for Machine Learning," 23 October 2019. [Online]. Available: https://machinelearningmastery.com/what-is-maximum-likelihood-estimation-in-machine-learning/. [Accessed 23 December 2021]. |
| [14] | Y. Nesterov, Introductory Lectures on Convex Optimization, Kluwer Academic Publishers, 2004. |
| [15] | N. V. Alger, 2021. [Online]. Available: https://math.stackexchange.com/questions/2285282/ relating-condition-number-of-hessian-to-the-rate-of-convergence. |
| [16] | M. A. T. Figueiredo, R. D. Nowak and S. J. Wright, "Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems," *IEEE Journal of Selected Topics in Signal Processing,* vol. 1, no. 4, pp. 586-597, 2007. |
| [17] | B. Su, "Convex Optimization Lecture notes," 2021. |
| [18] | J. Nocedal and S. J. Wright, Numerical Optimization, Springer Science+Business Media, 1999. |
| [19] | H. Mohimani, M. Babaie-Zadeh and I. Gorodnitsky, "Sparse Recovery using Smoothed l0 (SL0): Convergence Analysis," 2010. |