Advanced Topics in Operations Research Notes

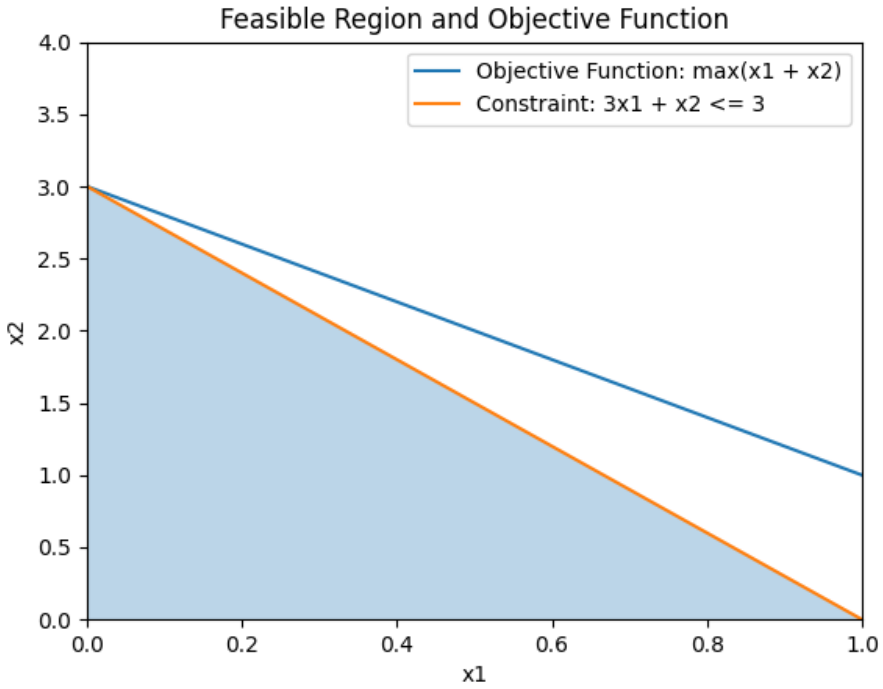
Andres Espinosa

November 4, 2024

Contents

1 Wednesday 08/28/2024

$$max\ x_1 + x_2 \tag{1}$$
$$s.t.\ 3x_1 + x_2 \leq 3 \tag{2}$$
$$x_1 \geq 0, x_2 \geq 0 \tag{3}$$



x^* is an extreme point if we cannot find $x_1, x_2 \in S$, such that $x_1 \neq x_2$ and $x^* = \lambda x_1 + (1-\lambda)x_2$ for some $\lambda \in (0, 1)$
For an n-dimensional problem, there are n-many constraints that are active/binding and their coefficients are linearly independent.
Optimal solutions can be found that are not basic feasible solutions when constraints and objective functions lie on the same plane.

2 Friday 08/30/2024

2.1 Complexity Introduction

Three elements for complexity analysis:

- Model or problem formulation - The known part of a problem. Includes the formulation and problem descriptions.
- Oracle - The smallest computing unit, whose details inside can be ignored.
- Target outcome - When to stop

Two types of complexity:

- Analytical complexity - The number of calls of the oracle which is necessary to solve a given problem formulation up to accuracy ϵ
- Numerical complexity - The number of arithmetic operations which is necessary to solve a given problem formulation up to accracy ϵ

2.2 P, NP, NP-hard

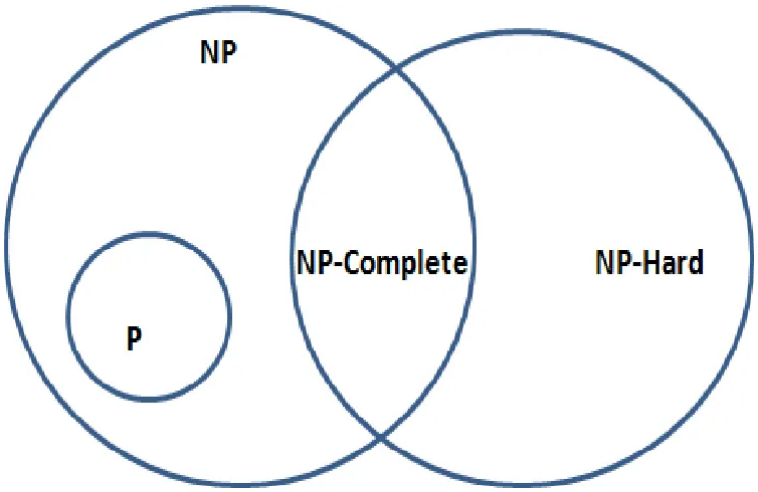
A **problem** is a function $F : I \rightarrow B$, where I is the set of instances encoded as strings of characters and B is the set of problem outputs.

For example, linear programming in $\mathbb{R}^{mxn}$ is a problem where the instance parameters are (m, n, A, c, b) and the output B is the optimal solution found to the objective function.
Another example, making a decision takes in an instance I and outputs a decision of yes or no.

$$I = \textit{Problem-specific information parameters}, B \in (yes, no) \tag{4}$$

3 Wednesday 09/04/2024

3.1 P, NP, NP-complete, and NP-hard



A **problem** is a function $F : I \to B$, where I is the set of instances encoded as strings of characters and B is the set of problem outputs.

A mapping from I to B by taking in the input parameters of what the specific problem actually entails, and then the solution or answer is parametrized by B.

For example, linear programming in $\mathbb{R}^{mxn}$ is a problem where the instance parameters are (m, n, A, c, b) and the output B is the optimal solution found to the objective function.

Another example, making a decision takes in an instance I and outputs a decision of yes or no.

$$I = \textit{Problem-specific information parameters}, B \in (yes, no) \tag{5}$$

3.1.1 Turing machine

A turing machine is a model of how a computer processes a program. There is a *tape* and *finite-state controller*

- **Tape**: A memory unit divided into tape cells. Each tape cell stores one symbol chosen from a finite set of symbols. The tape head is the sole cell currently processed by the controller.
- **Finite-state controller**: A collection of finite states and rules where at each time step, the controller takes an action of erase, move, update cell, and update state.

There is only one unique rule applied for each symbol-state combination

A turing machine is said to be able to solve a problem $F : I \to B$, if given any instance $x \in I$, the turing machine eventually produces the solution B and stops.

My understanding of this is that a turing machine has a controller that goes and selects individual cells based off of the symbol and then makes an action to change the state based off of a set of symbol-state pairs.

3.1.2 P-problem

The class P of problems is defined to be those problems $F : I \to yes, no$ such that a Turing machine M can compute F, and the number of steps required by M for this computation is bounded by $p(n)$ where p is a polynomial and n is the length of the input.

In other words, a problem P can be solved in polynomial time by a turing machine.

F is a decision problem

NP problem: Problems whose solution, if it is "yes", have proofs verifiable in polynomial time by a deterministic Turing machine.

P problems can be solved in polynomial time, NP problems are problems that can be verified in polynomial time. A non-deterministic Turing machine would be able to solve an NP problem in polynomial time. *A non-deterministic Turing machine is defined as the relaxation of the "unique rule per symbol-state" constraint.*
For example, in the exercise done in class, instead of having the unique rule of "if no one is to the right, say 'yes'", you could have a non-deterministic rule where you have a 50% chance of saying 'no' or 'yes'. Putting randomness in the rule would make it a non-deterministic machine.

3.1.3 NP-complete

A decision problem $F : I \to yes, no$ can be polynomial transformed into another decision problem $G : I' \to yes, no$ if there exists a function $f : I \to I'$. The function f can be computed by a polynomial-time Turing machine, and f has the property that $F(x) = yes \iff G(f(x)) = yes$
Suppose $F : I \to yes, no$ is a decision problem that lies in NP, and suppose that every problem in NP can be polynomially transformed to F . Then, F is said to be NP-complete

An algorithm can solve all NP problem efficiently if it can solve an NP-complete problem efficiently. If it solves an NP-complete problem, that means it is able to transform a problem into an NP problem.

4 Friday 09/06/2024

I did not attend this day so just notes over the slides.

4.1 NP-hard

4.1.1 Definitions

We say that a Turing machine can use F as an **oracle** if M is augmented with the following capability: M can compute $F(x)$ from x in one step.
We say that a problem G is **Cook-reducible** to problem F (aka, polynomial-time Turing reducible) if G can be computed in polynomial time by a Turing machine that uses F as an oracle.
A problem F is said to be **NP-hard** if an NP-complete problem is Cook-reducible to F.

4.1.2 Quadratic problem

General quadratic programming:

- Given symmetric matrix $H \in \mathbb{R}^{nxn}$, matrix $A \in \mathbb{R}^{mxn}$, and vectors **b,c** $\in \mathbb{R}^n$
- Solve

$$\begin{aligned} \min \mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ s.t. A\mathbf{x} \geq \mathbf{b} \end{aligned} \tag{GQP}$$

5 Monday 09/09/2024

I also did not attend today so these are slide notes

5.1 Nonlinear Programming Basic Concepts

5.1.1 Nonlinear problem formulation

$$\begin{aligned} &\min_x f(\mathbf{x}) \\ \text{s.t. } &h(\mathbf{x}) = 0 \\ &g(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{6}$$

where $f : \mathbb{R}^n \to \mathbb{R}, h : \mathbb{R}^n \to \mathbb{R}^{m_1},$ and $g : \mathbb{R}^n \to \mathbb{R}^{m_2}$

- Objective function, cost function, disutility function
- Inequality constraints, non-negativity constraints
- Equality constraints
- Feasible region, feasible solution
- Optimal solution, minimal solution
- Unconstrained problems

5.1.2 Global and local solutions definitions

Let $\Omega := \mathbf{x} : g(\mathbf{x} \leq \mathbf{0}, h(\mathbf{x} = \mathbf{0}))$

A feasible solution $\mathbf{x}^* \in \Omega$ is said to be globally minimal if and only if $f(\mathbf{x}^*) \leq f(\mathbf{x}) \forall \mathbf{x} \in \Omega$

A feasible solution $\mathbf{x}^* \in \Omega$ is said to be locally minimal if and only if $f(\mathbf{x}^*) \leq f(\mathbf{x}) \forall \mathbf{x} \in \Omega \cap N(\mathbf{x}^*, \epsilon)$ for some $\epsilon > 0$. Here, $N(\mathbf{x}^*, \epsilon) = \{\mathbf{x} : ||\mathbf{x}^* - \mathbf{x} \leq \epsilon||\}$

In other words, a global solution is global if it is the minimum of all values in the set. It is locally minimal if it is less than all points in a certain radius defined arbitarily by ϵ.

5.2 Necessary conditions of optimality

5.2.1 Gradients

Consider a function $f\mathbb{R}^n \to \mathbb{R}$, then the gradient of f is given as

$$\nabla f(\mathbf{x}) := \begin{bmatrix} \frac{df(\mathbf{x})}{dx_1} \\ \frac{df(\mathbf{x})}{dx_2} \\ \vdots \\ \frac{df(\mathbf{x})}{dx_n} \end{bmatrix} \tag{7}$$

Finding gradient of function $f(x_1, x_2) = 0.5x_1^2 + 0.5x_2^2$ at (1,-1)

$$\frac{df(\mathbf{x})}{dx_1} = x_1 \tag{8}$$

$$\frac{df(\mathbf{x})}{dx_2} = x_2 \tag{9}$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{10}$$

5.2.2 Examples

a.

$$\nabla 2x_1^2 + 2x_2^2 = \begin{bmatrix} 4x_1 \\ 4x_2 \end{bmatrix} \tag{11}$$

Solving for $\nabla f = 0$,

$$\mathbf{x}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{12}$$

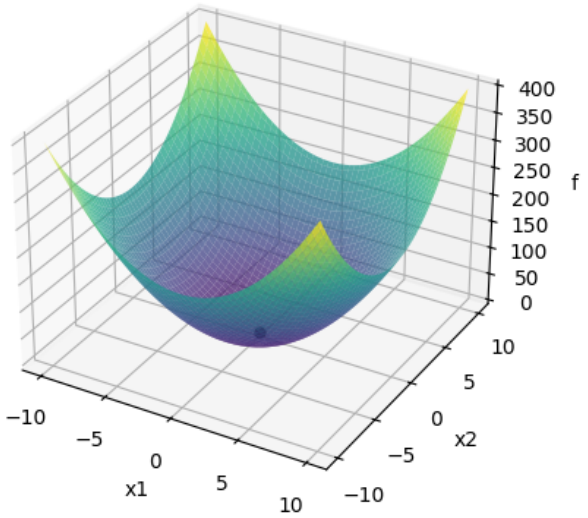


Figure 1: Graph of $2x_1^2 + 2x_1^2$ with solution at $(x_1, x_2) = (0, 0)$

b.

$$\frac{d\sin(x)}{dx} = \cos(x) \tag{13}$$

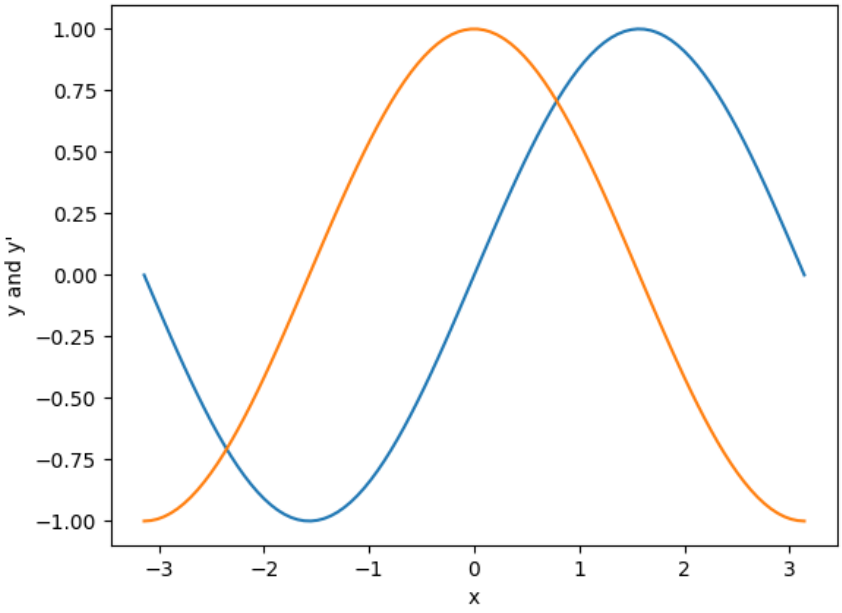


Figure 2: Graph of sin(x) and the solution cos(x)

Solution to this problem is the curve $\cos(x)$
c.

$$\frac{dx^3}{dx} = 3x^2 \tag{14}$$

Solution to this problem is the curve $3x^2$
d. $(x-3)(x^2-1)(x+2)$ is already in it's factored form so with product rule:

$$\nabla f = 4x^3 - 3x^2 - 14x + 1 \tag{15}$$

- x = -1.5
- x = 2
- x = 0

6 Wednesday 09/11/2024

6.1 Nonlinear programming Part II

6.1.1 Problem formulation

$$\begin{aligned} &\min_x f(x) \\ &s.t. g(x) \leq 0 \end{aligned} \tag{16}$$

Now there is a constraint on the problem

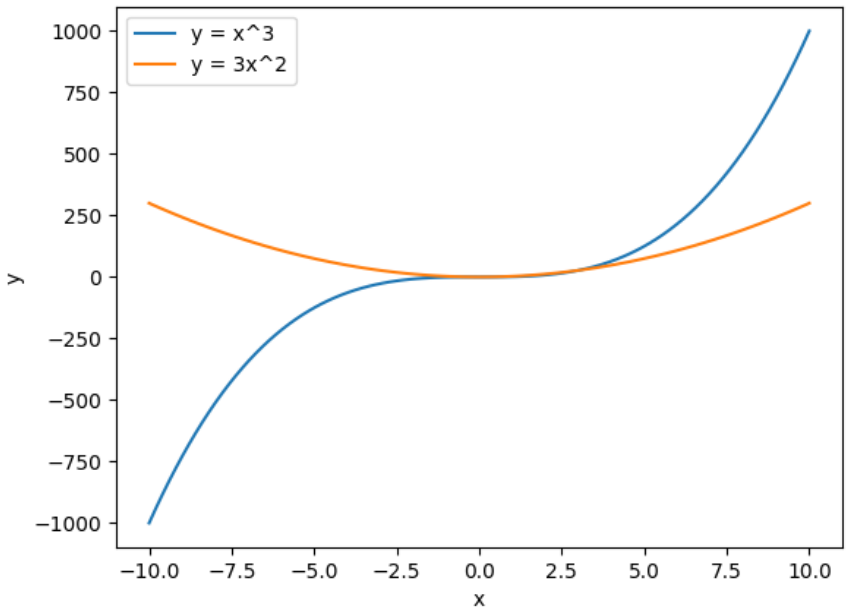


Figure 3: Graph of x^3 and it's solution $3x^2$

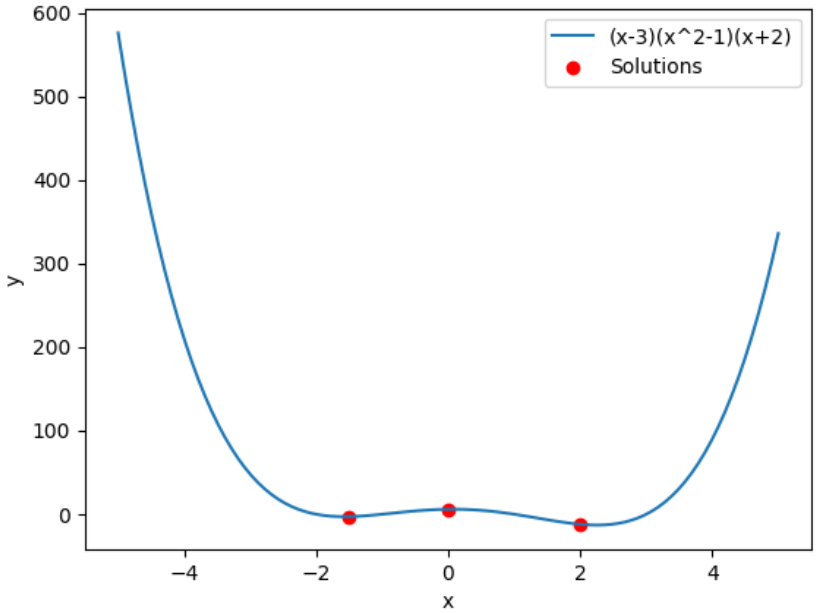


Figure 4: Graph of $(x-3)(x^2-1)(x+2)$ and it's solutions $x = 0, -1.5, 2$

6.1.2 Gradient descent intro

Dropping a coin into a bowl will cause the coin to travel in the direction of the negative gradient until it reaches $\nabla f = 0$

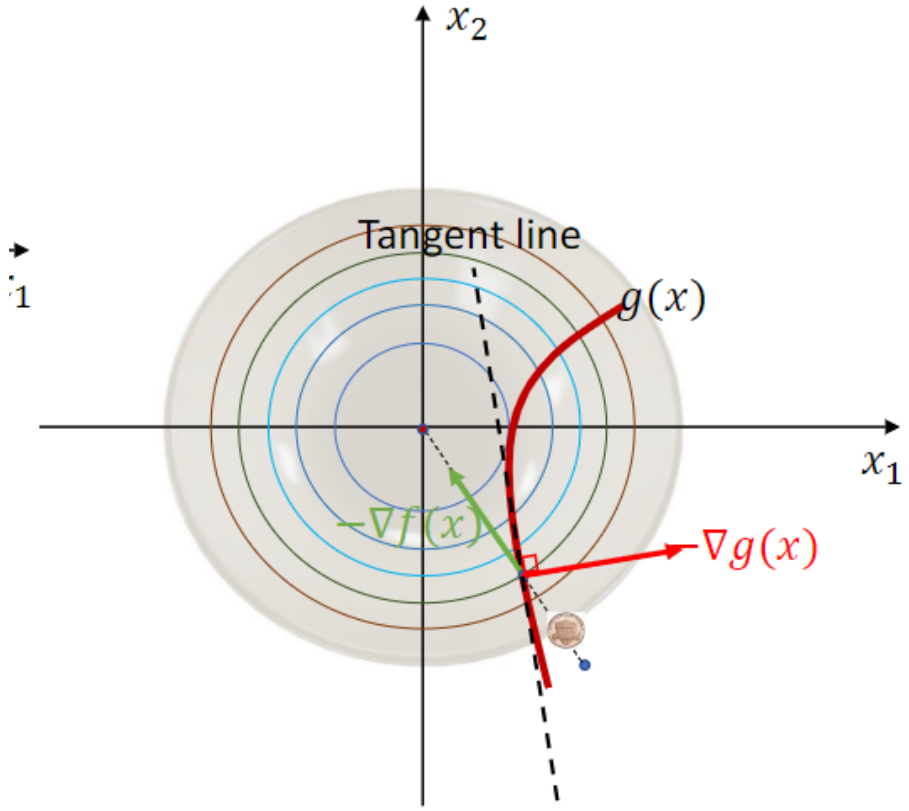


Figure 5: A coin traveling in the direction of $-\nabla$ to reach the bottom of the bowl

$$f(x + \delta d) - f(x) = \langle \nabla f(x), \delta d \rangle + 0(\delta^2) = \langle \nabla f(x), \delta d \rangle \tag{17}$$

In the event that $\langle \nabla f(x), \delta d \rangle < 0$, The step is in a direction that is away from the gradient. It also implies that the gradient is not zero at this point.

We can also introduce a constraint function $g(x)$ present in Figure 5 that defines the section (or in the figure's case, a line) that the function f is feasible on. With the introduction of this constraint, the coin can only travel along the path of the constraint g. For the solution to be optimal in the context of the function f subject to the constraint g, we would need to reach the point where $\nabla f(x^*) + \gamma \nabla g(x^*) = 0$, where $\gamma = \frac{\|\nabla f\|}{\|\nabla g\|}$

7 Friday 09/13/2024

7.1 Review

7.1.1 What is an Inner Product

An inner product defined as:

$$\langle u, v \rangle \tag{18}$$

is a generalization of the dot product, it effectively gives the answer to the question: How much of one vector is in the direction of the other vector?

7.1.2 Understanding $g(x^*) = 0$

If $g(x^*) \leq 0$, then the constraint is being obeyed. The constraint is not obeyed otherwise.

Taking the equation,

$$g(x^* + \delta d) = \delta \langle \nabla g(x^*), d \rangle - \epsilon \tag{19}$$

The step size δcan be moved, and as long as that term is less than zero and does not breach the feasible region.

This implies that for a solution to be optimal, $g(x^*) = 0$ must be true, (the converse does not hold).

7.2 Gradient bowl

7.2.1 Adding another constraint

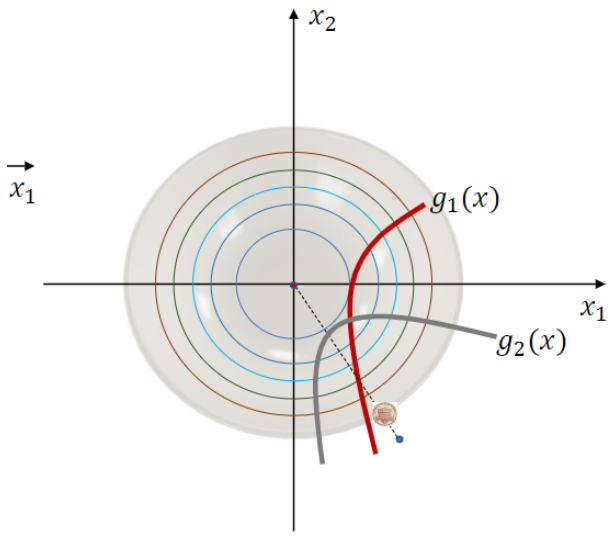


Figure 6: Image of the bowl with a second constraint

We want to walk opposite of the direction of ∇f while still walking perpendicular to the gradient

of the constraint.

$$\begin{aligned} &\min f(x) \\ s.t. &g_1(x) \leq 0 \\ &g_2(x) \leq 0 \end{aligned} \tag{20}$$

A cone is a collection of rays that start from a certain point.

In order to find the set of directions we can walk along that will be both improving the objective function while staying in the feasible region, we need to find the intersection of the halfspaces created by each.

- The feasible region based off of g_1 is the halfspace created by the set $\{x|\nabla g_1(x) \leq 0\}$.
- The feasible region based off of g_2 is the halfspace created by the set $\{x|\nabla g_2(x) \leq 0\}$.
- The "improving" region based off of f is the halfspace created by the set $\{x|\nabla f(x) \leq 0\}$.

The intersection of these sets is the cone of all possible directions that would result in an improvement to the function while obeying the constraints.

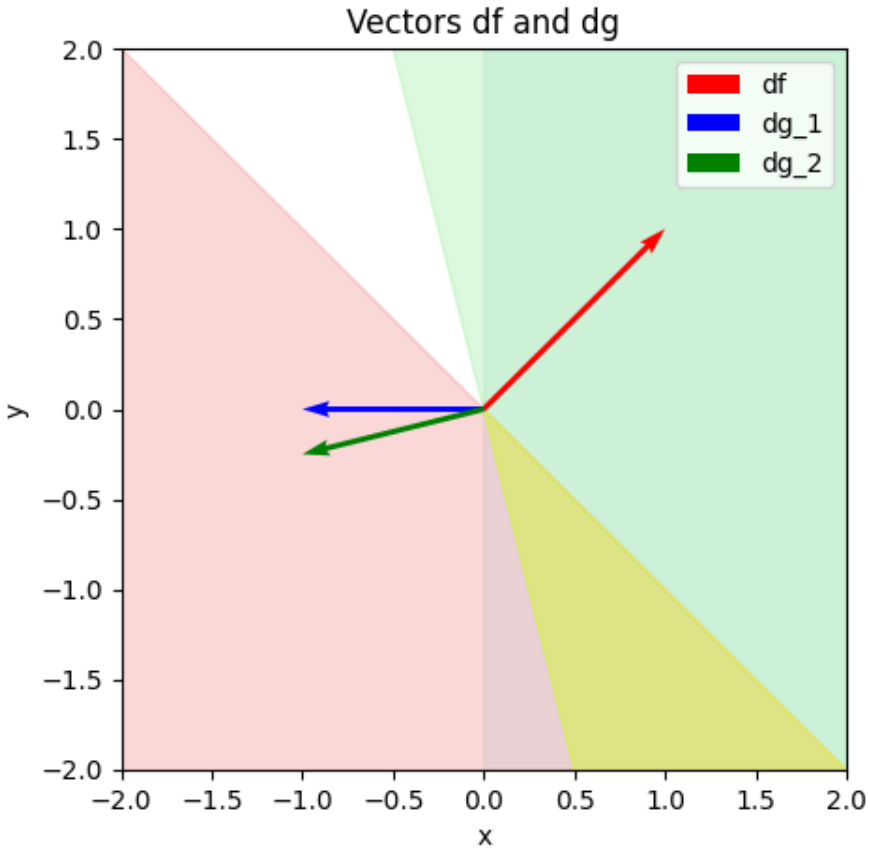


Figure 7: Depicting the walkable region as the intersection between the three regions

7.2.2 Finding optimality conditions

For two constraints g_1 and g_2, the solution x^* is optimal if

$$\nabla f(x^*) + \gamma_1 \nabla g_1(x^*) + \gamma_2 \nabla g_2(x^*) = 0 \tag{21}$$

8 Monday 09/16/2024

8.1 Review

8.1.1 Problem formulation for multiple constraints

$$\begin{aligned} \min f(x) \\ s.t. g_1(x) \leq 0 \\ g_2(x) \leq 0 \end{aligned} \tag{22}$$

The difference between a function and a tiny step of size δ in direction d can be approximated as the inner product between the gradient of the function and the step δd

$$\begin{aligned} f(x+\delta d) - f(x) &\approx \langle \nabla f(x), \delta d \rangle \\ g_1(x+\delta d) - g_1(x) &\approx \langle \nabla g_1(x), \delta d \rangle \\ g_2(x+\delta d) - g_2(x) &\approx \langle \nabla g_2(x), \delta d \rangle \\ g_1(x) = 0, g_2(x) &= 0 \end{aligned} \tag{23}$$

In order for the constraints set in the general minimization problem to be met, the inner product between the gradient of the constraint and the step should be non-positive. We can define the "walkable" region as the intersection of the halfspaces created by the negative gradients of the objective function and any constraints. For a solution to be optimal, this intersection must be empty.

8.1.2 Optimality

A minimal solution is reached when

$$\nabla f(x^*) + \gamma_1 \nabla g_1(x^*) + \gamma_2 \nabla g_2(x^*) = 0 \tag{24}$$

where $\gamma_1 \geq 0, \gamma_2 \geq 0$

This is a non-negative linear combination of the gradient vectors. The set of all non-negative linear combinations of two vectors u and v is a cone with edges on u and v. This cone can be parametrized by a single scalar w using the following

$$\begin{aligned} w(r_1 u + r_2 v) &= y \\ r_1 + r_2 &= 1 \\ r_1 \geq 0, r_2 \geq 0, w &\geq 0 \end{aligned} \tag{25}$$

Using that logic to explain the equation $\nabla f(x^*) + \gamma_1 \nabla g_1(x^*) + \gamma_2 \nabla g_2(x^*) = 0$, there is a vector ∇f that is opposite of any linear combination of the constraints.

8.2 Active constraints

In an optimization problem such as the bowl, it is possible that there is only one active constraint, to the problem. Therefore, although g_1 and g_2 are defined, only one is relevant to the optimization problem.

$$\nabla f(x^*) + \gamma_1 \nabla g_1(x^*) + \gamma_2 \nabla g_2(x^*) = 0 \tag{26}$$

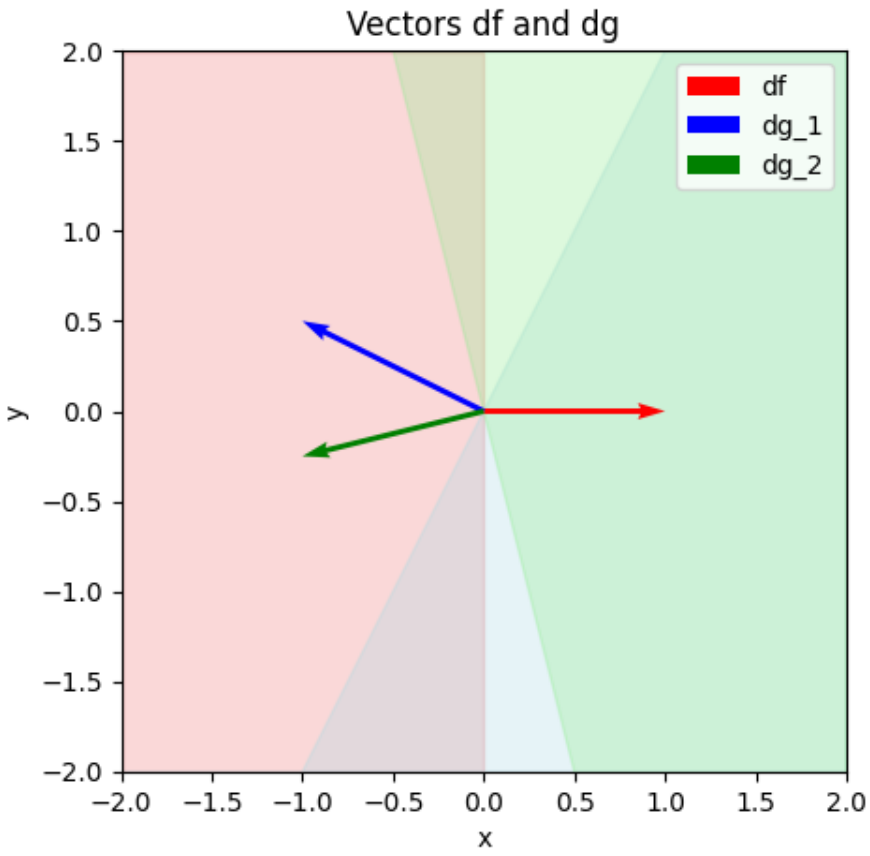


Figure 8: Image of an empty walkable region since ∇f is opposite of ∇g

where $\gamma_1 \geq 0, \gamma_2 \geq 0$ when there is only one active constraint, is equivalent to

$$\nabla f(x^*) + \gamma_1 \nabla g_1(x^*) = 0 \tag{27}$$

where $\gamma_1 \geq 0$

Can we alter the first rule to incorporate this new information of potentially inactive constraints? Adding binary variables.

9 Wednesday 09/18/2024

9.1 Multiple Constraints

9.1.1 Inactive constraints

Could have the case where two constraints are defined but only one is active optimal solution

$$\begin{aligned} &\min f(x) \\ s.t. &g_1(x) \leq 0 \\ &g_2(x) \leq 0 \end{aligned} \tag{28}$$

$$\text{Inactive Case: } \begin{aligned} \nabla f(x^*) + \gamma_1 \nabla g_1(x^*) &= 0 \\ g_1(x^*) = 0, \gamma_1 &\geq 0 \end{aligned} \tag{29}$$

$$\text{Active Case: } \begin{aligned} \nabla f(x^*) + \gamma_1 \nabla g_1(x^*) + \gamma_2 \nabla g_2(x^*) &= 0 \\ g_1(x^*) = 0, g_2(x^*) = 0, \gamma_1 \geq 0, \gamma_2 &\geq 0 \end{aligned} \tag{30}$$

Both cases can be captured when $\nabla g_2 < 0, \gamma_2 = 0$ or $\nabla g_2 = 0, \gamma_2 \geq 0$. Those two scenarios are captured by the equation

$$\gamma_2 g_2(x^*) = 0 \tag{31}$$

9.1.2 Karush-Kahn-Tucker Conditions

$$\begin{aligned} &\min f(x) \\ s.t. &g_1(x) \leq 0 \\ &g_2(x) \leq 0 \end{aligned} \tag{32}$$

is minimized when

$$\text{Final System: } \begin{aligned} \nabla f(x^*) + \gamma_1 \nabla g_1(x^*) + \gamma_2 \nabla g_2(x^*) &= 0 \\ g_1(x^*) \leq 0, g_2(x^*) \leq 0, \gamma_1 \geq 0, \gamma_2 &\geq 0, \\ \gamma_1 g_1(x^*) = 0, \gamma_2 g_2(x^*) &= 0 \end{aligned} \tag{33}$$

9.2 Adding an equality constraint

Can split an equality constraint into two inequalities.

$$\begin{aligned} &\min f(x) \\ &h(x) = 0 \end{aligned} \quad \text{or} \quad \begin{aligned} &\min f(x) \\ &h(x) \leq 0, h(x) \geq 0 \end{aligned} \tag{34}$$

is satisfied when

$$\begin{aligned} \nabla f(x^*) + \gamma_1 \nabla h(x^*) - \gamma_2 \nabla h(x^*) = 0 \\ h(x^*) \leq 0, h(x^*) \geq 0, \gamma_1 \geq 0, \gamma_2 \geq 0, \\ \gamma_1 h(x^*) = 0, \gamma_2 h(x^*) = 0 \end{aligned} \tag{35}$$

Which can be simplified to

$$\begin{aligned} \nabla f(x^*) + \hat{\gamma} \nabla h(x^*) = 0 \\ h(x^*) = 0 \end{aligned} \tag{36}$$

9.3 KKT Conditions General Case

Under a constraint qualification, a local optimal solution satisfies the conditions

$$\begin{aligned} \nabla f(x^*) + [\nabla h(x^*)]^T \Theta + [\nabla g(x^*)]^T \Gamma = 0, \\ [g(x^*)]^T \Gamma = 0, \\ \Gamma \geq 0, \\ h(x^*) = 0, g(x^*) \leq 0 \end{aligned} \tag{37}$$

9.3.1 KKT Solutions example

$$\begin{aligned} \min_{x_1, x_2} x_1^2 + x_2^2 \\ s.t. x_1 + 2x_2 \leq -1 \\ x_1 + x_2 = -3 \\ x_1 \geq 0 \end{aligned} \tag{38}$$

THe KKT solutions

$$\begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \gamma_1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \gamma_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \gamma_3 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0 \tag{39}$$

$$\gamma_1 \geq 0, \gamma_3 \geq 0, x_1 + 2x_2 \leq -1, x_1 + x_2 = -3 \tag{40}$$

10 Monday 09/23/2024

10.1 KKT Condition qualification

Drawing out the problem

$$\begin{aligned} & min - x_1 \\ s.t. & x_2 - (1 - x_1)^3 \leq 0 \\ & x_1 \geq 0, x_2 \geq 0 \end{aligned} \tag{41}$$

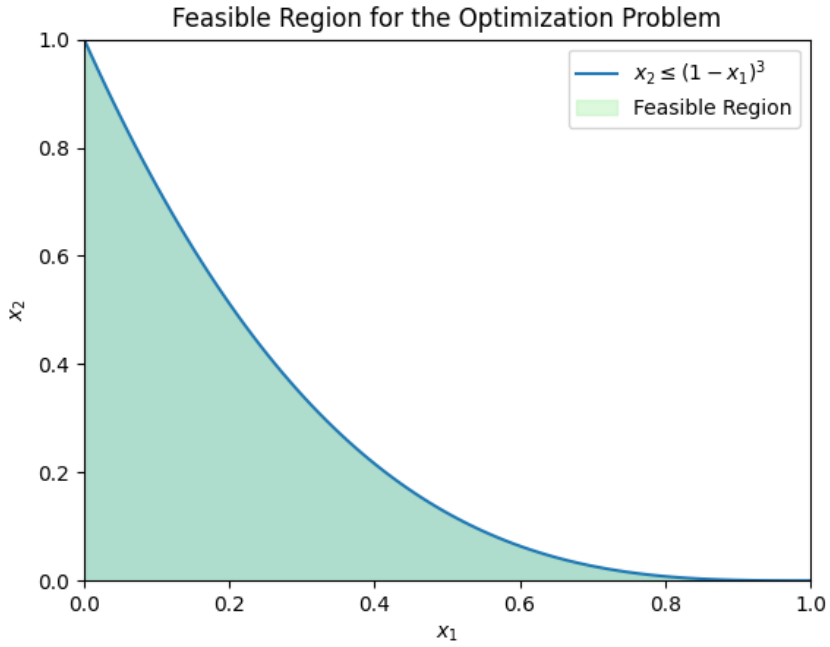


Figure 9: Graph of the feasible region of the problem above

The objective function is minimized at the point (1,0). To be rigorous, we check if this optimization problem satisfies the KKT conditions.

$$\text{Stationary condition: } \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \gamma_1 \begin{bmatrix} -3(1 - x_1)^2 \\ 1 \end{bmatrix} - \gamma_2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \gamma_3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 \tag{42}$$

This stationary condition is not true when $x_1, x_2 = (1, 0)$ because each $\gamma_i \geq 0$. There are no non-negative γ_i that satisfy the stationary condition.

10.2 Adding regularity condition

- Linear CQ: g and h are affine functions
- Linear indendepence CQ: The gradients of the active inequality constraints and the gradients of the equality constraints are linearly independent.

Only one of the above constraint qualifications should hold to allow the KKT conditions to work. For the previous example, since the objective function is cubic, the KKT conditions do not hold. The conditions are also solution specific.

10.3 Fritz John conditions

The Fritz John conditions are a slight modification to the KKT conditions

$$\begin{aligned} \lambda \nabla f(x^*) + [\nabla h(x^*)]^T \Theta + [\nabla g(x^*)]^T \Gamma = 0, \\ [g(x^*)]^T \Gamma = 0, \\ \lambda \geq 0, \Gamma \geq 0, \\ h(x^*) = 0, g(x^*) \leq 0 \end{aligned} \tag{43}$$

There is an introduction of the λ coefficient to the objective function that alleviates the regularity conditions to the KKT conditions.

10.4 NLP: Part III

Convex sets and functions.

A function $h : \mathbb{R}^n \to \mathbb{R}$ is said to be convex if and only if it satisfies

$$\lambda h(x_1) + (1 - \lambda) h(x_2) \geq h(\lambda x_1 + (1 - \lambda) x_2) \tag{44}$$

for any $\lambda \in [0, 1]$ and any $x_1, x_2 \in \textbf{dom } h$

A set $\Omega \subseteq \mathbb{R}^n$ is said to be convex if and only if it satisfies

$$\lambda x_1 + (1 - \lambda) x_2 \in \Omega \tag{45}$$

for any $\lambda \in [0, 1]$ and any $x_1, x_2 \in \Omega$

11 Monday 09/30/2024

11.1 Review

11.1.1 Convex functions and sets

$$\frac{1}{2}x^T Q x + c^T x \tag{46}$$

is the quadratic form and this function is convex as long as Q is positive semidefinite.

A function $h : \mathbb{R}^n \to \mathbb{R}$ is said to be convex if and only if it satisfies

$$\lambda h(x_1) + (1-\lambda)h(x_2) \geq h(\lambda x_1 + (1-\lambda)x_2) \tag{47}$$

for any $\lambda \in [0,1]$ and any $x_1, x_2 \in \textbf{dom } h$

A set $\Omega \subseteq \mathbb{R}^n$ is said to be convex if and only if it satisfies

$$\lambda x_1 + (1-\lambda)x_2 \in \Omega \tag{48}$$

for any $\lambda \in [0,1]$ and any $x_1, x_2 \in \Omega$

11.2 Convexity

11.2.1 Properties of a convex function

For a twice-differential function to be convex, it's Hessian matrix must be positive semidefinite. One definition for positive semidefinite is that all eigenvalues must be non-negative, meaning that the stretch of the matrix is positive or none in the direction of each eigenvector. Below we define what the hessian of a function f looks like.

$$\nabla^2 f(x) = \begin{bmatrix} \frac{d^2 f(x)}{dx_1^2} & \frac{d^2 f(x)}{dx_1 x_2} & \cdots & \frac{d^2 f(x)}{dx_1 x_n} \\ \frac{d^2 f(x)}{dx_2 x_1} & \frac{d^2 f(x)}{dx_2^2} & \cdots & \frac{d^2 f(x)}{dx_2 x_n} \\ & \vdots & & \\ \frac{d^2 f(x)}{dx_n x_1} & \frac{d^2 f(x)}{dx_n x_2} & \cdots & \frac{d^2 f(x)}{dx_n^2} \end{bmatrix} \tag{49}$$

Note: Diagonal matrices' eigenvalues are the values along the diagonal.

11.2.2 Examples

- The 1D function $f(x) = \frac{1}{2}x^2$ is a convex function because its hessian is $\nabla^2 f(x) = 1$. Since this is a positive value and this matrix is positive semidefinite, the function f is convex.
- The 1D function $f(x) = x^3$ has a hessian of $\nabla^2 f(x) = 6x$. This function is convex on $\mathbb{R}_{++}$ and concave otherwise (both at x =0).
- The 1D function $f(x) = \frac{1}{x}$ is convex on the domain $\mathbb{R}_{++}$. The hessian of this function is $\nabla^2 f(x) = \frac{2}{x^3}$, which is positive on $\mathbb{R}_{++}$
- The 1D function $f(x) = -log(x)$ is convex on the domain $\mathbb{R}_{++}$. The hessian of this function is $\nabla^2 f(x) = \frac{1}{x^2}$

11.2.3 Linear regression

Attempting to verify if the least squares equation is convex.

$$\min_x \frac{1}{2n} \sum_{i=1}^{n} (a_i^T x - b_i)^2 \tag{50}$$

$$\nabla f(x) = \frac{1}{2n} \sum_{i=1}^{n} \frac{d}{dx} (a_i^T x - b_i)^2 \tag{51}$$

$$= \frac{1}{2n} \sum_{i=1}^{n} (a_i^T x - b_i) a_i \tag{52}$$

$$\nabla^2 f(x) = \frac{1}{2n} \sum_{i=1}^{n} \frac{d}{dx} a_i^T x a_i - b_i a_i \tag{53}$$

$$= \frac{1}{2n} \sum_{i=1}^{n} a_i^T a_i \tag{54}$$

This is convex since $a_i^T a_i$ is a symmetric positive semi-definite matrix.

12 Wednesday 10/02/2024

12.1 Convex Functions

12.1.1 Verifying convexity

- $\frac{1}{2n} \sum_{i=1}^{n} (a_i^T x - b_i)^2$
- $a^T x + b$
- $-\log(x)$
- $\frac{1}{x}, x > 0$
- $\frac{1}{x}, x \neq 0$
- $\frac{1}{x}, x < 0$
- $g(x) + h(x)$
- $\frac{g(x)}{h(x)}$
- $g(h(x))$
- $\max\{f_1(x), f_2(x)\}$

12.1.2 composition rule for convexity

A logical conclusion for a composition rule $f(x) = g(h(x))$ would be that f is convex if both g and h are convex. However, this does not work because we can use the example $g(x) = -x$ and $h(x) = -\log(x)$. The composite would be $f(x) = \log(x)$ which is concave, not convex. Therefore in order to construct a set of rules for establishing convexity, we can take the second derivate of a composite

$$f(x) = g(h(x)) \tag{55}$$
$$\nabla f(x) = h'(x)g'(h(x)) \tag{56}$$
$$\nabla^2 f(x) = h''(x)g'(h(x)) + h'(x)^2 g''(h(x)) \tag{57}$$

g needs to be convex, and either h is convex and g is non-decreasing or h is concave and g is non-increasing.

12.2 Convex Sets

If a function f is convex, then the set

$$\Omega = \{x | f(x) \leq c\} \tag{58}$$

is convex. The notation above means "The set of all x given that a function $f(x)$ is less than c"

$$\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda)x_2) \tag{59}$$
$$\text{since } f(x_1), f(x_2) \leq c \tag{60}$$
$$\lambda c + (1-\lambda)c \geq f(\lambda x_1 + (1-\lambda)x_2) \tag{61}$$
$$c \geq f(\lambda x_1 + (1-\lambda)x_2) \tag{62}$$

The last line implies that any point on the line f that is between two arbitrary points $x_1, x_2 \in \Omega$ is less than c.
What about the set

$$\hat{\Omega} = \{x | f(x) = c\} \tag{63}$$

This set is not convex. Equality constraints are convex if f is a linear function.

13 Friday 10/04/2024

13.1 Preserving Convexity

13.1.1 Non-negative weighted sum

if $f_1(x), f_2(x)$ are convex functions. Then their non-negative weighted sum

$$g(x) = \alpha f_1(x) + \beta f_2(x) \tag{64}$$

is convex

13.1.2 affine composition

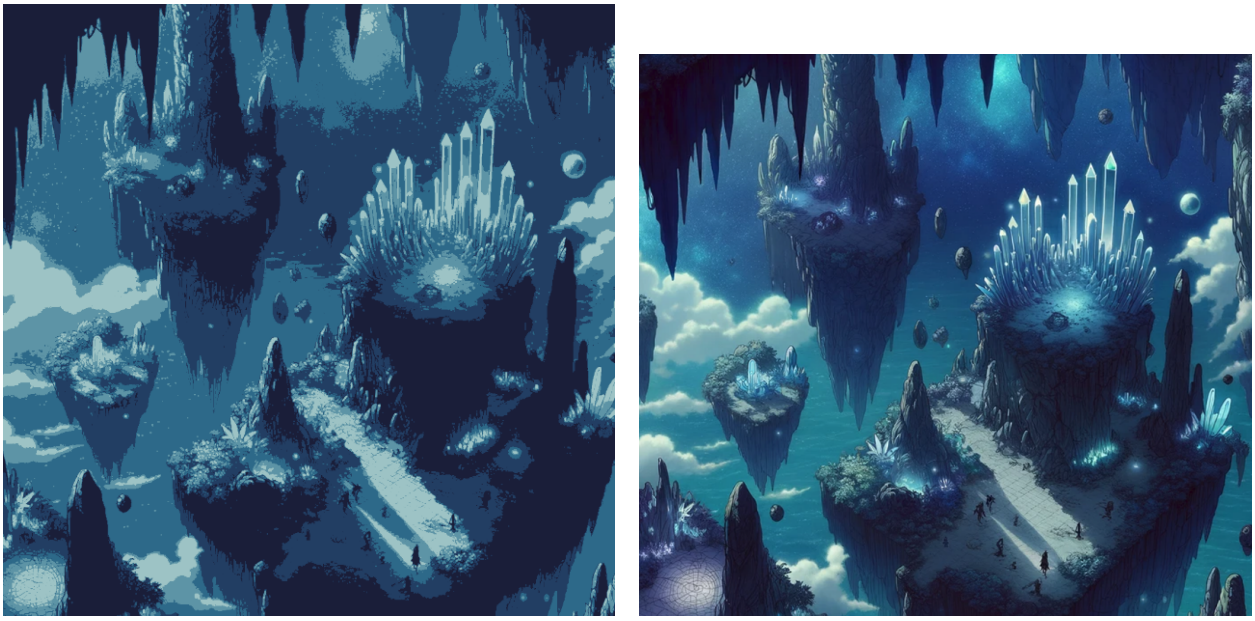
If $f(x)$ is convex and x can be written as the affine composition $x = Ay + b$, then $f(Ay + b)$ is also convex

13.2 Applications

Many real-world functions are not convex, so attempting to build convex representations can lose important signals and aspects of the original function. In the application of image reconstruction, taking an image that has been blurred and distorted and reconstructing it back into the original resolution of the image.

13.2.1 Application 1: Upscaling images

One application of convex optimization is through upscaling images that are noisy and blurry in attempts to recover the original detail of the image.



(a) Blurry and noisy image

(b) Original image being recovered

Figure 10: Comparison between the blurry and noisy image (left) and the original image (right).

13.2.2 Radiotherapy treatment planning

Convex optimization can be useful in radiotherapy treatment. Radiation damages the DNA of tumor cells. Properly optimized radiotherapy machines aim to maximize the amount of tumor damaged by the machine while minimizing the damage to other cells. The stochasticity in the partical/xray process introduces non-linearity into the problem. Some constraints are in the problem because there are critical organs that must be avoided in the process.

$$\begin{aligned} \min f(D) \\ s.t. D = F(x) \\ x \geq 0 \\ \text{other dose constraints} \end{aligned} \tag{65}$$

Where F is a function that maps the source of radiation to the distribution of dosage. f is better dose conforming given dose prescription. x is a vector with intensity, angle, and shape attributes. x

is the intensity of radiation from its sources. Each beam is separated into a grid, and then collected into the variable x where each entry in the vector corresponds to an intensity of a beam at a certain beamlet. D is dose distribution. The amount of energy on each region (voxel, a small cube unit) of the human body.

$$D = \begin{bmatrix} 1.4 \\ 3.3 \\ 5.2 \end{bmatrix} \tag{66}$$

Here, that means the region is partitioned into three voxels, and the amount of dose in each region is 1.4, 3.3, and 5.2 respectively.

$$x = \begin{bmatrix} 0 \\ 3 \\ 5 \\ 0 \\ 3 \\ 2 \end{bmatrix} \tag{67}$$

Here, the x is a flattened matrix that contains the intensity and direction of a bunch of beamlets collapsed into one vector.

Problem formulation:

- target: 95% of volume ≥ 72
- critical organ 1 average ≤ 10
- Critical organ 2 max dose ≤ 72

14 Monday 10/07/2024

14.1 Radiotherapy example

$$\begin{aligned} \min f(D) \\ s.t. D = F(x) \\ x \geq 0 \\ \text{other dose constraints} \end{aligned} \tag{68}$$

D is the dose distribution, x is a vector beamlet intensities, some clinical criteria are:

- Target region should have 95% of its bolume with 72 Gy or more
- Critical organ 1 should have the average less than or equal to 10 Gys
- The max of critical organ 2 should be less than or equal to 72 Gy

How can we model $D = F(x)$ while maintaining convexity? That constraint must be affine in order for the problem to still be a convex optimization problem. F is a mapping from $\mathbb{R}^m \to \mathbb{R}^p$ where m is the number of beamlet intensities and p is the number of voxels in the human body. To preserve convexity, we can approximate the original problem with a linear formulation of the first contraint.

$$\begin{aligned} \min f(D) \\ s.t. Ax - ID \\ x \geq 0 \\ \text{other dose constraints} \end{aligned} \tag{69}$$

Now modelling clinical constraints. We can define C_2 as the region of D pertaining to critical organ 2 and same for critical organ 1:

$$\begin{aligned} \min f(D) \\ s.t. Ax - ID \\ x \geq 0 \\ d_i \leq 72, i \in C_2 \\ \frac{1}{|C_1|} \sum_{i \in C_1} d_i \leq 10 \\ \text{other dose constraints} \end{aligned} \tag{70}$$

14.2 Big M

Can model a binary variable z_i that is the indicator function of $d_i \geq 72$

15 Friday 10/11/2024

15.1 Chemotherapy formulation

$$\begin{aligned} \text{minimize } & -\sum_{i \in T} d_i \\ \text{subject to } & Ax = D \\ & d_i \leq 72 \forall i \in C_1 \\ & \frac{1}{|C_2|} \sum_{i \in C_2} d_i \leq 10x \geq 0 \end{aligned} \tag{71}$$

How do we incorporate the constraint that 95% of voxels in the target region T must be greater than or equal to 72 radiation units? Hint given is that the maximum of a set of convex functions is convex. $g(d_i) = \max\{0, 72 - d_i\}$. The objective function can be changed to $\sum_{i \in T} g(d_i)$.

After this formulation, it is possible for the feasible region to be empty. Is there a way to introduce a trade-off so we can "pay" to violate constraints. This can be done by adding weights to constraints and adding them to the objective function.

$$\begin{aligned} \text{minimize } & w_1 \sum_{i \in T} g(d_i) + w_2 \sum_{i \in C_1} h(d_i) \\ \text{subject to } & Ax = D \\ & \frac{1}{|C_2|} \sum_{i \in C_2} d_i \leq 10x \geq 0 \end{aligned} \tag{72}$$

Where $h(d_i) = \max\{0, d_i - 72\}$

Summarizing the radiotherapy problem: It is a feasibility problem with many dose constraints where we want to find a radiotherapy treatment that doesn't violate the dose constraints. However, practically the feasible region tends to be empty for a lot of those problems. So instead of just returning no solution, the problem is reformulated with the constraints being placed in the objective function with weights so that the problem can be manipulated to relax constraints to attain feasibility.

16 Monday 10/14/2024

16.1 Gradient Descent

If we have a point x_0 of the differentiable function

$$\min_{x \in \mathbb{R}^n} f(x) \tag{73}$$

How do we improve on that to approach a better solution? We can use the gradient and travel opposite of it to the point $x_1 = x_0 - \delta \nabla f(x_0)$

$$f(x_0 + \delta d) - f(x_0) = \delta \langle \nabla f(x_0), d \rangle + o(\delta) \tag{74}$$

Iteratively plugging in to improve on an initially arbitrary point x_0 to push closer and closer to the optimal value. We can pick a value K to repeat the bottom iteration step K amount of times. We also specify the α as the step size that the gradient descent algorithm takes.

$$x_k = x_{k-1} - \alpha \nabla f(x_{k-1}) \tag{75}$$

In neural networks, gradient descent is used to push towards the optimal point and attain a good fit to the data. Backpropagation is used to smartly exploit the structure of the neural network to effiiciently calculate the gradient of the NN.

If the problem is convex and the gradient is continuous, then as K approaches infinity, the value $f(x_K)$ will approach $f(x^*)$. If convex, then $|f(x_K) - f(x^*)| \leq \mathbf{O}(\frac{1}{K})$

How do we pick α? When α is large, the algorithm diverges, but if it is too small, then the algorithm will practically take too long to converge as the step sizes don't make any meaningful progress. One method of picking α that will somewhat alleviate these cases is to decrease α over time at a rate $\alpha_k = \frac{\theta}{\sqrt{k}}$. Still need to pick the value for θ, but the chance of the algorithm diverging or taking too long to converge is much lower.
If $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, then the optimal choice for α is $\frac{1}{L}$.
We can reframe the choice of α as an optimization problem.

17 Wednesday 10/16/2024

17.1 Gradient descent review

Can formulate the gradient descent problem as an optimization problem while still preserving convexity. We want to find the optimal step size. If we know x^*, then we can pick step sizes that would cause us to get as close as possible to it. Locally, the only information that we have is the previous gradient $\nabla f(x^{k-1})$, the current iteration and previous iteration point x^k, x^{k-1} and the function values for those points $f(x^k), f(x^{k-1})$.

$$\min_{\alpha} f(x^{k-1} - \alpha \nabla f(x^{k-1})) \tag{76}$$

The above optimization problem is an unconstrained convex problem if the function f is convex since input to f is an affine combination with respect to α. For each problem, $x^{k-1}, \nabla f(x^{k-1})$ are fixed parameters. The bisection method is a very effective algorithm for searching for the appropriate value of α.

17.2 Bisection method

For a one-dimensional optimization problem, starting with a guess of α_1, can evaluate the derivative of the function and grab the sign. If the sign of the derivative is positive, then that is the "positive" upper bound to the problem and we know the minimal value is less than the guess α_1. We then guess α_2 and evaluate its sign, if it is negative, we know the optimal value is between α_1 and α_2.

18 Monday 10/28/2024

18.1 Nesterov's optimal method

Algorithm 1 Nesterov's Method

$\mathbf{y}^1 = \mathbf{x}^0 \in \mathbb{R}^n, t_1 = 1, k = 1$
for $k = 1, \ldots, K$ **do**
  $\mathbf{x}^k = \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k)$
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
  $y^{k+1} = x^k + \frac{t_k - 1}{t_{k+1}}(x^k - x^{k-1})$
  $k+ = 1$

18.2 Alternative Gradient Descent

Alternative intepretation of gradient Descent

$$\mathbf{x}^k = \arg\min_x f(\mathbf{x}^{k-1}) + \langle \nabla f(\mathbf{x}^{k-1}), \mathbf{x} - \mathbf{x}^{k-1} \rangle + \frac{1}{2\alpha^k}\|\mathbf{x} - \mathbf{x}^{k-1}\|_2^2 \tag{77}$$

18.3 Newton's Method

Newton's Method:

$$\mathbf{x}^k = \arg\min_x f(\mathbf{x}^{k-1}) + \langle \nabla f(\mathbf{x}^{k-1}), \mathbf{x} - \mathbf{x}^{k-1} \rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{k-1})^T \nabla^2 f(\mathbf{x}^{k-1})(\mathbf{x} - \mathbf{x}^{k-1}) \tag{78}$$

For Newton's Method, instead of using a manually chosen step-size of α, we boost the efficiency by changing the step size at each point to be the second derivative at that point. The drawback with Newton's method is that calculating the inverse of the Hessian can be time expensive. First order methods are more useful than second order (newton's) because less memory is needed for big calculations.

18.4 Zeroth-order methods

Blackbox/derivative-free optimization. There are algorithms built for optimizing based off of input and output instead of derivatives. Backpropagation also even becomes computationally difficult when there are a massive number of parameters and decision variables. So, how do we design an optimization algorithm when the gradient ∇f can not be calculated because we do not know f, we can only feed it inputs and evaluate its outputs.

$$f'(x) = \lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta} \tag{79}$$

In Zeroth-order methods, we can't take the limit to zero so instead we evaluate at small step sizes and approximate the derivative with

$$f'(x) = \frac{f(x+\gamma) - f(x)}{\gamma}, \text{ for a small } \gamma \neq 0 \tag{80}$$

Since zeroth order methods do not require a derivative, instead they require that the simulation or function is evaluated at each point chosen. If a simulation has 1,000 variables, then the simulation must be ran 1,001 times. This is because there is a different amount of step in each dimension.

Figure 11: Blackbox Optimization

19 Monday 11/04/2024

19.1 Optimization Algorithms for Unconstrained Problems 3

19.1.1 Meta-Heuristics

There are heuristic optimization algorithms that try to exploit some intuitive understanding of the original problem and use that to find the best solution instead of some problem-agnostic algorithm like gradient descent. Genetic algorithms are algorithms that create different generations of solutions with characteristics and a "fitness" score. At each iteration, the probability of an algorithm surviving is defined by its fitness score and then a new generation is created from the solutions that survived. At each generation, there is a mutation process that can introduce new characteristics into the solutions.

19.2 Optimization Algorithms for Constrained Problems

A constrained optimization problem can be transformed into an unconstrained problem by introducing a cost to each constraint in the objective function.

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g(x) \leq 0 \\ &h(x) = 0 \end{aligned} \tag{81}$$

Can be turned into

$$\text{minimize } f(x) + \lambda g(x) + \nu h(x) \tag{82}$$

19.2.1 Projection-based method

Mirror descent is an optimization method that allows us to alter the unconstrained case of gradient descent and change it into the constrained case. We introduce the domain and restrict it to a convex set $\Omega \subseteq \mathbb{R}^d$