EECS 127 Lecture Notes

Authored by Dun-Ming Huang, SID: 303******2

0.1 Preface

This section is directly copy-pasted from my GitHub repo for the note.

eecs127 moment.
Hi I'm Brandon. You might know me from work. If not, I think you should be glad you don't, or you would have had to deal with my EECS 16A Puns.
I update this note after every lecture and before exams (which I will notify about if I run a revision, or you can watch the repository).
Since I make these lecture notes by transcribing lecture contents on the fly, you would probably expect immature pedagogical formatting (which immediately appears) and typos in these efforts.

0.1.1 First of all, My Notes are Not Substitutes to the Course Readers

My notes are my own transcriptions for EECS 127 lecture notes. On a professional note, yes, you may use these for personal purposes and gaining extra insights on in-course contexts, but this is by no means a pedagogical substitute of the original course notes, even if it describes all the concepts of the course reader.

On a professional note, once again, the official course readers are structured in a pedagogical imperative: it connects concepts to optimize student understanding and the coherency of course content (even if it might have been confusing on the first reads). This course note does not. I was not obligated to connect concepts. I was only writing the work to organize information for my own sake. This work is not specifically written for the public just like my CSM resources were. My notes are by no means a substitute of EECS 127 readers.

But, thinking from the other angle, we may consider these notes I produced as a complementary resource for EECS 127.

0.1.2 How did Brandon Use This Note?

Great Question. I'm glad you asked. Cyrus.

I used this note extensively to lookup summaries of concepts when writing homeworks and mock exams, as well as to force myself to read through derivations on lecture notes via transcribing them onto LaTeX and running a rigorous revision later. I have also shared my notes to others (as you see, it is hosted as an open source project on Github), in the hope that they can use these notes as a lookup for summaries.

You can expect to see more formal course contents starting at the later half of Note 1.

0.1.3 What is the Last Four Digits of Your Social Security Number?

gXcQ

Contents

Chapter 1

Introduction and Least Squares

For this lecture, we will traverse through the Spring 2023 versions of course logistics, and then an introduction towards optimization.

1.1 An Introductory Prompt

In this course, we discuss a problem in Computer Science called, “optimization”. Understanding optimization is not limited to just understanding mathematical details, and has more to do with “problem formulation”: for whatever technology comes, optimization is an involved technique of critical thinking.

1.2 Administratives, Summary

Logistics. For Discussion, Homework policies, please check the lecture slides. Friday sections are equivalent to the subsequent Monday sections.
Advices from Instructor. Do homeworks, collaborate with people.
New: Projects. We will have an option to do a project at the end of the semester, where we complete a project of choice. This is an effort to bridge the gap between students and research experience; where, provided a research paper, the student will extend that literature at the end of project. This will count towards grade and remove weight from exams if the project grade helps. Schemes will be announced. Projects will be in groups, released after the midterm and due during RRR week.

1.3 Introduction to The Subject Topic: Optimization

Optimization is an approach to problems.
It is the technique where we attempt to optimize some statistic resembling of a result. For example, in machine learning from DATA C100 (or EECS 16A/B), we have chosen appropriate loss functions for our learning task to characterize the performance of a model:

> - Minimizing the MSE of a linear regresssion model.
> - Minimizing the cross-entropy loss of a logistic regression model.
> - Minimizing the distance traveled by an agent in a maze.

Here, we model learning tasks and attempt to optimize a metric. Depending on how we model and represent a problem, the model will perform differently on its learning task. Therefore, optimization is a study about “picking the right loss function” for some same objective. We consider these design choices, study how they affect learning process.

For another example, Air Traffic control is another optimization problem (at the point of writing, US has just experienced a flight paralysis); for another example, queueing and revenue computation are also common optimization problems. Optimization itself is omnipresent in modern applications.

In this course, we will learn about:

- Low rank approximation
- Ridge regression
- Stochastic Gradient Descent
- Dual Program (which always provides a lower bound for some optimization problem)
- Applications: LQR Control, Classification, SVM

Let's get into the Math now!!!

1.3.1 An Example of Problem Formulation

Say, we work in an oil production firm (I know, very US). We are allowed to make 10,000 barrels of crude oil, which may be produced into either *Jet Fuel* or *Gasoline*.
For every barrel of Jet Fuel produced, a revenue of 30 cents; for Gasoline, 20 cents. Meanwhile, 1 barrel of crude oil produces 0.6 barrels of Jet Fuel or 0.7 barrels of Gasoline.
Furthermore, the firm demands that we produce more than some amount of Jet Fuel and Gasoline (respectively, say, 1000 and 2000 barrels).
Finally, there are transportation capacities, where 180000 barrel miles are available. Distributing Gasoline costs 30 miles, and distributing Jet Fuel costs 10 miles.

We have now been presented a prompt: provided the above conditions, how can I maximize my revenue?
Let us first formulate this prompt mathematically:

$$\begin{cases} x_j = & \text{Quantity of Jet Fuel in barrels} \\ x_g = & \text{Quantity of Gasoline in barrels} \end{cases}$$

The revenue we generate would be formulated as

$$R(x_j, x_g) = 0.3x_j + 0.2x_g$$

such that we are presented the **constraints**:

$$\begin{cases} \text{Production Quantity Minimum:} & x_j \geq 1000, x_g \geq 2000 \\ \text{Total Available Resources:} & \frac{x_j}{0.7} + \frac{x_g}{0.6} \leq 10000 \\ \text{Transportation Capacity:} & 10x_j + 30x_g \leq 180000 \end{cases}$$

which are mathematically translated from the above English text.

In an optimization framework, we formualte a problem with the following frame:

Explain 1.3.1. Formulating Optimization Problems

In a general optimization problem, we attempt to minimize some function $f_0(\vec{x})$, such that some constraint exists:

$$\forall i \in \{1, \ldots, m\}, f_i(\vec{x}) \leq b_i$$

and here, $\vec{x}$ is listed as a representation of some system's state, existing within the domain of possible inputs (states).
The general objective of optimization is to find a state of system that minimizes $f_0(\vec{x})$, which we mathemati-

cally express the solution as:

$$\vec{x}^* = argmin_{\forall i \in \{1,...,m\}, f_i(\vec{x}) \leq b_i} f_0(\vec{x})$$

See the example from above, and try matching the parts of optimization problem with the above framework!

There are some more types of optimization problems! One of them is the famous Least Squares Regression:

1.4 Least Squares Regression

The problem statement:

For some matrix A and a vector $\vec{b}$, solve for:

$$\min_{\vec{x}} ||A\vec{x} - \vec{b}||_2^2$$

Such problem can be widely applied to many mathematical problems, such as regression and projection.

1.4.1 Formulating Least Squares Regression

Let us discuss least squares algorithm in the context of linear regression:

Provided a dataset of points:

$$\{(x_1, y_1), \ldots, (x_n, y_n)\}$$

let us attempt to model a linear equation

$$y = mx + c$$

First of all, what is the variable we attempt to search/optimize for?
It would be m, c that are the unknowns.
To formulate the Least Squares Problem, we would need to formulate our linear equation for points into some matrix-vector multiplication, and knowing that we are solving for m and c, the formulation follows:

$$A \begin{bmatrix} m \\ c \end{bmatrix} = \vec{b}$$

Let us now fill in the contents of A and $\vec{b}$ for formulation, where A and $\vec{b}$ are of known quantities:

$$\begin{bmatrix} \vec{x} & \vec{1} \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} = \vec{y}$$

1.4.2 Closed-Form Solution of Least Squares Problem

From prior coursework we also know that there is a closed-form solution:

$$\vec{x}^* = (A^T A)^{-1} A^T \vec{b}$$

for matrices A with nontrivial nullspaces (see EECS16A for a proof that $N(A) = N(A^T A)$).

To minimize the L2 norm of difference between $A\vec{x}$ and $\vec{b}$, we attempt to find a vector $\vec{x}$ that minimizes the differences between them.
Note that for any choice of $\vec{x}$, it is by definition that $A\vec{x} \in Col(A)$.

We now face two claims that we resolve to proceed on solving this problem:

> 1. The endpoint of $proj_{Col(A)}\vec{b}$ is the point closest to the endpoint of $\vec{b}$ (if they share a same starting point) in $Col(A)$.
> 2. $\vec{OP} = A\vec{x}^*$, where $\vec{x}^*$ has the closed-form solution as previously described.

Claim 1 can be proven by contradiction:

> Assume another point C closer than P to B (such that $\vec{OB}$ is closer to $\vec{b}$). However, Pythagorean Theorem argues against it.

Claim 2 is a series of algebraic manipulation as outlined in previous courseworks:
Based on that the error vector $\vec{e} = \vec{b} - A\vec{x}$ should be orthogonal to the columnspace $Col(A)$ (because of geometry):

$$\begin{aligned} A^T\vec{e} &= 0 \\ A^T(A\vec{x}^* - \vec{b}) &= 0 \\ A^TA\vec{x}^* &= A^T\vec{b} \\ \vec{x}^* &= (A^TA)^{-1}A^T\vec{b} \end{aligned}$$

Such result is also known as the **Normal Equation**.

Interestingly, Least Squares Algorithm has a quadratic property, causing it to be some parabolic object that performs a **convex function** to optimize along; that is, the local minimum of this function is a global minimum.

Chapter 2

Linear Algebra Bootcamp: Norms, Gram-Schmidt, QR, FTLA

2.1 Vectors and Norms

In the previous lecture, we have recognized the following symbol:

Symbol 2.1.1. Euclidean Norm

The Euclidiean Norm, otherwise known as a **2-Norm**, is a mathematical quantity for some vector $\vec{x}$:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

which measures the Euclidean distance (Pythagorean Theorem's results) between the starting and terminal point of a vector.

While the Euclidean Norm is an extremely common norm, there exist more norms to vectors.
Particularly, **norms** are functions that satisfy the following properties:

Definition 2.1.1. Norm

A norm is a function $f : \mathcal{X} \to \mathbb{R}$ such that:

1. Non-negativeness: $\forall \vec{x} \in \mathcal{X}, \|\vec{x}\| \geq 0$, and $\|\vec{x}\| = 0 \iff \vec{x} = \vec{0}$
2. Triangle Inequality: $\forall \vec{x}, \vec{y} \in \mathcal{X}, \|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$
3. Scalar Multiplication: $\forall \alpha \in \mathbb{R}, \vec{x} \in \mathcal{X}, \|\alpha\vec{x}\| = \alpha\|\vec{x}\|$

2.1.1 LP-Norms

It is noteworthy to mention that, almost every norm demonstrates one property of the vector. For the diversity of characteristics that a vector may have, there certainly exists a diversity of norms for vectors.

For example, a general family of norms that satisfy the above properties would be the **LP-norms**:

Definition 2.1.2. LP-Norms

LP-Norms are norm functions defined as:

$$\|\vec{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

for some natural number p.

Particularly, the Euclidean Norm, otherwise known as the *2-norm* is LP-norm with $p = 2$.
Meanwhile, the 1-norm and ∞-norm are defined as follows:

$$\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|$$
$$\|\vec{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|$$

where, the 1-norm of a vector can be stated as the Manhattan distance (Taxicab geometry) of between that vector's starting and terminal point.

2.2 Cauchy-Schwartz Inequality

This inequality was active in EECS 16A!

Definition 2.2.1. Cauchy-Schwartz Inequality

The inequality is phrased as:

$$|\vec{x}^T\vec{y}| \leq \|\vec{x}\|_2\|\vec{y}\|_2$$

And using the property,

$$\forall x \in \mathbb{R}, |\cos(x)| \leq 1$$

This inequality originates from the following algebraic work:

$$\begin{aligned} |\langle\vec{x},\vec{y}\rangle| &= |\vec{x}^T\vec{y}| = |\vec{y}^T\vec{x}| \\ &= |\|\vec{x}\|_2\|\vec{y}\|_2\cos\left(\theta_{\vec{x},\vec{y}}\right)| \leq |\|\vec{x}\|_2\|\vec{y}\|_2| \end{aligned}$$

We observe that the above derivation holds statement on equivalence between inner product and product of magnitudes, cosine.
That is justified by the mechanics along which we find the projection of $\vec{y}$ onto $\vec{x}$:

$$\text{proj}_{\vec{x}}(\vec{y}) = \vec{x}\frac{\vec{x}^T\vec{y}}{\|\vec{x}\|_2^2}$$

where, since the projection is a multiple of $\vec{x}$ such that $\text{proj}_{\vec{x}}(\vec{y}) = t\vec{x}$, we also recognize that,

$$\cos\left(\theta_{\vec{x},\vec{y}}\right) = \frac{\|t\vec{x}\|_2}{\|\vec{y}\|_2}$$

Upon matching the two equations, I acquire:

$$t = \cos\left(\theta_{\vec{x},\vec{y}}\right)\frac{\|\vec{y}\|_2}{\|\vec{x}\|_2} = \frac{\vec{x}^T\vec{y}}{\|\vec{x}\|_2^2}$$
$$|\|\vec{x}\|_2\|\vec{y}\|_2| = \langle\vec{x},\vec{y}\rangle$$

2.2.1 Holder's Inequality and Norm Ball

In fact, we find that the Cauchy-Schwartz is a general case of this inequality:

Definition 2.2.2. Holder's Inequality

Provided the quantities

$$\vec{x}, \vec{y} \in \mathbb{R}^n, \text{and some } p, q \geq 1 \text{ s.t. } \frac{1}{p} + \frac{1}{q} = 1$$

Then,

$$|\vec{x}^T \vec{y}| \leq \|\vec{x}\|_p \|\vec{y}\|_p$$

From this concept, let us consider the concept of *"norm ball"*: the geographical object containing all vectors such that their lp-norm is 1.

> - For a 2-norm, for example, the norm ball looks like a unit circle (containing all 2D vectors with length 1, hence a circle of radius 1).
> - For a 1-norm, similar logic guides us to a diagonally placed circle (rotated $45°$) centered at the origin with side lengths 2.
> - For the ∞-norm, has a norm ball of a square centered at origin with side length 2. This embeds the unit ball from 2-norm, which embeds the unit ball from 1-norm.

The last bullet point hints that the area of norm ball is larger for any increase in p such that it embeds the prior norm balls.

Now, let us attempt to solve for some optimization regarding a norm ball:

$$\max_{\|\vec{x}\|_p \leq 1} \vec{x}^T \vec{y}$$

p = 2:

The solution would be, by the Cauchy Schwartz's implication,

$$\vec{x}^* = \frac{\vec{y}}{\|\vec{y}\|_2}$$

p = 1:

The expression of dot product is equivalently

$$x_1 y_1 + \cdots + x_n y_n$$

Where the constraint is

$$|x_1| + \cdots + |x_n| \leq 1$$

For each value the components of $\vec{x}$, which is finite and upper bounded by 1, we should allocate maximum contribution to the maximum element of $\vec{y}$ to maximize this dot product (a weighted sum of $\vec{y}$'s component, essentially).
Therefore, let i be the index at which $\vec{y}$ has the component of largest absolute value,

> There is an achievable solution for $\vec{x}^*$, being the unit vector $\vec{e}_i$ multiplied by $sgn(y_i)$, so to counter for cases where $\vec{y}$ is negative.

In turn, we see that

$$\vec{x}^T \vec{y} = \max_i |y_i| = \|\vec{y}\|_\infty$$

Meanwhile, let us use the Holder's Inequality to achieve a more rigorous proof. Holder's Inequality states that,

$$|\vec{x}^T \vec{y}| \leq \|\vec{x}\|_1 \|\vec{y}\|_\infty = \max_i |y_i|$$

The Holder's Inequality expresses **an upper bound**, and the formulation in above section shows **achievability of upper bound**.

Alternative proof of p = 1 via Triangle Inequality:
Via the triangle inequality, we acquire:

$$\begin{aligned} |\vec{x}^T\vec{y}| &= |\sum_i x_i y_i| \\ &\underset{Triangle\ Inequality}{\leq} \sum_i |x_i y_i| \\ &= \sum_i |x_i||y_i| \\ &\leq \sum_i |x_i| \max_i |y_i| = \max_i |y_i| = \|\vec{x}\|_\infty \end{aligned}$$

We see the expression $|\vec{x}^T\vec{y}|$ has an **achievable upper bound**, assembling all necessary aspects of deriving the solution for the optimization problem.

p = ∞:

We can find upper bound of the expressionvia Holder's Inequality,

$$|\vec{x}^T\vec{y}| \leq \|\vec{x}\|_1\|\vec{y}\|_\infty = \sum_i |y_i|$$

The infinity norm of $\vec{x}$ shows the constraint that:

$$\max_i |x_i| = 1$$

and we may simply compute:

$$x_i = sgn(y_i)$$

to find and certify the achievability of this upper bound.

2.3 Gram-Schmidt Orthonormalization and QR Decomposition

Gram-Schmidt Orthonormalization is an algorithmic technique to find an orthonormal basis for a set of vectors. The procedure of such algorithm is portrayed as defined in the following cell:

Definition 2.3.1. The Procedure of Gram-Schmidt Orthonormalization

Let us have a set of vectors $\{\vec{a_1}, \dots, \vec{a_n}\}$.

```
$\vec{q_1} = \frac{\vec{a_1}}{\|\vec{a_1}\|_2}$
for $i$ in $\{2, \dots, n\}$:
    $\vec{z_i} = proj_{\{\vec{q_1}, \dots, \vec{q_{i-1}}\}}\vec{a_i} = \sum_{j=1}^{i-1} \vec{q_j}\langle\vec{a_i}, \vec{q_j}\rangle$
    $\vec{s_i} = \vec{a_i} - \vec{z_i}$
    $\vec{q_i} = \frac{\vec{s_i}}{\|\vec{s_i}\|_2}$
return $\{\vec{q_1}, \dots, \vec{q_n}\}$
```

Then, QR Decomposition is a technique to factorize a matrix A based on its orthonormal basis:

Definition 2.3.2. QR Decomposition

The QR decomposition of a matrix A is to factorize A as the product QR, where the two matrices are decided as:

$$\begin{cases} Q, & \text{An orthonormal matrix whose span is equal to that of } A \\ R, & R_{ij} = \vec{q_i} \cdot \vec{a_j} \end{cases}$$

where we may define vectors $\vec{a_i}$ as the i^{th} column of A, and set of vectors $\vec{q_i}$ may be found as the orthonormal base of $\mathcal{R}(A)$ via Gram-Schmidt.

Explain 2.3.1. Example of QR Decomposition

Problem: Solve for its QR Decomposition of

$$A = \begin{bmatrix} 3 & -3 & 1 \\ 4 & -4 & 7 \\ 0 & 3 & 3 \end{bmatrix}$$

Finding the first column of Q:

$$\vec{q_1} = \frac{1}{5} \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix}$$

Finding the second column of Q:

$$\begin{aligned} \text{proj}_{\vec{q_1}}(\vec{a_2}) &= (\vec{a_2} \cdot \vec{q_1})\vec{q_1} \\ \vec{z_2} &= \vec{a_2} - \text{proj}_{\vec{q_1}}(\vec{a_2}) \\ &= \begin{bmatrix} -3 \\ -4 \\ 3 \end{bmatrix} - \begin{bmatrix} -3 \\ -4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} \\ \vec{q_2} &= \frac{\vec{z_2}}{\|\vec{z_2}\|_2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

Finding the third column of Q:

$$\begin{aligned} \text{proj}_{\{\vec{q_1},\vec{q_2}\}}(\vec{a_3}) &= (\vec{a_3} \cdot \vec{q_1})\vec{q_1} + (\vec{a_3} \cdot \vec{q_2})\vec{q_2} \\ &= \begin{bmatrix} -3 \\ -4 \\ 3 \end{bmatrix} \\ \vec{z_3} &= \vec{a_3} - \text{proj}_{\{\vec{q_1},\vec{q_2}\}}(\vec{a_3}) \\ &= \begin{bmatrix} 1 \\ -7 \\ 3 \end{bmatrix} - \begin{bmatrix} -3 \\ -4 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \\ 0 \end{bmatrix} \\ \vec{q_3} &= \frac{\vec{z_3}}{\|\vec{z_3}\|_2} = \frac{1}{5} \begin{bmatrix} 4 \\ -3 \\ 0 \end{bmatrix} \end{aligned}$$

And now, finding out that we still have to find the matrix R (sob emoji):

$$R = \begin{bmatrix} \vec{q_1} \cdot \vec{a_1} & \vec{q_1} \cdot \vec{a_2} & \vec{q_1} \cdot \vec{a_3} \\ 0 & \vec{q_2} \cdot \vec{a_2} & \vec{q_2} \cdot \vec{a_3} \\ 0 & 0 & \vec{q_3} \cdot \vec{a_3} \end{bmatrix} = \begin{bmatrix} 5 & -5 & -5 \\ 0 & 3 & 3 \\ 0 & 0 & 5 \end{bmatrix}$$

Chapter 3

Linear Algebra: Symmetric Matrices

3.1 Fundamental Theorem of Linear Algebra

Spaces are sets of vectors that follow the ten commandments (or rules) of being a vector space.
Among such perspective, we find ourselves curious about the orthogonality of sets of vectors (vector space):

Theorem 3.1.1. Orthogonal Decomposition of Space

Let us consider some vector space $\mathcal{X}$ and some subspace $\mathcal{S} \subseteq \mathcal{X}$.
Then, we may state that,

$$\forall \vec{x} \in \mathcal{X}, \vec{x} = \vec{s} + \vec{r} \text{ uniquely, s.t. } \vec{s} \in \mathcal{S} \wedge \vec{r} \in \mathcal{S}^{\perp}$$

such that,

$$\mathcal{S}^{\perp} = \{\vec{y} : \langle \vec{x}, \vec{y} \rangle = 0, \vec{x} \in \mathcal{S}\}$$

Conseuqentially, we are stating that for the space $\mathcal{X}$,

$$\mathcal{X} = \mathcal{S} \oplus \mathcal{S}^{\perp}$$

In other words, it is a **direct sum** of the vector spaces $\mathcal{S}$ and $\mathcal{S}^{\perp}$. We may write any vector in $\mathcal{X}$ as the sum of two components where each component belongs to some specific subspace of the direct sum.

This leads to a theorem about decomposing a vector space from some given matrix:

Theorem 3.1.2. Fundamental Theorem of Linear Algebra

Theorem. Consider a matrix $A \in \mathbb{R}^{m \times n}$, then,

$$\mathbb{R}^n = N(A) \oplus R(A^T)$$

where, similarly,

$$\mathbb{R}^m = \mathcal{R}(A) \oplus N(A^T)$$

Proof. We may use Orthogonal Decomposition of Space (Theorem 3.1.1) to aid our proof. If so, we would only need to show the two following facts:

1. $N(A) \perp \mathcal{R}(A^T)$, or equivalently, $N(A) = \mathcal{R}(A^T)^{\perp}$
2. $N(A^T) \perp \mathcal{R}(A)$

Let us perform a proof on the first fact, which would be to prove the equivalence of two sets.
First, we may prove that $N(A) \subseteq \mathcal{R}(A^T)^{\perp}$:

Suppose that there is an arbitrary vector $\vec{u} \in N(A)$, where $A\vec{u} = \vec{0}$.
Then, we realize that,

$$(A\vec{u}) = \vec{0}, \vec{0}^T = \vec{u}^T A^T$$

then, for any arbitrary vector $\vec{u}'$, we may find that

$$\vec{u}^T A^T \vec{u}' = \vec{0}^T \vec{u}' = 0$$

Therefore, the vector $\vec{u}$ is orthogonal to any vector belonging to $\mathcal{R}(A^T)$, which would mathematically state,

$$\vec{u} \in \mathcal{R}(A^T)^\perp$$

Then, we may prove the opposite direction: $\mathcal{R}(A^T)^\perp \subseteq N(A)$

Suppose that there is an arbitrary vector $\vec{w} \in \mathcal{R}(A^T)$, and $\vec{x} \in \mathcal{R}(A^T)^\perp$; by which, we would state for the arbitrary pair $(\vec{w}, \vec{x})$ that

$$\forall \vec{w} \in \mathcal{R}(A^T), \vec{x} \in \mathcal{R}(A^T)^\perp, \ (\vec{x} \cdot \vec{w} = 0)$$

Thus, $\forall \vec{w} \in \mathcal{R}(A^T)$:

$$\begin{aligned} \langle \vec{x}, \vec{w} \rangle &= 0 \\ \langle \vec{x}, A^T \vec{w} \rangle &= 0 \\ \vec{x}^T A^T \vec{w} &= 0 \end{aligned}$$

Which qualifies us to state that $A\vec{x} = \vec{0}$.
Therefore, $\vec{x} \in N(A)$.

Applying a symmetrical proof to the second fact allows us to provide a complete proof for this theorem.

3.2 Minimum Norm Problem

This is a sister problem to the Least Squares Problem. Let the following table show their differences:

Least Squares	Minimum Norm
Solves an overdetermined system	Solves an underdetermined system
Formulation: $\min_{\vec{x}} \|A\vec{x} - \vec{b}\|_2^2$	Formulation: $\min_{A\vec{x}=\vec{b}} \|\vec{x}\|_2^2$

3.2.1 Solution to the Minimum Norm Problem

Let us provide some $\vec{x}$ that is a solution to $A\vec{x} = \vec{b} \neq \vec{0}$.
Some component of $\vec{x}$ might belong to the nullspace of A. Meanwhile, we also acknowledge that $\vec{x}$ is some arbitrary vector in the real space. Therefore, the vector $\vec{x}$ may be decomposed into $\vec{x} = \vec{n} + \vec{r}$, where

$$\vec{n} \in N(A), \vec{r} \in \mathcal{R}(A^T)$$

In that case, we now minimize for $\vec{r} \in \mathcal{R}(A^T)$ under the setup that,

$$A\vec{x} = A(\vec{n} + \vec{r}) = A\vec{r} = \vec{b}$$

We may recognize that $\vec{n} \cdot \vec{r} = 0$ due to the orthogonal decomposition's nature. Therefore,

$$\|\vec{x}\|_2^2 = \|\vec{n}\|_2^2 + \|\vec{r}\|_2^2$$

Therefore, let $\vec{x}^* = A^T\vec{w}$ for some specific $\vec{w}$, so to state that $\vec{x}^* \in \mathcal{R}(\mathcal{A}^\mathcal{T})$:

$$\begin{aligned} A\vec{x} &= \vec{b} \\ AA^T\vec{w} &= \vec{b} \\ \vec{w} &= (AA^T)^{-1}\vec{b} \\ \vec{x}^* &= A^T(AA^T)^{-1}\vec{b} \end{aligned}$$

3.3 Principal Component Analysis

Principal Component Analysis is a technique to reduce dimensionality in high-dimension datapoints, so to find underlying feature patterns and enable plotting. This can be achieved via projecting data onto a lower dimension structure to recover lower dimensional structure from the original high dimensional data.

Mathematically, such technique would be to project $\vec{x_1}, \ldots, \vec{x_n}$ onto $\vec{w} \in \mathbb{R}^p$ such that the projected vectors are all as close to the original vectors as possible.

Explain 3.3.1. Projections

For projecting some vector $\vec{x}$ onto $\vec{w}$, we would obtain

$$\text{proj}_{\vec{w}}(\vec{x}) = (\vec{w}^T\vec{w})^{-1}(\vec{w}^T\vec{x})\vec{w}$$

Projecting vector $\vec{x}$ onto a columnspace of matrix A then follows the least squares formula,

$$(A^TA)^{-1}A^T\vec{x}$$

Now, let's discuss how would PCA work:

Explain 3.3.2. PCA as an Optimization Problem

Formulation.
Let the individual projection errors be phrased as

$$\|\vec{x_i} - \langle\vec{w}, \vec{x_i}\rangle\vec{w}\|_2^2$$

Where, the average projection error can thus be quantified as,

$$R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}\|\vec{x_i} - \langle\vec{w}, \vec{x_i}\rangle\vec{w}\|_2^2 = MSE(\vec{w})$$

Therefore, the formulation of optimization problem is:

$$\min_{\vec{w}} R(\vec{w}) \text{ subject to } \|\vec{w}\|_2^2 = 1$$

And the assumption of PCA dataset is that it is zero-meaned, because de-meaned data will allow less bias when deciding the weight vector for PCA.
Now, let us gently poke at the problem, even if we might barely know the problem.
Let us consider the projection error,

$$\|\vec{x_i} - \langle\vec{w}, \vec{x_i}\rangle\vec{w}\|_2^2 = \|\vec{e_i}\|_2^2$$

The projection error can be algebraically simplified:

$$\begin{aligned}\|\vec{e_i}\|_2^2 &= \left(\vec{x_i} - \langle\vec{w}, \vec{x_i}\rangle\vec{w}\right)^T \cdot \left(\vec{x_i} - \langle\vec{w}, \vec{x_i}\rangle\vec{w}\right) \\ &= \|\vec{x_i}\|_2^2 - 2\langle\vec{x_i}, \vec{w}\rangle^2 + \langle\vec{x_i}, \vec{w}\rangle^2\|\vec{w_i}\|_2^2 \\ &= \|\vec{x_i}\|_2^2 - \langle\vec{x_i}, \vec{w}\rangle^2\end{aligned}$$

Therefore, removing the fixed costs in the error, we are essentially facing the following optimization problem:

$$\max_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n}\langle\vec{x_i}, \vec{w}\rangle^2$$

Chapter 4

Principal Component Analysis

4.1 Symmetric Matrix

Let us begin with a formal definition of Symmetric Matrix:

Definition 4.1.1. Symmetric Matrix

A matrix $A \in \mathbb{R}^{n \times n}$ is a **symmetric matrix** if

$$A = A^T$$

or equivalently,

$$\forall i, j \in \{1, \cdots, n\}, a_{i,j} = a_{j,i}$$

and in such case we express,

$$A \in \mathbb{S}^n$$

Symmetric Matrices have such properties summarized by the Spectral Theorem:

Theorem 4.1.1. Spectral Theorem

If $A \in \mathbb{S}^n$, then:

1. All of its eigenvalues are real.
2. Eigenspaces corresponding to distinct eigenvalues are orthogonal.
3. The algebraic multiplicity of an eigenvalue is equal to its geometric multiplicity (alternatively, the matrix is diagonalizable, such that there exists an orthonormal U and diagonal Σ to compose $A = U\Sigma U^T$).

An interlude here:

Definition 4.1.2. Multiplicity of Eigenvalues

A review from EECS 16B:

- The algebraic multiplicity of an eigenvalue (m_A) is the number of times such eigenvalue appears in its matrix's characteristic polynomial.
- The geometric multiplicity of an eigenvalue (m_G) is the number of lienarly independent eigenvectors that such eigenvalue corresponds to.

Meanwhile, symmetric matrices also enjoy special properties in their eigenvalues:

Theorem 4.1.2. Variational Characteristics of Rayleigh Coefficient

The Rayleigh Coefficient of some symmetric matrix $A \in \mathcal{S}^n$ and vector $\vec{x} \in \mathbb{R}^n$ is defined as:

$$R_{A,\vec{x}} = \frac{\vec{x}^T A\vec{x}}{\vec{x}^T\vec{x}}$$

Theorem. Then, we may acknowledge via proof that:

$$\forall\vec{x} \neq \vec{0}, \lambda_{min}(A) \leq R_{A,\vec{x}} \leq \lambda_{max}(A)$$

and alternatively we may formulate the above inequality such that,

$$\begin{cases} \lambda_{max}(A) = \max_{\|\vec{x}\|_2^2=1} \vec{x}^T A\vec{x} \\ \lambda_{min}(A) = \min_{\|\vec{x}\|_2^2=1} \vec{x}^T A\vec{x} \end{cases}$$

Proof. Let us define that $\vec{y} = U^T\vec{x}$, and we will proceed onto the algebraic work:

$$\begin{aligned} \vec{x}^T A\vec{x} &= \vec{x}U\Lambda U^T\vec{x} \\ &= \vec{y}^T\Lambda\vec{y} \\ \|\vec{y}\|_2^2 &= 1 \end{aligned}$$

And, furthermore,

$$\begin{aligned} \vec{y}^T\Lambda\vec{y} &= \sum \lambda_i y_i^2 \\ &\leq \sum \lambda_{max} y_i^2 = \lambda_{max} \\ \vec{y}^T\Lambda\vec{y} &= \sum \lambda_i y_i^2 \\ &\geq \sum \lambda_{min} y_i^2 = \lambda_{min} \end{aligned}$$

So we have now found the upper bound and lower bound of the optimization problem, justifying the boundaries of inequality.
And, we may also acknowledge that providing $\vec{x}$ as the normalized eigenvector of maximum and minimum eigenvalues provide the solution for achieving the lower and upper bound:

$$\begin{aligned} \vec{v}_{\lambda_{max}}^T A\vec{v}_{\lambda_{max}} &= \lambda_{max}\|\vec{v}_{\lambda_{max}}\|_2^2 \\ &= \lambda_{max} \\ \vec{v}_{\lambda_{min}}^T A\vec{v}_{\lambda_{min}} &= \lambda_{min}\|\vec{v}_{\lambda_{min}}\|_2^2 \\ &= \lambda_{min} \end{aligned}$$

Below, let us discuss how we may use the properties of symmetric matrices to solve Principal Component Analysis, which is a problem we were studying in the previous lecture.

4.2 PCA

Principal Component Analysis is a technique to reduce dimensionality in high-dimension datapoints, so to find underlying feature patterns and enable plotting. This can be achieved via projecting data onto a lower dimension structure to recover lower dimensional structure from the original high dimensional data.

We will pick up from last lecture with a new formulation for optimization problem:

Explain 4.2.1. Introduction of Principal Component Analysis as Problem

Suppose we have a set of vectors:

$$\{\vec{x_1},\dots,\vec{x_n}\},\forall i(\vec{v_i}\in\mathbb{R}^p)$$

And we would like to project our data into a lower dimensional subspace, such that we may reduce computational cost, and projected vectors are close to the original data.

Let us find the first principal component, $\vec{w}$, then, such that the dataset projected on this direction is as close to the original data as possible. Then, we will repeat the process of:

- Find principal component $\vec{w}_i$

↓ Remove from dataset (usually a matrix) all projected components onto principal component i

- Repeat process to find principal component $\vec{w}_{i+1}$

4.2.1 Formulating PCA as an Optimization Problem

Let us now formulate the mathematical aspects of PCA.

Suppose we have a set of vectors:

$$\{\vec{x_1},\dots,\vec{x_n}\},\text{and }\forall i(\vec{v_i}\in\mathbb{R}^p)$$

Let us find a vector $\vec{w}$ along the constraint and objective function that:

$$\begin{cases}\|\vec{w}\|_2 & =1\\ MSE(\vec{w}) & =\frac{1}{n}\sum_{i=1}^n e_i^2\end{cases}$$

where, the individual projection error terms were defined and derived into:

$$\begin{aligned}\|\vec{e_i}\|_2^2&=(\vec{x_i}-\langle\vec{w},\vec{x_i}\rangle\vec{w})^T\cdot(\vec{x_i}-\langle\vec{w},\vec{x_i}\rangle\vec{w})\\&=\|\vec{x_i}\|_2^2-2\langle\vec{x_i},\vec{w}\rangle^2+\langle\vec{x_i},\vec{w}\rangle^2\|\vec{w_i}\|_2^2\\&=\|\vec{x_i}\|_2^2-\langle\vec{x_i},\vec{w}\rangle^2\end{aligned}$$

Therefore,

$$MSE(\vec{w})=\frac{1}{n}\sum_{i=1}^n(\|\vec{x_i}\|_2^2-\langle\vec{w},\vec{x_i}\rangle^2)$$

To consider the optimization problem holistically, we should formulate that,

$$\underset{\vec{w}}{argmin}MSE(\vec{w})=\underset{\vec{w}}{argmin}-\frac{1}{n}\sum_{i=1}^n\langle\vec{w},\vec{x_i}\rangle^2$$

which is equivalently solving for

$$\underset{\vec{w}}{\operatorname{argmax}}\sum_{i=1}^n\langle\vec{w},\vec{x_i}\rangle^2$$

and we would recognize so becuase the ignored aspect of MSE, $\frac{1}{n}\sum_{i=1}^n\|\vec{x}\|_2^2$, is a nonnegative fixed cost.

4.2.2 Solution to PCA as an Optimization Problem

Now, let us attempt to solve for the above formulation:

Keep in mind that our data matrix would appear as:

$$X = \begin{bmatrix} \vec{x_1}^T \\ \vdots \\ \vec{x_n}^T \end{bmatrix}$$

We may begin with the algebraic work then.

$$X\vec{w} = \begin{bmatrix} \vec{x_1}^T \vec{w} \\ \vdots \\ \vec{x_n}^T \vec{w} \end{bmatrix}$$

Let us capture the information and make some algebraic derivation:

$$\begin{aligned} \frac{1}{n} \sum \langle \vec{w}, \vec{x_i} \rangle^2 &= \frac{1}{n} \sum \left(\vec{x_i}^T \vec{w} \right)^2 \\ &= \frac{1}{n} \| X\vec{w} \|_2^2 \\ &= \vec{w}^T \frac{X^T X}{n} \vec{w} \end{aligned}$$

We call the matrix in above intermediate form the covariance matrix:

$$C = \frac{X^T X}{n} = C^T$$

And, using the knowledge from Section 4.1 about Rayleigh coefficient, we obtain that the solution of:

$$\operatorname*{argmax}_{\vec{w}} \vec{w}^T C \vec{w} \text{ subject to } \|\vec{w}\|_2^2 = 1$$

would be the eigenvector of $\lambda_{max}(C)$.

Chapter 5

SVD and Low-Rank Approximation

5.1 Singular Value Decomposition (SVD)

Let us suppose that there is a matrix $A \in \mathbb{R}^{m \times n}$ whose rank is $rk(A) = r$, then there exists a decomposition of A in the shape of:

$$A = \sum_{i=1}^{n} \sigma_i \vec{u_i} \vec{v_i}^T$$

where,

> - The set $\{\vec{u_1}, \dots, \vec{u_n}\}$ is orthonormal
> - The set $\{\vec{v_1}, \dots, \vec{v_n}\}$ is orthonormal
> - $\forall i \in [1, r], \sigma_i > 0$, and $\forall i \in [1, n], \sigma_i \geq 0$

Symbol 5.1.1. Compact SVD

The above dyadic decomposition of SVD written in terms of matrix transformation would be:

$$\begin{aligned} A &= \begin{bmatrix} \vec{u_1} & \cdots & \vec{u_r} \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v_1}^T \\ \vdots \\ \vec{v_r}^T \end{bmatrix} \\ &= U_r \Sigma_r V_r^T \end{aligned}$$

Meanwhile, the full SVD that still considers zero-valued σ_i is written as follows:

Symbol 5.1.2. Full SVD

The formulation follows:

$$\begin{aligned} A &= \begin{bmatrix} \vec{u_1} & \cdots & \vec{u_m} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{v_1}^T \\ \vdots \\ \vec{v_n}^T \end{bmatrix} \\ &= U \Sigma V^T \end{aligned}$$

from which, we may observe that the shape of Σ is highly dependent on the shape of A:

> - If A is tall, so is Σ
> - If A is wide, so is Σ

The squareness of U and V in full SVD allows us to also decompose A^TA as:

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^TU^TU\Sigma V^T = V\Sigma^2V^T$$

which is more convenient than the compact form SVD that instead uses rectangular U and V.

5.1.1 Formation of SVD

As singular vectors are eigenvectors of symmetric matrices A^TA, by applying Spectral Theorem, we find that singular vectors of A would all be orthonormal.
Then, let's use this point as an opportunity to praise A^TA's symmetricness:

Theorem 5.1.1. The Formation of SVD

Let the nonzero eigenvalues of A^TA be

$$\lambda_1 \geq \cdots \geq \lambda_r > 0$$

where, upon dimming that $rk(A^TA) = r$, we may obtain $rk(A) = rk(A^TA) = r$ (using rank-nullity theorem, and the fact that $N(A^TA) = N(A)$).
Let us also suppose that orthonormal vectors in V form eigenpairs with the above eigenvalues:

$$\forall i \in [1, r], A^TA\vec{v_i} = \lambda_i\vec{v_i}$$

Now, define that,

$$\sigma_i := \sqrt{\lambda_i}$$

such that:

$$\vec{u_i} := \frac{A\vec{v_i}}{\sigma_i}$$

Let us now check, whether the above formation grants an orthonormal U:

Theorem 5.1.2. Orthonormality of Set of $\vec{u_i}$

Let us follow the definition of $\vec{u_i}$ from the last block.
Then,

$$A\vec{v_i} = \sigma_i\vec{u_i}$$

Suppose now we have some vectors $\vec{u_i}$ and $\vec{u_j}$, then

$$\begin{aligned}\vec{u_i} \cdot \vec{u_j} &= \frac{\vec{v_i}^TA^T}{\sigma_i}\frac{A\vec{v_j}}{\sigma_j} \\ &= \frac{\vec{v_i}^T\lambda_j\vec{v_j}}{\sigma_i\sigma_j} = 0\end{aligned}$$

Now that we have proven the orthonormality and note the definitions of $\vec{u_i}$ and $\vec{v_i}$, we arrive at the conclusion:

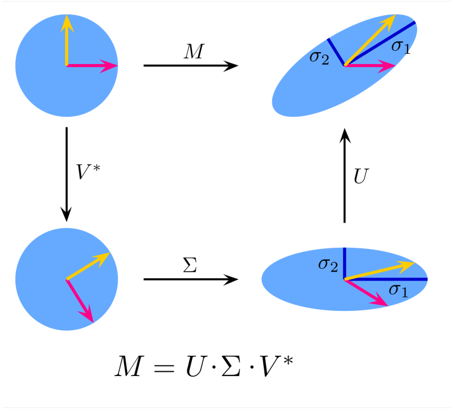
$$AV = U\Sigma, A = U\Sigma V^T$$

such arithmetic operation is only possible under the full SVD form (such that V can be noted as orthonormal at all). This is once again a demonstration of significance of full SVD.

5.1.2 Geometry of SVD

Figure 5.1.1. Geometric Property of SVD as Transformation

We may see from the below image's operations,



that SVD is, as priorly mentioned, a combination of three transformations. Where,

- The orthonormal matrices U, V are reflections or rotations.
- The matrix Σ is a dilation.

Furthermore, let us observe the vectors drawn on the unit circle.
Since V is orthonormal, we may conclude that $\vec{v_1} \perp \vec{v_2}$. However, bizzarely, we may also find that

$$A\vec{v_1} \perp A\vec{v_2}$$

5.2 Low Rank Approximation

Low Rank Approximation is the task of approximating a matrix A in a lower rank than $rk(A)$, to conserve computational cost and perform efficient computations.

5.2.1 Matrix Norms

Let us first discuss the optimization function of this task: matrix norm.

The norm of matrix also comes in diverse form, because matrices can be interpreted in many forms.
For example:

- Chunk of data (like an array)
- Operator of transformation

Therefore, along these interpretations, we may present respective definitions of matrix norms.

Definition 5.2.1. Frobenius Norm

The Frobenius Norm is defined as:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{i,j}^2} = \sqrt{Trace(A^T A)}$$

This is because the Frobenius inner product may be defined as:

$$\langle A, B\rangle_F = Trace(A^T B) = \sum_{i=1}^{n} \vec{A}_i \cdot \vec{B}_i$$

and, as we recognize the relation between norm and inner product,

$$\|A\|_F = \sqrt{\langle A, A\rangle_F} = \sqrt{\sum_{i=1}^{n} \vec{A}_i \cdot \vec{A}_i}$$

The Frobenius Norm has some interesting properties:

Theorem 5.2.1. Invariance of Frobenius Norm upon Orthonormal Matrices

Let $U_1 \in \mathbb{R}^{m\times m}$, $A \in \mathbb{R}^{m\times n}$, and $U_2 \in \mathbb{R}^{n\times n}$, where U_1, U_2 are orthonormal.
Then,

$$\begin{aligned}
\|U_1 A\|_F &= \sqrt{Trace(A^T U_1^T U_1 A)} \\
&= \sqrt{Trace(A^T A)} = \|A\|_F \\
\|AU_2\|_F &= \sqrt{Trace(U_2^T A^T A U_2)} \\
&\underset{\text{Cyclic Property}}{=} \sqrt{Trace(A^T A)} = \|A\|_F
\end{aligned}$$

Now, let's discuss another definition of matrix norm:

Definition 5.2.2. Operator Norm (aka. Spectral Norm, L2-Norm)

Such norm is defined as:

$$\|A\|_2 = \max_{\|\vec{x}\|_2=1} \|A\vec{x}\|_2 = \sigma_{max}(A)$$

The derivation follows:

$$\begin{aligned}
\max_{\|\vec{x}\|_2=1} \|A\vec{x}\|_2 &= \max_{\|\vec{x}\|_2=1} \sqrt{\vec{x}^T A^T A\vec{x}} \\
&\underset{\text{Rayleigh Coefficient}}{=} \sqrt{\lambda_{max}(A^T A)} = \sigma_{max}(A)
\end{aligned}$$

Alternatively, it may be stated as the maximum possible transformation some matrix A can offer a normal vector. Upon that above statement,

$$\forall \vec{x}, \|A\vec{x}\|_2 \leq \|A\|_2 \|\vec{x}\|_2$$

Furthermore, L2-Norms are also invariant upon orthonormal matrices.
Suppose we attach another orthonormal matrix U to the original matrix A, then the optimization problem develops a solution fairly similaly:

$$\begin{aligned}
\max_{\|\vec{x}\|_2=1} \|AU\vec{x}\|_2 &= \max_{\|\vec{x}\|_2=1} \sqrt{\vec{x}^T U^T A^T A U\vec{x}} \\
&= \max_{\|\vec{x}\|_2=1} \sqrt{\vec{y}^T A^T A\vec{y}} \\
&= \sqrt{\lambda_{max}(A^T A)} = \sigma_{max}(A) = \|A\|_2
\end{aligned}$$

where since we still find that $\vec{y}$ is normal, the attachment of U providing no change to the optimization problem. Meanwhile, we find this phenomenon to persist even if U is multiplied from an opposite direction:

$$\begin{aligned}\max_{\|\vec{x}\|_2=1} \|UA\vec{x}\|_2 &= \max_{\|\vec{x}\|_2=1} \sqrt{\vec{x}^T A^T U^T U A\vec{x}} \\ &= \max_{\|\vec{x}\|_2=1} \sqrt{\vec{x}^T A^T A\vec{x}} \\ &= \sqrt{\lambda_{max}(A^T A)} = \sigma_{max}(A) = \|A\|_2\end{aligned}$$

Therefore, we may also observe similar properties in Frobenius Norm, where:

$$\|A\|_2 = \|U_1 A\|_2 = \|AU_2\|_2 \text{ for orthonormal matrices } U_1, U_2$$

Chapter 6

Low-Rank Approximation

6.1 Discussion of L-RA

Suppose we have some matrix $A \in \mathbb{R}^{m\times n}$, where storing this matrix would take significant computational resources. Is there a method to store an approximation of this matrix, such that the approximation is more efficient to store for, and we would thus reduce computational cost of storage?

We have discussed the possibility of such technique within the previous lecture, where we have solved the optimization problem:

$$\max_{\|\vec{x}\|_2=1} \|A\vec{x}\|_2$$

Today, we will discuss the extension of such solution: Low-Rank Approximation.

LRA would be characterized from the optimization problem:

$$\underset{B\in\mathbb{R}^{m\times n}, rk(B)=k}{\text{argmin}} \|A-B\|_F$$

where we may see a contextually similar optimization problem:

$$\underset{B\in\mathbb{R}^{m\times n}, rk(B)=k}{\text{argmin}} \|A-B\|_2$$

Let us solve the low-rank approximation of matrices based on their Frobenius norm and L2 norm.

6.1.1 The LRA Optimization Problem on Spectral Norm

Problem.

$$\underset{B\in\mathbb{R}^{m\times n}, rk(B)=k}{\text{argmin}} \|A-B\|_2$$

Solution.
Keep in mind that our proof proceeds in the direction of:

> - Finding a possible upper bound
> - Finding the achievability of such upper bound
> - Finding the legitimacy of such upper bound

Now, let us move on with the subparts of a proof.

Finding a Possible Upper Bound.
Let us define from the SVD of A:

$$A_k = \sum_{i=1}^{k} \sigma_i \vec{u_i} \vec{v_i}^T$$

Then, the difference between such approximation and A would be:

$$\|A - A_k\|_2 = \|A_n - A_k\|_2 = \left\| \sum_{i=k+1}^{n} \sigma_i \vec{u_i} \vec{v_i}^T \right\|_2$$

which, following along the optimization problem phrased last lecture, we would obtain the solution to be:

$$\left\| \sum_{i=k+1}^{n} \sigma_i \vec{u_i} \vec{v_i}^T \right\|_2 = \sigma_{k+1}(A)$$

Finding the Legitimacy (and Achievability) of Upper Bound.
Then, let us show that for every B where $rk(B) \leq k$, the difference $\|A - B\|_2$ is larger than our current possible minimum, σ_{k+1}. By proving so we secure the result of maximization is as illustrated above.
Let us first define $\vec{w}$ that,

$$\|A - B\|_2 = \max_{\|\vec{w}\|=1} \|(A - B)\vec{w}\|_2$$

Where, if $\vec{w} \in N(B)$, then we would be able to simplify the above optimization problem into something easier to solve: along such above constraint, let us note that,

$$\|A - B\|_2^2 \geq \|(A - B)\vec{w}\|_2^2 = \|A\vec{w}\|_2^2$$

And decomposing A into SVD form, which contains orthonormal matrices,

$$\|A - B\|_2^2 \geq \|U\Sigma V^T \vec{w}\|_2^2 = \|\Sigma V^T \vec{w}\|_2^2$$

Provided that $\vec{w}$ is a unit vector, and V is invertible, we can guarantee that

$$\exists \vec{a}, V\vec{a} = \vec{w}$$

Therefore, let us now substitute such value in, and we will obtain that

$$\begin{aligned} \|A - B\|_2^2 &\geq \|\Sigma V^T \vec{w}\|_2^2 \\ &= \|\Sigma \vec{a}\|_2^2 \\ &= \vec{a}^T \Sigma^T \Sigma \vec{a} = \vec{a}^T \Lambda \vec{a} \end{aligned}$$

Back to the Achievability of Upper Bound, Final Step.
Let us come back to our assumption that $\vec{w} \in N(B)$, and extract more insights from it.
Now, we may define along the V of $A = U\Sigma V^T$ that:

$$V_{k+1} = \begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_{k+1} \end{bmatrix}$$

where along the property of SVD we understand that $rk(V_{k+1}) = k + 1$.
To find some $\vec{w}$ such that $\vec{w} \in N(B) \land \vec{w} \in \mathcal{R}(V_{k+1})$, we should notice that:

1. Along rank-nullity theorem, $\dim(N(B)) \geq n - k$.

2. Along above description, $\dim(\mathcal{R}(V_{k+1})) = k + 1$

At this point, we must note that the achievability in our proposed upper bound exists in the possibility of having such $\vec{w}$. Therefore, **the achievability of upper bound is protected by the existence of $\vec{w}$, which we must guarantee**. These two subspaces of $\mathbb{R}^n$ must have overlap. Therefore, there must exist some vector that belongs to both subspaces. This guarantees the existence of our $\vec{w}$.
Let us now follow along, and revisit the derivation that $\vec{a} = V^T\vec{w}$ on the change of basis.
Then, here we perform the final computations:

$$\begin{aligned}
\vec{a} &= V^T\vec{w} \\
&= V^T(\sum_{i=1}^{k+1} a_i\vec{V_i}) \\
&= \begin{bmatrix} a_1 & \cdots & a_{k+1} & 0 & \cdots & 0 \end{bmatrix}^T \\
\|A-B\|_2^2 \geq \vec{a}^T\Lambda\vec{a} &= \sum_{i=1}^{k+1} a_i^2\Lambda_i \\
&\geq \sum_{i=1}^{k+1} a_i^2\Lambda_{k+1} = \Lambda_{k+1}\sum_{i=1}^{k+1} a_i^2 \\
&= \Lambda_{k+1} = \sigma_{k+1}^2
\end{aligned}$$

Note, $\|\vec{a}\|_2 = \|\vec{w}\|_2 = 1$, which allows us to use the analysis of Rayleigh Coefficient on the last aligned equation. We have thus established that the minimal possible expression of our minimization problem is σ_{k+1}^2

6.1.2 The LRA Optimization Problem on Frobenius Norm

Problem.

$$\underset{B\in\mathbb{R}^{m\times n}, rk(B)=k}{\text{argmin}} \|A-B\|_F$$

Solution.
Let us consider A_k as priorly defined to be the solution of optimization again.

$$\begin{aligned}
\|A\|_F^2 &= \|U\Sigma V^T\|_F^2 \\
&= \|\Sigma\|_F^2 = \sum_{i=1}^{n} (\sigma_i(A))^2 \\
\|A-A_k\|_F^2 &= \sum_{i=k+1}^{n} (\sigma_i(A))^2 \\
\|A-B\|_F^2 &= \sum_{i=1}^{n} (\sigma_i(A-B))^2
\end{aligned}$$

Note that in above, we are expressing the singular values of difference matrices, not multiplying a singular value by a matrix.

Now, following the logic before, we want to prove a lowerbound:

> Prove that $\forall B \in \mathbb{R}^{m\times n}\ s.t.\ rk(B) = k, \|A-B\|_F \geq \|A-A_k\|_F$

Using the aligned equations above, let us find an equivalent proof prompt,

> Prove that $\forall i, \sigma_i(A-B) \geq \sigma_{k+i}(A)$

Note once again that these are singular values of different matrices. Where, along spectral norm's definition, we can also obtain that,

$$\sigma_{k+i}(A) = \|A - A_{k+i-1}\|_2$$

translated in text stating that A_{k+i-1} is the best rank $(k+i-1)$ approximation to A in spectral norm sense.
Now, let us return to the above equivalent prompt and define such that $C := A - B$.

$$\begin{aligned}\sigma_i(A-B) &= \sigma_i(C)\\ &= \|C - C_{i-1}\|_2 = \|A - B - C_{i-1}\|_2\end{aligned}$$

Pay attention to the rank of these matrices.

> - B must be some rank-k (or less) matrix, as we are looking for B to be a k-rank approximation of A.
> - C_{i-1} would have some rank less than $i-1$.
> - Define the combination $D = B + C_{i-1}$, $rank(D) \leq k+i-1$

We would thus reuse the equation addressed before and state,

$$\sigma_i(A-B) = \sigma_i(A-D)$$

Using the solution from spectral norm side proof of LRA, for any matrix $D = B + C_{i-1}$ with at most rank $k+i-1$,

$$\begin{aligned}\sigma_i(A-B) &= \|A - (B + C_{i-1})\|_2\\ &\geq \|A - A_{k+i-1}\|_2 = \sigma_{k+i}(A)\end{aligned}$$

Chapter 7

Vector Calculus

The motivation of vector calculus is to work with linear algebra problems more analytically, in more interpretable methods.
The tools that help to optimize linear algebra expression is the calculus of vector: vector calculus.

7.1 Function Expansion: Taylor Series

Let's start with functions, the basic of calculus.

Definition 7.1.1. Scalar Valued Function of Vector

Let a function f be such that:

$$f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$$

Such a function is by definition a scalar valued function of vector.

Then, we may begin our effort to generalize the derivative of such functions, as follows.

First of all, let us recap Calculus II knowledge of polynomial expansion of some fucntion:

Theorem 7.1.1. Taylor's Theorem (Expansion)

Let f be a scalar valued function for scalar:

$$f(x) : \mathbb{R} \to \mathbb{R}, x_0 \in \mathbb{R}$$

Then, Taylor expansion allows us to write such that:

$$f(x_0 + \Delta x) = f(x_0) + \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x=x_0} \Delta x + \cdots$$

Essentially,

$$f(x_0 + \Delta x) = \sum_{i=0}^{n} \frac{f^{(i)}(x_0)(\Delta x)^i}{i!}$$

where we call the Taylor expansion towards the n^{th} degree term be an n^{th} order Taylor expansion.

What is the purpose of introducing Taylor's expansion here? The fact is, we may approximate a value $f(x_0 + \Delta x)$ based on $f(x)$ along the product of Δx and a derivative. **The derivative affects how large the "perturbation" is that occurs in the approximation of** $f(x_0 + \Delta x)$. Similar logic is applied for upcoming n-order derivatives.

Then, along the above reintroduction of Taylor's Series, let us look at a vector edition of Taylor's Theorem:

Theorem 7.1.2. Taylor's Theorem for Vectors

Let f be a scalar valued function for vector.
Then, we may express that:

$$f(\vec{x_0}+\Delta\vec{x}) = f(\vec{x_0}) + \frac{\partial f}{\partial x}\bigg|_{\vec{x}=\vec{x_0}} \Delta\vec{x} + \frac{1}{2!}(\Delta\vec{x})^T \nabla^2 f(\vec{x})\bigg|_{\vec{x}=\vec{x_0}} \Delta\vec{x}$$

We usually stop at the second order approximation when using Taylor's Theorem for vectors, because this is usually a good equilibrium point for computational precision and simplification.

7.2 Function Expansion: Derivative of Vector Functions

In addition, we define the derivative of vectors as follows:

Definition 7.2.1. Derivatives of Scalar Valued Vector Function

Here, we observe that the first order derivative of such functions are row vectors, and the second order derivative is a matrix called **Hessian**:

$$\frac{\partial f}{\partial x} = (\nabla_{\vec{x}} f)^T = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$
$$(\nabla^2 f(\vec{x}))_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$
$$\nabla^2 f(\vec{x}) \in \mathbb{S}^n$$

Note that it is only for convex functions where the symmetric argument of Hessian applies.
This is beacuse:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{}{\partial x_i}\frac{\partial f}{\partial x_j}$$

where the order of variables put matters for functions.
The reason why convex functions can host the symmetric Hessian is because of Clairaut's Theorem (not explicitly mentioned in lecture):

Theorem 7.2.1. Clairaut's Theorem

If the second partial derivatives of a function are continuous, then the order of differentiation is immaterial.
Alternatively (quoted from Purdue University),

> Let $f : \mathbb{R}^2 \to \mathbb{R}$ have all partial derivatives up to second derivative be continuous near (a, b), then:
> $$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$$

7.2.1 Examples of Polynomial Expansion

Now, let us provide an example in approximating a scalar valued vector function via Taylor's Theorem:

Explain 7.2.1. Polynomial Expansion of 2-Norm

Let us attempt to find a Taylor Expansion for the function

$$f(\vec{x}) = \|\vec{x}\|_2^2$$

We will define the level sets of this function as:

$$\{\vec{x}|f(\vec{x}) = C\}$$

which would be circles centered at the origin with radius $\sqrt{C}$.
The gradient of such function would be:

$$\nabla_{\vec{x}} f = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$$

And the Hessian:

$$H_{\vec{x}} f = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Therefore, upon Taylor's Theorem,

$$\begin{aligned} f(\vec{x_0} + \Delta\vec{x}) &= f(\vec{x_0}) + (\nabla_{\vec{x}} f)^T \Delta\vec{x} + \frac{1}{2!}(\Delta\vec{x})^T H_{\vec{x}} f \Delta\vec{x} \\ &= x_{01}^2 + x_{02}^2 + 2x_{01}\Delta x_1 + 2x_{02}\Delta x_2 + (\Delta x_1)^2 + (\Delta x_2)^2 \\ &= (x_{01} + \Delta x_1)^2 + (x_{02} + \Delta x_2)^2 = \|\vec{x_0} + \Delta\vec{x}\|_2^2 \end{aligned}$$

Since the Hessian is symmetric, we can perform a lot of interesting mathematics with it.
A third order derivative is known as a tensor, but this is out of scope for EECS 127.

Let's look at some more examples of Taylor Expansion:

Explain 7.2.2. Polynomial Expansion of Euclidean Inner Product

Let us find a Taylor Expansion to the function:

$$f(\vec{x}) = \vec{x}^T \vec{a}, \vec{a} \in \mathbb{R}^n$$

The gradient of such function is then,

$$\nabla_{\vec{x}} f = \vec{a}$$

Then, the Hessian of such function would be a zero matrix.
Therefore,

$$\begin{aligned} f(\vec{x_0} + \Delta\vec{x}) &= f(\vec{x_0}) + (\nabla_{\vec{x}} f)^T \Delta\vec{x} \\ &= \sum_{i=1}^{n} a_i(x_{0i} + \Delta x_i) = (\vec{x_0} + \Delta\vec{x})^T \vec{a} \end{aligned}$$

7.2.2 Matrix in Vector Calculus

Let us discuss how the roles of matrices are when we take the derivative of a scalar-valued function by a vector.

Explain 7.2.3. Derivatives of Matrix-Involving Inner Product

Let us find the derivative expressions to the function:

$$f(\vec{x}) = \vec{x}^T A\vec{x}, A \in \mathbb{R}^{n\times n}$$

We can derive the gradient as follows:

$$\begin{aligned}
\frac{\partial}{\partial x_i}\vec{x}^T A\vec{x} &= \frac{\partial}{\partial x_i}\vec{x}^T \begin{bmatrix}\vec{A_1} & \cdots & \vec{A_n}\end{bmatrix}\vec{x} \\
&= \frac{\partial}{\partial x_i}\begin{bmatrix}\vec{x}\cdot\vec{A_1} & \cdots & \vec{x}\cdot\vec{A_n}\end{bmatrix}\vec{x} \\
&= \frac{\partial}{\partial x_i}\sum_{k=1}^{n}\sum_{j=1}^{n} A_{j,k}x_k x_j \\
&= \frac{\partial}{\partial x_i}\left(\sum_{j=1}^{n} A_{j,i}x_i x_j + \sum_{k=1}^{n} A_{i,k}x_i x_k\right) \\
&= \sum_{j=1}^{n} A_{j,i}x_j + \sum_{k=1}^{n} A_{i,k}x_k = (\vec{A^T})_i\cdot\vec{x} + \vec{A}_i\cdot\vec{x} \\
\frac{\partial}{\partial x}\vec{x}^T A\vec{x} &= (A + A^T)\vec{x}
\end{aligned}$$

And, uh, Hessians.

$$\nabla^2 f(\vec{x}) = \frac{\partial}{\partial x}(A + A^T)\vec{x} = A + A^T$$

Then, let us define the derivative of a vector-valued function with respect to some vector variable:

Definition 7.2.2. Jacobian (Derivative) of Vector-Valued Vector Function

For some function $f(\vec{x}) = \vec{y}$ such that:

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

Then,

$$J = \begin{bmatrix}\frac{\partial\vec{f}}{\partial x_1} & \cdots & \frac{\partial\vec{f}}{\partial x_n}\end{bmatrix}$$

Explain 7.2.4. The Jacobian of Polar-Cartesian Coordinate Translator

Let f be the function:

$$f(\vec{v}) = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta)\end{bmatrix}$$

Then the Jacobian of it would be:

$$J = \begin{bmatrix}\cos(\theta) & -r\sin(\theta) \\ \sin(\theta) & r\cos(\theta)\end{bmatrix}$$

whose determinant would then be $\det(J) = r$.

Last but not least, let us explore chain rule in matrix calculus:

Definition 7.2.3. Chain Rule

Let a function f be

$$f(\vec{x}) = g(h(\vec{x}))$$

Then, we may express that

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}g}{\mathrm{d}y}\frac{\mathrm{d}h}{\mathrm{d}x}$$

Example.
Now, let's look at the least squares problem.
Originally, we perform this computation:

$$\begin{aligned}
f(x) &= \|A\vec{x} - \vec{b}\|_2^2 \\
&= (A\vec{x} - \vec{b})^T((A\vec{x} - \vec{b})) \\
&= \vec{x}^T A^T A\vec{x} - 2\vec{x}^T A\vec{b} + \vec{b}^T\vec{b} \\
\nabla_{\vec{x}} f(\vec{x}) &= (A^T A + A^T A)\vec{x} - 2A^T\vec{b} \\
&= 2A^T A\vec{x} - 2A\vec{b} \\
\vec{x}^* &= (A^T A)^{-1} A^T\vec{b}
\end{aligned}$$

Instead, we may use chain rule:

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}x} &= \frac{\mathrm{d}g}{\mathrm{d}y}\frac{\mathrm{d}h}{\mathrm{d}x} \\
&= (\nabla_{\vec{y}} g)^T \frac{\mathrm{d}}{\mathrm{d}x}(A\vec{x}) \\
&= 2(A\vec{x} - \vec{b})^T \cdot A
\end{aligned}$$

We may thus obtain the gradient to be:

$$\nabla_{\vec{x}} f = 2A^T(A\vec{x} = \vec{b})$$

Chapter 8

The Extension of Vector Calculus

Note: This lecture's content will be shorter than other notes because half of the lecture was put on reviewing Lecture 7

8.1 The Main Theorem

The. Main.

Theorem 8.1.1. The Main Theorem

Theorem. Let Ω be an open subset of $\mathbb{R}^n$, and let function f be differentiable and such that

$$f : \mathbb{R}^n \to \mathbb{R}$$

then, for an optimal solution $\vec{x}^*$ that solves the optimization problem:

$$\min_{\vec{x}\in\Omega} f(\vec{x})$$

Then, the Main Theorem states that,

$$\frac{\mathrm{d}f}{\mathrm{d}\vec{x}}\Big|_{\vec{x}=\vec{x}^*} = 0$$

Proof. We realize that Ω is an open set, meaning $\vec{x}^* + \Delta\vec{x}$ can be guaranteed to be involved in Ω, for the definition of an open set is such that,

$$\forall x \in \Omega, \exists \epsilon > 0 \; (|x - y| < \epsilon \implies y \in \Omega)$$

Then, by the Taylor expansion and optimality assumption, we may state that:

$$f(\vec{x}^* + \Delta\vec{x}) = f(\vec{x}^*) + \frac{\mathrm{d}f}{\mathrm{d}x}\bigg|_{\vec{x}=\vec{x}^*}(\Delta\vec{x}) + \frac{1}{2}(\Delta\vec{x})^T\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\bigg|_{\vec{x}=\vec{x}^*}(\Delta\vec{x}) \geq f(\vec{x}^*)$$

Manipulating the above inequality,

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}x}\bigg|_{\vec{x}=\vec{x}^*}(\Delta\vec{x}) + \frac{1}{2}(\Delta\vec{x})^T\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\bigg|_{\vec{x}=\vec{x}^*}(\Delta\vec{x}) &\geq 0 \\
\frac{\mathrm{d}f}{\mathrm{d}x}\bigg|_{\vec{x}=\vec{x}^*} + \frac{sum(H.O.T.)}{\Delta\vec{x}} &\geq 0 \\
\lim_{\Delta\vec{x}\to 0}\left(\frac{\mathrm{d}f}{\mathrm{d}x}\bigg|_{\vec{x}=\vec{x}^*} + \frac{sum(H.O.T.)}{\Delta\vec{x}}\right) = \frac{\mathrm{d}f}{\mathrm{d}x}\bigg|_{\vec{x}=\vec{x}^*} &\geq 0
\end{aligned}$$

We may then provide a symmetric argument on the value $\vec{x}^* - \Delta\vec{x}$:

$$\begin{aligned} \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*}(-\Delta\vec{x}) + \frac{1}{2}(-\Delta\vec{x})^T \left.\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right|_{\vec{x}=\vec{x}^*}(-\Delta\vec{x}) && \geq 0 \\ -\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*} + \frac{sum(H.O.T.)}{\Delta\vec{x}} && \geq 0 \\ \lim_{\Delta\vec{x}\to 0} -\left(\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*} + \frac{sum(H.O.T.)}{\Delta\vec{x}}\right) = -\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*} && \geq 0 \end{aligned}$$

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*} \leq 0$$

Consequently, we reach the following conclusion:

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*} \geq 0 \wedge \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*} \leq 0 \implies \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{\vec{x}=\vec{x}^*} = 0$$

Note: H.O.T. means “Higher Order Terms” of Taylor Expansion, starting from the second-order term

8.2 Perturbation Analysis, Effect of Noise

In this problem, we consider the following concern:

> Let $A\vec{x} = \vec{y}$, where A is a square invertible matrix.
> Then, for some change in $\vec{y}$ (characterized as $\Delta\vec{y}$), how would this affect $\vec{x}$?
>
> In other words, we compare $\frac{\|\Delta\vec{x}\|_2}{\|\vec{x}\|_2}$ with $\frac{\|\Delta\vec{y}\|_2}{\|\vec{y}\|_2}$

Let us now set up the problem, by priorly noting what we know about the relationship between matrices, measurements, and perturbations:

$$\begin{aligned} A\vec{x} &= \vec{y} \\ A(\vec{x} + \Delta\vec{x}) &= \vec{y} + \Delta\vec{y} \end{aligned}$$

Then, we may manipulate the expressions to find that,

$$\begin{aligned} A\Delta\vec{x} = \Delta\vec{y} \implies \Delta\vec{x} &= A^{-1}\Delta\vec{y} \\ \|\Delta\vec{x}\|_2 &= \|A^{-1}\Delta\vec{y}\|_2 \leq \|A^{-1}\|_2\|\Delta\vec{y}\|_2 \end{aligned}$$

The last line is certified because the spectral norm of A^{-1} measures how much can it transform the norm of a vector.

Then, we may provide a minimization problem to attempt find some expression about the ratio of perturbation and actual measurement:

$$\min \frac{\|\Delta\vec{x}\|_2}{\|\vec{x}\|_2}$$

Now, let us observe the following work of upperbounding such denominator:

$$\begin{aligned} A\vec{x} = \vec{y} \implies \|A\vec{x}\|_2 &\leq \|A\|_2\|\vec{x}\|_2 \\ \|\vec{y}\|_2 &\leq \|A\|_2\|\vec{x}\|_2 \\ \frac{1}{\|\vec{x}\|_2} &\leq \frac{\|A\|_2}{\|\vec{y}\|_2} \end{aligned}$$

And therefore,

$$\begin{aligned}\frac{\|\Delta\vec{x}\|_2}{\|\vec{x}\|_2} &\leq \|A^{-1}\|_2\|\vec{\Delta y}\|_2\frac{\|A\|_2}{\|\vec{y}\|_2}\\ &= \|A^{-1}\|_2\|A\|_2\frac{\|\Delta\vec{y}\|_2}{\|\vec{y}\|_2}\\ &= \sigma_{\max}(A)\sigma_{\max}(A^{-1})\frac{\|\Delta\vec{y}\|_2}{\|\vec{y}\|_2} = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}\frac{\|\Delta\vec{y}\|_2}{\|\vec{y}\|_2}\end{aligned}$$

Consequently, we arrive at the conclusion (summary) of:

$$\frac{\|\Delta\vec{x}\|_2}{\|\vec{x}\|_2} \leq \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}\frac{\|\Delta\vec{y}\|_2}{\|\vec{y}\|_2}$$

where we call the fraction $\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$ the **condition number**.

Chapter 9

Ridge Regression

9.1 Perturbation Analysis Guides into Ridge Regression

In the concept of perturbation analysis, we ask that, for some system

$$A\vec{x} = \vec{y}$$

with a square, invertible A, how much would $\vec{x}$ change provided some small change $\vec{y} \rightarrow \vec{y} + \vec{\partial y}$?

Then, our solution (cited in Chapter 8, or Lecture 8) is as follows:

$$\frac{\|\vec{\partial x}\|_2}{\|\vec{x}\|_2} \leq \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \frac{\|\vec{\partial y}\|_2}{\|\vec{y}\|_2}$$

From whcich, we discover the characteristic of a matrix called **condition number**:

$$\frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

Let us connect the above discussion to the Least Squares Problem.
In the context of least squares problem, we are provided a more robust normal equation via the use of pseudoinverse:

$$(A^{\dagger}A)\vec{x} = A^T\vec{b}$$

As our least square system is sensitive to noise, we may investigate the condition number of A^TA to observe the system's change in its solution provided the perturbation in system measurements.
This is significant in computations. A high condition number makes a system's matrix highly unstable for numerical precisions to persist provided noise in measurements and systems. Such property is, in fact, demonstrated by the convergence warnings in Jupyter iPython notebooks!

But, provided the significance of condition number, we may also discuss ways to alleviate its problems. To reduce the condition number of some matrix, we may attempt to reduce the ratio of singular values via adding some multiple of identity matrix (λI) to it. This allows us to greatly reduce the condition number, shifting it away from instability and being less prone to variance in training data:

$$A^TA + \lambda I$$

As a side note, in CS189, we discover similar approaches that can help tackle training data that leads to singular covariance matrices.
Let us investigate below why adding a multiple of diagonal matrix does not make our regression problem have a very deviated solution (despite the fact we somehow alter the design matrix of our problem).

9.2 Ridge Regression

In the least squares problem, we have a formulated optimization problem of:

$$\min_{\vec{x}} \|A\vec{x} - \vec{b}\|_2$$

to offer insight to the optimization problem to prevent its divergence from true parameter value, is to present a new formulation of the problem:

$$\min_{\vec{x}} \|A\vec{x} - \vec{b}\|_2^2 + \lambda^2\|\vec{x}\|_2^2$$

where, we regularize the least squares solution by stating that large solutions of $\vec{x}$ must lead to unidealistic increase in the cost (loss) function. We penalize a large least squares solution $\vec{x}$.

Here, using different norms in the regularizer will provide different properties. The L1 regularizer has the LASSO property (as outlined in DATA C100), while the L2 regularizer we use now provides a convex function as well as a closed-form solution unlike the L1 effort (once again, as outlined in DATA C100).

Let us first compute the gradient of such loss function:

$$\begin{aligned}\nabla_{\vec{x}}\|A\vec{x} - \vec{b}\|_2^2 + \lambda^2\|\vec{x}\|_2^2 &= 2A^T(A\vec{x} - \vec{b}) + 2\lambda^2 I\vec{x} \\ \vec{x}^* &= (A^TA + \lambda^2 I)^{-1}A^T\vec{b}\end{aligned}$$

The eigenvalues of $A^TA + \lambda^2 I$, meanwhile, would be the eigenvalues of A^TA added the value λ^2 (by manipulating the definition of eigenvalues and eigenvectors).
This is also known as the shift property of eigenvalues:

Theorem 9.2.1. Shift Property of Eigenvalues

Let $\vec{v}$ be an eigenvector of A^TA.
Then, provided that:

$$A^TA\vec{v} = \mu\vec{v}$$

we may see that,

$$(A^TA + \lambda^2 I)\vec{v} = (\mu + \lambda^2)\vec{v}$$

Thus determine that:

> for any eigenpair of A^TA being $(\mu, \vec{v})$, a corresponding eigenpair exists in $A^TA + \lambda^2 I$ being $(\mu + \lambda^2, \vec{v})$.

Therefore, small λ corresponds to less regularizaiton effort, and vice versa.

9.2.1 Development of Ridge Regression

Now, suppose we have a least square system where $\vec{x}$ has a smaller norm, where $\lambda I\vec{x} \sim \vec{0}$.
Then, by that close-to-zero property we observe in $\lambda I\vec{x}$, adding the information of λI into the least squares problem formulation would barely alter the original formulation too much, due to the close-to-zero-ness of ridge regression's current least squares solution.
This allows us to use larger λ (have greater regularization).
We have two ways of incorporating such information into the least squares problem now:

- Vertically concatenate λI below A.
- Use Ridge Regression's format, which uses a similar idea to preserve the original system as much as possible. We will prove later that this is exactly the first idea.

In the concatenation idea, our system of least squares problem is reformulated as:

$$\begin{bmatrix} A \\ \lambda I \end{bmatrix} \vec{x} = \begin{bmatrix} \vec{b} \\ \lambda I \vec{x} \end{bmatrix}$$

Let's use both block matrix and normal equation to further explore the idea of concatenating λI:

$$\begin{aligned} &\left(\begin{bmatrix} A^T & \lambda I \end{bmatrix} \begin{bmatrix} A \\ \lambda I \end{bmatrix} \right)^{-1} \begin{bmatrix} A^T & \lambda I \end{bmatrix} \begin{bmatrix} \vec{b} \\ \lambda I \vec{x} \end{bmatrix} \\ &= (A^T A + \lambda^2 I)(A^T \vec{b} + \lambda^2 I \vec{x}) \\ &\sim (A^T A + \lambda^2 I) A^T \vec{b} \end{aligned}$$

and we have therefore demonstrated the similarity between the two ideas of ridge regression.
It is noteworthy that the common notation of ridge regression does not use the notation λ^2 when addressing regularization parameter. We are doing so in the context of demonstrating how ridge regression is developed, through a simpler set of mathematical notations.

Once we expand the idea to adding weights for matrices, creating the system:

$$\begin{bmatrix} W_1 A \\ W_2 I \end{bmatrix} \vec{x} = \begin{bmatrix} W_1 \vec{b} \\ W_2 \vec{x_0} \end{bmatrix}$$

we end up with a new optimization problem:

$$\min_{\vec{x}} \|W_1(A\vec{x} - \vec{b})\|_2^2 + \|W_2(\vec{x} - \vec{x_0})\|_2^2$$

which we call the **Tikhonov Regularization** technique.
Note: $\vec{x_0}$ is an arbitrary piece of information.

9.3 Probabilistic Information from Ridge Regression

Suppose that we instead now have probabilistic information:

$$(\vec{x_i}, y_i), \text{ where } y_i = g(x_i) + z_i$$

and, $z_i \sim \mathcal{N}(0, \sigma_i^2)$ is a noise defined on a Gaussian distribution.
And, suppose we have a linear model,

$$y_i = \vec{w}^T \vec{x} + z_i$$

Then, we may state that,

$$f_{z_i}(z_i) = \frac{\exp\left(-\frac{z_i^2}{2\sigma_i^2}\right)}{\sqrt{2\pi}\sigma_i}$$

and we attempt to learn the weights $\vec{w}$ to complete the model for some related problem.
Therefore, with multivariate Gaussian noise $\vec{z}$ and datapoint matrices X, we formulate this as a least square system,

$$X\vec{w} + \vec{z} = \vec{y}$$

9.3.1 Maximum Likelihood Estimation and Maximum A-Posteriori

Now, we may attempt to estimate the parameters that makes the observed data most likely.
This is related to the technique of Maximum Likelihood Estimation. Let me shamelessly copy a segment of my CS189 notes here to provide a brief explanation:

Explain 9.3.1. MLE from CS189

Suppose that we go back to the coin flip example (just like 126 does), where heads appear with a probability p (and otherwise for tails).
Then, statisticians world ask, provided the real data of coin, what value of p (the parameter of coin flip probability distribution) is closest to its true inherent value.
Let us suppose that the number of heads we obtain is a discrete random variable, $X \sim Binomial(n, p)$:

$$\mathbb{P}(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$$

Then, let us propose that the real data presents k heads, and we would defined the Likelihood function $\mathcal{L}$ as:

$$\mathcal{L}(p) = \mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

where it is a function of distribution parameters.

Then, Maximum Likelihood Estimation (MLE) is the method of estimating the parameters of a statistical distribution by picking the parameters that maximize $\mathcal{L}$. Furthermore, it would be a method of density estimation, where we estimate some probability density function from the provided dataset.

In this case, we are performing MLE to obtain weight $\vec{w}$ provided the Gaussian noise $\vec{z}$:

Explain 9.3.2. Solving Maximum Likelihood Estimation Phrased Optimization

Let us begin from the prompt:

$$\begin{aligned}
\operatorname*{argmax}_{\vec{w}_0} f(\vec{y}|\vec{w} = \vec{w_0}) &= \operatorname*{argmax}_{\vec{w}_0} \prod_{i=1}^{n} f(Y_i = y_i|\vec{w} = \vec{w_0}) \\
&= \operatorname*{argmax}_{\vec{w}_0} \prod_{i=1}^{n} f(z_i = y_i - \vec{x_i}^T \vec{w}|\vec{w} = \vec{w_0}) \\
&= \operatorname*{argmax}_{\vec{w}_0} \prod_{i=1}^{n} \frac{\exp\left(-\frac{(y_i - \vec{x_i}^T)^2}{2\sigma_i^2}\right)}{\sqrt{2\pi}\sigma_i} \\
&= \operatorname*{argmax}_{\vec{w}_0} \frac{1}{\sqrt{2\pi}^n} \exp\left(\sum_{i=1}^{n} \frac{-(y_i - \vec{x_i}^T \vec{w})}{2\sigma_i^2}\right) \prod_{i=1}^{n} \frac{1}{\sigma_i} \\
&= \operatorname*{argmin}_{\vec{w}_0} \sum_{i=1}^{n} \frac{(y_i - \vec{x_i}^T \vec{w})^2}{2\sigma_i^2} \\
&= \operatorname*{argmin}_{\vec{w}_0} \|S(X\vec{w_0} - \vec{y})\|_2^2
\end{aligned}$$

where,

$$S = \begin{bmatrix} \frac{1}{\sqrt{2}\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sqrt{2}\sigma_n} \end{bmatrix}$$

and we end up on a weighted least squares setup.
Furthermore, upon IID uniform Gaussians, we will end up with $S = \frac{1}{\sqrt{2}\sigma_i}$.

Meanwhile, the Maximum A-Posteriori approach attempts to provide a priori distribution on $\vec{w}$.

We solve the optimization problem of:

$$\underset{\vec{w_0}}{\operatorname{argmax}}\, f(\vec{w} = \vec{w_0}|\vec{y})$$

As you may observe, both techniques offer the most likely weights for some condition, but the conditions are framed quite differently:

> - MLE find the most likely $\vec{w}$ to **lead to current observation**.
> - MAP find the most likely $\vec{w}$ **given the current observation** that occurs.

9.3.2 A Personal Learning on MLE vs MAP

While this is not entirely the focus of exam scope, I'd like to address the difference between MLE and MAP with a few lines.
First of all, We may notice the relationship between objective function of MLE and MAP to have the following relationship:

$$f(\vec{y}|\vec{w} = \vec{w_0}) = \pi_y f(\vec{w} = \vec{w_0}|\vec{y})$$

If we consider the priori π_y to be uniform (for example, all possible components of $\vec{y}$ address an event of uniform distribution like coin flips or dice rolls), then maximizing the MLE objective function indeed maximizes the MAP objective function (as they are scalar multiples of each other). This means MLE and MAP, under a uniform priori for any possible component of $\vec{y}$, are equal optimization problems.

Chapter 10

Convexity

10.1 Convex Set

Let us first define a convex set geometrically,

Definition 10.1.1. Convex Set

A set $C \subseteq \mathbb{R}^n$ is convex if the line segment between any two points in C is contained in C.
For example, convex polygons resemble a convex set of points.

Then, to express such concept algebraically, we would arrive at the algebraic addenda to definition:

Definition 10.1.2. Convex Set in Algebraic Perspective

The set of points within a line segment terminated by points $\vec{x}, \vec{y} \in C$ would be expressable as

$$L_{\vec{x},\vec{y}} := \{\theta\vec{x} + (1-\theta)\vec{y} | \theta \in [0,1]\}$$

And, a set C is convex iff

$$\forall \vec{x}, \vec{y} \in C, L_{\vec{x},\vec{y}} \subseteq C$$

Upon the algebraic interpretation of convexity in sets, we may also define convexity of combinations:

Definition 10.1.3. Convex Combination

The combination $\vec{x} = \sum_i \theta_i \vec{x_i}$ is a convex combination iff it satisfies the following qualities:

1. $\forall i, \theta_i \geq 0$
2. $\sum_i \theta_i = 1$

Along which, we may then define convexity on numerous mathematical objects. Another example is shown here,

Definition 10.1.4. Convex Hull

The convex hull of some set $S \subseteq \mathbb{R}^n$ is the set of all convex combinations of members of S.
This formulates a concept very similar to “span”.

10.1.1 Proof of Convexity

Sets of mathematical objects (like matrices) can be convex as well. Let us demonstrate with the following proofs, where we in unison practice to prove convexity of sets.

Explain 10.1.1. The Convexity of Set of Rank-1 Matrices

Problem. Let us find a set of all rank-1 Matrices:

$$\{M_1 = \{A \in \mathbb{R}^{m\times n} | rk(A) = 1\}\}$$

decide whether it is convex or not.

Proof. Let us observe whether for any arbitrary matrices $X, Y \in M_1$, the "line segment" along these matrices are included in M_1.
In other words, we ask, whether the matrix $\theta X + (1-\theta)Y$ belongs to M_1.
The answer would be no. Suppose X_0 and Y_0 each have a basis $\{\vec{x}\}$, $\{\vec{y}\}$, and let $\vec{x}$ and $\vec{y}$ be linearly independent vectors.

$$X_0 = \begin{bmatrix} 2\vec{x} & \vec{x} \end{bmatrix}, Y_0 = \begin{bmatrix} \vec{y} & \vec{y} \end{bmatrix}$$

Consequently,

$$\forall \theta \in [0,1], rk(\theta X_0 + (1-\theta)Y_0) > 1$$

And thus,

$$\neg(\forall X, Y \in C, L_{\vec{x},\vec{y}} \subseteq C)$$

Similalry, let's explore another example of proof/disproof for convexity:

Explain 10.1.2. The Convexity of Set of PSD Matrices

Problem. Let us find a set of all rank-1 Matrices:

$$\{P = \{A | A \succcurlyeq 0 \wedge A \in \mathbb{S}^n\}\}$$

decide whether it is convex or not.

Proof. To prove that a matrix is positive-semidefinite by definition, we need only prove it suits one of the three definitions of positive-semidefiniteness.
For most convenience, I'd like to use the definition that:

$$A \succcurlyeq 0 \iff \forall \vec{x} \in \mathbb{R}^n, \vec{x}^T A \vec{x} \geq 0$$

Now, let me work with two arbitrary matrices $X, Y \in P$, and attempt to prove the following equivalent problem:

Equivalent Prompt. Prove that $\forall \theta \in [0,1], \theta X + (1-\theta)Y \in P$.

Fortunately, we may observe that,

$$\forall \vec{x}, \vec{x}^T(\theta X + (1-\theta)Y)\vec{x} = \vec{x}^T\theta X\vec{x} + \vec{x}^T(1-\theta)Y\vec{x}$$
$$\forall \vec{x}, \vec{x}^T X\vec{x} \geq 0 \wedge \vec{x}^T Y\vec{x} \geq 0 \implies \forall \vec{x}, \vec{x}^T\theta X\vec{x} + \vec{x}^T(1-\theta)Y\vec{x} \geq 0$$

Furthermore,

$$\begin{aligned}(\theta X + (1-\theta)Y)^T &= \theta X^T + (1-\theta)Y^T \\ &= \theta X + (1-\theta)Y\end{aligned}$$

Therefore,

$$\forall \theta \in [0,1], \theta X + (1-\theta)Y \succcurlyeq 0 \wedge \forall \theta \in [0,1], \theta X + (1-\theta)Y \in \mathbb{S}^n$$

then by definition it must belong to the set P. Via this subproof, we conclude that P is convex.

10.1.2 Hyperplanes

Hyperplanes are prevalent mathematical objects in Computer Science applications (such as the infamous Machine Learning), and its formal definition follows:

Definition 10.1.5. Hyperplane

Hyperplanes are sets of points that have the two following alternative forms:

$$H = \{\vec{x} \in \mathbb{R}^n | \vec{a}^T \vec{x} = \vec{b}\}$$
$$H = \{\vec{x} \in \mathbb{R}^n | \vec{a}^T (\vec{x} - \vec{x_0}) = 0\}, \vec{b} = \vec{a}^T \vec{x_0}$$

in the above notes, you may see the equivalence of two forms via adjusting the names of variables in the set-builder notation.

Upon defining it, let us then discuss the properties of such mathematical object. Is the hyperplane per se a convex set?

Explain 10.1.3. Convexity of Hyperplane

Proof. For a hyperplane defined as,

$$H = \{\vec{x} \in \mathbb{R}^n | \vec{a}^T (\vec{x} - \vec{x_0}) = 0\}$$

Let us take two arbitrary points $\vec{y}, \vec{z} \in H$.
Then, for some $\theta \in [0, 1]$,

$$\begin{aligned}\vec{a}^T (\theta\vec{y} + (1-\theta)\vec{z} - \vec{x_0}) &= \vec{a}^T \theta\vec{y} + \vec{a}^T (1-\theta)\vec{z} - \vec{a}^T \vec{x_0} \\ &= \theta\vec{a}^T \vec{x_0} + (1-\theta)\vec{a}^T \vec{x_0} - \vec{a}^T \vec{x_0} = 0\end{aligned}$$

Therefore, by definition,

$$\forall \vec{y}, \vec{z} \in H, L_{\vec{y},\vec{z}} \subseteq H$$

and we have proven the convexity of hyperplanes by its definition.

Furthermore, we can define specific subsets of a space defined by some hyperplane:

Definition 10.1.6. Halfspace

For a hyperplane

$$H = \{\vec{x} \in \mathbb{R}^n | \vec{a}^T (\vec{x} - \vec{x_0}) = 0\} \in \mathbb{R}^n$$

it partitions the real space $\mathbb{R}^n$ into two halves, which we formally call the halfspace.
The halfspaces can then be defined as follows:

$$H_+ = \{\vec{x} \in \mathbb{R}^n | \vec{a}^T (\vec{x} - \vec{x_0}) \geq 0\}$$
$$H_- = \{\vec{x} \in \mathbb{R}^n | \vec{a}^T (\vec{x} - \vec{x_0}) \leq 0\}$$

We call H_+ the positive halfspace, and H_- the negative halfspace.

And, along this concept of halfspace, we have developed the following theorem:

Theorem 10.1.1. Separating Hyperplane Theorem

Let $C, D \in \mathbb{R}^n$ be nonempty disjoint convex sets, then

$$\exists \vec{a} \neq \vec{0}, \vec{x_0} \in \mathbb{R}^n \big(\forall \vec{x} \in C, \vec{a}^T (\vec{x} - \vec{x_0} \geq 0) \wedge \forall \vec{x} \in D, \vec{a}^T (\vec{x} - \vec{x_0} \leq 0)\big)$$

In other words, there must exist a hyperplane that separates the members of C and D.

Such theorem is incredibly relevant to linear classifiers (specifically, this is an insight extracted from SVMs in CS189).
On a side note, the proof of this theorem is essentially the mathematical work to construct such a separating hyperplane, whose normal vector would ideally be some vector between a point of C and a point of D and the hyperplane crosses the midpoint of $\overline{pq}$.

10.2 Convex Functions

Once again, we will begin with a definition:

Definition 10.2.1. Convex Functions

A scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff the following qualities are satisfied:

- The domain of f is a convex set.
- The function f obeys the Jensen's Inequality:

$$\forall \vec{x}, \vec{y} \in Domain(f), \theta \in [0, 1], f(\theta \vec{x} + (1 - \theta)\vec{y}) \leq \theta f(\vec{x}) + (1 - \theta) f(\vec{y})$$

Essentially, the Jensen's Inequality states that:

> Any line segment between two points $(x, f(x)), (y, f(y))$ of a convex function would not be below the function curve between those two points $(x, f(x)), (y, f(y))$.

Chapter 11

Convex Optimization Problems

11.1 Convex Functions, Continued

We have defined last time that

Definition 11.1.1. Convex Functions

A scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff the following qualities are satisfied:

- The domain of f is a convex set.
- The function f obeys the Jensen's Inequality:

$$\forall \vec{x}, \vec{y} \in Domain(f), \theta \in [0,1], f(\theta\vec{x} + (1-\theta)\vec{y}) \leq \theta f(\vec{x}) + (1-\theta)f(\vec{y})$$

Essentially, the Jensen's Inequality states that:

> Any chord between two points $(x, f(x)), (y, f(y))$ of a convex function would not be below the function curve between those two points $(x, f(x)), (y, f(y))$.

Concave functions are, on the other hand, the version of convex functions where many theories apply on maximization. For example, while convex functions' critical points are directly their global minimum, concave functions' critical points are directly their global maximum.
By studying convex optimization, we equivalently study many aspects of concave optimization. (Moreover, inverting the sign of some concave function f already make a convex objective function).
Interestingly, linear functions are both concave and convex: they obey Jensen's Inequality by having the chord overlap with the objective function curve itself.

Now, let us discuss the properties of functions further:

Definition 11.1.2. Epigraph

An epigraph of a function f is the set:

$$epi(f) = \{(\vec{x}, t) \big| \vec{x} \in dom(f), f(\vec{x}) \leq t\}$$

Reviewing the above definition, we discover that an epigraph is the space above the objective function value, the "volume above curve".

Such figure is related to the convexity of a function:

Theorem 11.1.1. Epigraph and Convexity

Theorem. Function f is convex iff $epi(f)$ is a convex set.

Proof. Let us perform proof(s) for such theorem.
Direction 1: f is convex $\implies$ $epi(f)$ is convex.
A set C is convex if:

$$\forall \vec{x}, \vec{y} \in C, L_{\vec{x},\vec{y}} = \{\theta\vec{x} + (1-\theta)\vec{y} | \theta \in [0,1]\} \subseteq C$$

A function f is convex if:

$$\forall \vec{x}, \vec{y} \in Domain(f), \theta \in [0,1], f(\theta\vec{x} + (1-\theta)\vec{y}) \leq \theta f(\vec{x}) + (1-\theta) f(\vec{y})$$

Let us take two arbitrary points from $epi(f)$ be $(\vec{x}, t_x), (\vec{y}, t_y)$ such that $t_x \geq f(\vec{x})$ and $t_y \geq f(\vec{y})$.
Let us consider some point in $L_{\vec{x},\vec{y}}$ and see if it belongs to C:

$$\theta(\vec{x}, t_x) + (1-\theta)(\vec{y}, t_y) = (\theta\vec{x} + (1-\theta)\vec{y}, \theta t_x + (1-\theta)t_y)$$

Where, via the property of a convex function, we know that $\theta\vec{x} + (1-\theta)\vec{y}$ is included in its convex domain. Furthermore, via Jensen's Inequality

$$\begin{aligned} \theta t_x + (1-\theta)t_y &\geq \theta f(\vec{x}) + (1-\theta) f(\vec{y}) \\ &\geq f(\theta\vec{x} + (1-\theta)\vec{y}) \end{aligned}$$

Therefore, by definition, a convex function f has a convex epigraph $epi(f)$.
Direction 2: $epi(f)$ is convex $\implies$ f is convex.
Take two arbitrary points in $epi(f)$ to be $(\vec{x}, t_x = f(\vec{x})), (\vec{y}, t_y = f(\vec{y}))$, where by the definition of convexity, we realize that:

$$\forall \theta \in [0,1], (\theta\vec{x} + (1-\theta)\vec{y}, \theta t_x + (1-\theta)t_y) \in epi(f)$$

Since such point exists in an epigraph, it must infer that,

$$\theta t_x + (1-\theta)t_y = \theta f(\vec{x}) + (1-\theta) f(\vec{y}) \geq f(\theta\vec{x} + (1-\theta)\vec{y})$$

which is the Jensen's Inequality: what characterizes the definition of a convex function f.

This thus connects the definitions of convex sets and convex functions. But, ready or not, there are more definitions of convexity:

Definition 11.1.3. First Order Condition of Convexity

Let f be a differentiable function, then f is convex iff the following condition is satisfied:

$$\forall \vec{x}, \vec{y} \in Domain(f), f(\vec{y}) \geq f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x}|_{\vec{x}} \cdot (\vec{y} - \vec{x})$$

The implication of such definition is:

$$\nabla f(\vec{x}) = 0 \implies \forall \vec{y}, f(\vec{y}) \geq f(\vec{x}) + 0$$

Proof.
Direction 1: f is convex implies the first order condition of convexity
The definition of convex function allows us to state for some $\theta \in [0,1]$ that

$$\forall \vec{x}, \vec{y} \in Domain(f), \theta \in [0,1], f((1-\theta)\vec{x} + \theta\vec{y}) \leq (1-\theta) f(\vec{x}) + \theta f(\vec{y})$$

Rearranging the above inequality grants

$$\begin{aligned}\theta f(\vec{y}) &\geq f((1-\theta)\vec{x}+\theta\vec{y}) - f(\vec{x}) + \theta f(\vec{x}) \\ f(\vec{y}) &\geq \frac{1}{\theta} f(\vec{x}+\theta(\vec{y}-\vec{x})) - \frac{1}{\theta} f(\vec{x}) + f(\vec{x})\end{aligned}$$

Upon aligning the above equation with the definition of derivative, we may discover that:

$$\lim_{t\to 0} \frac{f(\vec{x}+\theta(\vec{y}-\vec{x})) - f(\vec{x})}{\theta(\vec{y}-\vec{x})} = \frac{\mathrm{d}f}{\mathrm{d}x}$$

Therefore, let us substitute the above derivative term into the function:

$$f(\vec{y}) \geq \frac{\mathrm{d}f}{\mathrm{d}x} \cdot (\vec{y}-\vec{x}) + f(\vec{x})$$

which is exactly the first order condition of convexity.
Direction 2: the first order condition of convexity implies f is convex.
Suppose we have two arbitrary points $\vec{x}, \vec{y}$, and that:

$$\vec{z} = \theta\vec{x} + (1-\theta)\vec{y}$$

Let us use the first order condition of convexity to state that,

$$\begin{aligned}f(\vec{x}) &\geq f(\vec{z}) + f'(\vec{z})(\vec{x}-\vec{z}) \\ f(\vec{y}) &\geq f(\vec{z}) + f'(\vec{z})(\vec{y}-\vec{z})\end{aligned}$$

Then,

$$\begin{aligned}\theta f(\vec{x}) + (1-\theta)f(\vec{y}) &\geq f(\vec{z}) + \theta f'(\vec{z})(\vec{x}-\vec{z}) + (1-\theta)f'(\vec{z})(\vec{y}-\vec{z}) \\ &= f(\vec{z}) + \theta f'(\vec{z})(\theta\vec{x} - \theta\vec{z} + (1-\theta)\vec{y} - (1-\theta)\vec{z}) \\ &= f(\vec{z})\end{aligned}$$

Upon the first order condition, we also have a second order condition of convexity:

Definition 11.1.4. Second Order Condition of Convexity

Let f be a twice-differentiable function, then f is convex iff the following condition is satisfied:

$$\nabla^2 f(x) \succcurlyeq 0$$

11.2 Convex Optimization Problem

Let us have some problem:

$$p^* = \min f_0(\vec{x}) \text{ s.t. } f_i(\vec{x}) \leq 0$$

A problem is a convex optimization problem if for all i, $f_i(x)$ is a convex function.

Chapter 12

Descent Methods

12.1 Strict Strong Convexity

Convexity also comes in different strictness. These more refined definition provide us flexibility and rigor in proofs. For example, we may first consider that, the zeroth order definition of convexity would be:

> For a function $f : \mathbb{R}^n \to \mathbb{R}$, the domain of f needs to be convex, and Jensen's Inequality applies to any two arbitrary points on the domain.

However, we fail to classify linear functions as either convex or concave. Linear functions were concluded to be both convex and concave, which is very confusing.

Definition 12.1.1. Strict Convexity

The zeroth order condition of strict convexity is altered to:

> For a function $f : \mathbb{R}^n \to \mathbb{R}$, the domain of f needs to be convex, and:

$$f(\theta\vec{x} + (1-\theta)\vec{y}) < \theta f(\vec{x}) + (1-\theta)f(\vec{y})$$

The first order condition is altered to:

$$\forall \vec{x}, \vec{y} \in Domain(f), f(\vec{y}) > f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x} \cdot (\vec{y} - \vec{x})$$

The second order condition is altered along a very similar logic:

$$\nabla^2 f(\vec{x}) > 0$$

And in summary, a strictly convex function must have a unique minimum (unlike, say, a ReLU function).

But just like humans, not only can convexity have strict variants, they can also have strong variants. Strong convexity implies strict convexity, and strict convexity implies convexity. These definitions are increasingly difficult to satisfy:

Definition 12.1.2. Strong Convexity

If f is differentiable, then for some $\mu > 0$, it is μ-strongly convex if domain of f is convex and

$$\forall \vec{x}, \vec{y} \in domain(f), f(\vec{y}) \geq f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x} \cdot (\vec{y} - \vec{x}) + \frac{\mu}{2}\|\vec{y} - \vec{x}\|_2^2$$

The insight such variant provides is, for some μ-strongly convex function:

$$\nabla^2 f(x) \succcurlyeq \mu I \implies \nabla^2 f(x) - \mu I \succcurlyeq 0$$

The smallest possible eigenvalue of the Hessian of f is then μ. The larger such value μ is, the stronger its convex; in other words, the further the function is from being nonconvex.

12.2 Gradient Descent

As you might have learned in prior courseworks (during college, or, high school given Berkeley), there are many variants of Gradient Descent.
Let's think about the vanilla version of gradient descent first, as an unconstrained optimization problem:

12.2.1 Inventing Gradient Descent

$$p^* = \min_{\vec{x} \in \mathbb{R}^n} f_0(\vec{x})$$

The foundation of gradient descent algorithm is an "oracle" that guides us at the direction of some greatest descent along the provided function f.
Now, we should understand when does the "oracle" guide us to convergence, such that we do not get lost in the search space.

Let's find the "oracle" via some perspectives on linear perturbation:

$$f(\vec{x} + \Delta\vec{x}) = f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x} \cdot (\Delta\vec{x}) + \text{H.O.T.}$$

If $\Delta\vec{x}$ is a good direction to perturb $\vec{x}$ to, then we wish as well that $f(\vec{x} + \Delta\vec{x}) < f(\vec{x})$.
Suppose that $\Delta\vec{x} = s\vec{v}$, where s is a real scalar and $\vec{v}$ is some direction-resembling vector.
Then, let us rewrite $f(\vec{x} + s\vec{v})$ as:

$$\begin{aligned} f(\vec{x} + s\vec{v}) &= f(\vec{x}) + s\frac{\mathrm{d}f}{\mathrm{d}x}\vec{v} \\ &= f(\vec{x}) + s\langle \nabla f(\vec{x}), \vec{v} \rangle < f(\vec{x}) \\ \langle \nabla f(\vec{x}), \vec{v} \rangle &< 0 \end{aligned}$$

This means $\nabla f(\vec{x})$ and $\vec{v}$ should be in opposite direction.
Furthermore, to maximize such above expression for finding a greatest descent, we may let

$$\vec{v} = -\nabla f(\vec{x})$$

and we conclude that the opposite direction of gradient is the great direction of descent– the "oracle".

Now, let us write up a draft for the gradient descent algorithm (which I will follow the minimal-energy principle of Physics and shamelessly copy from my notes in prior courseworks), that:

Definition 12.2.1. Gradient Descent Algorithm

Let the initial guess of minimum be $\vec{x_0}$.
Then, provided a stepsize η, and defining $\vec{x_k}$ to be the k^{th} guess of minimum, the iterative approach of algorithm follows as:

$$\vec{x}_{k+1} = \vec{x}_k - \eta \nabla f(\vec{x_k})$$

To get better constant stepsizes, we can perform hyperparamter tuning; or, solve the next subsection.

12.2.2 Finding η

Let us first define the function we perform gradient descent on, to demonstrate the process of finding η.
Let

$$\begin{aligned} f_0(\vec{x}) &= \|A\vec{x} - \vec{b}\|_2^2 \\ \nabla f_0(\vec{x}) &= 2A^T(A\vec{x} - \vec{b}) \end{aligned}$$

The iterative rule of gradient descent on function f_0 is thus:

$$\begin{aligned} \vec{x}_{k+1} &= I\vec{x}_k - 2\eta A^T(A\vec{x} - \vec{b}) \\ &= (I - 2\eta A^T A)\vec{x}_k + 2\eta A^T\vec{b} \end{aligned}$$

Let us discuss the recursion in the above form.
First of all, we require that the matrix coefficient of guess term: $(I - 2\eta A^T A)$, to provide some scaling that does not consistently scale the vector larger, so to prevent divergence. In matrices, we may consider eigenvalue as the indicator of such divergence or convergence. We require that the eigenvalues of $(I - 2\eta A^T A)$ satisfy some condition such that it does not lead to a consistently larger guess.

Now, let us look at the guess rule:

$$\begin{aligned} \vec{x}_{k+1} - \vec{x}^* &= (I - 2\eta A^T A)\vec{x}_k + 2\eta A^T\vec{b} - (A^T A)^{-1}A^T\vec{b} \\ &= (I - 2\eta A^T A)\vec{x}_k + (2\eta A^T A - I)(A^T A)^{-1}A^T\vec{b} \\ &= (I - 2\eta A^T A)[\vec{x}_k - (A^T A)^{-1}A^T\vec{b}] \\ &= (I - 2\eta A^T A)[\vec{x}_k - \vec{x}^*] \end{aligned}$$

Solve the recursion above,

$$\begin{aligned} \vec{x}_{m+n} - \vec{x}^* &= (I - 2\eta A^T A)[\vec{x}_{m+(n-1)} - \vec{x}^*] \\ &\vdots \\ &= (I - 2\eta A^T A)^n[\vec{x}_{m+(n-n)} - \vec{x}^*] = (I - 2\eta A^T A)^n[\vec{x}_m - \vec{x}^*] \end{aligned}$$

Upon the fact that $I - 2\eta A^T A$ is a symmetric matrix (i.e., it is diagonalizable), we are granted the following statements:

$$\begin{aligned} I - 2\eta A^T A &= U\Lambda U^T \\ \vec{x}_k - (A^T A)^{-1}A^T\vec{b} &= (I - 2\eta A^T A)^k[\vec{x}_0 - (A^T A)^{-1}A^T\vec{b}] \\ &= U\Lambda^k U^T[\vec{x}_0 - (A^T A)^{-1}A^T\vec{b}] \end{aligned}$$

Hinting at what a good choice of η is such that we may guarantee the subsequent guesses decrease as timestep increase. Particularly, we choose some η such that the eigenvalues of $(I - 2\eta A^T A)$ are less than 1 in magnitude. This allows us to upperbound the scaling effect of a matrix on some vector to not increase the vector's size (in the sense that, our guess does not travel away from the optimum we attempt to find via gradient descent).

12.2.3 Generalizations

For functions that are μ-strongly convex, their gradients do not change too slowly.
For functions that are L-smooth, such that

$$f(\vec{y}) \leq f(\vec{x}) + \nabla f(\vec{x}) \cdot (\vec{y} - \vec{x}) + \frac{L}{2}\|\vec{y} - \vec{x}\|_2^2$$

their gradients do not change too fast.

Theorem 12.2.1. Lemma on Gradients of L-smooth Functions

Lemma: For an L-smooth function f

$$\|\nabla f(\vec{x})\|_2^2 \leq 2L(f(\vec{x}) - f(\vec{x}^*))$$

Using the L-smooth property of a function,

$$f(\vec{y}) \leq f(\vec{x}) + \nabla f(\vec{x}) \cdot (\vec{y} - \vec{x}) + \frac{L}{2}\|\vec{y} - \vec{x}\|_2^2$$
$$f(\vec{x} - \eta\nabla f(\vec{x})) \leq f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x}(-\eta\nabla f(\vec{x})) + \frac{L}{2}\|\eta\nabla f(\vec{x})\|_2^2$$

Chapter 13

Descent Methods and Convex Optimizations

13.1 Continued, Gradient Descent Convergence Proof

While there was a splendid convergence rate for the least squares problem, we want to discuss how gradient descent operates for other functions.
To have some similar quadratic form for the gradient descent convergence proof, we may work with smooth and strongly convex function, and it would offer the interval at which a gradient descent convergence lies at. In lecture, this was phrased as a “quadratic sandwich”.

> *Puts two bread across the two side of a strongly convex, smooth function
> **WHAT ARE YOU**
> *A quadratic sandwich, chef*

sorry for the digression.

13.1.1 Breads of Quadratic Sandwich

Here, let us review two definitions introduced in the prior lecture:

Definition 13.1.1. Review: Strong Convexity

If f is differentiable, then for some $\mu > 0$, it is μ-strongly convex if domain of f is convex and

$$\forall \vec{x}, \vec{y} \in domain(f), f(\vec{y}) \geq f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x} \cdot (\vec{y} - \vec{x}) + \frac{\mu}{2} \|\vec{y} - \vec{x}\|_2^2$$

Definition 13.1.2. L-smooth Functions

For an L-smooth function,

$$f(\vec{y}) \leq f(\vec{x}) + \nabla f(\vec{x}) \cdot (\vec{y} - \vec{x}) + \frac{L}{2} \|\vec{y} - \vec{x}\|_2^2$$
$$f(\vec{x} - \eta \nabla f(\vec{x})) \leq f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x}(-\eta \nabla f(\vec{x})) + \frac{L}{2} \|\eta \nabla f(\vec{x})\|_2^2$$

The L-smoothness of a function guarantees the gradient doesn’t change too fast, while the μ-strong convexity guarantees the gradient doesn’t change too slow.

Let us prove the lemma addressed in prior lecture regarding the L-smooth functions:

Theorem 13.1.1. Lemma on Gradients of L-smooth Functions

Lemma: For an L-smooth function f

$$\|\nabla f(\vec{x})\|_2^2 \leq 2L(f(\vec{x}) - f(\vec{x}^*))$$

Proof: Suppose the current guess is some vector $\vec{x}$, such that the next guess is $\vec{y} = \vec{x} - \eta\nabla f(\vec{x})$. Provided that $f(\vec{x}^*)$ is a global minimum, we may determine that,

$$\forall\vec{x}, f(\vec{x}^*) \leq f(\vec{y}) = f(\vec{x} - \eta\nabla f(\vec{x}))$$

Meanwhile, we may find via the L-smoothness condition that,

$$\begin{aligned} f(\vec{x} - \eta\nabla f(\vec{x})) &\leq f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x}(-\eta\nabla f(\vec{x})) + \frac{L}{2}\|\eta\nabla f(\vec{x})\|_2^2 \\ &= f(\vec{x}) - \eta\|\nabla f(\vec{x})\|_2^2 + \frac{\eta^2 L}{2}\|\nabla f(\vec{x})\|_2^2 \\ &= f(\vec{x}) + \eta\left(\frac{\eta L}{2} - 1\right)\|\nabla f(\vec{x})\|_2^2 \end{aligned}$$

To produce the most efficient upperbound, we should minimize the coefficient of gradient's squared L2 norm. This brings us to a convex optimization problem:

$$\min_\eta \frac{\eta^2 L}{2} - \eta$$

which we may solve via calculus to see,

$$\eta^* = \frac{1}{L}$$

Let us substitute this back into the above derivation,

$$\begin{aligned} f(\vec{x} - \eta\nabla f(\vec{x})) &= f(\vec{x}) + \eta\left(\frac{\eta L}{2} - 1\right)\|\nabla f(\vec{x})\|_2^2 \\ &= f(\vec{x}) - \frac{L}{2}\|\nabla f(\vec{x})\|_2^2 \end{aligned}$$

Once again, via the property of $f(\vec{x}^*)$ to be the global minimum,

$$\begin{aligned} f(\vec{x}^*) &\leq f(\vec{x} - \eta\nabla f(\vec{x})) \\ &= f(\vec{x}) - \frac{L}{2}\|\nabla f(\vec{x})\|_2^2 \end{aligned}$$

This can then be derived into the aforementioned lemma.

13.1.2 Quadratic Sandwich

Let us now demonstrate the "quadratic sandwich" we discussed in prior:

Theorem 13.1.2. Quadratic Sandwich

Theorem. For function f that is L-smooth and μ-strongly convex, then for some appropriate α,

$$\|\vec{x}_{t+1} - \vec{x}^*\|_2^2 \leq \alpha\|\vec{x}_t - \vec{x}^*\|_2^2$$

Proof. For a μ-strongly convex function, we may see that,

$$f(\vec{x}') \leq f(\vec{x}) + \frac{\mathrm{d}f}{\mathrm{d}x} \cdot (\vec{x}' - \vec{x}) + \frac{\mu}{2}\|\nabla f(\vec{x})\|_2^2$$

Be a little bit manipulative, and we will obtain:

$$\frac{\mathrm{d}f}{\mathrm{d}x} \cdot (\vec{x}' - \vec{x}) \leq f(\vec{x}') - f(\vec{x}) - \frac{\mu}{2}\|\nabla f(\vec{x})\|_2^2$$

Now, to find a relationship (such that we may continue a proof), we will have to dedicate the following algebraic effort.
Particularly, we attempt to find a relationship between different guesses that are one timestamp across:

$$\begin{aligned}
\|\vec{x}_{t+1} - \vec{x}^*\|_2^2 &= \|\vec{x}_t - \eta\nabla f(\vec{x}_t) - \vec{x}^*\|_2^2 \\
&= \|(\vec{x}_t - \vec{x}^*) - \eta\nabla f(\vec{x}_t)\|_2^2 \\
&= \big((\vec{x}_t - \vec{x}^*) - \eta\nabla f(\vec{x}_t)\big)^T\big((\vec{x}_t - \vec{x}^*) - \eta\nabla f(\vec{x}_t)\big) \\
&= \|\vec{x}_t - \vec{x}^*\|_2^2 + \|\eta\nabla f(\vec{x}_t)\|_2^2 + 2(\eta\nabla f(\vec{x}_t))^T(\vec{x}^* - \vec{x}_t) \\
&\leq \|\vec{x}_t - \vec{x}^*\|_2^2 + 2L\eta^2(f(\vec{x}_t) - f(\vec{x}^*)) + 2\eta(f(\vec{x}^*) - f(\vec{x}_t) - \frac{\mu}{2}\|\nabla f(\vec{x})\|_2^2) \\
&= (1 - \eta\mu)\|\vec{x}_t - \vec{x}^*\|_2^2 + (2L\eta^2 - 2\eta)(f(\vec{x}_t) - f(\vec{x}^*))
\end{aligned}$$

Here, by letting $\eta = \frac{1}{L}$, we can cancel out the latter term.
We thus obtain that the appropriate α is:

$$\alpha = 1 - \eta\mu = 1 - \frac{\mu}{L}$$

Therefore, our interval of convergence for L is:

$$\begin{gathered}
-1 < 1 - \frac{\mu}{L} < 1 \\
0 < \mu < 2L
\end{gathered}$$

The implication of quadratic sandwich is that it bounds any function's convergence in gradient descent via some working quadratic forms.
If boundable by quadratics, convergence can nicely occur.

13.2 Stochastic Gradient Descent

There is a variety of other names, but SGD is perhaps the most popular choice.
Essentially, SGD is "lazy gradient descent". This name comes from its algorithmic property of reducing computational cost to avoid computing an entire gradient for some large vector or dataset.

The loss function of a dataset is very frequently written in the form:

$$L(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m} L_i(\vec{x})$$

For example, in least squares algorithm,

$$L(\vec{x}) = \frac{1}{m}\|A\vec{x} - \vec{b}\|_2^2 = \frac{1}{m}\sum_{i=1}^{m}(\vec{a}_i^T\vec{x} - \vec{b}_i)^2$$

Fortunately, by the additive property of derivatives, we may also state that:

$$\nabla f(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(\vec{x})$$

In SGD, instead of stepping in the direction of $\nabla f(\vec{x})$, we take a step in $\nabla f_i(\vec{x})$.
This is a legitimate approach, because

$$\mathbb{E}[\nabla f_i(\vec{x})] = \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(\vec{x}) = \nabla f(\vec{x})$$

The features of SGD involve:

- Computationally efficient!
- Can be comfortably applied on online data
- Is a more noisy algorithm, and will thus help to escape local minimum

Chapter 14

Applications and Extensions of Gradient Descent

14.1 Stochastic Gradient Descent, Continued

From last lecture, we have obtained the guess rule of gradient descent, and produced another guess rule named "Stochastic Gradient Descent" that is a comparatively effortless guess version of gradient descent.
For objective functions that can be decomposed into some components:

$$f(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m} f_i(\vec{x})$$

the SGD guess rule is that

$$\vec{x}_{k+1} = \vec{x}_k - \eta\nabla f_i(\vec{x}_k)$$

where we may, via prior derivation, see that:

$$\mathbb{E}[\nabla f_i(\vec{x}_k) = \nabla f(\vec{x}_k)]$$

SGD is a rather noisy algorithm that doesn't use the true gradient.
Consequently, SGD may not converge with a constant stepsize, and it also loses the property of gradient descent where: upon convergence, the increment between guesses becomes zero due to the gradient at optimum points being $\nabla f(\vec{x}^*) = \vec{0}$. For SGD, then, there needs to have a decaying stepsize over time to portray some similar property of convergence.

14.1.1 A Demonstration of SGD

Let us consider an example: Let us discuss the objective function:

$$f(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\|\vec{x} - \vec{p}_i\|_2^2$$

which provides the optimum:

$$\vec{x}^* = \frac{1}{m}\sum_{i=1}^{n}\vec{p}_i$$

Suppose we initiate our guess at $\vec{0}$, and provide $\eta = \frac{1}{k}$.
SGD may either randomly choose one component of the objective function to take guess rule along, or do so circularly

(which is easier to implement). The circular approach comes with occasional benefits and elegance.
Let us discuss its iterations below.

$$\begin{aligned}
\vec{x}_1 &= \vec{x}_0 - \eta_1 \nabla f_1(\vec{x}_0) = -(\vec{x}_0 - \vec{p}_1) = \vec{p}_1 \\
\vec{x}_2 &= \vec{x}_1 - \eta_2 \nabla f_2(\vec{x}_1) = \vec{p}_1 - \frac{1}{2}(\vec{x}_1 - \vec{p}_2) = \frac{1}{2}(\vec{p}_1 + \vec{p}_2) \\
\vec{x}_3 &= \vec{x}_2 - \eta_3 \nabla f_3(\vec{x}_2) = \frac{1}{2}(\vec{p}_1 + \vec{p}_2) - \frac{1}{3}(\vec{x}_2 - \vec{p}_3) = \frac{1}{3}(\vec{p}_1 + \vec{p}_2 + \vec{p}_3)
\end{aligned}$$

where we may see the guess is the mean of all points $\vec{p}_i$ involved until the current guess.

Choosing the stepsize decay pattern is a dark art of itself. Its not a story the Jedi would tell you. Its a Sith legend of Hyperparameter Tuning. The dark side of the Force is a pathway to many abilities some consider to be unnatural, but convex-optimal.

14.1.2 Mathematical Case Study on Convergence of SGD

Let us perform a case study for the least squares problem applied with SGD. We will use a variant of MSE for the loss function to optimize on:

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2}(a_i x - b_i)^2$$

where,

$$\begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} x = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, \vec{x}^* = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2^2}$$

We may observe:

$$\nabla f_i(x) = a_i(a_i x - b_i), \vec{x}_i^* = \frac{b_i}{a_i}$$

Then, we would want to show that,

$$\min_i \vec{x}_i^* \leq \vec{x}^* \leq \max_i \vec{x}_i^*$$

Let us perform some algebraic manipulation as follows:

$$\begin{aligned}
\vec{x}^* &= \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2^2} = \frac{\sum_{i=1}^m a_i^2 \frac{b_i}{a_i}}{\|\vec{a}\|_2^2} \\
&\leq \frac{\max_i \vec{x}_i^* \sum_{i=1}^m a_i^2}{\|\vec{a}\|_2^2} = \max_i \vec{x}_i^*
\end{aligned}$$

observing that such logic can be applied onto the case of $\min_i \vec{x}_i^*$ as well.

The implication of such phenomenon aids us towards convergence. Provided that our gradient of least squares loss function is:

$$\nabla f_i(x) = a_i^2(x - \frac{b_i}{a_i})$$

If we have $x > \frac{b_i}{a_i}$, which means it would be larger than the maximum possible optimum $\max_i \vec{x}_i^*$, and we would be pushed towards the interval marked by $[\min_i \vec{x}_i^*, \max_i \vec{x}_i^*]$.
Furthermore, a similar logic applies to the case of $x < \frac{b_i}{a_i}$.
Our oracle (gradient) is generally correct (albeit occasionally inelegant). Now, we only require the step size to be some appropriate amount such that, provided the correct direction towards which the next guess should exist, we do not overshoot outside the aforementioned proper interval of optimization.

Keep in mind that such interval is still highly theoretical and will not be realistically attained, so the tuning of step size highly depends on whether the phenomenon of convergence is truly observed (via some plotting measure).

14.2 Gradient Descent with Prior Optimization Constraint

Suppose that we attempt to solve the following problem:

$$\min_{\vec{x}\in\mathcal{X}} f(\vec{x})$$

where, $\mathcal{X}$ is a convex, compact set upon which we perform gradient descent on.
However, using conventional methods of the general, unconstrained gradient descent would possibly put the convergence outside $\mathcal{X}$. The concept of solution is **Projected Gradient Descent**, where we define the projection of one point $\vec{y}$ onto another point $\vec{x}$ as:

$$\Pi_{\mathcal{X}}(\vec{y}) = \underset{\vec{x}\in\mathcal{X}}{\operatorname{argmax}} \|\vec{y}-\vec{x}\|_2$$

Therefore, as we obtain a guess that is outside $\mathcal{X}$, we project that guess back into $\mathcal{X}$ to satisfy the constraint.
In other words, upon the usual guess rule computation, we would also add the computational cost of projections onto each iterative step of the original gradient descent algorithm. The guess rule of PGD is therefore:

$$\vec{x}_{k+1} = \Pi_{\mathcal{X}}(\vec{x}_k - \eta f(\nabla f(\vec{x}_k)))$$

Such problem is rarely used in practice, however, because we would be solving another optimization problem per step. The effect of such operation is inefficiency. Hence, although PGD works conceptually, it is not employed practically.

14.2.1 Conditional Gradient Descent

We may also propose an alternative variation from Frank Wolfe, where suppose we are solving for:

$$\vec{y_k} \in argmin_{\vec{y}\in\mathcal{X}} \nabla f(\vec{x}_k)\cdot\vec{y}$$

and γ is a fixed decating stepsize sequence.
Then the guess rule of CGD would be phrased as:

$$\begin{aligned}\vec{x}_{k+1} &= (1-\gamma_k)\vec{x}_k + \gamma_k\vec{y}_k\\ &= \vec{x}_k + \gamma_k(\vec{y}_k - \vec{x}_k)\end{aligned}$$

This setup allows the iterated guess to be a convex combination within $\mathcal{X}$.

Chapter 15

Interlude: Logistic Regression

insert something about taking 127 and 189 together here

15.1 Monotone Transformations

In prior work, we have minimized a positive function by minimizing its squared value.
For example:

$$argmin_{\vec{x}}\|A\vec{x}-\vec{b}\|_2 = argmin_{\vec{x}}\|A\vec{x}-\vec{b}\|_2^2$$

This is an example of monotone transformation:

Definition 15.1.1. Monotone Transformation

For a function $\Phi(x)$ that is continuous and strictly increasing, then

$$p^* = \min_{\vec{x}} f_0(\vec{x}) \text{ s.t. } \forall i \in [1,n] f_i(\vec{x}) \leq 0$$

Then, such problem is equivalent to the following:

$$g^* = \min_{\vec{x}} \Phi(f_0(\vec{x})) \text{ s.t. } \forall i \in [1,n] f_i(\vec{x}) \leq 0$$

There are many continuous and strictly increasing functions, including but not excluded to:

- $\Phi(x) = x^2$, with domain restructed to non-negative x
- $\Phi(x) = \log(x)$
- $\Phi(x) = e^x$

15.1.1 Building Logistic Regression

Suppose we have datapoints $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$, such that:

$$\forall i \in [1,n], y_i \in \{-1, 1\}$$

The task of logistic regression is to build a classifier between class labels 0 and 1 via predicting the probability that some datapoint belongs to a certain class:

$$\mathbb{P}(y_i = 1 | x = \vec{x}) = q(\vec{x})$$

Now, suppose that we construct such function as an affine transformation:

$$q(\vec{x}) = \vec{w}^T\vec{x} + \beta$$

However, the probability of an event must be between 0 and 1 (inclusively), so we must apply some other transformation such that $\vec{w}^T\vec{x}$ is no longer unbounded as it is now. Here, q is some other function we will involve in our estimation for probability, where maximizing q should grant a maximization of $\vec{w}^T\vec{x}$ as well.

Concretely, performing the monotone transformation $\Phi(x) = \log(x)$ onto $\vec{w}^T\vec{b}$ would make its range just unbounded in one single direction: $(\log(0), \log(1)) = (-\infty, 0)$.
And similarly, performing the monotone transformation $\Phi(x) = \frac{x}{1-x}$ makes $\vec{w}^T\vec{b}$ unbounded in anotehr direction: $(0, \infty)$.
Combining the above result, let us perform the monotone transformation:

$$\log\left(\frac{q(\vec{x})}{1-q(\vec{x})}\right) = \vec{w}^T\vec{x} + \beta$$

such that $\vec{w}^T\vec{x}$ is indeed a suitable function for predicting probability.

From above, we may perform some algebraic manipulation, and would eventually find that:

$$q(\vec{x}) = \frac{\exp\bigl(\vec{w}^T\vec{x} + \beta\bigr)}{1 + \exp\bigl(\vec{w}^T\vec{x} + \beta\bigr)}$$

To maximize the probability at which we obtain our current dataset, we also have the option of performing MLE. Suppose we apply MLE (Maximum Likelihood Estimation) on our above hypothesis for probability function, then we would be solving the optimization problem of maximizing the following expression:

$$\mathbb{P}(y_1, \ldots, y_n | \vec{w}^T, \beta)$$

Furthermore, suppose the probability at which $\mathbb{P}(y_i = 1)$ is as noted above, then we may dictate that:

$$\mathbb{P}(y_i = -1 | x = \vec{x}_i) = \frac{1}{1 + \exp\bigl(\vec{w}^T\vec{x}_i + \beta\bigr)} = \frac{\exp\bigl(-(\vec{w}^T\vec{x}_i + \beta)\bigr)}{1 + \exp\bigl(-(\vec{w}^T\vec{x}_i + \beta)\bigr)}$$

Then, substituting such expression:

$$\mathbb{P}(Y = y_i) = \frac{\exp\bigl(y_i(\vec{w}^T\vec{x}_i + \beta)\bigr)}{1 + \exp\bigl(y_i(\vec{w}^T\vec{x}_i + \beta)\bigr)}$$

such that the expressions of $\mathbb{P}(Y = y_i)$ is coherent with our algebraic works above:

$$\mathbb{P}(Y = y_i) = \begin{cases} \frac{\exp\bigl((\vec{w}^T\vec{x}_i + \beta)\bigr)}{1 + \exp\bigl((\vec{w}^T\vec{x}_i + \beta)\bigr)}, & y_i = 1 \\ \frac{\exp\bigl(-(\vec{w}^T\vec{x}_i + \beta)\bigr)}{1 + \exp\bigl(-(\vec{w}^T\vec{x}_i + \beta)\bigr)}, & y_i = -1 \end{cases}$$

Finally, applying maximum likelihood, we solve:

$$(\vec{w}^*, \beta^*) = \operatorname*{argmax}_{\vec{w}, \beta} \prod_{i=1}^{n} \frac{\exp\bigl(y_i(\vec{w}^T\vec{x}_i + \beta)\bigr)}{1 + \exp\bigl(y_i(\vec{w}^T\vec{x}_i + \beta)\bigr)}$$

to build the parameters of logistic regression classifier.
Now, performing a monotone transformation with $\Phi(x) = \log(x)$ will grant us a convex objective function to optimize for:

$$(\vec{w}^*, \beta^*) = \operatorname*{argmax}_{\vec{w}, \beta} \log\left(\prod_{i=1}^{n} \frac{\exp\bigl(y_i(\vec{w}^T\vec{x}_i + \beta)\bigr)}{1 + \exp\bigl(y_i(\vec{w}^T\vec{x}_i + \beta)\bigr)}\right)$$

Namely, while monotone transformations may change the convexity of a function, it will still provide an equivalent optimization problem.

Thanks for staying through all the maths.

Chapter 16

Weak Duality

Ok yeah, last time's lecture is kind of chill. Today's, it's not. – Professor.

16.1 Active Constraints

Let us consider this optimization problem:

$$\min_{\substack{x\geq 1\\ x\leq 2}} x^2$$

If moving the constraint causes a shift in the solution to this problem, such constraint is what we define as an **active constraint**. For example, the constraint $x \geq 1$ is an active constraint as it controls the solution of our optimization problem.
On the other hand, for another problem:

$$\min_{\substack{x\geq -1\\ x\leq 2}} x^2$$

none of the constraints directly influence the minimum of our optimization problem, so there are no active constraints.
Then, observing another problem below:

$$\min_{\substack{x_1^2\leq 2^2\\ x_2^2\leq 1^2}} x_1 + x_2$$

both of the constraints occur to be active.

There is no virtual limit on the number of active constraints an optimization problem can have (perhaps except the number of variables it involves), and active constraints can be observed by its definition as we observe whether changing one constraint changes the solution to our optimization problem.

16.2 Infimum vs. Minimum

Consider the following optimization problem:

$$\min_{\substack{x\geq 0\\ x\in\mathbb{R}}} e^{-x}$$

where we, despite knowing that the smallest possible value of such expression is 0, also acknowledge that it is not an achievable upperbound.
In such situation, we use the term **infimum**: the largest lower bound findable for an expression, to perform some mathematical conveniences similar to what a minimum provides.
The formal definition of it relies on real analysis.

16.3 Introduction to Lagrangian

16.3.1 Phrasing Optimization Problems

Let us consider the general optimization problem of:

$$p^* = \min_{\substack{\forall i \in [1,m],\ f_i(\vec{x}) \leq 0 \\ \forall j \in [1,n],\ h_j(\vec{x}) = 0}} f_0(\vec{x})$$

and a simplified version of it:

$$s^* = \min_{\substack{f_1(\vec{x}) \leq 0 \\ h_1(\vec{x}) = 0}} f_0(\vec{x})$$

We consider these expressions as **The Primal**.

Let us also introduce an indicator variable:

$$\mathbb{1}_{<0}\{f_i(\vec{x})\} = \begin{cases} 0, & f_i(\vec{x}) \leq 0 \\ \infty, & f_i(\vec{x}) > 0 \end{cases}$$

$$\mathbb{1}_0\{h_i(\vec{x})\} = \begin{cases} 0, & h_i(\vec{x}) = 0 \\ \infty, & h_i(\vec{x}) \neq 0 \end{cases}$$

such that we will re-introduce the simplified version of above minimization problem as:

$$s^* = \min f_0(\vec{x}) + \mathbb{1}_{<0}\{f_1(\vec{x})\} + \mathbb{1}_0\{h_1(\vec{x})\}$$

We may thus write a constrained optimization problem into its unconstrained form.

$$p^* = \min f_0(\vec{x}) + \sum_{i=1}^{m} \mathbb{1}_{<0}\{f_i(\vec{x})\} + \sum_{j=1}^{n} \mathbb{1}_0\{h_j(\vec{x})\}$$

However, these problems present non-continuous objective functions, and we thus cannot directly take the gradient of it to find critical points.

We should then discuss methods of finding some similar expression to the indicator variable.
One choice is as follows:

$$\mathbb{1}_{<0}\{f_i(\vec{x})\} \longleftrightarrow \lambda_i f_i(\vec{x}), \lambda_i > 0$$

such that our objective function decreases when $f_i(\vec{x}) < 0$ and achieves a similar effect to having such constraint.
Similarly,

$$\mathbb{1}_0\{h_i(\vec{x})\} \longleftrightarrow \nu_i h_i(\vec{x}), \nu_i > 0$$

(note: ν, v are different notations.)

16.3.2 Lagrangian of an Optimization Problem

The Lagrangian of an optimization problem is then defined as:

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) = f_0(\vec{x}) + \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \sum_{i=1}^{n} \lambda_i h_i(\vec{x})$$

where, λ_i, ν_i are called **dual variables**. This technique extends into what we call "Lagrange Multipliers".

We may notice that, $L(\vec{x}, \vec{\lambda}, \vec{\nu})$ is an affine function of $\vec{\lambda}$, $\vec{\nu}$, which makes it concave.
Then, we may define some function as follows to help with mathematical convenience:

$$g(\vec{\lambda}, \vec{\nu}) = \inf_{\vec{x}} L(\vec{x}, \vec{\lambda}, \vec{\nu})$$

Affine functions are convex and concave, and g is a pointwise maximum.
The infimum is a pointwise minimum, which introduces g as a concave function (pointwise minimum of multiple concave functions).
The function g thus extracts the minimum possible value of L based on what $\vec{x}$ minimizes it, when given a pair $(\vec{\lambda}, \vec{\nu})$.
This function g is called a **Dual Function**, which is corresponding to its primal.

The purpose of g is esoteric.

Theorem 16.3.1. g as Lower Bound of Optimization Solution

Theorem. We may consider g as a lower bound on the primal problem's solution p^*:

$$\forall \vec{\lambda} \geq 0, \vec{\nu},\ g(\vec{\lambda}, \vec{\nu}) \leq p^*$$

Proof. Consider some $\tilde{\vec{x}}$ as a feasible point for the primal. Then,

$$f_i(\tilde{\vec{x}}) \leq 0, h_i(\tilde{\vec{x}}) = 0$$

Therefore,

$$\lambda_i f_i(\tilde{\vec{x}}) \leq 0, \nu_i h_i(\tilde{\vec{x}}) = 0$$

So, when considering the value of $L(\vec{x}, \vec{\lambda}, \vec{\nu})$,

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) = f_0(\vec{x}) + \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \sum_{i=1}^{n} \lambda_i h_i(\vec{x}) \leq f_0(\vec{x})$$

The Lagrangian itself is always the lower bound of the function value itself whenever we discuss a feasible point for the primal.
Furthermore, conceptually, g is a function to extract the theoretical minimum Lagrangian possible– it must be smaller than the true minimum of objective function.

Definition 16.3.1. Dual Problem

We thus phrase that,

$$d^* = \max_{\vec{\lambda} \geq 0} g(\vec{\lambda}, \vec{\nu})$$

is a **dual problem** of the primal, which is the maximization of a concave function g.

Chapter 17

Weak, Strong Duality

17.1 Introduction to Weak vs. Strong Duality via Problems

17.1.1 Minimum-Norm Problem, Continued

The problem is phrased as:

$$\min_{\substack{A\vec{x}=\vec{b} \\ A\vec{x}-\vec{b}=\vec{0}}} \vec{x}^T\vec{x}$$

where, A is full rank and thus have infinite solutions.
We may write the lagrangian of such problem as a convex, quadratic function of $\vec{x}$:

$$L(\vec{x},\vec{\nu}) = \vec{x}^T\vec{x} + \vec{\nu}^T(A\vec{x}+\vec{b})$$

and we may therefore derive that,

$$g(\vec{\nu}) = \min_{\vec{x}} L(\vec{x},\vec{\nu})$$

Furthermore, as the function L is convex quadratic, we can minimize such function over $\vec{x}$ by setting gradient to be $\vec{0}$:

$$\begin{aligned} \nabla_{\vec{x}} L(\vec{x},\vec{\nu}) &= 2\vec{x} + A^T\vec{\nu} \\ 2\vec{x}^* + A^T\vec{\nu} &= \vec{0} \\ \vec{x}^* &= -\frac{1}{2}A^T\vec{\nu} \end{aligned}$$

Therefore,

$$\begin{aligned} g(\vec{\nu}) &= L(-\frac{1}{2}A^T\vec{\nu},\vec{\nu}) \\ &= \frac{1}{4}\vec{\nu}^T AA^T\vec{\nu} + \vec{\nu}^T(-\frac{1}{2}AA^T\vec{\nu} - \vec{b}) \\ &= -\frac{1}{4}\vec{\nu}^T AA^T\vec{\nu} - \vec{\nu}^T\vec{b} \end{aligned}$$

is a concave quadratic function of $\vec{\nu}$.

We also understand that $g(\vec{\nu})$ is a lowerbound to the true minimum norm solution (see note 16).
Furthermore, via vector calculus, we may obtain that:

$$\operatorname*{argmax}_{\vec{\nu}} g(\vec{\nu}) = -2(AA^T)^{-1}\vec{b}$$

And therefore, the true lower lower bound to minimum solution is:

$$\vec{x}^* = -\frac{1}{2}A^T\vec{\nu}^* = A^T(AA^T)^{-1}\vec{b}$$

As we have so obtained in the first notes.

Definition 17.1.1. Strong vs. Weak Duality

We phrase that,

$$d^* = \max_{\vec{\nu}} g(\vec{x}^*, \vec{\nu})$$

is a **dual problem** of the primal, which is the maximization of a concave function g. In such situation, where p^* is equal to its tightest lowerbound d^*, we understand this as **"strong duality"**.
Otherwise, when $p^* \geq d^*$, we phrase this situation as **weak duality**.

17.1.2 Partitioning Problem

Problem statement:

For some symmetric matrix $W \in \mathbb{S}^n$, solve:

$$\min_{\substack{\vec{x} \\ x_i^2=1}} \vec{x}^T W \vec{x}$$

this is not necessarily a convex optimization problem, and its combinatorial bruteforce solution for finding optimum has an exponential complexity in computation.

Let us first phrase the Lagrangian of this function:

$$\begin{aligned} L(\vec{x}, \vec{\nu}) &= \vec{x}^T W \vec{x} + \sum_i \nu_i (x_i^2 - 1) \\ &= \vec{x}^T (W + diag(\vec{\nu}))\vec{x} - \sum_i \nu_i \end{aligned}$$

Then, remind that,

$$g(\vec{\nu}) = \inf_{\vec{x}} L(\vec{x}, \vec{\nu})$$

where, if $W + diag(\vec{\nu}) \succcurlyeq 0$, then $L(\vec{x}, \vec{\nu})$ is a convex functino of $\vec{x}$, and because the quadratic term of g must be nonnegative, the minimum value of this function g becomes $-\sum_{i=1}^n \mu_i$.
Otherwise, in the case that $W + diag(\vec{\nu}) \preccurlyeq 0$, the minimum of this function becomes $-\infty$.
This phrases the function g as a piecewise function:

$$g(\vec{\nu}) = \begin{cases} -\sum_{i=1}^n \mu_i & \text{, if } Q \succcurlyeq 0 \\ -\infty & \text{, otherwise} \end{cases}$$

Consequently,

$$\begin{aligned} d^* &= \max_{\vec{\nu}} g(\vec{x}^*, \vec{\nu}) \\ &= \max_{\substack{\vec{\nu} \\ W+diag(\vec{\nu})\succcurlyeq 0}} -\sum_{i=1}^n \nu_i \end{aligned}$$

To maximize the above expression, we need that

$$\vec{\nu} = -\lambda_{\min}(W)\vec{1}$$

Meanwhile, we understand that $W - \lambda_{\min}(W)I \succcurlyeq 0$ by the eigenvalue shift property. Therefore, we may state that,

$$p^* \geq d^* = n\lambda_{\min}(W)$$

and conclude a case of weak duality.

17.2 Minmax Inequality

Let there be a primal problem p, and a dual problem d, such that:
Primal.

$$p^* = \min f_0(\vec{x}) + \sum_{i=1}^{m} \mathbb{1}_{<0}\{f_i(\vec{x})\} + \sum_{j=1}^{n} \mathbb{1}_0\{h_j(\vec{x})\}$$

Dual.

$$g(\vec{x}^*, \vec{\nu}) = L(\vec{x}, \vec{\lambda}, \vec{\nu})d^* = \max_{\substack{\vec{\lambda} \geq 0 \\ \vec{\nu}}} g(\vec{x}^*, \vec{\nu})$$

Then, the Minmax Inequality states:

Theorem 17.2.1. Minmax Inequality

For any sets $\mathcal{X}, \mathcal{Y}$, function f,

$$\min_{\vec{x} \in \mathcal{X}} \max_{\vec{y} \in \mathcal{Y}} f(\vec{x}, \vec{y}) \geq \max_{\vec{y} \in \mathcal{Y}} \min_{\vec{x} \in \mathcal{X}} f(\vec{x}, \vec{y})$$

Essentially, in a minimax perspective, going second is always more advantageous.

Proof. Consider some $\vec{x}_0 \in \mathcal{X}$, $\vec{y}_0 \in \mathcal{Y}$, then define

$$h(\vec{y}_0) := \min_{\vec{x} \in \mathcal{X}} f(\vec{x}, \vec{y}_0)$$
$$g(\vec{x}_0) := \max_{\vec{y} \in \mathcal{Y}} f(\vec{x}_0, \vec{y})$$

Then, let us perform the following algebraic process:

$$\begin{aligned} h(\vec{y}_0) &= \min_{\vec{x} \in \mathcal{X}} f(\vec{x}, \vec{y}_0) \\ &\leq f(\vec{x}_0, \vec{y}_0) \\ &\leq \max_{\vec{y} \in \mathcal{Y}} f(\vec{x}_0, \vec{y}) = g(\vec{x}_0) \end{aligned}$$

Then, along that inequality which works for any pair of points $(\vec{x}, \vec{y})$,

$$\max_{\vec{y}_0 \in \mathcal{Y}} h(\vec{y}_0) \leq \min_{\vec{x}_0 \in \mathcal{X}} g(\vec{x}_0)$$

The above translates to exactly the Minmax Inequality.

While we consider in an optimization problem that its dual is phrased as:

$$d^* = \max_{\vec{\lambda}, \vec{\nu}} g(\vec{\lambda}, \vec{\nu}) = \max_{\vec{\lambda}, \vec{\nu}} \min_{\vec{x}} L(\vec{x}, \vec{\lambda}, \vec{\nu})$$

we also consider the primal of an optimization problem to phrase,

$$\begin{aligned}
p^* &= \max_{\vec{\lambda},\vec{\nu}} L(\vec{x},\vec{\lambda},\vec{\nu}) \\
&= \max_{\vec{\lambda},\vec{\nu}} (f_o(\vec{x}) + \sum_i \lambda_i f_i(\vec{x}) + \sum_i \nu_i h_i(\vec{x})) \\
&= \begin{cases} \infty & \text{, upon any constraints } f_i, h_i \text{ violated} \\ f_0(\vec{x}) & \text{, if all constraints are unviolated} \end{cases}
\end{aligned}$$

So, provided that the constraints are unviolated in the primal problem,

$$p^* = \min_{\vec{x}} \max_{\vec{\lambda},\vec{\nu}} (f_o(\vec{x}) + \sum_i \lambda_i f_i(\vec{x}) + \sum_i \nu_i h_i(\vec{x}))$$

Therefore, along the Minmax Inequality, we may observe that:

$$p^* \geq d^*$$

Chapter 18

Strong Duality and Optimality Conditions

18.1 Strong Duality

We may observe a duality gap of some optimization problem as $p^* - d^*$; then, when is this duality gap 0, such that we achieve strong duality? The cndition of strong duality is phrased as follows:

Definition 18.1.1. Slater's Condition

Assume that the objective function $f_0(x)$ is convex, constraints $f_i(x)$ are convex, and $h_i(x)$ are affine.
Then, suppose

$$(\exists \text{ a } \textit{feasible } \vec{x}_0 \in ReInt(D) \wedge \forall i \in \{1, \dots, m\}(f(\vec{x_0}) < 0)) \implies (p^* = d^*)$$

where $ReInt(D)$ stands for the "relative interior" of the domain of f.
Here, *feasible* can also be phrased as fitting another statement existing in the constraint, say $A\vec{x}^* = \vec{b}$; in other words, being in the *feasible* region of an optimization.

We can further refine the above condition:

Definition 18.1.2. Refined Slater's Condition

Let us have an optimization problem:

$$p^* = \min_{\substack{\forall i \in \{1,\dots,k\}, f_i(x) \leq 0 \text{ and are affine} \\ \forall i \in \{k+1,\dots,m\}, f_l(x) \leq 0 \text{ and are non-affine} \\ \forall j \in \{1,\dots,n\}, h_j(x) = 0 \text{ and are affine}}} f_0(x)$$

Then, if there exists some $\vec{x}_0$ in the relative interior of domain of f and also within the feasible region of optimization such that $\forall l$, $f_l(\vec{x}) < 0$, we may determine that strong duality holds.

18.2 Optimality Conditions

A certificate helps to pose a upperbound to the difference between the primal and dual feasible point at a current setting $(point, parameter)$.

Definition 18.2.1. Certificate

Suppose we have a dual feasible point (λ, ν) and primal feasible point $\hat{x}$, then we may suppose that:

$$g(\lambda,\nu) \leq d^* \leq p^* \leq f(\hat{x})$$
$$f(\hat{x}) - p \leq f(\hat{x}) - g(\lambda,\nu) \leq \epsilon$$

Then, we obtain the following conclusion:

$$p^* \in [g(\lambda,\nu), f_0(\hat{x})]$$

and similarly for d^* upon strong duality.

We, in convex optimization, deal with errors between the primal and dual optimum of some optimization problem. This requires us to bound the error, and implies that our answers might not be the optimal solution. In other words, its optimality is not guaranteed.
So how do we find optimality in a solution?

We may attempt to find a defining condition for optimality to occur, where the primal and dual optimal solutions are the same (in the case we simplify a constrained primal optimization problem into an unconstrainted dual optimizartion problem).
Suppose our primal optimal point is $\vec{x}^*$ and dual optimal point is $\vec{\lambda}^*, \vec{\nu}^*$, then:

$$\begin{aligned} d^* &= \min_{\vec{x}} f_0(\vec{x}) + \sum \lambda_i^* f_i(\vec{x}) + \sum \nu_j^* h_j(\vec{x}) \\ &\leq f_0(\vec{x}^*) + \sum \lambda_i^* f_i(\vec{x}^*) + \sum \nu_j^* h_j(\vec{x}^*) \\ &\leq \min_{\vec{x}} f_0(\vec{x}) = p^* \end{aligned}$$

If the second line of inequality holds, then we observe $\vec{x}^*$ as the minimizer of L.

$$\nabla_{\vec{x}} L(\vec{x}, \vec{\lambda}, \vec{\nu}) = \vec{0}$$

Furthermore, suppose the third line of inequality that $\sum_i \lambda_i^* f_i(\vec{x}) = 0$. Since this is a sum of nonpositive values, if strong duality holds such that $p^* = d^*$, the following condition called **complementary slackness** us be satisfied:

$$\forall i \in \{1, \ldots, n\} \lambda_i^* f_i(\vec{x}) = 0$$

Complementary slackness can be utilized in the case where strong duality holds, and the constraints are differentiable; therefore, this technique is also suited for non-convex problem.

Now, we further extend upon these arguments to form the KKT conditions:

Definition 18.2.2. KKT Condition

KKT (Karush Kuhn Tucker) Condition is a series of conditions where, assuming strong duality holding, we may observe the following statements being satisfied at an optimal point, along each partition of the conditions:
Stationarity:

$$\forall i \in \{1, \ldots, m\}, \nabla f_0(x^*) + \sum \lambda_i^* \nabla f_i(x^*) + \sum \nu_j^* \nabla h_j(x^*) = 0$$

Primal Feasibility:

$$\begin{cases} \forall i \in \{1, \ldots, m\}, & f_i(x^*) \leq 0 \\ \forall j \in \{1, \ldots, n\}, & h_j(x^*) = 0 \end{cases}$$

Dual Feasibility:

$$\forall i \in \{1, \ldots, m\}, \lambda_i^* \geq 0$$

Complementary Slackness:

$$\forall i \in \{1, \ldots, m\}, \lambda_i^* f_i(\vec{x}) = 0$$

If the point is optimal, then KKT condition holds; however, the converse is not necessarily true. Then, restricting the space of points satisfying KKT is not really a restirction for optimality, but every point that satisfies KKT need not be optimal.
However, if our optimization problem is convex, then KKT conditions are sufficient to indicate optimality such that:

If $(\tilde{x}, \tilde{\lambda}, \tilde{\nu})$ satisfies KKT condition, then $\tilde{x}$ is a primal optimum and $(\tilde{\lambda}, \tilde{\nu})$ is a dual optimum.

Summary. As we ensure the strong duality of a problem by Slater's Condition, we then check for KKT condition to restrict ourselves onto a subset with possibly optimal points; and, in convex optimization problems, KKT conditions guarantee the optimality of its satisfying points.

18.3 Linear Program

Definition 18.3.1. Linear Program

A linear program is an optimization in the form of:

$$p^* = \min_{\substack{\forall i \in \{1,\ldots,m\}, \vec{a}_i^T \vec{x} \leq b_i \\ \forall j \in \{1,\ldots,n\}, \vec{u}_j^T \vec{x} = v_j}} (\vec{c}^T \vec{x})$$

where its constraints are all affine. Equivalently, we may writea linear program as

$$p^* = \min_{\substack{A\vec{x} \leq \vec{b} \\ U\vec{x} = \vec{v}}} (\vec{c}^T \vec{x})$$

Then, the Lagrangian of a linear program can be derived as:

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) = \vec{c}^T \vec{x} + \sum_i \lambda_i (A\vec{x} - \vec{b}) + \sum_i \nu_j (U\vec{x} - \vec{v})$$
$$g(\vec{\lambda}, \vec{\nu}) = \min_{\vec{x}} L(\vec{x}, \vec{\lambda}, \vec{\nu})$$

Then, at some optimal point $\vec{x}^*$, we observe that

$$\nabla_{\vec{x}} L(\vec{x}, \vec{\lambda}, \vec{\nu}) = c + A^T \lambda + U^T \nu = 0$$

Therefore, to derive for the optimal solution, we may find an equivalent optimization problem to work on:

$$\max_{\substack{\forall i \in \{1,\ldots,m\}, \lambda_i \geq 0 \\ c + A^T \lambda + U^T \nu = 0}} -(b^T \vec{\lambda} + \vec{\nu}^T \vec{v})$$

Thus, the dual of a linear program can also be phrased as a linear program.

Chapter 19

Duality and Shadow Prices

19.1 A Review of KKT Condition

Suppose the following necessary conditions apply:

> - Strong duality holds
> - The objective function and constraint functions are differentiable

And suppose that $\vec{x}^*$ is primal optimal, and $(\vec{\lambda}^*, \vec{\nu}^*)$ is dual optimal.
Then, their optimality implies the following "KKT Conditions":

Definition 19.1.1. KKT Condition

KKT (Karush Kuhn Tucker) Condition is a series of conditions where, assuming strong duality holding, we may observe the following statements being satisfied at an optimal point, along each partition of the conditions:
Stationarity (alternatively, First Order Condition):

$$\forall i \in \{1, \ldots, m\}, \nabla f_0(x^*) + \sum \lambda_i^* \nabla f_i(x^*) + \sum \nu_j^* \nabla h_j(x^*) = 0$$

Primal Feasibility:

$$\begin{cases} \forall i \in \{1, \ldots, m\}, & f_i(x^*) \leq 0 \\ \forall j \in \{1, \ldots, n\}, & h_j(x^*) = 0 \end{cases}$$

Dual Feasibility:

$$\forall i \in \{1, \ldots, m\}, \lambda_i^* \geq 0$$

Complementary Slackness:

$$\forall i \in \{1, \ldots, m\}, \lambda_i^* f_i(\vec{x}) = 0$$

Different from KKT Conditions are the sufficient conditions of optimality:

Definition 19.1.2. Sufficient Condition of Optimality

Suppose we have some pair of solutions $(\vec{\tilde{x}}, \vec{\tilde{\lambda}}, \vec{\tilde{\nu}})$, and all functions f are convex, all functions h are affine. Then, if all five KKT conditions are satisfied on such solution, then we may conclude that,

$$\begin{cases} \vec{\tilde{x}} & \text{must be primal optimal} \\ (\vec{\tilde{\lambda}}, \vec{\tilde{\nu}}) & \text{must be dual optimal} \end{cases}$$

The difference between KKT condition per se and sufficient condition lies on the direction of their implication statements.

19.2 Dual of a Linear Program

Following from last lecture, we obtained that the dual of a linear program can also be phrased as a linear program:

$$\begin{cases} \text{Primal:} \quad p^* = \min\limits_{\substack{A\vec{x}\leq\vec{b} \\ U\vec{x}=\vec{v}}} (\vec{c}^T\vec{x}) \\ \text{Dual:} \quad d^* = \max\limits_{\substack{\forall i\in\{1,\ldots,m\},\lambda_i\geq 0 \\ c+A^T\lambda+U^T\nu=0}} -(b^T\vec{\lambda}+\vec{\nu}^T\vec{v}) \end{cases}$$

Now, consider the case where we maximize a linear program instead:

$$\text{Primal: } p^* = \max_{\substack{A\vec{x}\leq\vec{b} \\ U\vec{x}=\vec{v}}} (\vec{c}^T\vec{x})$$

this is a convex optimization problem.
Meanwhile, noticing that for a concave function f,

$$\max f(\vec{x}) = -(\min -f(\vec{x}))$$

we may successfully optimize for a concave maximization problem via converting it into a concave minimization problem.
Following such logic, we may phrase the Lagrangian of linear program maximization as:

$$L(\vec{x},\vec{\lambda}) = \vec{c}^T\vec{x} + \vec{\lambda}^T(-(A\vec{x}-\vec{b}))$$
$$g(\vec{\lambda}) = \max_{\vec{x}} L(\vec{x},\vec{\lambda})$$

such that, if $A\vec{x} > \vec{b}$, then the Lagrangian's value decreases and cannot reach the maximum of objctive function. Therefore, we may perform the following primal-dual translation:

$$\begin{cases} \text{Primal:} \quad p^* = \max\limits_{A\vec{x}\leq\vec{b}} (\vec{c}^T\vec{x}) \\ \text{Dual:} \quad d^* = \min\limits_{\substack{\forall i\in\{1,\ldots,m\},\lambda_i\geq 0 \\ c-A^T\lambda=0}} (b^T\vec{\lambda}) \end{cases}$$

19.3 Shadow Prices

Suppose that we currently consider the following optimization problem:

> - We have 200 units of merlot grapes, where it earns λ_1 currency unit per grape unit.
> - We have 300 units of shiraz grapes, where it earns λ_2 currency unit per grape unit.
> - One red blend (q_1) costs 4 units of merlot and 1 unit of shiraz, while earning 20 currency units.
> - One blue blend (q_2) costs 2 units of merlot and 3 unit of shiraz, while earning 15 currency units.

We may phrase the business on wine mathematically as:

$$p^* = \max_{\substack{4q_1+2q_2\leq 200 \\ q_1+3q_2\leq 300}} 20q_1 + 15q_2$$

However, we may also phrase the business on grapes (instead of selling wine) as:

$$\begin{aligned}
&\max_{q_1,q_2} 20q_1 + 15q_2 + \lambda_1(200 - 4q_1 - 2q_2) + \lambda_2(300 - q_1 - 3q_2) \\
&\quad = \max_{q_1,q_2} (20 - 4\lambda_1 - \lambda_2)q_1 + (15 - 2\lambda_1 - 3\lambda_2)q_2 + 200\lambda_1 + 300\lambda_2
\end{aligned}$$

Note that, suppose both coefficients $(20 - 4\lambda_1 - \lambda_2)$ and $(15 - 2\lambda_1 - 3\lambda_2)$ are 0, then the values of q_1 and q_2 poses no significance as their coefficient is zero.
This leads to the minimization problem:

$$\min_{\substack{20-4\lambda_1-\lambda_2=0 \\ 15-2\lambda_1-3\lambda_2=0}} 200\lambda_1 + 300\lambda_2$$

which is exactly the dual of what optimization problem suggests p^*.

Chapter 20

Linear Programs

20.1 Mathematically Expressing Linear Programs

Definition 20.1.1. Linear Program

A linear program is an optimization problem with a linear objective function and linear constraints.

LPs (linear programs) are genreally phrased as:

$$\min_{A\vec{x}\leq\vec{b}} \vec{c}^T\vec{x}$$

and standardly phrased as:

$$\min_{\substack{\vec{x}\geq 0\\ A\vec{x}=b}} \vec{c}^T\vec{x}$$

Interestingly, any linear program can be written in a standard form.

20.1.1 Translating General Forms into Standard Forms

Here, we provide a part-by-part process on how to translate LPs in General Forms into Standard Forms.

Translating Constraints.

For any constraint f_i:

$$\sum_{j=1}^{n} A_{i,j}x_j \leq b_j$$

we may introduce a slack variable $\xi_i \geq 0$, such that we rephrase the above constraint as:

$$\sum_{j=1}^{n} a_{i,j}x_j + \xi_i = b_i$$

such that the dimensionality of our problem has increased (which increases the computational cost), but we may rephrase inequality constraints as equality constraints.

Dealing with x_i that are not constrained to be positive.

Any nonpositive number can be written as the difference of two positive numbers.
Using this mathematical fact, we may state that:

$$x_i = x_i^+ - x_i^-, \text{ where } x_i^+, x_i^- \geq 0$$

Let us demonstrate the translation process with the following examples:

Explain 20.1.1. Example in Translating LP Across Forms

For the following provided problem:

$$\min_{\substack{x_1+x_2\geq 3\\ 2x_1+2x_2=14\\ x_1\geq 0}} \quad 2x_1 + 4x_2$$

translate it into its standard form.

Let us first rephrase that,

$$x_2 = x_2^+ - x_2^-, \text{ where } x_2^+, x_2^- \geq 0$$

and introduce a new variable x_3 as a slack variable such that:

$$x_1 + x_2 - x_3 = 3$$

Therefore, we translate the linear program into its standard form as:

$$\min_{\substack{x_1,x_2^+,x_2^-,x_3\geq 0\\ x_1+x_2^+-x_2^--x_3=3\\ 3x_1+2x_2^+-2x_2^-=14}} \quad (2x_1 + 4x_2^+ - 4x_2^-)$$

Now, let us look at a slightly more complicated example:

Explain 20.1.2. Example in Translating LP Across Forms

For the following presented problem:

$$\min_{\substack{x_1+2x_2\leq 3\\ 2x_1+x_2\leq 3}} \begin{bmatrix} -1 \\ -1 \end{bmatrix}^T \vec{x}$$

translate such LP into its standard form.

We may introduce slack variables $\xi_1, \xi_2 \geq 0$, such that we rephrase the inequality constraints as

$$\begin{cases} x_1 + 2x_2 \leq 3 & \rightarrow x_1 + 2x_2 + \xi_1 = 3 \\ 2x_1 + x_2 \leq 3 & \rightarrow 2x_1 + x_2 + \xi_2 = 3 \end{cases}$$

Furthermore, as both x_1 and x_2 are not constrained to be nonnegative, we would add four more variables to rephrase them as the arithmetic results of nonnegative numbers:

$$x_1 = x_1^+ - x_1^-, \text{ where } x_1^+, x_1^- \geq 0x_2 \qquad = x_2^+ - x_2^-, \text{ where } x_2^+, x_2^- \geq 0$$

The resulting standard form of LP would then be:

$$\min_{\substack{x_1^+,x_1^-,x_2^+,x_2^-,\xi_1,\xi_2\geq 0\\ x_1^+-x_1^-+2(x_2^+-x_2^-)+\xi_1=3\\ 2(x_1^+-x_1^-)+x_2^+-x_2^-+\xi_2=3}} \begin{bmatrix} -1 \\ -1 \end{bmatrix}^T \vec{x}$$

Furthermore, as we observe the graph of the above linear program, we will discover that the minimum lies on the intersection.
Alternatively, we may find the dual problem of this linear program, and discover that

We will phrase the observation again as follows:

> For linear programs, its optimum lies on what we call an extreme point, where the two constraints of our aforementioned LP in Example 20.1.2 intersect.
> Here is a temporary definition to satiate your curiosities:
>
> **Definition 20.1.2. Extreme Point**
>
> A point p of a contex set $\mathcal{S}$ is an extreme point if each line segment that lies completely in $\mathcal{S}$ and contains p has p as an endpoint.
> An extreme point is also called a corner point.
> Quoted per written this source.

But that's for next section. Let's come back on track.

20.2 Interior Point

Let us define the interior points of a linear program's feasible region:

Definition 20.2.1. Interior Point

We may say $\vec{x}$ is an interior point of some set $\mathcal{X}$ if

$$\exists \epsilon > 0, (\forall \vec{z} \in \mathcal{X}, \|\vec{x} - \vec{z}\|_2 \leq \epsilon)$$

Hereby, we introduce the following theorem:

Theorem 20.2.1. Optimality of Extreme Point

Theorem. For a linear program:

$$\min_{\vec{x} \in \mathcal{X}} \vec{c}^T \vec{x}$$

where $\mathcal{X}$ is a closed set such that it contains its boundary, if $\vec{x}^*$ is an optimal solution, then $\vec{x}^*$ exists on the boundary of $\mathcal{X}$.

Proof. Suppose that $\vec{x}^*$ is the optimizing argument of the linear program. For the sake of contradiction, suppose that $\vec{x}^*$ is an interior point of the feasible region $\mathcal{X}$, such that

$$\exists \epsilon > 0, (\forall \vec{z} \in \mathcal{X}, \|\vec{x} - \vec{z}\|_2 \leq \epsilon)$$

Now, consider $\vec{z} = \vec{x}^* - \alpha \vec{c}$, such that $\|\vec{x}^* - \vec{z}\|_2 = \|\alpha \vec{c}\|_2 < \epsilon$.
Then,

$$\begin{aligned}\vec{c}^T \vec{z} &= \vec{c}^T (\vec{x}^* - \alpha \vec{c}) \\ &= \vec{c}^T \vec{x}^* - \alpha \|\vec{c}\|_2^2 \\ &\leq \vec{c}^T \vec{x}^*\end{aligned}$$

We reach the contradiction against the assumption that $\vec{x}^*$, the optimizing argument, is an interior point.
Consequently, the optimal argument $\vec{x}^*$ can never be an interior point, and must lie on the boundary of $\mathcal{X}$.

Furthermore, linear programs' optimal arguments are at what we call an extreme point.
To prove this fact, let us first provide some geometrical definitions to work along:

Definition 20.2.2. Polyhedron

A polyhedron is a set:

$$\mathcal{P} = \{\vec{x} \in \mathbb{R}^n | A\vec{x} \geq \vec{b}, C\vec{x} = \vec{d}, A \in \mathbb{R}^{m\times n}, \vec{b} \in \mathbb{R}^m\}$$

in other words, some set whose points are defined by a linear constraint.
Keep in mind that not all polyhedrons are not closed sets, but are always convex.

Definition 20.2.3. Extreme Point

Let $\mathcal{P}$ be some polyhedron.
Then, $\vec{x}$ is an extreme point if we cannot find two vectors $\vec{y}, \vec{z} \in \mathcal{P}$ (that are distinct form $\vec{x}$ itself) and $\theta \in [0, 1]$ such that $\vec{x} = \theta\vec{y} + (1-\theta)\vec{z}$.
Written as an implication:

$$\left[\ \nexists \vec{y}, \vec{z} \in (\mathcal{P} - \{\vec{x}\}), \theta \in [0,1]\ \big(\vec{x} = \theta\vec{y} + (1-\theta)\vec{z}\big)\ \right] \Leftrightarrow \vec{x} \text{ is an extreme point}$$

Therefore, this is to say that an extreme point is any point in the polyhedron that cannot be expressed as the convex combination of any other two vectors within the polyhedron.

Theorem 20.2.2. Unbounded Polyhedron and Extreme Points

Theorem. Let $\mathcal{P}$ has an extreme point iff $\mathcal{P}$ does not contain a line.

Argument. Suppose that we have a polyhedron that involves an extreme point where at least two boundaries intersect, then any line must intersect at least one boundary that the polyhedron $\mathcal{P}$ involves.

Chapter 21

Quadratic Program

21.1 Linear Programs, continued

Theorem 21.1.1. Existence of Solution in Linear Program

Theorem. Consider some linear program:

$$\min_{\vec{x}\in\mathcal{P}} \vec{c}^T\vec{x}$$

Assume $\mathcal{P}$ has at least one extreme point, and an optimal solution exists and is finite. Then, there exists an optimal solution that is an extreme point of $\mathcal{P}$.

Proof. Because $\mathcal{P}$ has at least one extreme point, it does not contain a line.
Then, suppose that v^* is an optimal solution of the addressed linear program, where the solution exists by assumption of theorem. We may phrase all possible points presenting this optimal solution to be:

$$Q = \{\vec{x}|\vec{x}\in\mathcal{P}, \vec{c}^T\vec{x} = \vec{v}^*\}$$

which is a polyhedron and a subset of $\mathcal{P}$. Consequently, Q cannot contain a line, and must contain an extreme point.
Suppose $\vec{x}^*$ is an extreme point of Q.
Now, suppose for the sake of contradiction that $\vec{x}^*$ is not an extreme point of $\mathcal{P}$, such that:

$$\exists\vec{y},\vec{z}\in(\mathcal{P}-\{\vec{x}^*\}), \theta\in(0,1), \left(\vec{x}^* = \theta\vec{y}+(1-\theta)\vec{z}\right)$$

Therefore,

$$\begin{aligned}\vec{c}^T\vec{x}^* &= v^*\\ \vec{c}^T[\theta\vec{y}+(1-\theta)\vec{z}] &= v^*\\ \vec{c}^T\vec{x}^* &= \vec{c}^T[\theta\vec{y}+(1-\theta)\vec{z}]\\ &\geq \theta\vec{c}^T\vec{y}+(1-\theta)\vec{c}^T\vec{z} = v^*\end{aligned}$$

Therefore, for the equality to be satisfied, it must be that

$$\vec{y} = \vec{z} = \vec{x}^*$$

which breaks the premise that $\vec{x}^*$ is an interior point (that $\vec{y}\neq\vec{x}^*$ and so for $\vec{z}$).
Thus, by the above constraint, $\vec{x}^*$ is not an interior, but an extreme point of $\mathcal{P}$.

21.2 Quadratic Programs

Definition 21.2.1. Quadratic Program

A quadratic program is an optimization problem with a quadratic objective function and linear constraints.

QPs (quadratic programs) are genreally phrased as:

$$\min_{\substack{A\vec{x}\leq\vec{b}\\ F\vec{x}=\vec{d}}} \frac{1}{2}\vec{x}^T H\vec{x} + \vec{c}^T\vec{x}$$

where H need be positive-semidefinite and symmetric to guarantee convexity.

21.2.1 Solving an Unconstrained QP

Consider the quadratic program:

$$p^* = \min_{\vec{x}\in\mathbb{R}^n} \frac{1}{2}\vec{x}^T H\vec{x} + \vec{c}^T\vec{x} + d$$

Case 1. Suppose one case where,

$$H \succcurlyeq 0, \vec{c} \in \mathcal{R}(H)$$

Then, let $H\vec{x}_0 = -\vec{c}$, and keep in mind that H is summetric.

$$\begin{aligned} f(\vec{x}) &= \frac{1}{2}(\vec{x}-\vec{x}_0)^T H(\vec{x}-\vec{x}_0) + \alpha \\ &= \frac{1}{2}\vec{x}^T H\vec{x} + \frac{1}{2}\vec{x}_0^T H\vec{x}_0 - \vec{x}_0^T H\vec{x} + \alpha \end{aligned}$$

Then, we may figure that the minimizing argument of this objective function is $\vec{x}_0$ and $p^* = \alpha$.
Then, suppose H is invertible, we find that $\vec{x}_0 = -H^{-1}\vec{c}$

Case 2. Suppose one other case where,

$$H \succcurlyeq 0, \vec{c} \notin \mathcal{R}(H)$$

In such case, we may phrase that $\vec{c} = -H\vec{x_0} + \vec{r}$, where $\vec{r} \in \mathcal{R}(H)^\perp = N(H^T) = N(H)$ by the Fundamental Theorem of Linear Algebra.
Then,

$$\begin{aligned} f(k\vec{r}) &= \frac{1}{2}(k\vec{r})^T H(k\vec{r}) + \vec{c}^T(k\vec{r}) + d \\ &= 0 + (-H\vec{x}_0 + \vec{r}^T k\vec{t} + d \\ &= k\|\vec{r}\|_2^2 - k\vec{x_0}^T H^T\vec{r} + d \\ &= k\|\vec{r}\|_2^2 + d \\ \lim_{k\to-\infty} f(k\vec{r}) &= -\infty \end{aligned}$$

Case 3. Suppose one last case where, there exists a negative eigenvalue in H.

Then, as the objective function is concave, its minimum is $-\infty$.

Chapter 22

More on Quadratic Programs, and SOCPs

I know the way chapters are divided is disastrous in terms of topic-wise content organization, but I'll continue the current division style becuase I want to number each chapter by their respective lecture number.

22.1 More on Quadratic Programs

22.1.1 Categories of Quadratic Program

Let us formally define an unconstrained quadratic program:

Definition 22.1.1. Unconstrained Quadratic Programs

An unconstrained QP is generally stated as:

$$\min_{A\vec{x}\leq\vec{b}} \frac{1}{2}\vec{x}^T H\vec{x} + \vec{c}^T\vec{x} + \vec{d}$$

where, since this is a QP, we have a quadratic objective function and linear constraints.
For any QP with a convex objective function, the optimization problem itself is convex.

Furthermore, we discovered that, in the above definition,

> If $H \succcurlyeq 0$ and $\vec{c} \in \mathcal{R}(H)$, then we may have an explicit solution.
> Else, the primal solution of this QP is $p^* = -\infty$.

Meanwhile, a constrained QP may be defined as follows:

Definition 22.1.2. Constrained Quadratic Program

A constrained QP is generally expressed as:

$$\min_{A\vec{x}=\vec{b}} \frac{1}{2}\vec{x}^T H\vec{x} + \vec{c}^T\vec{x} + \vec{d}$$

For a constrained QP:

$$\min_{A\vec{x}=\vec{b}} \frac{1}{2}\vec{x}^T H\vec{x} + \vec{c}^T\vec{x} + \vec{d}$$

Let A have a nullspace such that N is the basis for $\mathcal{N}(A)$, such that if $\vec{x}_0$ is some solution of the optimization problem, the feasible set is expressed as

$$\{\vec{x} | \vec{x} = \vec{x}_0 + N\vec{z}\}$$

Therefore, we may rewrite the constrained QP as:

$$\begin{aligned}
&\min_{\substack{\vec{x}=\vec{x}_0+N\vec{z} \\ \vec{z}}} \frac{1}{2}(\vec{x}_0 + N\vec{z})^T H(\vec{x}_0 + N\vec{z}) + \vec{c}^T(\vec{x}_0 + N\vec{z}) + d \\
&= \min_{\vec{z}} \frac{1}{2}\vec{z}^T(N^T HN)\vec{z} + (\vec{x}_0^T HN + \vec{c}^T N)\vec{z} + \frac{1}{2}\vec{x}_0^T H\vec{x}_0 + \vec{c}^T\vec{x}_0 + d
\end{aligned}$$

which is an unconstrained QP.

22.1.2 Applications of QP

LQR. One famous application of Quadratic Programs is the Linear-Quadratic Regulator (LQR) Problem, where

> Provided some model $\vec{x}(t+1) = A\vec{x}(t) + B\vec{u}(t)$, we minimize the cost of operation:

$$\min \sum_{t=0}^{T} \|\vec{u}(t)\|_2^2 + \|\vec{x(T)} - \vec{g}\|_2^2$$

LASSO. Another famous application of Quadratic Program is Least Squares (and Ridge Regression) Algorithm (an unconstrained quadratic program).
And notably, the LASSO regularization,

$$\min_{\vec{x}} \|A\vec{x} - \vec{b}\|_2^2 + \lambda\|\vec{x}\|_1$$

is also a quadratic program, as it can be rephrased into a problem of two variables x_i and $t_i = |x_i|$.

$$\min_{\substack{\vec{x},\vec{t} \\ |x_i|=t_i \\ t_i \geq 0}} \|A\vec{x} - \vec{b}\|_2^2 + \lambda \sum_{i=1}^{n} t_i$$

and then relaxed as:

$$\min_{\substack{\vec{x},\vec{t} \\ \forall i, -t_i \leq x \leq t_i \\ t_i \geq 0}} \|A\vec{x} - \vec{b}\|_2^2 + \lambda \sum_{i=1}^{n} t_i$$

Why does this relaxation preserve the original minimum?
Suppose that for some i, $\exists \epsilon > 0, t_i = x_i + \epsilon$, such that we have some nonzero slack, and find that another better optimal exists at ${t_i}' = t_i - \epsilon$ where the regularizing term becomes smaller.
However, we would in turn find a better optimum than x. Therefore, it must be that the true optimum exists where there is zero slack: where the pre-relaxed constraints hold.

22.1.3 Another Example of Relaxing an Optimization Problem

Let us look at the piecewise linear filling problem, where we observe that for a set $\{y_1, \ldots, y_n\}$ and piecewise constants $\{x_1, \ldots, x_n\}$,

$$\forall i, y_i = x_i + noise$$

The problem deamnds us to find $\hat{\vec{x}}$ such that $\hat{\vec{x}}$ does not change on consecutive timesteps.

To solve this problem, let us define a matrix

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$

such that,

$$-D\vec{x} = \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ \vdots x_n - x_{n-1} \end{bmatrix}$$

and we would like that the cardinality of $D\vec{x}$ is $Card(D\vec{x}) \leq k$. Therefore, we may define the Piecewise Linear Filling problem as:

$$\min_{card(D\hat{\vec{x}}) \leq k} \|\vec{y} - \hat{\vec{x}}\|_2^2$$

We will then relax the problem to have a L1-norm cosntraint instead of a relaxed one:

$$\min_{\|\vec{x}\|_1 \leq k} \|\vec{y} - \hat{\vec{x}}\|_2^2$$

A class of methods is proficient at solving these problems efficiently, called the Interior Point Methods.

22.2 Second Order Cone Programs (SOCP)

Let us first define a "con":

Definition 22.2.1. Cone

A set $\mathcal{K}$ is a cone if:

$$\forall \alpha \geq 0, (\vec{x} \in \mathcal{K} \implies \alpha\vec{x} \in \mathcal{K})$$

For example,

$$\{(x_1, x_2) \big| |x_1| \leq x_2\}, \{(x_1, x_2) \big| x_2 \geq 0\}$$

Furthermore, we may define a convex cone as a cone that is also convex:

Definition 22.2.2. Convex Cone

Cone but convex. – Prof. Ranade, 10:40AM April 6th 2023 at Lewis 100

Next, we may define a polyhedral cone as follows:

Definition 22.2.3. Polyhedral Cone

Suppose we have a polyhedron $\mathcal{P} = \{\vec{x} \big| A\vec{x} - \vec{b}\}$.
Then, a polyhedral cone is the set:

$$\mathcal{K} = \{(\vec{x}, t) \big| A\vec{x} \leq \vec{b}t, t \in \mathbb{R}, t \geq 0\}$$

And, a second order cone may thus be defined as:

$$\mathcal{K} = \{(\vec{x}, t) \big| \|\vec{x}\|_2 \leq t\} \subset \mathbb{R}^{|\vec{x}|+1}$$

Now, let us consider this general class of problem:

$$\min \vec{g}^T\vec{x} \text{ s.t. } \forall i \in \{1,\dots,m\}, \|A_i\vec{x}+b_i\|_2 \le \vec{c}_i^T\vec{x}+d_i$$

It can be in fact summarized as a SOCP, where suppose:

$$\begin{cases} \vec{y} & = A_i\vec{x}+b_i \\ t & = \vec{c}_i^T\vec{x}+d_i \end{cases}$$

then the problem has a constraint equivalent to $(\vec{y},t)$ being contained by some second order cone.
With traingle inequality we may prove that the feasible set of this problem, $\{\vec{x}\big|\|A_i\vec{x}+b_i\|_2 \le \vec{c}_i^T\vec{x}+d_i\}$ is convex.
You can't imagine how long I've waited to drop this line, but,

> This is left as an exercise to the reader.

22.2.1 Facility Location Problem

Suppose we have a few asylums spread across the country (eg., University of California, Berkeley; University of Currying, Brocolli; University of Critish Bolumbia...), how may we allocate resources across these centers to minimize the amount of commute in distributing resources?
We may formulate this problem as:

$$\min_{\vec{x}} \frac{1}{m}\sum_{i=1}^{m} \|A_i\vec{x}-b_i\|_2$$

and the argument along which we may debate this is a SOCP is that we may first state,

$$z_i = \|A_i\vec{x}-b_i\|_2$$

such that we now have a linear objective function, and the problem is now rephrased as:

$$\min_{\substack{x \\ \|A_i\vec{x}-b_i\|_2 \le z_i}} \sum_{i=1}^{n} z_i$$

Furthermore, we have thus developed a constraint in the form of $\|A_i\vec{x}+b_i\|_2 = \vec{c}_i^T\vec{x}+d_i$. We may then relax the constraint to obtain inequality constraints instead, with optimum preserved.

Furthermore, we may perform customizations to maximize specific utilities, where, for example, the asylum with least resources available (my university in this case?) would need the better treatment from this problem.
Through this expression of "min-max utility", upon which we rephrase the problem as:

$$\min_x \max_i \|\vec{x}_i - \vec{y}_i\|_2$$

we may then rephrase this as a SOCP problem.
Once again, the context of the problem is to minimize $\vec{x}$ on the maximum possible distance (the worst-off asylum's situation of receiving no funding for education).
Now, suppose that $\max_i \|\vec{x}_i - \vec{y}_i\|_2 = R$. Then, we may simplify the problem into

$$\min_{\substack{x,R \\ \forall i, \|A_i\vec{x}-b_i\|_2 \le R}} R$$

which is a SOCP problem.

Chapter 23

LASSO

23.1 L1-norm Optimization

We have performed regularization using L2 norm, and arrived at Ridge Regression. What if, instead, we use the L1 norm to regularize a cost function, where the problem is phrased as follows:

$$\min_{\substack{\vec{x} \\ A\vec{x}=b}} \|\vec{x}\|_1$$

Recognizing that essentially, $|x| = x \times sgn(x)$, we may transform the above optimization function into the following linear program:

$$\min_{\substack{\vec{x} \\ A\vec{x}=b}} \sum_{i=1}^{n} sgn(x_i) \times x_i$$

And we may introduce variables:

$$|x_i| = x_i^+ + x_i^-, \text{ where: } \begin{cases} x_i^+ & = x_i \text{ if } x_i > 0\text{; otherwise it is 0} \\ x_i^- & = x_i \text{ if } x_i < 0\text{; otherwise it is 0} \end{cases}$$

to rewrite the above problem as

$$\min_{\substack{\vec{x} \\ A(\vec{x}^+ - \vec{x}^-)=b}} \sum_{i=1}^{n} x_i^+ + \sum_{i=1}^{n} x_i^-$$

This problem now introduces $\vec{x}^+$ and $\vec{x}^-$ without constraining them, but we understand that for their sum to capture the L1-norm, it must be that the solution $\vec{x}^{+*}, \vec{x}^{-*}$ cannot be at the same time nonzero.
Here, we may realize that if $x_i^+ > 0$ and $x_i^- > 0$, then, for continuing to conform the constraint, we may present a new solution $x_i^{+\prime}, x_i^{-\prime}$ such that,

$$\text{For some } \epsilon \in \mathbb{R}, \begin{cases} x_i^{+\prime} = x_i^+ - \epsilon \\ x_i^{-\prime} = x_i^- - \epsilon \end{cases}$$

$$x_i^{+\prime} - x_i^{-\prime} = x_i^+ - x_i^-$$

$$A(x_i^{+\prime} - x_i^{-\prime}) = A(x_i^+ - x_i^-) = b_i$$

Furthermore, while this new solution satisfies the constraint, the objective function value decreases by 2ϵ.
Therefore, we may always create a more optimal solution than a supposed optimum at where both of x_i^+, x_i^- are nonzero.
Consequently, the optimum exists where for all i, either x_i^+ or x_i^- is 0; there must be one non-zero and one zero component.

23.1.1 L1 Least Squares

Now, let's let the least squares problem participate in the discussion:

$$\min_{\vec{x}} \|A\vec{x} - \vec{b}\|_1$$

where, we may rephrase this problem as:

$$\min_{\substack{\vec{x},\vec{e} \\ A\vec{x}-\vec{b}=\vec{e}}} \quad \|\vec{e}\|_1$$

to solve it in a aforementioned way.

Alternatively, let us also consider the following optimization problem that is related to the above:

$$\min_{\vec{x}} \|\vec{x} - \vec{b}\|_1$$

where each element of $\vec{x}$ are equal. To minimize such function, we may analyze the gradient of the above objective function with respect to $\vec{x}$:

$$\frac{\partial \|\vec{x} - \vec{b}\|_1}{\partial x_i} = \begin{cases} 1 & , x_i \geq b_i \\ -1 & , x_i < b_i \\ DNE & , x_i = b_i \end{cases}$$

Additionally, let us consider the following notion:

Definition 23.1.1. Critical Point

Critical point of a function is a point where the derivative of that function is either 0, undefined, or at the boundary of its function.

The derivative of the objective function, $\sum_{i=1}^{n} |x - b_i|$, is thus equal to 0 if we let x become the median of $\vec{b}$.
The implication of this is that median is a more robust solution than the mean, because it is less easily influenced by outliers of a dataset (particularly, outliers influence the mean as they participate in its value, but outliers rarely participate in the detemrination of a median).

An alternative persepctive of the above problem follows. Suppose that $\vec{b} \in n$, where the index m signifies the location of median of $\vec{b}$.
Then, as we phrase the optimization problem above as:

$$\min_{x} |x - b_m| + \sum_{\substack{i=1 \\ i \neq m}}^{n} |x - b_i|$$

we find that the latter term remains invariant in its derivative when $x \in (b_{m-1}, b_{m+1})$.
Therefore, to minimize the entire objective function, we may focus on minimizing the former term of the objective function; consequently, we arrive once again at the solution that $x = b_m$.

23.2 LASSO (L1) Regularization

Now, we attempt to solve the following regularized least squares version, known as LASSO:

$$\min_{\vec{x}} \|A\vec{x} - \vec{b}\|_2^2 + \lambda \|\vec{x}\|_1$$

where we regularize least squares solution with the L1-norm of it.
Alternatively, we phrase the problem as:

$$\min_{\substack{\vec{x} \\ \|\vec{x}\|_1 \leq t}} \quad \|A\vec{x} - \vec{b}\|_2^2$$

We would see that the second constraint of the above problem, $\|\vec{x}\|_1 \leq t$, form polytopes whose vertices locate on the dimensions of a coordinate system.
Consequent of the polytope's relationship with coordinate dimensions, the result of LASSO usually yield zero-components in its solution $\vec{x}^*$. In other words, LASSO encourages sparsity (where the solution vector contains many zero). In the machine learning context, this is by itself a selection on some subset of feature.

23.2.1 A Demonstration of LASSO in Scalar Case

Let us consider the scalar version of LASSO problem:

$$\min_x \frac{1}{2}\sum_{i=1}^{n}(a_i x - b_i)^2 + \lambda|x|$$

Then, the derivative of such problem can be computed as:

$$\begin{cases} \sum_i a_i(a_i x - b_i) + \lambda & \text{, if } x > 0 \\ \sum_i a_i(a_i x - b_i) - \lambda & \text{, if } x < 0 \\ DNE & \text{, if } x = 0 \end{cases}$$

Then, we find minimizing argument x^* where,

$$x^* = \begin{cases} \frac{\vec{a}\cdot\vec{b}-\lambda}{\|\vec{a}\|_2^2} & \text{, if } x > 0 \\ \frac{\vec{a}\cdot\vec{b}+\lambda}{\|\vec{a}\|_2^2} & \text{, if } x < 0 \\ DNE & \text{, if } x = 0 \end{cases}$$

and in other words,

$$x^* \begin{cases} > 0 & \text{, if } \vec{a}\cdot\vec{b} > \lambda \\ < 0 & \text{, if } \vec{a}\cdot\vec{b} < -\lambda \end{cases}$$

Plotting where the solution exists let us see that the solution x^* is 0 when $\vec{a}\cdot\vec{b} \in (-\lambda, \lambda)$.

Chapter 24

Descent Methods

24.1 Coordinate Descent

Coordinate Descent is another descent method that comes in the form of some simple algorithm.
This algorithm requires a convex objective function, and the idea of coordinate descent is to minimize an objective function one coordinate at a time.

The procedure of this algorithm follows:

1. Initiate a guess
$$\vec{x}^{(0)} = \begin{bmatrix} x_1^{(0)} \\ \vdots \\ x_n^{(0)} \end{bmatrix}$$

2. The update rule:
$$x_i^{(k)} = \underset{x_i}{\text{argmin}} f(x_1^{(k)}, \ldots, x_i, x_{i+1}^{(k-1)}, \ldots, x_n^{(k-1)})$$
iterates for i from 1 to n.

   For example,
$$\begin{aligned} x_1^{(k)} &= \underset{x_1}{\text{argmin}} f(x_1, x_2^{(k-1)}, \ldots, x_n^{(k-1)}) \\ x_n^{(k)} &= \underset{x_n}{\text{argmin}} f(x_1^{(k)}, \ldots, x_{n-1}^{(k)}, x_n) \end{aligned}$$

This above algorithm holds for differentiable, convex scalar-valued functions.
Mind that we recognize the optimum argument $\vec{x}^*$ such that $f(\vec{x}^*)$ minimizes along all coordinate directions.
Therefore, where we find $\vec{x}^*$, for some objective function f, $\nabla f(\vec{x}^*) = \vec{0}$. As the algorithm updates along each coordinate axis, we gradually approach such point where the gradient becomes 0.

24.1.1 The Descent of LASSO

For some function that is formatted as:
$$f(\vec{x}) = g(\vec{x}) + \sum_{i=1}^{n} h_i(x_i)$$

where g, h_i are convex but h_i is not necessarily differentiable. In such case, we call h_i separables, and there is a proof that performing coordinate descent on f would still converge to optimum.

An example of some function along the above format would be Least Squares algorithm with LASSO Regularization applied onto it, where $h_i(x_i) = |x_i|$.
To minimize this function using coordinate descent, we may first attempt to compute the gradient of the objective function to obtain the derivative of it w.r.t. x_i:

$$\nabla\frac{1}{2}\|A\vec{x} - \vec{b}\|_2^2 = A^T(A\vec{x} - \vec{b})$$

and find that,

$$\frac{\partial}{\partial x_i}\frac{1}{2}\|A\vec{x} - \vec{b}\|_2^2 = \vec{A_i}^T(A\vec{x} - \vec{b})$$

where $\vec{A_i}$ denotes the i^{th} column of A.

Therefore, for the entire LASSO objective function f,

$$\frac{\partial f(\vec{x})}{\partial x_i} = \begin{cases} \vec{A_i}^T(A\vec{x} - \vec{b}) + \lambda & , x_i > 0 \\ \vec{A_i}^T(A\vec{x} - \vec{b}) - \lambda & , x_i < 0 \\ DNE & \text{, otherwise} \end{cases}$$

24.2 Newton's Method

Newton's method is an iterative descent method, such that for the formulation

$$\min_{\vec{v}} \frac{1}{2}\vec{v}^T H\vec{v} + \vec{c}^T\vec{v}$$

Then if H is positive definite and symmetric, such that $\vec{c} \in \mathcal{R}(H)$, we observe that $\vec{v}^* = H^{-1}\vec{c}$.
Newton's Method is surprisingly prevalent due to the capability of Taylor's Theorem to approximate any function as a quadratic polynomial.

Now, suppose we attempt to optimize some function f, where we realize via Taylor's Theorem that,

$$f(\vec{x} + \vec{v}) = f(\vec{x}) + \frac{\partial f}{\partial \vec{x}}\vec{v} + \frac{1}{2}\vec{v}^T H_f\vec{v}$$

to find the minimum along the approximation curve (defined by $\vec{v}$ in this Taylor context),

$$\underset{\vec{v}}{\text{argmin}} f(\vec{x} + \vec{v}) = -H_f(\vec{x})^{-1}\nabla f(\vec{x})$$

The update rule of Newton's Rule will then be used:

$$\vec{x}_{k+1} = \vec{x}_k + \vec{v} = \vec{x}_k - H_f(\vec{x})^{-1}\nabla f(\vec{x})$$

where, the learning rate is now decided by the Hessian rather than being a customizable hyperparameter. This fact makes Newton's Method a second-order method.

Along our discussion above, here are some conclusions on Newton's Method:

- If the Hessian of objective function is positive definite, the convergence of Newton's Method occurs much faster than the convergence of Gradient Descent.
- It is computationally expensive, because it has to computer Hessians and its inverses.
- Prone to accumulating perturbation errors.

24.2.1 Equality Constrained Quadratic Program

Provided some quadratic objective function,

$$\min_{A\vec{x}=\vec{b}} \frac{1}{2}\vec{x}^T H\vec{x} + \vec{c}^T\vec{x} + d$$

We may see that the Lagrangian of such objective function may be written as:

$$\mathcal{L}(\vec{x},\vec{\nu}) = \frac{1}{2}\vec{x}^T H\vec{x} + \vec{c}^T\vec{x} + d + \vec{\nu}^T(A\vec{x} - \vec{b})$$

where we may find the KKT conditions:

$$\begin{cases} A\vec{x}^* = \vec{b} \\ H\vec{x}^* + \vec{c} + A^T\vec{\nu}^* = 0 \end{cases}$$

Let us summarize the above KKT condition in matrix notation:

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \vec{x}^* \\ \vec{\nu}^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

Then, by casting the above constraints to a Newton's Method, we can enable Newton's Method to solve equality constrained quadratic programs.

Chapter 25

Optimization Applications I: LQR (Linear-Quadratic Regulator) Control

25.1 Introduction to Model-Based Control

Control problems have been approached on some model-based method; for example, we mathematically model the relationship of states and controls as:

$$\vec{x}_{t+1} = A\vec{x}_t + B\vec{u}_t$$

where $\vec{x}_t$ represents the state of a system at timestamp t.

We would like to assess the cost of an LQR to be:

$$\sum_{t=0}^{N-1} \frac{1}{2}(\vec{x}_t^T Q\vec{x}_t + \vec{u}_t^T R\vec{u}_t) + \frac{1}{2}\vec{x}_N^T Q_F \vec{x}_N$$

where Q, Q_F, R are symmetric and positive-semidefinite. Here, the term $\vec{x}_t^T Q\vec{x}_t$ accounts for the penalty of a current state, and $\vec{u}_t^T R\vec{u}_t$ provides a cost on the control terms u.
We perform a minimization on this control, where we also pose the following constraints to phrase the minimization problem as:

$$\min_{\substack{\vec{u}_t \\ \forall t=0,\ldots,N-1, \vec{x}_{t+1}=A\vec{x}_t+B\vec{u}_t \\ \vec{x}_0=\vec{x}_{\text{init}}}} \sum_{t=0}^{N-1} \frac{1}{2}(\vec{x}_t^T Q\vec{x}_t + \vec{u}_t^T R\vec{u}_t) + \frac{1}{2}\vec{x}_N^T Q_F \vec{x}_N$$

For one of the methods, we may employ dynamic programming with back propagation of information. This is among one of the standard methods to resolve LQR Control problem.
On a simialr logic, as each control and state at a timestamp is dependent upon some term of a timestamp before it, we can formulate large optimization problem as we write every expression in terms of $\vec{x}_0$ and $\vec{u}$s. However, this is computationally expensive. We must find a more elegant method: **A Duality-Based, Iterative Trick**.

25.2 LQR Control Via Duality and Certificate

Suppose that we rewrite the Lagrangian of our above objectiev function as:

$$\mathcal{L}(\vec{x}, \vec{u}, \vec{\nu}) = \sum_{t=0}^{N-1} \frac{1}{2}(\vec{x}_t^T Q\vec{x}_t + \vec{u}_t^T R\vec{u}_t) + \frac{1}{2}\vec{x}_N^T Q_F \vec{x}_N + \vec{\nu}_0^T(\vec{x}_{\text{init}} - \vec{x}_0) + \sum_{t=0}^{N-1} \vec{\nu}_{t+1}^T(A\vec{x}_t + B\vec{u}_t - \vec{x}_{t+1})$$

The associated KKT Conditions for such problem can be organized as follows.
First of all, let us derive the gradient of our Lagrangian w.r.t. its inputs at some specific timestamp t:

$$\begin{aligned}\nabla_{\vec{u}_t}\mathcal{L} &= \begin{cases} Q\vec{x}_t + A^T\vec{\nu}_{t+1} - \vec{\nu}_t & ,0 \leq t < N \\ Q_F\vec{x}_N - \vec{\nu}_N & ,t = N \end{cases} \\ \nabla_{\vec{x}_t}\mathcal{L} &= R\vec{u}_t + B^T\vec{\nu}_{t+1}\end{aligned}$$

Consequently, we acquire that,

$$\begin{cases} \vec{u}_t & = -R^{-1}B^T\vec{\nu}_{t+1} \\ \text{Adjoint System:} & \begin{cases} \vec{\nu}_t = A^T\vec{\nu}_{t+1} + Q\vec{x}_t \\ \vec{\nu}_N = Q_F\vec{x}_N \end{cases} \end{cases}$$

We now make a reasonable guess that there exists some linear relationship between $\vec{x}$ and $\vec{\nu}$.

$$\vec{\nu}_t = P_t\vec{x}_t, P_N = Q_F$$

Then, using backwards induction, we would like to prove the guess:

Explain 25.2.1. Proof of the Ricatti Equation

Proof:
Base Case. We know that $\vec{\nu}_N = Q_F\vec{x}_N$.
Induction Hypothesis. Suppose that for some timestamp t, $\vec{\nu}_t = P_t\vec{x}_t$.
Inductioon Step. Then, let us substitute the equations we found above for $\vec{\nu}$ and $\vec{x}$ into the above expression:

$$\begin{aligned}\vec{\nu}_t &= P_t\vec{x}_t \\ &= P_t(A\vec{x}_{t-1} + B\vec{u}_{t-1}) \\ &= P_t(A\vec{x}_{t-1} - BR^{-1}B^T\vec{\nu}_t) \\ (I + P_tBR^{-1}B^T)\vec{\nu}_t &= P_tA\vec{x}_{t-1} \\ \vec{\nu}_t &= (I + P_tBR^{-1}B^T)^{-1}P_tA\vec{x}_{t-1} \\ A^T\vec{\nu}_t &= A^T(I + P_tBR^{-1}B^T)^{-1}P_tA\vec{x}_{t-1} \\ &= \vec{\nu}_{t-1} - Q\vec{x}_{t-1} \\ \vec{\nu}_{t-1} &= (A^T(I + P_tBR^{-1}B^T)^{-1}P_tA + Q)\vec{x}_{t-1}\end{aligned}$$

Consequently, we obtain the Ricatti Equation:

$$P_{t-1} = A^T(I + P_tBR^{-1}B^T)^{-1}P_tA + Q$$

Therefore, upon our prior work,

$$\begin{aligned}\vec{u}_{t-1} &= R^{-1}B\vec{\nu}_t \\ &= R^{-1}B(I + P_tBR^{-1}B^T)^{-1}P_tA\vec{x}_{t-1} \\ &= K_{t-1}\vec{x}_{t-1} \\ K_{t-1} &= R^{-1}B(I + P_tBR^{-1}B^T)^{-1}P_tA\end{aligned}$$

Chapter 26

Support Vector Machine

This note contains two lectures worth of content, and is put in one chapter for coherency.

26.1 Introduction to Classifiers

In the scheme of classification vectors, we attempt to classify some datapoint $(\vec{x}, y)$ among specific classes, where we attempt to predict the label of this datapoint, $y \in \{1, -1\}$, based on its feature vector, $\vec{x}$. Support Vector Machine is a classification algorithm. Unlike logistic regression, which offers the probability that a point belonging to a class, SVM is an algorithm that directly provides the prediction of a datapoint based on geometric properties.
This geometric property we attempt to find is "separating hyperplane", where we find a hyperplane that can separate each class of points on one side of the hyperplane.

Remember that this hyperplane, which becomes the "decision boundary" of our classifier (because it is the boundary upon which prediction is decided) can be written as:

$$f(\vec{x}) = \vec{w}^T \vec{x} + b = \vec{w}^T(\vec{x} + \vec{x}_0)$$

Therefore, our prediction rule becomes:

$$y_i = \begin{cases} 1 & , f(\vec{x}_i) > 0 \\ -1 & , f(\vec{x}_i) < 0 \end{cases}$$

Compactly rewriting the above expression, we find that we actually require

$$y_i f(\vec{x}_i) > 0$$

26.2 SVM as an Optimization Problem

If our dataset can be separated by a hyperplane in such manner, we call our data "linearly separable".
However, there can exist many hyperplanes to result in a linear separation of some dataset. So which hyperplane is better? Per convention of optimization problem, we can come up with a heuristic that rates how good each hyperplane works as a decision boundary.

26.2.1 Hard-Margin SVM

Conceptually, to permit for noise perturbations in datapoints, we would like our decision boundary to have as much distance from each point as possible. Here we define this concept as:

Definition 26.2.1. Margin

Margin is the distance between a hyperplane and the closest datapoint to it.

We would like our decision boundary to have the maximum margin. This result is called a **max margin separator**.

What is the distance from a point to a hyperplane?
Fortunately, we have done a similar analysis in the most recent CSM16A worksheet. Let me shamelessly quote a section of that (which is still written by myself):

Explain 26.2.1. Distance from Point to Hyperplane

Problem Statement. Provided a point X and a hyperplane $\mathcal{H} = \{\vec{x} | \vec{w}^T\vec{x} + b = 0\}$, what is the distance between X and $\mathcal{H}$?

Solution. Let $\vec{x}$ represent the point X, and $\vec{y}$ be $proj_{\mathcal{H}}(\vec{x})$.
Then, such difference should also be a multiple of $\vec{w}$ as well, which is orthogonal to $\mathcal{H}$, such that:

$$(\vec{x} - \vec{y}) = \alpha\vec{w}$$

Therefore, the arithmetic follows:

$$\begin{aligned}
(\vec{x} - \vec{y}) &= \alpha\vec{w} \\
(\vec{x} - \vec{y}) \cdot \vec{w} &= \alpha\vec{w} \cdot \vec{w} \\
\vec{y} \in \mathcal{H} &\implies \vec{y} \cdot \vec{w} + b = 0 \\
\vec{x} \cdot \vec{w} + b &= \alpha\vec{w} \cdot \vec{w} \\
\alpha &= \frac{\vec{w}^T\vec{x} + b}{\|\vec{w}\|_2^2}
\end{aligned}$$

Then, the shortest distance between X and $\mathcal{H}$ may be expressed as:

$$\|\vec{x} - \vec{y}\|_2 = \alpha\|\vec{w}\|_2 = \frac{\vec{w}^T\vec{x} + b}{\|\vec{w}\|_2}$$

The optimization problem for finding the maximal margin separator would thus be:

$$\max_{\substack{\vec{w},b,m \\ \forall i, \frac{\vec{w}^T\vec{x}_i+b}{\|\vec{w}\|_2} \geq m \\ \forall i, y_i(\vec{w}^T\vec{x}_i+b)>0}} \quad m$$

This problem is known as the "Hard-Margin SVM Problem", more popularly formulated as:

$$\max_{\substack{\vec{w},b,m \\ \forall i, \frac{y_i(\vec{w}^T\vec{x}_i+b)}{\|\vec{w}\|_2} \geq m \\ m\geq 0}} \quad m$$

To let the problem have a unique solution, we may normalize $\vec{w}$ by choosing $\|\vec{w}\|_2 = m$.
This causes our problem to be further reduced as we eliminate m and change our maximization problem into an equivalent minimization problem:

$$\min_{\substack{\vec{w},b \\ \forall i, y_i(\vec{w}^T\vec{x}_i+b)\geq 1}} \quad \frac{1}{2}\|\vec{w}\|_2^2$$

This is a Quadratic Program.

26.2.2 Soft-Margin SVM

In soft-margin SVM, we allow for margin violation, such that the problem is rephrased as:

$$\min_{\substack{\vec{w},b \\ \forall i, y_i(\vec{w}^T\vec{x}_i+b)\geq 1-\xi_i \\ \forall i, \xi_i \geq 0}} \quad \frac{1}{2}\|\vec{w}\|_2^2$$

with slack variables ξ_i.
However, notice that we would want ξ_i to be small as well.
Therefore, we should regularize our obejctive function with some norm of $\vec{\xi}$:

$$\min_{\substack{\vec{w},b,\vec{\xi} \\ \forall i, y_i(\vec{w}^T\vec{x}_i+b)\geq 1-\xi_i \\ \forall i, \xi_i \geq 0}} \quad \frac{1}{2}\|\vec{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i$$

where $C > 0$ and controls the strength of regularization just as λ does in LASSO Regression:

> - Small C: Less Sensitive to Margin Violation
> - Large C: More Sensitive to Margin Violation

Now we may also consider the loss function of SVM.
Suppose we illustrate our loss function to be a $0 - 1$ loss based on mispredictions, then we would be facing a non-convex loss function. We should then attempt on the convexization of such function, resulting in a function that we call **hinge loss**:

$$L_{\text{hinge}} = \max(1 - y_i(\vec{w}^T\vec{x}_i + b), 0)$$

Furthermore, combining the two optimization problem constraints above does grant us that:

$$\forall i, \xi_i \geq \max(1 - y_i(\vec{w}^T\vec{x}_i + b), 0)$$

Therefore, our formulation for soft-margin SVM can be further paraphrased as its **Hinge-Loss Formulation**:

$$\min_{\substack{\vec{w},b \\ \forall i, y_i(\vec{w}^T\vec{x}_i+b)\geq 1-\xi_i \\ \forall i, \xi_i \geq 0}} \quad \frac{1}{2}\|\vec{w}\|_2^2 + C\sum_{i=1}^{n} L_{\text{hinge}}(y_i, \vec{w}^T\vec{x}_i + b)$$

26.2.3 Support Vector Identification via Duality

Suppose that we let the dual variables α_i, β_i correspond to the soft margin SVM formulation's first and second constraint.
Then, we obtain a Lagrangian:

$$\begin{aligned}
&\mathcal{L}(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \beta) \\
&\quad = \frac{1}{2}\|\vec{w}\|_2^2 + c\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\big((1-\xi_i) - y_i(\vec{w}^T\vec{x}_i + b)\big) \\
&\quad = \frac{1}{2}\|\vec{w}\|_2^2 - \sum_{i=1}^{n}\alpha_i y_i(\vec{w}^T\vec{x}_i + b) + \sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}(C - \alpha_i - \beta_i)\xi_i
\end{aligned}$$

The KKT Condition becomes necessary and sufficient for strong duality to hold, as we have convex, aggine constraint with a feasible set of possible solutions (a more precise version of Slater's Condition will not be readdressed here).

Then, the KKT Conditions should be as quoted in the following:

First-Order Conditions:

$$\begin{cases} \nabla_{\vec{w}}\mathcal{L} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0 \\ \frac{\partial \mathcal{L}}{\partial b} = -\sum_i \alpha_i y_i = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \end{cases}$$

From which we can already extract insights on:

> - The expression of $\vec{w}$ and C
> - Some α_i becoming non-zero shows its respective optimization constraint to be active.

Complementary Slackness:

$$\begin{cases} \alpha_i\big((1-\xi_i) - y_i(\vec{w}^T\vec{x}_i + b)\big) = 0 \\ \beta_i \xi_i = 0 \end{cases}$$

Here, if $0 < \alpha_i < C$ (such that it is nonzero), then $\big((1-\xi_i) - y_i(\vec{w}^T\vec{x}_i + b)\big) = 0$. Therefore, α_i desides which datapoints are the support vectors of a dataset, in that any non-zero α_i indicates its corresponding point to have an active constraint.

Meanwhile, β_i decides the non-zero-ness of slack variables, such that upon the first-order KKT condition, β_i not being 0 informs us that ψ_i is 0. Hence:

$$(0 < \alpha_i < C \land \beta_i \neq 0) \implies y_i(\vec{w}^T\vec{x}_i + b) = 1$$

such vector that lies on the margin is known as a support vector.

On the other hand, having a β_i provides that such vector is a support vector, but unknown whether on the margin or not. In that regard, any vector that forms an active constraint is a support vector.

A vector with $\alpha_i = 0$ is not a support vector then; we may observe this by the KKT condition, and see that any point with $\alpha_i = 0$ does not contribute to the formulation of $\vec{w}$. While they can be on the margin, they are “coincidentally” on the margin and do not themselves decide the margin.

26.3 Computing the Dual of SVM

Because Slater’s Condition holds for the dual problem of SVM, strong duality holds.

Let us then consider that our solution obeys the KKT Conditions listed in the previous section, such that

$$\begin{aligned} \mathcal{L}_{\text{KKT}}(\vec{w}, \vec{b}, \vec{\xi}, \vec{\alpha}, \vec{\beta}) &= \frac{1}{2}\|\vec{w}\|_2^2 - \sum_i \alpha_i y_i \vec{w}^T\vec{x}_i + \sum_i \alpha_i \\ &= \frac{1}{2}\|\vec{w}\|_2^2 - \vec{w}^T \sum_i \alpha_i y_i \vec{x}_i + \sum_i \alpha_i \\ &= \frac{1}{2}\|\vec{w}\|_2^2 - \|\vec{w}\|_2^2 + \sum_i \alpha_i \\ &= -\frac{1}{2}\|\vec{w}\|_2^2 + \sum_i \alpha_i \\ &= -\frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x}_i\right)^T \left(\sum_i \alpha_i y_i \vec{x}_i\right) + \sum_i \alpha_i \\ &= -\frac{1}{2}\vec{\alpha}^T \text{diag}(\vec{y}) X X^T \text{diag}(\vec{y}) \vec{\alpha} + \sum_i \alpha_i \\ &= -\frac{1}{2}\vec{\alpha}^T Q \vec{\alpha} + \sum_i \alpha_i \\ Q &= \text{diag}(\vec{y}) X X^T \text{diag}(\vec{y}) \end{aligned}$$

where

$$X = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_n^T \end{bmatrix}$$

is out design matrix.
Then, to solve for the max-min of this KKT-substituted Lagrangian, we may perform the following procedure:

$$d^* = \max_{\substack{\vec{\alpha},\vec{\beta} \\ \sum \alpha_i y_i = 0 \\ \forall i, C - \alpha_i - \beta_i = 0 \\ \forall i, \alpha_i \geq 0 \\ \forall i, \beta_i \geq 0}} -\frac{1}{2}\vec{\alpha}^T Q \vec{\alpha} + \sum_i \alpha_i$$

26.4 Kernel Trick

Suppose that our dataset is "linearly separable" by another coordinate system (for example, a circle in polar coordinates).
Therefore, regarding the question of "is a dataset linearly separable", we ask more on "is there a coordinate system by which a dataset is linearly separable". In doing so, we are effectively lifting features. In the case of circles, we are lifting our feature via a feature function of second-order terms:

$$\vec{x} = \Phi(\vec{x})$$

We would thus change the constraint on hyperplane separability of planes in soft-margin SVM into:

$$\forall i, y_i(\vec{w}^T \Phi(\vec{x}_i) + b) \geq 1 - \xi_i$$

This brings us to the question: suppose we lift features to achieve better performance in SVM, how does the computational cost fare, and how do we reduce such cost?

Let us propose the idea of a kernel function:

$$\{\Phi(\vec{x}_i)\}^T \Phi(\vec{x}_j) = \mathcal{K}(\vec{x}_i, \vec{x}_j)$$

The kernel function's purpose is to reduce the computational cost of inner products of lifted features, such that we may compute the matrix Q relatively efficiently.
There are many choices of the kernel function $\mathcal{K}$, which we introduce in COMPSCI 189.

For those who are interested, I introduce a description of Kernel Trick from COMPSCI 189 below.

26.5 Kernel Trick (from COMPSCI 189)

26.5.1 Motivation

When we have d input features, degree-p polynomial regression presents $\mathcal{O}(d^p)$ features. This is a lot. However, there is a magic that allows us to use these features without computing all of them.

Such magical trick is based on two observations that, in every learning algorithm:

1. The weights can be written as a linear combination of the sample points:

$$w = X^T a = \sum_{i=1}^{n} a_i X_i, a \in \mathbb{R}^n$$

2. We can use inner products of $\Phi(x)$ (lifted features) only, and not have to compute the lifted features themselves. These inner products can be computed rather efficiently.

For observation (1), we would substitute the above expression of w into an objective function, and now optimize within the algorithm with respect to a (instead of to w).

26.5.2 Kernel Ridge Regression

We may first center X and y (without interfering with the fictitious dimension) such that they are zero-meaned.
This lets us replace I' (identity matrix that doesn't punish fictitious dimension) with I in normal equations:

$$(X^T X + \lambda I)w = X^T y$$

Then, suppose a of the kernel trick is a solution to:

$$(X^T X + \lambda I)a = y$$

we obtain:

$$X^T y = (X^T X + \lambda I)X^T a$$

such that $w = X^T a$ is a solution to the normal equations, and w is a linear combo of sample points.
Here, then a is a dual solution that solves the dual form of ridge regression:

$$\min_w \|XX^T a - y\|^2 + \lambda \|X^T w\|^2$$

When we train, we solve for the a that suits the training set. Then, when we test, we compute the regression function as:

$$h(z) = w^T z = a^T X z = \sum_{i=1}^{n} a_i (X_i^T z)$$

Here, we define a kernel function to speed up the computation.
Let $k(x, z) = x^T z$ be a kernel function, then let the kernel matrix be defined as $K = XX^T$.
In other words, $K_{i,j} = k(X_i, X_j)$.

The dual ridge regression algorithm thus involves the following steps:

> - $\forall i, j, K_{i,j} \leftarrow k(X_i, X_j)$ (This is $\mathcal{O}(n^2 d)$ times)
> - Solve $(K + \lambda I)a = y$ for a (This is $\mathcal{O}(n^3)$ times)
> - For each test point z, $h(z) \leftarrow \sum_i a_i k(X_i, z)$ (This is $\mathcal{O}(nd)$ times)

Here, we see that the dual algorithm solves an $n \times n$ lnear system with $\mathcal{O}(n^3 + n^2 d)$ time. The primal algorithm, on the other hand, sovles a $d \times d$ linear system with $\mathcal{O}(d^3 + d^2 n)$ time. We prefer dual when $d > n$.

26.5.3 The Kernel Trick (Kernelization), Polynomial

The polynomial kernel function of degree-p polynomial is:

$$k(x, z) = (x^T z + 1)^p$$

Essentially,

$$(x^T z + 1)^p = \Phi(x)^T \Phi(z)$$

where, $\Phi(x)$ contains every monomial in X of degree $o, \ldots, p$.

Example. For $d = 2$, $p = 2$,

$$\begin{aligned}
\left(x^T z + 1\right)^2 &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 z_1 x_2 z_2 + 2x_1 z_1 + 2x_2 z_2 + 1 \\
&= \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1 x_2 & \sqrt{2}x_1 & \sqrt{2}x_2 & 1 \end{bmatrix} \begin{bmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1 z_2 \\ \sqrt{2}z_1 \\ \sqrt{2}z_2 \\ 1 \end{bmatrix} \\
&= \Phi(x)^T \Phi(z)
\end{aligned}$$

Here, we have computed $\Phi(x)^T \Phi(z)$ in $\mathcal{O}(d)$ time instead of $\mathcal{O}(d^p)$. Therefore, kerneled ridge regression undderlyingly works with lifted feature vectors instead of feature vectors per se, buy we use the polynomial kernel function such that we do not need to compute the feature vectors themselves.

Chapter 27

Interior Point Methods

27.1 Introduction to Interior Point Method

When working with unconstrained QPs and linear equality constrained QPs, we have generally convenient formulas to solve such problems. For any program that is not a QP but can be phrased as one, we may approximate it in a second-order Taylor polynomial under unconstrained settings. Meanwhile, equality constrained Newton's Method was developed in prior lectures via the help of KKT Conditions and matrix inversion.
Well, what about solving other constrained programs?

Interior Point Method is a powerful tool that we can use to solve constrained optimization problems, specifically on programs with linear equality and linear inequality constraints.

27.2 Barrier Function

Suppose that we face an optimization problem:

$$\min_{\substack{A\vec{x}=\vec{b} \\ \forall i, f_i(\vec{x})\leq 0}} \quad f_o(\vec{x})$$

We may add an indicator function to our optimization problem, where the indicator function outputs infinity for any point that does not satisfy the inequality constraint, and 0 for any point that satisfies the inequality constraint.
Then, we may rewrite the problem as:

$$\min_{A\vec{x}=\vec{b}} f_o(\vec{x}) + \sum_i \mathbb{1}_{f_i(\vec{x})\leq 0}$$

Such indicator function is known as the barrier function.

Definition 27.2.1. Barrier Function

In constrained optimization, a field of mathematics, a barrier function is a continuous function whose value on a point increases to infinity as the point approaches the boundary of the feasible region of an optimization problem (Wikipedia).

Suppose rather than using a piecewise indicator function, we work with an approximated one, such that:

$$I_{\text{approx}}(u) = -\frac{1}{t}\log(-u)$$

is the logarithmic barrier function.
Then, our new optimization problem becomes

$$\begin{aligned} p^* &= \min_{A\vec{x}=\vec{b}} f_o(\vec{x}) + \frac{1}{t}\Phi(\vec{x}) \\ \Phi(\vec{x}) &= \sum_{i=1}^{n} -\log\big(-f_i(\vec{x})\big) \end{aligned}$$

Here, the selection of t depends on the observed primal-dual gap. If such gap is large, we increase t (usually doubling). This allows us to warm-start Newton's Method computations as well.

Chapter 28

Mixed: Integer Programming

This lecture is not in scope.
In this chapter, we will be discussing non-linear, integer programming, such that we may step into the non-linear regime of optimization via techniques that we have built up in this class so far.
As a personal note, in the adversarial machine learning project of EECS 127, we have seen problems (such as finding images with discrete values for pixels) that require integer outputs.

28.1 Integer Programming

In integer programming, our objective need have integer constraints: the solution must be an integer. These problems are generally phrased as:

$$p^* = \min_{\substack{A\vec{x}=\vec{b} \\ \vec{l}\leq\vec{x}\leq\vec{u} \\ x_j\in\mathbb{Z}}} \vec{c}^T\vec{x}$$

An integer decision variable comes in many contexts.
For example, an indicator variable itself is an integer decision variable (where 1 expresses "True", 0 expresses "False").
A mixed integer program, meanwhile, is an integer program that also involves floating number decision variables.

Let us explore integer programming with an example problem below.

28.1.1 Exploration by Example

Suppose UC Berkeley wants to build new buildings. It can build new buildings, or new dorms.
Both buildings and dorms can be built on either southside or northside, but we want to build at most one dorm, and if we build a dorm it should be close to buildings. We observe constraints, but they are more qualitative than quantitative, but as in any optimization value, we will handle this optimization problem by expressing the value and cost of each investment:

Building Type	Location	Value	Cost	Representative Variable
Building	North	9M	6M	$x_1 \in \{0, 1\}$
	South	5M	3M	$x_2 \in \{0, 1\}$
Dorm	North	6M	5M	$x_3 \in \{0, 1\}$
	South	4M	2M	$x_4 \in \{0, 1\}$

Suppose our budget is 10M, then we may mathematically phrase this optimization as follows:

$$\max_{\substack{\forall i, 0 \le x_i \le 1 \land x_i \in \mathbb{Z} \\ 6x_1+3x_2+5x_3+2x_4 \le 10 \\ x_3+x_4 \le 1 \\ x_3=1 \implies x_1=1 \\ x_4=1 \implies x_2=1}} \quad 9x_1 + 5x_2 + 6x_3 + 4x_4$$

As a side note, $x_3 = 1 \implies x_1 = 1$ states one of the constraints above, that "if we build a dorm it should be close to buildings". Now, let us further simplify such constraint into:

$$x_3 - x_1 \le 0$$

Because:

- In the case that $x_1 = 1$, our expression $x_3 - x_1$ is 0.
- In the case that $x_3 = 0$, our expression $x_3 - x_1$ must be negative.
- The only case at which this value becomes positive is that $x_3 = 1, x_1 = 0$, but that is against the implication.

Along such logic, we may then rephrase our problem as:

$$\max_{\substack{\forall i, x_i \in \{0,1\} \\ 6x_1+3x_2+5x_3+2x_4 \le 10 \\ x_3+x_4 \le 1 \\ x_3-x_1 \le 0 \\ x_4-x_2 \le 0}} \quad 9x_1 + 5x_2 + 6x_3 + 4x_4$$

28.2 Disjunction of Constraints

The application of integer programming is wide. For example, as indirectly demonstrated above, enables us to write "either/or" expressions within the constraint.

Suppose we attempt to express the following constraint in an optimization problem:

$$3x_1 + 2x_2 \le 18 \oplus x_1 + 4x_2 \le 16$$

Note, $\oplus$ stands for either-or, and is mathematically called "xor".

We can rephrase the above constraint by introducing a large constant M.
Now, for a binary variable y, I reexpress my constraint as follows:

$$\begin{cases} 3x_1 + 2x_2 \le 18 + yM \\ x_1 + 4x_2 \le 16 + (1-y)M \end{cases}$$

that way, introducing the binary variable y lets us toggle between different constraints on an either-or fashion.

When we have more constraints within the xor sequence, we will make a decision tree of constraints via multiplying more binary variables.

28.3 Algorithmic Works

Integer Programs are useful not just for the above capabilities. It also has the following properties:

> - There are few feasible solutions. However, this number can explode very quickly, essentially exponentially in number of decision variables.
> - However, with more constraints, integer programs can lead to faster solutions.

Integer programs have many solving softwares, such as Cplex and Gurobi. They use an algorithm known as "Branch and Bound" (Land and Dong, 1960s), which is still the state-of-the-art strategy for solving these programs.

28.3.1 Demonstration of BB Algorithm

Let us attempt solving the aforementioned building problem with Branch and Bound Algorithm.

We would begin by solving the optimization problem without the integer constraint on decision variables. In such effort, we acquire that:

$$\begin{cases} p^* = 16.5 \\ \vec{x}^T = \begin{bmatrix} 0.833 & 1 & 0 & 1 \end{bmatrix} \end{cases}$$

We see that such solution is infeasible. One simple solution to bring back the integer constraint is to round a decision variable; however, in doing so, we may violate other constraints. What we perform instead, then, is to build a branch on the non-integral decision variable: x_1.
Consider a branch of the problem based on the value of x_1, such that for branches b_0, b_1:

$$p^*_{b_i} = \max_{\substack{x_1=0 \\ other constraints as is}} f_0(x)$$

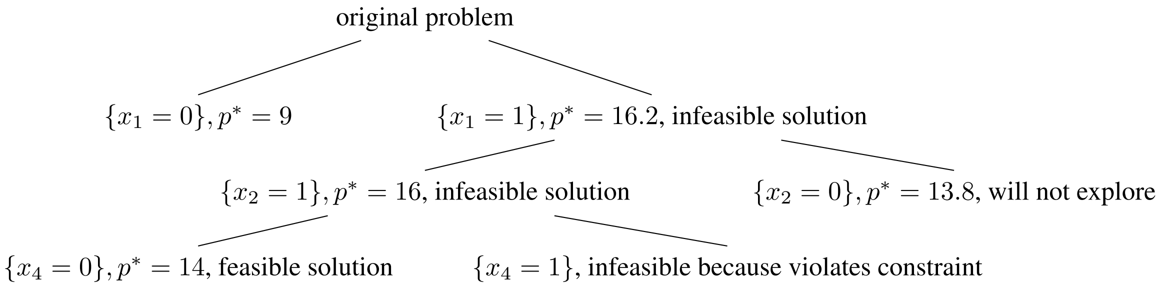
we solve such linear program without integer constraints again. In such case, we find our incumbent solution as follows:

$$\begin{cases} p^*_{b_0} = 9 \\ \vec{x}^T = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \end{cases}$$

Meanwhile, on the other branch we would find:

$$\begin{cases} p^*_{b_1} = 16.2 \\ \vec{x}^T = \begin{bmatrix} 1 & 0.8 & 0 & 0.8 \end{bmatrix} \end{cases}$$

As we encounter non-integral solutions, we continue to traverse along such direction (because there's no branches that is supposed to occur under integral solutions). The overall operation of our BB efforts is as traced follows:


original problem
$\{x_1 = 0\}, p^* = 9$
$\{x_1 = 1\}, p^* = 16.2$, infeasible solution
$\{x_2 = 1\}, p^* = 16$, infeasible solution
$\{x_2 = 0\}, p^* = 13.8$, will not explore
$\{x_4 = 0\}, p^* = 14$, feasible solution
$\{x_4 = 1\}$, infeasible because violates constraint

We make the conclusion that the incumbent solution is determined at the branch of $x_4 = 0$.

We did not explore the branch of $x_2 = 0$ because as we traverse the tree, the value of p^* only becomes smaller (since we are exploring sub-optimal values that are less than their non-integer optimization results). If the branches only yield smaller variables, we don't explore it. More specifically, we should recognize that the primitive version of BB is a depth-first-traversal algorithm, and we ended up finding $p^* = 14$ before encountering the $x_2 = 0$ branch that has a smaller argument, therefore pruned that branch.
That means, if we have to explore the $x_2 = 0$ branch first, we would actually explore that branch because we didn't find a better maximum than 9, the first incumbent solution, yet.

Chapter 29

Revision Log

> - A week before first midterm: Revised all notes in scope of first midterm.
> - April 27th, 2023: All first drafts of in-scope contents are released on GitHub.