EECS 127: Optimization Models in Engineering

Tyler Zhu

March 10, 2021

"A good stock of examples, as large as possible, is indispensable for a thorough understanding of any concept, and when I want to learn something new, I make it my first job to build one."

– Paul Halmos.

These are course notes for the Spring 2021 rendition of EECS 127, Optimization Models in Engineering, taught by Professor Laurent El Ghaoui.

Contents

1 Tuesday, January 19th: Introduction

1.1 Introduction

In this course, we'll be talking primarily about *optimization*. A standard form of optimization is the following:

$$p^* = \min_{\boldsymbol{x}} f_0(\boldsymbol{x}) \quad \text{subject to: } f_i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, m,$$

where

- vector $\boldsymbol{x} \in \mathbb{R}^n$ is the *decision variable*;
- $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the *objective* function, or *cost*;
- $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, represent the *constraints*;
- p^* is the *optimal value*.

Realistically, $\boldsymbol{x} = [x_1 \ \ldots \ x_n]$ represents different decisions, i.e. x_2 would be our decision at time $t = 2$. Also note that we can easily solve instead to maximize some $r(x)$ by setting $f_0(x) = -r(x)$. This above setup is known as the *standard form*.

Oftentimes we will have multiple optimal solutions to the constraint, in which case any $x^* \in \arg\min f_0(x)$ for which $f_i(x^*) \leq 0, i = 1, \ldots, m$ is satisfied acts as an optimizer.

In this class, we're not as concerned with algorithms for optimization, but more so translating problems from the real world into this language.

Example 1.1 (Least-squares regression)**.** A classic example in machine learning is when we have a given vector y and we're trying to express it as a linear function of an input vector z, i.e. data points. The goal is the solve the objective

$$\min_x \sum_{i=1}^m (y_i - x^\top z^{(i)})^2$$

where i think it's a racha's angle for me

- $z^{(i)} \in \mathbb{R}^n$, $i = 1, \ldots, n$ are data points;
- $y \in \mathbb{R}^m$ is a "response" vector;
- $x^\top z$ is the scalar product $z_1 x_1 + \cdots + z_n x_n$ b/w the two vectors $x, z \in \mathbb{R}^n$.

One example of constraints we could be working with are $x \geq 0$ and $x^\top \mathbb{1} = 1$, which corresponds to modeling a discrete distribution.

Example 1.2 (Support Vector Machines (SVMs))**.** In SVMs, we instead are trying to optimize a "hinge" loss, i.e.

$$\min_{x,b} \sum_{i=1}^m \max(0, 1 - y_i(x^\top z^{(i)} + b))$$

where

- $z^{(i)} \in \mathbb{R}^n$, $i = 1, \ldots, n$ are data points;
- $y \in \{-1, 1\}^m$ is a *binary* response vector;
- $x^\top z + b = 0$ defines a "separating hyperplane" in data space.

We could imagine that our points are colored green and red. Then at a conceptual level, we're trying to create a hyperplane that separates our data points into two different classes as clearly as possible. Once we find the best x, b, we can predict the binary output $\hat{y}$ corresponding to a new point's predicted class.

While we just gave a few machine learning examples which were problems without constraints, we can often use optimization to act on certain situations. One example is energy production. There's a **lot** of other ones.

1.2 Optimization Problems

There is more nomenclature we need to know:

- *Feasible set*, i.e. the set of possible values satisfying the constraints.
- *Unconstrained minimizer*:x_0, i.e. minimizing the cost function without constraints.
- *Optimal Point*: x^*.
- *Level sets* of objective functions, i.e. sets $\{x|R(x) = c\}$ for some c.
- *Sub-level sets*, i.e. sets $\{x|R(x) \leq c\}$ for some c.

Usually our optimal points will be some intersection with the smallest level sets and the feasible set.

Similar to neural networks, we can have an issue in optimization of local vs. global optimal points. A point z is *locally optimal* if there exists a value $R > 0$ such that z is optimal for problem

$$\min_x f_0(x) \text{ s.t. } f_i(x) \leq 0,\ i = 1, \dots, m \text{ and } |x_i - z_i| \leq R,\ i = 1, \dots, n.$$

A local minimizer x minimizes f_0, but only compared to nearby points on the feasible set. The value of the objective function at that point is *not* necessarily the (global) optimal value of the problem. Locally optimal points might be of no practical interest to the user. Visually, you could imagine that we could find ourselves in certain pits that locally seem like optima but globally are far from it.

Due to these problems (and others), often times even coming up with any minimizer can be difficult. However, there's a special class of problems called *convex problems* which have the nice property that *all* local optimums are also global optimums. Usually your objective function when plotted looks something like a bowl, i.e. there's a single point at the bottom which is the previously mentioned global optimum.

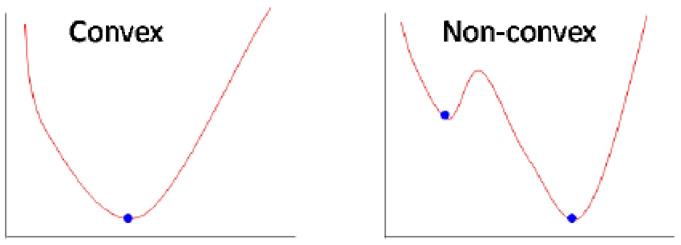


Figure 1: Convex and Non-convex functions

Funny enough, even though most problems are non-convex, we could find a convex function that lower bounds our objective function and hope its global optimum is a decent solution to our original objective. Pushing this further, if we could find the tightest convex function which is a lower bound (think convex hulls), then its optimal point *is* the same as our original's! It looks like we just turned a hard problem into a much easier one, but we unfortunately have no idea how to find this convex-hull. Back to square one.

1.3 Course Outline

In this course, we shall deal specifically with convex optimization problems with special structure, such as:

- Least-Squares (LS)
- Linear Programs (LP)
- Convex Quadratic Program (QP)
- Second-order cone programs (SOCP)

For such specific models, very efficient solution algorithms exist with high quality implementations/software (CVX, for example). Most of the problems we come across can be categorized into one of the above structures, as we'll come to see.

A large part of our time will be spent talking about affine subspaces, and normal vectors to hyperplanes to geometrically construct feasible sets.

We'll also discuss a few non-convex problems that come up often in the real world:

- *Boolean/integer optimization*: $x_i \in \{0,1\}^n$; some variables are constrained to be Boolean or integers. Convex optimization can be used for getting (sometimes) good approximations.
- *Cardinality-constrained problems*: we seek to bound the number of non-zero elements in a vector variable. Convex optimization can be used for getting good approximations.
- *Non-linear programming*: usually non-convex problems with differentiable objective and functions. Algorithms provide only local minima.

It goes without saying that most (but not all) non-convex problems are *hard*!

For example, let's look at boolean optimization. We would like to solve $\min_{\boldsymbol{x}} c^\top \boldsymbol{x}$ subject to $A\boldsymbol{x} \leq b$ and $\boldsymbol{x} \in \{0,1\}^n$. One way of relaxing this into a problem that's easier to solve is to consider instead $\boldsymbol{x} \in [0,1]^n$, i.e. going from discrete to continuous feasible set. This is an important question because this allows us to do sparsity and feature-selection.

What this course is for:

- Learning to model and efficiently solve problems arising in Engineering, Management, Control, Finance, ML, etc.
- Learning to prototype small- to medium- sized problems on numerical computing platforms.
- Learning basics of applied linear algebra, convex optimization.

What this course is NOT:

- A course on mathematical convex analysis.
- A course on details of optimization algorithms.

Here's a high level overview of what we're talking about:

- Linear algebra models
  - Vectors, projection theorem, matrices, symmetric matrices.
  - Linear equation, least-squares and minimum-norm problems.
  - Singular value decomposition (SVD), PCA, and related optimization problems.
- Convex optimization models
  - Convex sets, convex functions, convex problems.
  - Optimality conditions, duality.
  - Special convex models: LP, QP, SOCP.
- Applications
  - Machine Learning.
  - Control.
  - Finance.
  - Engineering desisgn.

There will only be six homeworks in this course.

2 Thursday, January 21st: Vectors and Functions

Today's lecture went pretty fast and sped through sections, so likewise these notes are pretty rough. Also I don't have time to add pictures (unless I rip them off from somewhere), so I hope you have Beth Harmon level visualization powers.

2.1 Introduction

We usually write vectors in column format, i.e.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Element x_i is the ith component, and the number n of components is the *dimension* of x. Importantly, in math, we always 1-index, not 0-index.

We can use vectors for example in bag of words frequency matching.

Example 2.1 (Time series)**.** We can model a time series, i.e. the evolution of a physical or economical quantity. We can represent it like $x = [x(1)\ x(2)\ \ldots\ x(T)]^\top \in \mathbb{R}^T$ where each $x(k)$ is the value of the quantity at time k.

One example of a model for time series is an "auto-regressive" model, where we assume that the output depends linearly on certain previous terms and some stochasticity (i.e. error). For example, if we think it only depends on the previous two days, we could write that

$$x_t \approx \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \text{error}.$$

Vectors have a few special properties, in that the operations of sum, difference, and scalar multiplication all hold and are closed, i.e. that they return other vectors. These properties define a *vector space*. The simplest example is $\mathcal{X} = \mathbb{R}^n$.

From a vector space $\mathcal{X}$, a nonempty subset $\mathcal{V}$ of $\mathcal{X}$ is a *subspace* if, for any scalars α, β,

$$x, y \in \mathcal{V} \implies \alpha x + \beta y \in \mathcal{V}.$$

For example, the set of all possible linear combinations of vectors in $S = \{x^{(1)}, \ldots, x^{(m)}\}$ forms a subspace called the *span* of S, denoted as $\text{span}(S)$.

Finally, given two subspaces $\mathcal{X}, \mathcal{Y}$ in $\mathbb{R}^n$, the *direct sum* of $\mathcal{X}, \mathcal{Y}$, denoted by $\mathcal{X} \oplus \mathcal{Y}$, is the set of vectors of the form $x + y$ with $x \in \mathcal{X}, y \in \mathcal{Y}$. One can easily check that $\mathcal{X} \oplus \mathcal{Y}$ is a subspace.

We also have the familiar concepts of linear independence and hence the *basis* for a subspace $\mathcal{S}$, the smallest set of vectors which spans $\mathcal{S}$, whose cardinality defines the dimension of the subspace.

Definition 1 (Affine sets)**.** An affine set is a set of the form

$$\mathcal{A} = \{x \in \mathcal{X} \mid x = v + x^{(0)}, v \in \mathcal{V}\},$$

where $x^{(0)}$ is a given point and $\mathcal{V}$ is a given subspace of $\mathcal{X}$. Subspaces are affine spaces containing the origin (i.e. $x^{(0)}$ is the origin).

Naturally, a *line* is a one-dimensional affine set. We start with some point x_0 which hinges the line, and then some vector v which determines the direction (i.e. including all multiples of it).

Quick quiz: how many values do we need to uniquely determine lines in 2D? You might think it's 2, but we need to identify the vertical line as well, so it's 3. Think of $ax + by + c = 0$.

One question we might have is how do we measure the *size* of a vector; norms are the answer to this question. In short, a *norm* simply assigns real numbers to vectors in a consistent manner, i.e. it satisfies the following properties:

Definition 2 (Norm)**.** A function $\|\cdot\|$ from $\mathcal{X}$ to $\mathbb{R}$ is a *norm* if

1. $\|x\| \geq$ for all $x \in \mathcal{X}$, and $\|x\| = 0$ iff $x = 0$;
2. $\|x + y\| \leq \|x\| + \|y\|$, for any $x, y \in \mathcal{X}$ (triangle inequality);
3. $\|\alpha x\| = \alpha\|x\|$, for any scalar α and any $x \in \mathcal{X}$.

A classic example of a norm which we'll use often are the ℓ_p norms, defined by

$$\|x\|_p := \left(\sum_{k=1}^{n} |x_k|^p\right)^{1/p}, \quad 1 \leq p < \infty.$$

We will primarily use three norms:

- For $p = 2$, we obtain the standard Euclidean length
$$\|x\|_2 = \sqrt{\sum_{k=1}^{n} x_k^2},$$
- or $p = 1$ which gives the sum-of-absolute-values length (or Manhattan distance)
$$\|x\|_1 = \sum_{k=1}^{n} |x_k|.$$
- When $p = \infty$, it defines the ℓ_∞ norm (max absolute value norm)
$$\|x\|_\infty = \max_{k=1,\dots,n} |x_k|.$$
- The cardinality of a vector x is often called the ℓ_0 (pseudo-) norm, which

2.2 Inner product

Inner products are interesting since we can think of vectors now as functions acting on other vectors, i.e. like $\langle x, \cdot\rangle$.

Definition 3 (Inner product)**.** An *inner product* on a (real) vector space $\mathcal{X}$ is a real-valued function which maps pairs $x, y \in \mathcal{X}$ into a scalar denoted by $\langle x, y\rangle$. The inner product satisfies the following axioms (for $x, y, z \in \mathcal{X}$ and scalar α):

$$\begin{aligned}
\langle x, x\rangle &\geq 0;\\
\langle x, x\rangle &= 0 \text{ iff } x = 0;\\
\langle x + y, x\rangle &= \langle x, z\rangle + \langle y, z\rangle;\\
\langle \alpha x, y\rangle &= \alpha\langle x, y\rangle;\\
\langle x, y\rangle &= \langle y, x\rangle.
\end{aligned}$$

A vector space with an inner product is called an *inner product space*, and the standard inner product in $\mathbb{R}^n$ is the row-column product of two vectors,

$$\langle x, y\rangle = x^\top y = \sum_{k=1}^n x_k y_k.$$

The inner product induces a norm given by $\|x\| = \sqrt{\langle x, x\rangle}$ (which you may notice is identical to the ℓ_2-norm in $\mathbb{R}^n$).

With inner products, we can define the angle θ between two vectors x, y by

$$\cos\theta = \frac{x^\top y}{\|x\|_2\|y\|_2},$$

which results from doing some easy geometry. This lets us characterize when x, y are *orthogonal* (i.e. $\theta = \pm 90°$), and when x, y are *parallel* (i.e. $\theta = 0°, \pm 180°$).

Since $|\cos\theta| \le 1$, we easily get the following inequality.

Theorem 4 (Cauchy-Schwarz Inequality). For vectors $x, y \in \mathbb{R}^n$,

$$|x^\top y| \le \|x\|_2\|y\|_2$$

where equality holds when $x = \alpha y$ for scalar α (i.e. when $|\cos\theta| = 1$).

We also have the following generalization with ℓ_p norms.

Theorem 5 (Holder's Inequality). For any vectors $x, y \in \mathbb{R}^n$ and for any $p, q \ge 1$ such that $1/p + 1/q = 1$, it holds that

$$|x^\top y| \le \sum_{k=1}^n |x_k y_k| \le \|x\|_p\|y\|_q.$$

2.3 Maximization of inner product over norm balls

Let's solve our first optimization problem and discuss maximizing the inner product over norm balls. We're trying to solve the optimization problem

$$\max_{\|x\|_p\le 1} x^\top y.$$

For $p = 2$, we can use the Cauchy-Schwarz inequality to get $|x^\top y\| \le \|x\|_2\|y\|_2$, where equality holds when $x = \alpha y$. Since $\|x\|_2 \le 1$ and we want the largest positive value, we should take

$$x_2^* = \frac{y}{\|y\|_2},$$

so $\max_{\|x\|_2\le 1} x^\top y = \|y\|_2$.

For $p = \infty$, each entry of x should be ± 1 to satisfy the norm condition, which then maximizes the inner product by making everything positive, so

$$x_\infty^* = \operatorname{sgn}(y).$$

So $\max_{\|x\|_\infty\le 1} x^\top y = \|y\|_1$.

For $p = 1$, we simply set

$$[x_1^*] = \begin{cases} \text{sgn}(y_i) & \text{if } i = m \\ 0 & \text{o/w} \end{cases}, \quad i = 1, \dots, n,$$

where m is the index of an entry in y which has maximum absolute value. Then $\max_{\|x\|_1 \le 1} x^\top y = \|y\|_\infty$.

Notice that all of these results are in line with what we'd expect by applying Holder's inequality to each case.

2.4 Orthogonalization and Projections

In our favorite vector space $\mathbb{R}^2$, two vectors are perpendicular when they have a dot product 0. For generic inner product spaces, we say that two vectors x, y in an inner product space $\mathcal{X}$ are *orthogonal* if $\langle x, y\rangle = 0$, which we denote by $x \perp y$. Nonzero vectors are *mutually orthogonal* if they are pairwise orthogonal. It is a simple exercise to show that mutually orthogonal vectors are linearly independent.

Given a subset $\mathcal{S}$ of an inner product space $\mathcal{X}$, a vector $x \in \mathcal{X}$ is orthogonal to $\mathcal{S}$ if $x \perp s$ for all $s \in \mathcal{S}$. The set of vectors orthogonal to $\mathcal{S}$ is called the *orthogonal complement* of $\mathcal{S}$, and is denoted by $\mathcal{S}^\perp$.

Theorem 6 (Orthogonal Decomposition). If $\mathcal{S}$ is a subspace of an inner-product space $\mathcal{X}$, then any vector $x \in \mathcal{X}$ can be uniquely written as the sum of an element in S and an element in $S^\perp$, i.e.

$$\mathcal{X} = \mathcal{S} \oplus \mathcal{S}^\perp \quad \text{for any subspace } \mathcal{S} \subseteq \mathcal{X}.$$

Now we will talk about projections. The idea of projection is central in optimization, which is finding a point on a given set that is closest (in norm) to a given point.

Finish writing about projections

We have an important theorem concerning projections.

Theorem 7 (Projection Theorem). Let $\mathcal{X}$ be an inner product space, x be a given element in $\mathcal{X}$, and $\mathcal{S}$ a subspace of $\mathcal{X}$. Then, there exists a unique vector $x^* \in \mathcal{S}$ which is the solution to the problem

$$\min_{y \in \mathcal{S}} \|y - x\|.$$

Moreover, a necessary and sufficient condition for x^* being the optimal solution for this problem is that $x^* \in \mathcal{S}$, $(x - x^*) \perp \mathcal{S}$.

Let $p \in \mathbb{R}^n$ be a given point. We want to compute the Euclidean projection p^* of p onto a line $L = \{x_0 + \text{span}(u)\}$, $\|u\|_2 = 1$.

Now say we want to solve the harder problem of projecting onto a span of vectors, say $\mathcal{S} = \text{span}(x^{(1)}, \dots, x^{(d)}) \subseteq \mathcal{X}$. By the projection theorem, we can convert this into a system of equations and solve.

2.5 Functions and maps

In this class, we usually reserve the term *function* to denote $f : \mathbb{R}^n \to \mathbb{R}$, and use the term *map* when $f : \mathbb{R}^n \to \mathbb{R}^m$ and $m > 1$.

A *linear* function is simply a function that perserves scaling and additivity, i.e. that

$$f(\alpha x) = \alpha f(x) \quad \forall x \in \mathbb{R}^n, \alpha \in \mathbb{R}$$
$$f(x + y) = f(x) + f(y) \quad \forall x, y \in \mathbb{R}^n.$$

The gradient of a function can be intepreted in the context of level sets. Geometrically, the gradient of f at a point x_0 is a vector $\nabla f(x_0)$ perpendicular to the contour line of f at level $\alpha = f(x_0)$, pointing from x_0 outwards the α-sublevel set (i.e. points towards higher values of the function).

3 Tuesday, January 26th: Matrices and Linear Maps

3.1 Matrix Basics

In this class, we will use the convention that if A is a data matrix, then every column of A is a single data point. Also, note that $x^\top Ay = y^\top A^\top x$, since the inner product is symmetric and the result is a scalar. This will be useful later on when we discuss duality.

There's many ways we can think of a matrix-vector product. Let $A \in \mathbb{R}^{m,n}$ with columns $a_1, \ldots, a_n \in \mathbb{R}^m$, and $b \in \mathbb{R}^n$ a vector. Then we can define the product by

$$Ab = \sum_{k=1}^{n} a_k b_k,$$

which is essentially just a weighted sum of the columns of A with the elements of b as coefficients.

We can do the same definition for multiplication on the right by A. Let $\alpha_1, \ldots, \alpha_m$ be the rows of A. If instead we had some $c \in \mathbb{R}^m$, we can multiply on the left by it's transpose to get

$$c^\top A = \sum_{k=1}^{m} c_k \alpha_k^\top,$$

which says that this is a weighted sum of the rows α_k of A using the elements of c as coefficients.

From matrix-vector products, we can then define matrix products. We can view a matrix $A \in \mathbb{R}^{m,n}$ in two different ways; either as a collection of columns or rows.

$$A = \begin{bmatrix} a_1 & a_2 & \ldots & a_n \end{bmatrix}, \text{ or } A = \begin{bmatrix} \alpha_1^\top \\ \alpha_2^\top \\ \vdots \\ \alpha_m^\top \end{bmatrix},$$

where $a_1, \ldots, a_n \in \mathbb{R}^m$ denote the columns of A and $\alpha_1^\top, \ldots, \alpha_m^\top \in \mathbb{R}^n$ denote the rows of A.

We have three main interpretations of matrix-matrix products. One is obtained by letting A act as a linear map on the columns of B, where $B = \mathbb{R}^{n,p}$ and $B = [b_1 \ \ldots \ b_p]$. This gives

$$AB = A \begin{bmatrix} b_1 & \ldots & b_p \end{bmatrix} = \begin{bmatrix} Ab_1 & \ldots & Ab_p \end{bmatrix}.$$

Similarly, we can also view the product as the linear map B operating on the rows of A, which gives

$$AB = \begin{bmatrix} \alpha_1^\top \\ \vdots \\ \alpha_m^\top \end{bmatrix} B = \begin{bmatrix} \alpha_1^\top B \\ \vdots \\ \alpha_m^\top B \end{bmatrix}.$$

Finally, we can also write AB as a sum of *dyadic* matrices (i.e. rank one matrices of the form $uv^\top$). Letting $\beta_i^\top$ denote the rows of B, we can write

$$AB = \sum_{i=1}^{n} a_i \beta_i^\top.$$

3.2 Matrices as Linear Maps

Assume that A is an $m \times n$ matrix unless stated otherwise. Recall that we can interpret matrices as *linear maps*, i.e. maps where $f(ax + by) = af(x) + bf(y)$ for scalars a, b. Affine maps are simply linear functions plus a constant, i.e. $f(x) = Ax + b$.

Also recall that we have the familiar concepts of range, rank, and nullspace. Specifically, the set of linear combinations of the columns a_i's of a matrix A are of the form Ax, for $x \in \mathbb{R}^n$. We call this the *range* of A, and denote it as

$$\mathcal{R}(A) = \{Ax | x \in \mathbb{R}^n\}.$$

The dimension of $\mathcal{R}(A)$ (i.e. cardinality of a basis for A or the number of lin. ind. rows of A) is called the *rank* of A. We also have that $\operatorname{rank}(A) = \operatorname{rank}(A^\top)$.

On the other hand, the null space of A gives the vectors which are mapped to zero by A (i.e. the *kernel*) and is denoted by

$$\mathcal{N}(A) = \{x \in \mathbb{R}^n | Ax = 0\}.$$

The null space of a matrix gives you the "ambiguity" of the solutions to $Ax = 0$. For any two solutions $Ax = 0$ and $Ay = 0$, $A(x + y) = A(x - y) = 0$, so $x + y, x - y \in \mathcal{N}(A)$ for example.

Importantly, $\mathcal{R}(A^\top)$ and $\mathcal{N}(A)$ are mutually orthogonal subspaces, i.e. $\mathcal{N}(A) \perp \mathcal{R}(A^\top)$, meaning that every vector in one space is orthogonal to every vector in the other. This leads us to the following fundamental theorem.

Theorem 8 (Fundamental Theorem of Linear Algebra)**.** For any given matrix $A \in \mathbb{R}^{m,n}$, it holds that $\mathcal{N}(A) \perp \mathcal{R}(A^\top)$ and $\mathcal{R}(A) \perp \mathcal{N}(A^\top)$, hence

$$\mathcal{N}(A) \oplus \mathcal{R}(A^\top) = \mathbb{R}^n$$
$$\mathcal{R}(A) \oplus \mathcal{N}(A^\top) = \mathbb{R}^m.$$

Consequently, we can decompose any vector $x \in \mathbb{R}^n$ into a vector in the range of $A^\top$ and another in the nullspace of A:

$$x = A^\top \xi + z, \quad z \in \mathcal{N}(A).$$

Similarly, we can decompose any vector $w \in \mathbb{R}^m$ into a vector in the range of A and another in the nullspace of $A^\top$:

$$w = A\varphi + \zeta, \quad \zeta \in \mathcal{N}(A^\top).$$

This theorem makes much more sense geometrically. We did a derivation of least squares using the fundamental theorem which is cool; we revisit this anyways later on in the linear regression lecture.

3.3 Matrix Concepts

Things we didn't cover in lecture that I need to review and note on (for my own sake): determinants, inverses, similar matrices, eigenvalues and diagonalizability, ...

Briefly cover eigenvalues, determinants. Similarity for spectral thm.

Some matrices with special structure:

- Square, diagonal, triangular (upper or lower).
- Symmetric: a square matrix A such that $A = A^\top$ (more on this later; in fact, an *entire class* on it)
- **Orthogonal**: a square matrix A such that $AA^\top = A^\top A = I$. These are interesting since this implies $A^{-1} = A^\top$. So for any vector x, $\|Ax\|_2^2 = x^\top A^\top Ax = x^\top x = \|x\|_2^2$, so orthogonal matrices don't change the norm of x. Hence, they actually represent rotations.

- **Dyads** (i.e. "outer products"): matrices of the form $uv^\top$. Let $A = uv^\top$. Then the (i, j)-th entry of A is
$$A_{ij} = e_i^\top (uv^\top)e_j = (e_i^\top u)(v_j^\top e_j) = u_i y_j,$$
(where e_i, e_j are the elementary vectors) which is reminiscent of the inner product.

Outer products are quite useful in practice, for multiple reasons. If you have a data matrix like a time series, and observe that it is close to an outer product, then you can tell that you have factors which are scaled versions of each other.

We can also find what the rank of an outer product is. If $A = xy^\top$, then an element of its range looks like $Az = (xy^\top)z = x(y^\top z)$. This last term is a scalar (as it's an inner product), so all outputs are scaled versions of x, and thus A is rank one. In fact, all dyads are rank one.

Here are some other matrix concepts that I personally find helpful. The first is using the standard basis vectors $e_1, \dots, e_n$ to express information. We already saw this idea earlier when analyzing the (i, j)th position of the dyad $A = uv^\top$. If we want the ith row of a matrix A or the ith entry of a column vector, we can multiply on the left by $e_i^\top$. If we want the jth row of the matrix or the jth entry of a row vector, we can multiply on the right by e_j

Here's an example of where it's helpful. Recall from earlier that given matrices $A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{n,p}$, their matrix product can be written as $AB = \sum_{i=1}^n a_i \beta_i^\top$. If we were to insert a diagonal matrix $\mathrm{diag}(c_1, \dots, c_n)$ in between this product, we can break it down as

$$\begin{aligned}
A \ \mathrm{diag}(c_1, \dots, c_n) \ B &= [A(c_1 e_1) \ \dots \ A(c_n e_n)]B && \text{(using } e_1, \dots, e_n \text{ for the columns)} \\
&= [c_1(Ae_1) \ \dots \ c_n(Ae_n)]B \\
&= [c_1 a_1 \ \dots \ c_n a_n]B && \text{(using } a_1, \dots, a_n \text{ as columns of } A\text{)} \\
&= \sum_{i=1}^n c_i a_i \beta_i^\top && \text{(using the dyadic matrix product above)}
\end{aligned}$$

In short, introducing a diagonal matrix turns this product into a weighted sum of dyads. Useful to know for later when we discuss spectral decomposition and SVD!

3.4 Matrix Norms

Definition 9 (Matrix norm). A function $f : \mathbb{R}^{m,n} \to \mathbb{R}$ is a *matrix norm* if, analogously to the vector case, it satisfies three standard axioms. Namely, for all $A, B \in \mathbb{R}^{m,n}$ and $\alpha \in \mathbb{R}$,

- $f(A) \geq 0$, and $f(A) = 0$ if and only if $A = 0$;
- $f(\alpha A) = |\alpha| f(A)$;
- $f(A + B) \leq f(A) + f(B)$.

Many of the popular matrix norms also satisfy a fourth condition called *sub-multiplicativity*: for any conformably sized matrices A, B, $f(AB) \leq f(A)f(B)$.

Probably the most common example of a matrix norm is the Frobenius norm.

Definition 10 (Frobenius norm). The *Frobenius norm* $\|A\|_F$ is simply the standard Euclidean ℓ_2 vector norm applied to the vector formed by all elements of $A \in \mathbb{R}^{m,n}$, i.e.
$$\|A\|_F = \sqrt{\mathrm{tr}\ AA^\top} = \sqrt{\sum_{i=1}^m \sum_{j=1}^m |a_{ij}|^2}.$$

From this definition, it can also be interpreted as the sum of the eigenvalues of $AA^\top$.

For any $x \in \mathbb{R}^n$, it holds that $\|Ax\|_2 \leq \|A\|_F \|x\|_2$ (as a result of the Cauchy-Schwarz inequality applied to $|a_i^\top x|$). Finally, the Frobenius norm is also sub-multiplicative: for any $B \in \mathbb{R}^{n,p}$, it holds that $\|AB\|_F \leq \|A\|_F \|B\|_F$.

Another type of matrix norm are the operator norms, which give a characterization of the *maximum* input-output *gain* of the linear map $u \mapsto Au$. Each choice of ℓ_p norm to measure the inputs and outputs in gives rise to a different norm.

Definition 11 (Induced Operator norm)**.** The *operator norm* induced by a given ℓ_p norm is defined as

$$\|A\|_p = \max_{u \neq 0} \frac{\|Au\|_p}{\|u\|_p} = \max_{\|u\|=1} \|Au\|_p.$$

By definition, $\|Au\|_p \leq \|A\|_p \|u\|_p$ since $\|A\|_p$ is the max gain achieved with any choice of u. From this, it follows that any operator norm is sub-multiplicative, since for two operators B, A,

$$\|Bu\| \leq \|B\|_p \|u\|_p \implies \|ABu\|_p \leq \|A\|_p \|Bu\|_p \leq \|A\|_p \|B\|_p \|u\|_p,$$

and $\max_{\|u\|=1} \|ABu\|_p = \|AB\|_p$ by definition, so $\|AB\|_p \leq \|A\|_p \|B\|_p$.

For the usual values of $p = 1, 2, \infty$, we have the following results (not obvious):

- The ℓ_1-induced norm corresponds to the largest absolute column sum

  $$\|A\|_1 = \max_{\|u\|_1=1} = \max_{j=1,\ldots,n} \sum_{i=1}^m |a_{ij}|,$$

  since $Au = u_1 a_1 + \cdots + u_n a_n$, so maximizing the weight of the largest absolute column maximizes the norm.

- The ℓ_∞-induced norm corresponds to the largest absolute column sum

  $$\|A\|_\infty = \max_{\|u\|_\infty=1} = \max_{i=1,\ldots,m} \sum_{i=1}^n |a_{ij}|.$$

- The ℓ_2-induced norm (sometimes called the *spectral* norm corresponds to the square root of the largest eigenvalue $\lambda_{\max}$ of $A^\top A$:

  $$\|A\|_2 = \max_{\|u\|_2=1} \|Au\|_2 = \sqrt{\lambda_{\max}(A^\top A)}.$$

  This follows from the spectral decomposition of symmetric matrices, which we'll see later.

There's also something called the *spectral radius* $\rho(A) = \max_i |\lambda_i(A)|$ which measures the maximum modulus of the eigenvalues of A. It turns out that $\rho(A) \leq \|A\|_p$ for any induced matrix norm $\|\cdot\|_p$, and that $\rho(A) \leq \min(\|A\|_1, \|A\|_\infty)$, i.e. that the radius is no larger than the maximum row or column sum of $|A|$.

4 Thursday, January 28th: Matrices II

4.1 Orthogonalization: Gram-Schmidt

Recall that a basis $(u_i)_{i=1}^n$ is said to be *orthogonal* if $u_i^\top u_j = 0$ for $i \neq j$. If $\|u_i\|_2 = 1$, the basis is said to be *orthonormal.*

Orthogonalization is the process where we find an orthonormal basis of the span of given vectors (which could be linearly dependent!). Specifically, given vectors $a_1, \ldots, a_k \in \mathbb{R}^n$, an orthogonalization procedures computes vectors $q_1, \ldots, q_n \in \mathbb{R}^n$ such that

$$S := \text{span}\{a_1, \ldots, a_k\} = \text{span}\{q_1, \ldots, q_r\},$$

where $r = \dim S$ and $(q_1, \ldots, q_r)$ is an orthonormal basis for S.

One useful concept for this process is that of a projection onto a line. We have a point $a \in \mathbb{R}^n$ which we want to project onto a line $L(q) := \{tq : t \in \mathbb{R}\}$, where q is a unit vector in $\mathbb{R}^n$. This amounts to solving the optimization problem

$$\min_{t \in \mathbb{R}} \|a - tq\|_2$$

for the closest point on $L(q)$ to a. The resulting vector $a_{\text{proj}} := t^* q$ (t^* being the optimal value) is this projection onto the line $L(q)$. This optimization problem has a simple solution (by geometry):

$$a_{\text{proj}} = (q^\top a)q.$$

A simple interpretation of this is that we are *removing the component* of a along q, since $a - a_{\text{proj}}$ and a_{proj} are orthogonal. Gram-Schmidt uses this idea to then create a set of vectors which are orthogonal to each other. The big idea is to first orthogonalize each vector w.r.t. the previous ones then normalize it to have norm one.

Theorem 12 (Gram-Schmidt Procedure). Let $a_1, \ldots, a_n$ be linearly independent. The Gram-Schmidt procedure below will output an orthonormal basis for span$\{a_1, \ldots, a_n\}$.

- Set $\tilde{q}_1 = a_1$.
- Normalize: set $q_1 = \tilde{q}_1 / \|\tilde{q}_1\|_2$.
- Remove component of q_1 in a_2:: set $\tilde{q}_2 = a_2 - (a_2^\top q_1)q_1$.
- Normalize: set $q_2 = \tilde{q}_2 / \|\tilde{q}_2\|_2$.
- Remove component of q_1, q_2 in a_3: set $\tilde{q}_3 = a_3 - (a_3^\top q_1)q_1 - (a_3^\top q_2)q_2$.
- Normalize: set $q_3 = \tilde{q}_3 / \|\tilde{q}_3\|_2$.
- etc.

We start by projecting each a_i onto the previous normalized basis vectors, which makes it orthogonal to them. Then we normalize so that our basis becomes orthonormal.

This is well-defined, since if $\tilde{q}_i$ is ever 0, this means that the vectors are linearly dependent, which we assumed against. This means we can easily modify the above to produce a basis in the linearly dependent case; simply skip a_i if the result $\tilde{q}_i$ turns out to be 0.

4.2 QR Decomposition

The basic goal of QR decomposition is to factor a matrix as a product of two matrices which hopefully have simple structure to exploit. This will make it easier to solve equations $Ax = y$, since we can instead analyze $Cx = z$ and then $Bz = y$ as

$$Ax = B(Cx) = Bz = y.$$

For example, if we could factor $A = BC$ where B is some rotation, then we could easily solve this by taking its inverse rotation.

The QR decomposition is just the Gram-Schmidt procedure applied to the columns of the matrix, with the result applied in matrix form.

Consider an $m \times n$ matrix $A = (a_1, \dots, a_n)$ with columns a_i, and assume the columns are linearly independent. Then A is full column-rank, so each step of G-S can be written as

$$a_i = (a_i^\top q_1)q_1 + \cdots + (a_i^\top q_{i-1})q_{i-1} + \|\tilde{q}_i\|_2 q_i, \quad i = 1, \dots, n$$

where recall that $\tilde{q}_i = a_i - (a_i^\top q_1)q_1 - \cdots - (a_i^\top q_{i-1})q_{i-1}$ (the remaining components of a_i, per se). We write this as

$$a_1 = r_{i1}q_1 + \cdots + r_{i,i-1}q_{i-1} + r_{ii}q_i, \quad i = 1, \dots, n,$$

where $r_{ij} = (a_i^\top q_j)$ for $1 \le j \le i-1$ and $r_{ii} = \|\tilde{q}_i\|_2$.

Since we got this output from G-S, we know the q_i's form an orthonormal basis, so the matrix $Q = (q_1, \dots, q_n)$ satisfies $Q^\top Q = \mathbf{I}_n$ (i.e. is an *orthonormal* matrix). The QR decomposition thus allows us to write the matrix in a *factored* form

$$A = QR, \quad Q = \begin{bmatrix} q_1 & \dots & q_n \end{bmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & 0 & r_{nn} \end{pmatrix}$$

where Q is an $m \times n$ matrix with $Q^\top Q = \mathbf{I}_n$ and R is $n \times n$ upper-triangular.

With this decomposition, this makes it very efficient to solve linear equations, since any equation of the form $Ax = y$ can be written as $QRx = y$. We can then multiply by $Q^\top$ to get $Rx = Q^\top y$, and R is upper-triangular, so we can efficiently solve this using back-substitution.

When the columns of A are not independent, G-S simply skips over any zero vectors $\tilde{q}_j$ that we encounter. In matrix form, this means we obtain a factorization $A = QR$, with $Q \in \mathbb{R}^{m \times r}$, $r = \text{rank}(A)$, and R having an upper staircase form like

$$R = \begin{pmatrix} * & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}$$

This is just as simple for us to solve with back-substitution, but we can also permute the columns of R so that we bring all the pivot entries to thhe front.

4.3 Kernels in Machine Learning

This section is quite rough since it was on the fly during lecture. We revisit this later on in the least-squares lecture, so I'd just read that section instead.

Recall that in least-squares, our training problem is solving the optimization problem

$$\min_{w,b} \mathcal{L}(X^\top w + b \cdot 1, y),$$

where the pairs of poinits (X_i, y_i) with $X_i \in \mathbb{R}^n, y_i \in \mathbb{R}$ form our training set.

Our predictions are then given by $\hat{y}(x) = w^\top x + b$ once we find these optimal w, b. This means we can compute an error between the *actual* measured output y_i with our predictions $\hat{y}_i$, which forms our loss function that we try to optimize:

$$\min_{w,b} \sum_{i=1}^{n} (y_i - (w^\top x_i + b))^2.$$

The shape of our data is $X = \begin{bmatrix} x_1 & \dots & x_m \end{bmatrix}$, i.e. an $n \times m$ matrix, so

$$X^\top w + b \cdot 1 = \begin{bmatrix} x_1^\top \cdot w + b \\ \vdots \\ x_m^\top \cdot w + b \end{bmatrix}.$$

Analyzing the term $X^\top w$ is the key to understanding the impact of kernels.

Suppose we simplify our problem to

$$\min_{w} \mathcal{L}(X^\top w) + \lambda \|w\|_2^2$$

where $\lambda > 0$ is a penalty/regularization term on our weights trying to force our training process to find a more "robust example for weights w." It helps account for noise in your dataset, so that small changes in the x_i won't result in large changes in our loss.

Let's analyze the simple case $m = 2$ (# of data points) and $n = 3$ (dimension of data points). Invoking the fundamental theorem of linear algebra, we can write our weight vector in terms of a vector in the span of our x_1, x_2, i.e. $w = Xv + r$ where $v \in \text{span}\{x_1, \dots, x_m\}$ and $r \in \mathcal{N}(X^\top)$, so $X^\top r = 0$. So $X^\top w = X^\top X v + X^\top r = X^\top X v$, which let's us rewrite the optimization problem as

$$\min_{w,v} \mathcal{L}[X^\top X v, y] + \lambda v^\top X^\top X v.$$

In other words, we've converted the optimization in terms of X instead into an optimization in terms of the matrix $X^\top X$. We refer to this matrix as the *kernel* matrix $K = X^\top X$, or the Gram matrix of our data. Note that

$$K_{ij} = e_i^\top K e_j = e_i^\top X^\top X e_j = x_i^\top x_j = \langle x_i, x_j \rangle.$$

This inspires us to think of other inner products that aren't just the usual scalar product $\langle x_i, x_j \rangle = x_i^\top x_j$. We could use nonlinear products instead which let use use richer models on transformed data instead.

5 Tuesday, February 2nd: Symmetric Matrices

Today is about symmetric matrices.

5.1 Symmetric Matrices

Recall that a square $n \times n$ matrix A is *symmetric* if $A = A^\top$. The set of symmetric $n \times n$ matrices is a subspace of $\mathbb{R}^{n,n}$, and is denoted by $\mathbb{S}^n$.

Example 5.1 (Sample Covariance Matrix)**.** For one dimensional data $z_1, \dots, z_m \in \mathbb{R}$, we can define the mean and variance of the data as

$$\hat{z} = \frac{1}{m}\sum_{i=1}^m z_i$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^m (z_i - \hat{z})^2.$$

We can extend this notion to multi-dimensional points, which results in an important symmetric matrix we will see many times: the covariance matrix.

The covariance matrix is a way for us to understand the variance along any direction u ($\|u\|_2 = 1$). Let $x_1, \dots, x_m \in \mathbb{R}^n$, which has sample mean $\hat{x} = \frac{1}{m}\sum_{i=1}^m x_i$. If we project our points x onto this line by using $x' = (x^\top u)u$, we reduce this to the previous one dimensional case and analyze the mean and variance again. This creates "scores" of our points along the line u defined by $z_i = (x_i - \hat{x})^\top u$. Then following the same procedure from before, we can compute variance as

$$\begin{aligned}\sigma^2(u) &= \frac{1}{m}\sum_{i=1}^m (z_i - \hat{z})^\top (z_i - \hat{z}) \\ &= \frac{1}{m}\sum_{i=1}^m u^\top (x_i - \hat{x})(x_i - \hat{x})^\top u \\ &= u^\top C u\end{aligned}$$

where $C = \frac{1}{m}\sum_{i=1}^m (x_i - \hat{x})(x_i - \hat{x})^\top$ is the covariance matrix. From this, one can see that we can use it to compute arbitrary sample variances.

Example 5.2 (Hessian matrix)**.** The Hessian of a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at a point x is the matrix containing the second derivates of the function at that point. It is given by

$$H_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j},\ 1 \le i, j \le n.$$

We usually denote the Hessian by $\nabla^2 f(x)$. Since the second-derivative is the same regardless of order of derivative, $H_{ij} = H_{ji}$, so it is symmetric.

As a quick aside, we can talk about quadratic functions being written in terms of matrices and vectors. Supposes we have a quadratic function

$$q(x) = x_1^2 + 2x_1x_2 + 3x_2^2 + 4x_1 + 5x_2 + 6,$$

which has Hessian

$$H = \begin{bmatrix} 2 & 2 \\ 2 & 6 \end{bmatrix}.$$

We can write the degree two monomials stricly in terms of the Hessian as

$$x_1^2 + 2x_1x_2 + 3x_2^2 = \frac{1}{2}x^\top Hx.$$

Taking this further, any quadratic function can be written as the sum of a quadratic term involving the Hessian, and an affine term:

$$q(x) = \frac{1}{2}x^\top Hx + c^\top x + d, \quad c^\top = [4\ 5],\ d = 6.$$

A **quadratic form** is a quadratic function with no linear and constant terms, that is $c = 0, d = 0$. Dropping the scalar, they generally look like

$$q(x) = x^\top Hx, \quad H \in \mathbb{S}^n.$$

A special quadratic form is when H is diagonal, which leads to the form

$$q(x) = x^\top \mathrm{diag}(a_1, \ldots, a_n)x = \sum_{i=1}^n a_i x_i^2,$$

that is $q(x)$ is a linear combination of pure squares x_i^2 (i.e. no cross terms x_ix_j).

We can finally get to this most important result, which states that every symmetric matrix is orthogonally similar to a diagonal matrix.

Theorem 13 (Spectral Theorem)**.** Let A be a symmetric $n \times n$ matrix and let $\lambda_i \in \mathbb{R}, (i = 1, \ldots, n)$ be its eigenvalues with multiplicty. Then there exists a set of orthogonal vectors $u_i \in \mathbb{R}^n$, $i = 1, \ldots, n$ such that $Au_i = \lambda_i u_i$. Equivalently, there exists an orthogonal matrix $U = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}$ (i.e. $UU^\top = U^\top U = \mathbf{I}_n$) such that

$$A = U\Lambda U^\top = \sum_{i=1}^n \lambda_i u_i u_i^\top, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n).$$

In other words, A can be written as the weighted sum of dyads which are mutually orthogonal.

Write up Rayleigh quotient and matrix gain

5.2 Positive-Semidefinite Matrices

Definition 14 (Positive Semidefiniteness)**.** A symmetric matrix $A \in \mathbb{S}^n$ is said to be *positive semidefinite* (PSD) if the associated quadratic form is nonnegative, i.e.,

$$x^\top Ax \geq 0, \quad \forall x \in \mathbb{R}^n.$$

If, moreover,

$$x^\top Ax > 0, \quad \forall x \in \mathbb{R}^n \setminus \{0\},$$

then A is said to be *positive definite* (PD). We denote a symmetric positive semidefinite (resp. positive definite) matrix by $A \succeq 0$ (resp. $A \succ 0$).

Some immediate properties are that a PSD matrix is actually positive definite if and only if it is invertible. Also, it holds that

$$A \succeq 0 \iff \lambda_i(A) \geq 0,\ i = 1, \ldots, n.$$

$$A \succ 0 \iff \lambda_i(A) > 0,\ i = 1, \ldots, n.$$

Matrix square-root, Cholesky decomp, and PSD's with ellipsoids. Maybe schur complements?

5.3 Principal Component Analysis

The motivation behind PCA is that if we have a low-dimensional data set existing in a high-dimensional space, then we should instead view it from an angle where the "variance" of thhe data is highest. In other words, we want to project our data onto a direction which maximizes the resulting variance of our points, since that lets us tell the points apart the most.

When we do this, recall that the variance of the projections of our points onto a direction u is given by the covariance matrix (Example 5.1) This gives us the optimization problem

$$\max_u u^\top C u : \quad \|u\|_2 = 1$$

where C is the empirical covariance matrix and we impose the unit vector constraint so that the optimization is actually tractable.

To solve this, we can look at the spectral decomposition of C given by

$$C = \sum_{i=1}^{p} \lambda_i u_i u_i^\top,$$

with $\lambda_1 \geq \cdots \geq \lambda_p$ and $U = [u_1, \ldots, u_p]$ is orthogonal ($U^\top U = I$). Then a solution to the quadratic form

$$\max_{u : \|u\|_2 = 1} u^\top C u$$

is $u^* = u_1$ where u_1 is the eigenvector of C that corresponds to the largest eigenvalue λ_1 as we discussed before.

Later we will see how to arrive at this using SVD.

6 Thursday, February 4th: Singular Value Decomposition

Add in more understandable motivation for SVD

6.1 The Singular Value Decomposition (SVD)

One motivation for SVD is that we'd like to analyze data better and understand potential clusters that may arise from it. First, we need to understand what *dyads* are. A matrix $A \in \mathbb{R}^{m,n}$ is called a *dyad* if it can be written as

$$A = pq^\top,$$

for some vectors $p \in \mathbb{R}^m, q \in \mathbb{R}^n$, i.e. the product of a column and a row vector. Element-wise the above reads

$$A_{ij} = p_i q_j, \quad 1 \leq i \leq m,\ 1 \leq j \leq n.$$

Breaking this down, notice that the jth column of A is given by

$$A(\cdot, j) = Ae_j = q_j p,$$

so all the columns are copies of p scaled by a number, namely q_j.

Similarly for the rows, the ith row of A is

$$A(i, \cdot) = e_i^\top A = p_i q^\top,$$

so the rows are scalar multiples of $q^\top$.

The singular value decomposition theorem states that any matrix can be written as a sum of dyads with the special property that if

$$A = \sum_{i=1}^r p_i q_i^\top,$$

then p_i and q_i are mutually orthogonal, i.e. $p_i^\top p_j = q_i^\top q_j = 0$ for all $i \neq j$. This special property is particularly important; it's actually really easy to write any arbitrary matrix as the sum of dyads. For example, we can easily break down the following matrix as

$$\begin{aligned} A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} &= \sum p_i q_i^\top \\ &= 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} + 2 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} + \dots \\ &= 1 \cdot e_1 e_1^\top + 2 e_1 e_2^\top + 3 e_1 e_3^\top + \dots. \end{aligned}$$

So the main contribution of SVD is that the dyads are mutually orthogonal, i.e. the p_i and q_i respectively are mutually orthogonal. It provides a three-term factorization which is similar to spectral factorization, but holds for any, possibly non-symmetric and rectangular, matrix A.

Theorem 15 (Singular Value Decomposition). Any matrix $A \in \mathbb{R}^{m,n}$ can be factored as

$$A = U\tilde{\Sigma}V^\top = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$$

where $U \in \mathbb{R}^{m,m}$ and $V \in \mathbb{R}^{n,n}$ are orthogonal matrices (i.e. $U^\top U = I_m$ and $V^\top V = I_n$), and $\tilde{\Sigma} \in \mathbb{R}^{m,n}$ is a matrix having the first $r = \operatorname{rank}(A)$ diagonal entries $(\sigma_1, \ldots, \sigma_r)$ positive and decreasing in magnitude, and all other entries zero:

$$\tilde{\Sigma} = \begin{bmatrix} \Sigma & 0_{r,n-r} \\ 0_{m-r,r} & 0_{m-r,n-r} \end{bmatrix}, \quad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \succ 0.$$

Before we discuss the proof, let's analyze in further detail what SVD is really doing. We look at its effect as a matrix operation

Add in proof of SVD

$$x \mapsto Ax = U\tilde{\Sigma}V^\top x.$$

Recall that orthogonal matrices represent rotations. Then the operation $V^\top x =: \bar{x}$ represents a rotation of x, since $V^\top$ is orthogonal.

Next, how does $\tilde{\Sigma}$ operate on $\bar{x}$? It will scale some dimensions of $\bar{x}$ by the singular values, and could either crush some dimensions of $\bar{x}$ or add more. For example, if we have $m = 3, n = 5$, we could have the following example of crushing the dimensions $\bar{x}$:

$$\tilde{\Sigma} = \left[\begin{array}{ccc|cc} 1 & & & \vert & \vert \\ & 0.2 & & 0 & 0 \\ & & 0 & \vert & \vert \end{array}\right], \quad \bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \implies \tilde{\Sigma}\bar{x} = \begin{bmatrix} x_1 \\ 0.2x_2 \\ 0 \end{bmatrix}.$$

On the other hand, if $m = 5, n = 4$, we get an example of adding dimensions to $\bar{x}$.

$$\tilde{\Sigma} = \left[\begin{array}{ccc|c} 1 & & & \vert \\ & 0.2 & & 0 \\ & & 0 & \vert \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad \bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \implies \tilde{\Sigma}\bar{x} = \begin{bmatrix} x_1 \\ 0.2x_2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

In summary, $\tilde{\Sigma}$ scales $\bar{x}$ and acts as a dimension adjustment.

Since we don't care about these last few columns of U corresponding to the zeroes in $\tilde{\Sigma}$, we get a compact form of SVD.

Theorem 16 (Compact-form SVD). Any matrix $A \in \mathbb{R}^{m,n}$ can be expressed as

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^\top = U_r \Sigma V_r^\top, \quad \Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_r)$$

where $r = \operatorname{rank}(A), U_r = [u_1 \ \dots \ u_r]$ is such that $U_r^\top U_r = I_r$, $V_r = [v_1 \ \dots \ v_r]$ is such that $V_r^\top V_r = I_r$, and $\sigma_1 \geq \cdots \geq \sigma_r > 0$.

The positive number σ_i are the *singular values* of A, vectors u_i are the *left singular vectors* of A, and v_i the *right singular vectors*. These quantities satisfy

$$Av_i = \sigma_i u_i, \quad u_i^\top A = \sigma_i v_i, \quad i = 1, \dots, r.$$

Morever, $\sigma_i^2 = \lambda_i(AA^\top) = \lambda_i(A^\top A), \ i = 1, \dots, r$, and u_i, v_i are the eigenvectors of $AA^\top$ and of $A^\top A$, respectively.

Let's unpack this last statement more. Starting with the full form SVD of an $m \times n$ matrix $A = U\tilde{\Sigma}V^\top$ with $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n \times n}$, we can write out $A^\top A$ as

$$\begin{aligned} A^\top A &= (V\tilde{\Sigma}^\top U^\top)(U\tilde{\Sigma}V^\top) = V\tilde{\Sigma}^\top (U^\top U)\tilde{\Sigma}V^\top \\ &= V\tilde{\Sigma}^\top\tilde{\Sigma}V^\top = V \left(\begin{array}{c|c} \operatorname{diag}(\sigma_1^2, \dots, \sigma_r^2) & 0 \\ \hline 0 & 0 \end{array}\right) V^\top \qquad \text{(since } U \text{ is orthogonal)} \end{aligned}$$

where the inside matrix is $n \times n$. This is the spectral decomposition of $A^\top A$, which is a symmetric matrix.

Similarly,

$$AA^\top = U \left(\begin{array}{c|c} \operatorname{diag}(\sigma_1^2, \dots, \sigma_r^2) & 0 \\ \hline 0 & 0 \end{array}\right) U^\top \qquad \text{(since } V \text{ is orthogonal)}$$

where the inside matrix is $m \times m$. This is the spectral decomposition applied to the symmetric matrix $AA^\top$. As a result, $AA^\top \succeq 0$ and $A^\top A \succeq 0$, since all the eigenvalues are squared values and must be non-negative.

To summarize, the SVD theorem allows us to write any matrix as a sum of dyads $A = \sum_{i=1}^r \sigma_i u_i v_i^\top$ where the σ_i provide the "strength" of the corresponding dyad and the rank is how many dyads we need to reconstruct A.

In practice, we'll employ algorithms to compute the SVD of matrices for us. The complexity of the algorithm which computes it via a sequence of linear transformation is $O(nm \min(n, m))$, but this can be sped up for sparse matrices if we're only interested in the largest few singular values.

6.2 Matrix properties via SVD

We can infer many things from the SVD. For example, the rank r of A is the number of nonzero singular values, i.e. nonzero entries on the diagonal of $\tilde{\Sigma}$.

The SVD also gives us a basis for the range and nullspace of A. Recall that the range of A is $\mathcal{R}(A) = \{Ax : x \in \mathbb{R}^n\}$. Then the first r columns of U give an orthonormal basis spanning the range of A, i.e.

$$\mathcal{R}(A) = \mathcal{R}(U_r), \quad U_r = \begin{bmatrix} u_1 & \dots & u_r \end{bmatrix}.$$

Since $r = \operatorname{rank} A$, by the fundamental theorem the dimension of the nullspace is $n - r$. An orthonormal basis spanning $\mathcal{N}(A)$ is given by the last $n - r$ columns of V, i.e.

$$\mathcal{N}(A) = \mathcal{R}(V_{nr}), \quad V_{nr} = \begin{bmatrix} v_{r+1} & \dots & v_n \end{bmatrix}.$$

These two facts are not immediately obvious, but here's some intuition. Let A be a 3×5 matrix with entries

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The rank of A is clearly 2, and the range is $\mathcal{R}(A) = \{Ax : x \in \mathbb{R}^5\}$, which expands to

$$\left\{ Ax = \begin{bmatrix} x_1 \\ .1x_2 \\ 0 \end{bmatrix} : x_1, x_2 \in \mathbb{R} \right\} = x_1 e_1 + (.1x_2)e_2,$$

meaning that $\{e_1, e_2\}$ is an orthonormal basis for the range. When we add back in the U and $V^\top$, all that happens to the basis is it gets rotated.

(I will say it's still not clear to me personally yet why this means the columns of U and V should be the basis, but I'm thinking about it).

We can also look at matrix norms using SVD. The *squared Frobenius* matrix norm of a matrix $A \in \mathbb{R}^{m,n}$ can be defined as

$$\|A\|_F^2 = \sum_{i,j} A_{i,j}^2 = \text{tr } A^\top A = \sum_{i=1}^n \lambda_i(A^\top A) = \sum_{i=1}^n \sigma_i^2,$$

where σ_i are the singular values of Aa. Hence the squared Frobenius norm is just the sum of the squares of the singular values.

The *squared spectral* matrix norm $\|A\|_2^2$, defined below, is equal to the maximum eigenvalue of $A^\top A$, so

$$\|A\|_2^2 = \sigma_1^2,$$

i.e. the spectral norm coincides with the maximal singular value of A.

The *nuclear* norm of a matrix A is defined as

$$\|A\|_* = \sum_{i=1}^r \sigma_i, \quad r = \text{rank} A.$$

This appears in problems related to low-rank matrix completion of rank minimization problems.

We can actually prove that the squared spectral norm above is equal to the maximum eigenvalue of $A^\top A$ as follows. Recall that $\|A\|_2^2 = \max_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2}$, i.e. the largest matrix gain or the induced operator norm. Then using $A = U\tilde{\Sigma}V^\top$,

$$\begin{aligned} \|A\|_2^2 = \max_{x\neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2} &= \max_{x \neq 0} \frac{\|\tilde{\Sigma}V^\top x\|_2^2}{\|x\|_2^2} && (\text{since } U^\top U = I) \\ &= \max_{z \neq 0} \frac{\|\tilde{\Sigma}z\|_2^2}{\|z\|_2^2} && (\text{with } z = V^\top x) \\ &= \max_z \|\tilde{\Sigma}z\|_2^2 : \quad z^\top z = 1 \\ &= \max_z \sum_{i=1}^r \sigma_i^2 z_i^2 : \quad z^\top z = 1 \\ &\leq \max_{1 \leq i \leq r} \sigma_i^2, \end{aligned}$$

which is achieved at the largest singular value, σ_1^2. So σ_1 gives us a measure of how much amplification can happen multiplying by A.

We can use this idea to define a useful matrix property.

Definition 17 (Condition Number). The *condition number* of an invertible matrix $A \in \mathbb{R}^{n,n}$ is defined as the ratio between the largest and the smallest singular value

$$\kappa(A) = \frac{\sigma_1}{\sigma_n} = \|A\|_2 \cdot \|A^{-1}\|_2.$$

The condition number provides a quantitative measure of how close A is to being singular. The larger $\kappa(A)$ is, the more close to singular A is.

It also provides a measure of the sensitivity of the solution of a system of linear equations to changes in the equatin coefficients.

There are other matrix properties we could derive, such as the Moore-Penrose pseudo-inverse, but we did not go over them in lecture (although I added condition number here for my own notes).

6.3 Low-rank matrix approximation

One problem we may consider frequently is that of approximating a matrix $A \in \mathbb{R}^{m,n}$ with a matrix of lower rank. For example, if we have a camera taking a still video of a scene, and then a person walks by, the image matrix suddenly goes from rank 1 to nonzero rank. Looking for a one-rank approximation is a way we could try to remove the person from the picture.

In particular, we consider the following rank-constrained approximation problem

$$\min_{A_k \in \mathbb{R}^{m,n}} \|A - A_k\|_F^2 \quad s.t. \text{rank}(A_k) = k,$$

where $1 \leq k \leq r$ is given.

Let $A = U\tilde{\Sigma}V^\top = \sum_{i=1}^r \sigma_i u_i v_i^\top$ be an SVD of A. An optimal solution to the above problem is to simply truncate this summation to the first k terms, i.e. take the k dyads with largest singular values,

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^\top.$$

The ratio

$$\eta_k = \frac{\|A_k\|_F^2}{\|A\|_F^2} = \frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_r^2}$$

indicates what fraction of the total *variance* (Frobenius norm) in A is explained by the rank k approximation of A. Plotting η_k as a function of k can give helpful guidance to what k would be best to approximate A with.

6.4 Link with PCA

PCA is the same as low rank approximations, except you center your data first, as we'll see shortly.

Let $x_i \in \mathbb{R}^n$, $i = 1, \ldots, m$ be the data points, so $X = \begin{bmatrix} x_1 & \ldots & x_m \end{bmatrix}$. Then replace each point with its centered version by subtracting the mean $\bar{x} = \frac{1}{m}\sum_{i=1}^m x_i$ to get

$$\tilde{X} = \begin{bmatrix} \tilde{x}_1 & \ldots & \tilde{x}_m \end{bmatrix}, \quad \tilde{x}_i = x_i - \bar{x}.$$

As we discussed last lecture (see Example 5.1), we want to look for a normalized direction in data space $z \in \mathbb{R}^n$, $||z||_2 = 1$, such that the variance of the projections of the centered data pointst on the line determined by z is maximal. These components are given by $\alpha_i = \tilde{x}_i^\top z$, $i = 1, \ldots, m$, so the mean-square variation of the data along this direction is

$$\frac{1}{m}\sum_{i=1}^m \alpha_i^2 = \sum_{i=1}^m z^\top \tilde{x}_i \tilde{x}_i^\top z = z^\top \tilde{X}\tilde{X}^\top z.$$

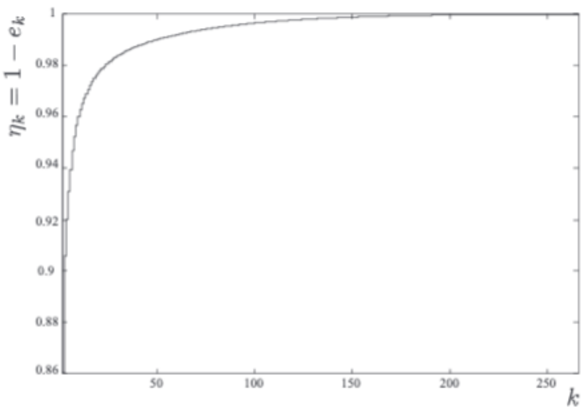


Figure 2: A plot of η_k vs. k taken from El Ghaoui's slides. Here, $k = 9$ captures a good amount of variance.

Hence, there's a very simple relationship between PCA and the covariance matrix $C = \frac{1}{m}\sum_{i=1}^m (x_i - \bar{x})(x_i - \bar{x})^\top$ once you've centered the data.

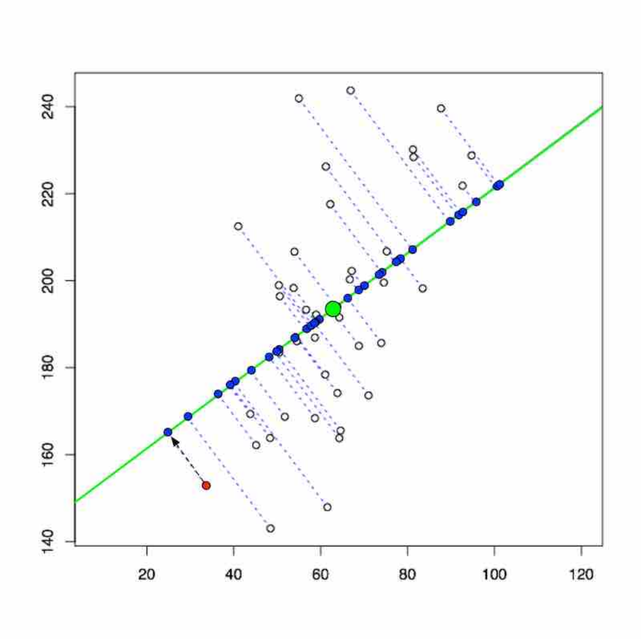


Figure 3: Illustration of projecting points onto the direction z. Taken from Lior Pachter's blog.

Finding the maximal variance direction reduces to solving the following optimization problem:

$$\max_{z \in \mathbb{R}^n} z^\top (\tilde{X}\tilde{X}^\top) z \quad \text{s.t. } \|z\|_2 = 1.$$

We can directly employ SVD to solve this problem. Let

$$\tilde{X} = U_r \Sigma V_r^\top = \sum_{i=1}^r \sigma_i u_i v_i^\top.$$

Then, $H = \tilde{X}\tilde{X}^\top = U_r \Sigma^2 U_r^\top$. We already know that this problem $z^\top H z$ is maximized by the eigenvector corresponding to the largest eigenvalue of H, which is σ_1^2. Hence, the direction is $z = u_1$, and the mean-square variation is proportional to σ_1^2.

Successive principal axes can be found by "removing" the first principal components and repeating these approaches on the deflated data matrix.

In summary, there are two ways to do PCA:

1. Form the covariance matrix C, find it's spectral decomposition, and take its top eigenvector (as we saw last lecture).

2. Form the centered data matrix $\tilde{X}$, and find an svd of $\tilde{X} = U\tilde{\Sigma}V^\top$. Then U contains the eigenvectors of $C = \frac{1}{m}\tilde{X}\tilde{X}^\top$ since $\frac{1}{m}\tilde{X}\tilde{X}^\top = \frac{1}{m}U\tilde{\Sigma}^2U^\top$ is the spectral decomposition for C (as we just saw).

7 Tuesday, February 9th: Linear Equations

7.1 Motivation for Linear Equations

Linear equations are one of the most basic form of relationship among variables in an engineering problem. There's all sorts of applications of linear equations that come up in science, and they form the building block for many other optimization methods, so it's important to know how to solve them.

Here is an example of a system in 3 equations in 2 unknowns:

$$\begin{aligned} x_1 + 4.5x_2 &= 1 \\ 2x_1 + 1.2x_2 &= -3.2 \\ -0.1x_1 + 8.2x_2 &= 1.5. \end{aligned}$$

We can write this in vector format as $Ax = y$, where

$$A = \begin{bmatrix} 1 & 4.5 \\ 2 & 1.2 \\ -0.1 & 8.2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ -3.2 \\ 1.5 \end{bmatrix}.$$

Solutions are $x \in \mathbb{R}^2$ which satisfy the equation. In this case, the equation has no solution, i.e. the system is *infeasible.*

We then went over a lot of examples. I prefer theory so I did not take notes on them.

7.2 Set of Solutions of Linear Equations

Our general setup is that we want to solve the linear equation

$$Ax = y, \quad A \in \mathbb{R}^{m,n}.$$

There are many issues we concern ourselves with, such as existence and uniqueness of solutions, or the characterization of the *solution set*

$$S = \{x \in \mathbb{R}^n : Ax = y\}.$$

Notice that Ax always lies in $\mathcal{R}(A)$, so $S \neq 0 \iff y \in \mathcal{R}(A)$. Hence, there is a solution if and only if $\text{rank}([A\ y]) = \text{rank}(A)$.

When existence is satisfied, the set of solutions is the affine set

$$S = \{\bar{x} + Nz, z \in \mathbb{R}^{n-r}\},$$

where $\bar{x}$ is any vector such that $A\bar{x} = y$, $r = \text{rank}(A)$, and $N \in \mathbb{R}^{n,n-r}$ is a matrix whose columns span the nullspace of A (hence $AN = 0$).

Geometrically, the null space represents the span of the "free variables" in the solutions.

Finding the solution set in the above manner can be useful in optimization. We can use this by turning our usual optimization problem into an unconstrained one, i.e.

$$\min_x f_0(x) : \ Ax = b \implies \min_z f_0(x_0 + Nz),$$

using notation from above.

7.3 Solving Linear Systems

We can also use SVD to easily analyze our linear equation system. As usual, suppose we're solving $Ax = y$ where $A \in \mathbb{R}^{m,n}$ and $y \in \mathbb{R}^m$. If $A = U\tilde{\Sigma}V^\top$ is the SVD of A (see Theorem 15), then $Ax = y$ is equivalent to

$$U\tilde{\Sigma}V^\top x = y \implies \tilde{\Sigma}V^\top x = U^\top y \implies \tilde{\Sigma}\tilde{x} = \tilde{y},$$

where $\tilde{x} = V^\top x, \tilde{y} = U^\top y$.

In other words, we're using U and V to rotate our inputs and outputs to make the system diagonal (as per the usual SVD interpretation). Since $\tilde{\Sigma}$ is a diagonal matrix

$$\tilde{\Sigma} = \begin{bmatrix} \Sigma & 0_{r,n-r} \\ 0_{m-r,r} & 0_{m-r,n-r} \end{bmatrix}, \quad \Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_r) \succ 0,$$

this makes the system very easy to solve or analyze. Note that $\mathcal{N}(A) = m - r$.

Let's analyze the resulting system $\tilde{\Sigma}\tilde{x} = \tilde{y}$, which reduces to

$$\begin{aligned} \sigma_i \tilde{x}_i &= \tilde{y}_i \quad i = 1, \dots, r \\ 0 &= \tilde{y}_i \quad i = r+1, \dots, m. \end{aligned}$$

Clearly we need $\tilde{y}_i$ to be zero for its last $m - r$ components, which happens when $y \in \mathcal{R}(A)$. Otherwise the solution set is empty.

If $y \in \mathcal{R}(A)$, then we can solve for $\tilde{x}$ with our first set of conditions: $\tilde{x}_i = \tilde{y}_i/\sigma_i$ for $i = 1, \dots, r$, with the last $n - r$ components being free (which correspond to elements in the nullspace of A).

When A is full column rank, it has trivial nullspace and hence there is a unique solution.

Example 7.1. Suppose we have a system $Ax = y$ with

$$A = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

This system has no solution if $(y_3, y_4) \neq 0$. If $y_3 = y_4 = 0$ so that $y \in \mathcal{R}(A)$, we then get the solution sest $S = \{x_0 + Nz : z \in \mathbb{R}^2\}$ with

$$x_0 = \begin{bmatrix} y_1/\sigma_1 \\ y_2/\sigma_2 \\ 0 \\ 0 \end{bmatrix}, \quad N = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Finally, we can also use QR decomposition to analyze a linear system. The idea is to reduce the system to a triangular one and then solve using "back-substitution." For example, if A is square and $A = QR$ with $Q^\top Q = I_n$, then $Ax = y$ is equivalent to $Rx = \tilde{y} = Q^\top y$, which is triangular. Then we just have to back substitute to find a solution.

Similarly to using SVD, we're rotating our outputs here to make the system triangular and easier to solve.

8 Thursday, February 11th: Least-Squares and Variants

8.1 Ordinary Least-Squares and Minimum-Norm Solutions

One classical **goal** of modeling is the following: given $A \in \mathbb{R}^{m \times n}, y \in \mathbb{R}^m$, find x such that $Ax \approx y$. In other words, if $A = [a_1 \; \dots \; a_n]$, then we are asking if y can be written as the linear combination of the columns of A, i.e. $y \approx \sum_{i=1}^n x_i a_i$.

The least squares approach uses the Euclidean norm to rate our approximation, solving the optimization problem

$$\min_x \|Ax - y\|_2.$$

Since the objective function is always ≥ 0, we can solve the "ordinary least-squares" problem

$$\min_x \|Ax - y\|_2^2 = \sum_{i=1}^m r_i^2, \quad r := Ax - y,$$

i.e. minimizing the sum of squared residuals.

Geometrically, since $Ax \in \mathcal{R}(A)$, the problem amounts to finding the point $\tilde{y} = Ax^*$ in $\mathcal{R}(A)$ at *minimum distance* from y. By the Projection Theorem (Theorem 7), this is just the orthogonal projection of y onto the subspace $\mathcal{R}(A)$. Notice the similarity to projections on a line, where we minimize $\|ax - y\|_2^2$ for a single vector.

Another interpretation of this problem can be in terms of the rows $a_i^\top$ of A, as we were alluding to above. The problem then becomes

$$\min_x \sum_{i=1}^m (y_i - a_i^\top x)^2.$$

Here, we're trying to fit every component y_i as some linear combination of the input a_i, where x determines the coefficients.

This interpretation hails from machine learning, where this is often useful for prediction. We have some training set where we find what the solution x^* to the above problem is, and then we can predict the output corresponding to a new vector $a \in \mathbb{R}^n$ (a *test point*) by computing $\hat{y} = a^\top x^*$.

We usually write least-squares in the above form because it leads us to an easy proof of the closed-form solution. As $y - Ax^* \in \mathcal{R}(A)^\perp = \mathcal{N}(A^\top)$, we have $A^\top(y - Ax^*) = 0$. This leads us to the following result.

Theorem 18 (Closed-Form Least Squares)**.** Any solution to the least-squares problem must satisfy the **Normal Equations**

$$A^\top A x = A^\top y.$$

This system always admits a solution, and if A is additionally full column rank, then the solution is unique and is given by

$$x^* = (A^\top A)^{-1} A^\top y.$$

Recall that $A^\top A$ is invertible since if it is full rank, $\|Ax\|_2^2$ is never 0 if $x \neq 0$, so $x^\top A^\top A x > 0$ meaning $A^\top A$ is positive definite.

As a corollary, notice that the set of optimal solutions is

$$\mathcal{X}_{\text{opt}} = A^\dagger y + \mathcal{N}(A),$$

where $A^\dagger y$ is the minimum norm point in the optimal set. Adding anything from the null space will still make it a solution that is optimal, just not necessarily minimum norm.

We can also find solutions of OLS via QR decomposition. Assume the columns of $A \in \mathbb{R}^{m \times n}$ are linearly independent so that the QR factorization $A = QR$ exists, with

- $m \times n$ matrix Q satisfies $Q^\top Q = I$,
- $n \times n$ matrix R invertible,
- then $A^\dagger = (A^\top A)^{-1} A^\top = R^{-1} Q^\top$.

This gives us the following algorithm.

- compute the QR factorization of $A = QR$ ($2mn^2$ flops);
- form $z = Q^\top y$ ($2mn$ flops);
- solve the triangular system $Rx = z$ via backward substitution.

In total, our algorithm is identical for solving $Ax = b$ for square invertible A and has complexity is $2mn^2$ flops.

8.2 Variants of Least-Squares

A generalization of the basic LS problem allows for the addition of linear equality constraints on x, resulting in the constrained problem

$$\min_x \|Ax - y\|_2^2, \quad \text{s.t. } Cx = d,$$

where $C \in \mathbb{R}^{p,n}$ and $d \in \mathbb{R}^p$.

This may look difficult to solve now, but we can convert it into a standard LS problem by "eliminating" the equality constraints via a standard procedure. Assume the problem is feasible, and let $\bar{x}$ be such that $C\bar{x} = d$, and let N have columns which are a basis for $\mathcal{N}(C)$. Then all of the feasible points can be expressed as $x = \bar{x} + Nz$, so our unconstrained problem in the variable z becomes

$$\min_z \|\bar{A}z - \bar{y}\|_2^2,$$

where $\bar{A} = AN, \bar{y} = y - A\bar{x}$.

Another variant is *Weighted Least-Squares*, where we want to give different weights to our residuals, instead minimizing $f_0(x) = \sum_{i=1}^m w_i^2 r_i^2$ where $w_i \geq 0$ are the given weights. We can easily convert this as

$$f_0(x) = \|W(Ax - y)\|_2^2 = \|A_w x - y_w\|_2^2,$$

where $W = \operatorname{diag}(w_1, \dots, w_m)$, $A_w = WA$, and $y_w = Wy$. This is again our original LS problem with row-weighted matrix A_w and vector y_w.

One popular variant of LS is the class of regularized LS problems, which are of the form

$$\min_x \ \|Ax - y\|_2^2 + \phi(x),$$

where a "regularization," or *penalty* term $\phi(x)$ is added to our usual LS objective. Typically $\phi(x)$ is proportional to either the ℓ_1 or ℓ_2 norm of x. In the former case, the problem is known as LASSO. We will instead discuss the latter case, known as *ridge regression.*

Consider the problem

$$\min_x \ \|Ax - y\|_2^2 + \lambda\|x\|_2^2, \quad \lambda \geq 0.$$

We can use the fact that the squared ℓ_2 norm of block vectors is the sum of the squared norms of the blocks to rewrite our problem as

$$\|Ax - y\|_2^2 + \lambda\|x\|_2^2 = \|\tilde{A}x - \tilde{y}\|_2^2,$$

where

$$\tilde{A} = \begin{bmatrix} A \\ \sqrt{\lambda} I_n \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} y \\ 0_n \end{bmatrix}.$$

By construction $\tilde{A}$ is full column rank due to the I_n. Here, $\lambda \geq 0$ is a tradeoff parameter between output accuracy and input effort. Since λ is a hyperparameter, we typically try out a bunch of different values to find the one with the best test accuracy on unseen data.

Applying our solution from above, we find that the optimal solution is just

$$x^* = (A^\top A + \lambda I_n)A^\top y,$$

which always exists and is unique since $\tilde{A}$ is always full column rank.

8.3 Kernels for Least-Squares

Before we talk about kernel learning, we're going to try and motivate it. Suppose we're making a nonlinear auto-regressive model for time-series, where y_t is a *quadratic* function of y_{t-1} and y_{t-2}. This can be modeled by

$$y_t = w_1 + w_2 y_{t-1} + w_3 y_{t-2} + w_4 y_{t-1}^2 + w_5 y_{t-1} y_{t-2} + w_6 y_{t-2}^2.$$

We can't exactly model this by a LS formulation with just $x_t = \begin{bmatrix} y_{t-1} & y_{t-2} \end{bmatrix}^\top$, but if we augment our data with a function $\phi(x_t)$, where

$$\phi(x_t) := (1, y_{t-1}, y_{t-2}, y_{t-1}^2, y_{t-1}y_{t-2}, y_{t-2}^2),$$

then we can write it as $y_t = w^\top \phi(x_t)$. Thus, we can just solve the classic LS problem

$$\min_w \sum_{t=1}^{T} (y_t - w^\top \phi(y_t))^2,$$

where our model is nonlinear in the data, but is linear in the model coefficients. If it was nonlinear in the coefficients, this would be a neural network.

With this realization, we can create non-linear (e.g. quadratic) decision boundaries, such as

$$w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 = b = 0.$$

This can be written as $w^\top \phi(x) + b = 0$ where $\phi(x) := (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$.

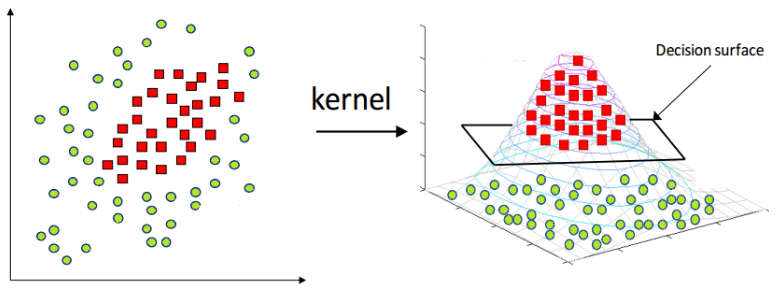


Figure 4: An example of moving to a higher dimension to create a linear decision boundary separating the data. Image taken from this medium article by Grace Zhang.

If we can do this, then in theory it seems we could always augment our input with enough dimensions to make any data separable. The question is how to do this in a computationally feasible manner.

First, let's look at the regularized learning setting for inspiration. The setup is to find

$$w^* = \min_w \|X^\top w - y\|_2^2 + \lambda \|w\|_2^2$$

where

- $X = [x_1, \dots, x_n]$ is a $p \times n$ matrix of data points,
- $y \in \mathbb{R}^n$ is the response vector (or labels),
- $w \in \mathbb{R}^p$ is the classifier or regression coefficients,
- $\lambda \geq 0$ is a regularization parameter.

With the optimal coefficients w^*, prediction simply requires computing $\hat{y} = (w^*)^\top x$ where $x \in \mathbb{R}^p$ is a new data point.

In general, the key result is that for any generic problem

$$\min_w \mathcal{L}(X^\top w, y) + \lambda \|w\|_2^2$$

where $\mathcal{L}$ is any loss function, the optimal w lies in the span of the data points $(x_1, \dots, x_n)$, i.e. $w = Xv$ for some $v \in \mathbb{R}^n$. In the above problem, we used $\mathcal{L}(z, y) = \|z - y\|_2^2$, i.e. the squared ℓ_2-norm.

Proof. Invoke the fundamental theorem of linear algebra, and write $w \in \mathbb{R}^p$ as the sum of two orthogonal vectors, one in the range of X and another orthogonal to it. Then

$$w = Xv + r,$$

where $v \in \mathbb{R}^n$ and $X^\top r = 0$ (since $r \in \mathcal{R}(X)^\perp = \mathcal{N}(X^\top)$). Applying this to our problem above and instead minimizing over v and r gives

$$\min_{v,r} \mathcal{L}(X^\top(Xv + r), y) + \lambda(\|Xv + r\|_2^2) = \min_{v,r} \mathcal{L}(X^\top Xv, y) + \lambda(\|Xv\|_2^2 + \|r\|_2^2),$$

where we used the Pythagorean theorem in the last step since Xv and r are orthogonal. Hence, setting $r = 0$ is optimal, and hence $w = Xv$ as we claimed. □

Using this result, we can show that our training problem rewrites as

$$\min_v \mathcal{L}(Kv, y) + \lambda v^\top Kv$$

which depends only on the $n \times n$ (PSD) “kernel” matrix $K := X^\top X$. Note that

$$K_{ij} = x_i^\top x_j, \quad 1 \leq i, j \leq n,$$

that is, K contains the scalar products between all data point pairs.

Now, our prediction/classification rule depends solely on the scalar products between any new point $\tilde{x}$ and our training data points $x_1, \dots, x_n$. Since $w = Xv$, we have

$$w^\top \tilde{x} = v^\top X^\top \tilde{x} = v^\top k, \qquad \text{where } k := X^\top \tilde{x} = (x_1^\top \tilde{x}, \dots, x_n^\top \tilde{x}).$$

Note the computational advantage here! Once K is formed (this takes $O(n^2 p)$), then the training problem only has n variables. When $p \gg n$, this leads to a dramatic reduction in problem size.

We haven't touched upon how this lets us create a richer class of models yet, so let's revisitt that. In the nonlilnear case, we can replace the feature vectors x_i by some “augmented” feature vectors $\phi(x_i)$, with ϕ a non-linear mapping.

For example, recall in classification with a quadratic decision boundary, we optimized

$$\min_w \sum_{i=1}^n (y_i - \phi(x_i)^\top w)^2 \text{ where } \phi(x) := (x_1, x_2, x_1^2, x_1 x_2, x_2^2).$$

This leads to the modified kernel matrix

$$K_{ij} = \phi(x_i)^\top \phi(x_j),\ 1 \le i, j \le n,$$

which, like the old kernel matrix, gives us scalar products between data points, except now the points are augmented.

Since we only need to know the scalar products, actually all we need is the *kernel function* associated with mapping ϕ, which is

$$k(x, z) = \phi(x)^\top \phi(z).$$

It gives us information about the metric in the feature space, e.g.

$$\|\phi(x) - \phi(z)\|_2^2 = k(x, x) - 2k(x, z) + k(z, z).$$

However, this function $k(\cdot, \cdot)$ can't be constructed arbitrarily; it has to satisfy our above condition for some ϕ.

Finally, suppose we have two different augmented points $\phi(x)$ and $\phi(z)$ for some given two vectors $x, z \in \mathbb{R}^2$. Then it turns out (and is easily verified upon inspection) that

$$k(x, z) = \phi(x)^\top \phi(z) = (1 + x^\top z)^2.$$

So instead of needing to expand $\phi(x)$ which is an expensive operation scaling exponentially, we can instead just work with this dot product formula which is more efficient (i.e. linear in the dimension of our vectors, in this case 2).

More generally, when $\phi(x)$ is the vector formed with all the products between the components of $x \in \mathbb{R}^n$ up to degree d, then for any two vectors $x, z \in \mathbb{R}^n$,

$$\phi(x)^\top \phi(z) = (1 + x^\top z)^d.$$

Our computational effort grows linearly in n since we just need to form this dot product.

Compare this with the "brute force" approach, where we first form $\phi(x), \phi(z)$ and then evaluate $\phi(x)^\top \phi(z)$. Here, the computational effort grows as n^d! In other words, we found a way to reduce an exponential effort to a linear one; pretty good I'd say.

This is what most people will refer to as the *kernel trick*, which allows us to pseudo-augment our data points into infinite-dimensional spaces. For example, we could also consider using the Gaussian kernel function (radial basis function)

$$k(x, z) = \exp\left(-\frac{\|x - z\|_2^2}{2\sigma^2}\right),$$

where $\sigma > 0$ is a scale parameter. In order to approximate this kernel function by some ϕ, we'd need to use a series for the exponential that goes on infinitely, i.e. ϕ is mapping into an infinite-dimensional space.

9 Tuesday, February 16th: Linear Algebra Review

Today is a linear algebra review.

Recall that projections are solving the optimization problem where given a point x, and some line going through x_0 in the direction of v ($\|v\|_2 = 1$), we want to find the point z on that line closestt to x. The line is defined by $\mathcal{L} = \{x_0 + tv : t \in \mathbb{R}\}$, and our optimal z will look like $z = x_0 + t^* v$. The problem to be solved is

$$\min_t \|x_0 + tv - x\|_2^2.$$

To solve this, we expand the objective as

$$(x_0 + tv - x)^\top (x_0 + tv - x) = t^2 v^\top v + 2tv^\top (x_0 - x) + (x_0 - x)^\top (x_0 - x),$$

and go about optimizing with gradients. Using $v^\top v$, the optimal value of t simplifies to $t^* = v^\top (x - x_0)$, i.e. the scalar product between v and $x - x_0$ as expepcted. This is one of the most important optimization problems to know, which is why we've gone over it so many times.

Hyperplanes are also very important to understand. Suppose we have the plane $\{x : a^\top x = b\}$, and we want to project 0 onto the hyperplane, i.e. find the closest point to 0. We can rewrite this by first finding an x_0 for which $a^\top x_0 = b$, $x_0 \neq a$. We know that $x_0 = \alpha a$ (by the fundamental theorem of linear algebra), so

$$\alpha a^\top a = b \implies \alpha = \frac{b}{\|a\|_2^2},\ x = \frac{ba}{a^\top a}.$$

The rest of today was review of previous days, so I didn't take notes.

10 Thursday, February 18th: Quiz

We just took a quiz today. That is all.

11 Tuesday, February 23rd: Linear Programs

We're starting the next portion of the course, which will be focused on "conic optimization." The main motivation comes from some problems which we've already seen can be solved by SVD/PCA, such as

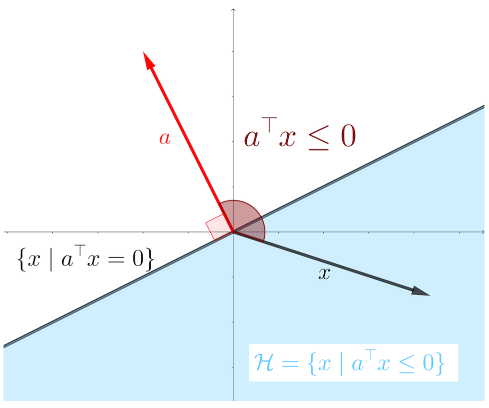
- $\min_X \|Ax - y\|_2^2 + \lambda\|x\|_2^2$ for $\lambda > 0$ (Ridge Regression)
- $\min_{L,R} \|X - LR^\top\|_F^2$ where L, R have k columns ($k < \min(n, m)$).

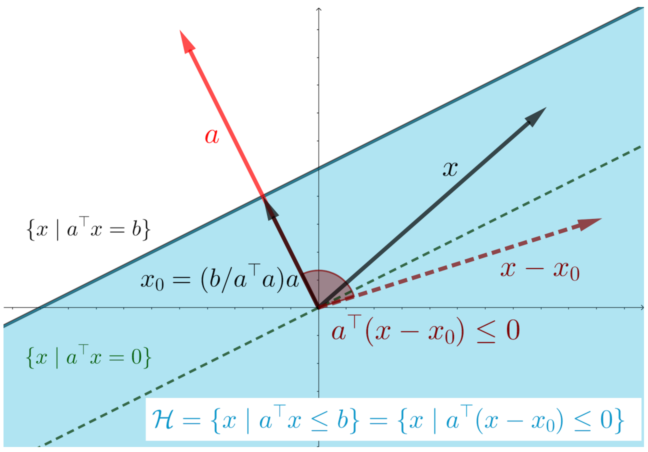
Our goal will be to discuss an entire class of optimization problems called conic optimization, which will lead us into discussing linear programming.

11.1 Polyhedrons

To talk about linear programs, we first need to discuss the geometry of the set of points we want to work with. For a given $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, the set $\mathcal{H}$ of points satisfying the linear inequality $a^\top x \leq b$ is a closed half-space. Its boundary is the hyperplane defined by the *equality* $a^\top x = b$.

The way to think about this is to start with the boundary $a^\top x = 0$, which is simply the plane normal to a passing through the origin. Then if $\mathcal{H} = \{x \mid a^\top x \leq 0\}$, then $\mathcal{H}$ is the set of points forming an obtuse angle with a (by properties of inner products).


a
$a^\top x \leq 0$
$\{x \mid a^\top x = 0\}$
x
$\mathcal{H} = \{x \mid a^\top x \leq 0\}$
Figure 5: The half-space $a^\top x \leq 0$.


a
x
$\{x \mid a^\top x = b\}$
$x_0 = (b/a^\top a)a$
$x - x_0$
$a^\top(x - x_0) \leq 0$
$\{x \mid a^\top x = 0\}$
$\mathcal{H} = \{x \mid a^\top x \leq b\} = \{x \mid a^\top(x - x_0) \leq 0\}$
Figure 6: The half-space $a^\top x \leq b$ (figures my own).

Now if we consider $a^\top x \leq b$, our boundary plane instead passes through the point $x_0 = (b/a^\top a)a$ (see the previous lecture), so the value b just tells us how far along a the boundary sits.

We can interpret this half-space as being the set of points whose inner product with a is less than b (obviously), which occurs below the boundary. However, a more illuminating intepretation is to rewrite the half-space as $a^\top x - b = a^\top(x - x_0) \leq 0$, which reduces to the case we just had. Hence the desired region is the one from before but shifted in the direction of a by x_0.

For given $a_1, \ldots, a_m \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$, the collection of m linear inequalities

$$a_i^\top x \leq b_i, \quad i = 1, \ldots, m$$

defines a region in $\mathbb{R}^m$ which is the intersection of m half-spaces and it is named a *polyhedron*. Depending on the actual inequalities, this region could be unbounded or bounded; in the latter case, we call it a *polytope*.

For convenience, we will often group several linear inequalities together using matrix notation. As such, if

$$A = \begin{bmatrix} a_1^\top \\ a_2^\top \\ \vdots \\ a_m^\top \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

then we can write the inequalities in equivalent matrix form $Ax \leq b$ (inequality taken component-wise).

Example 11.1 (Probability Simplex). The probability simplex is the polytope defined as

$$P = \{x \in \mathbb{R}^n : x \geq 0, \sum_{i=1}^n x_i = 1\}.$$

The name lends itself to an obvious interpretation. Every point in the polytope represents a potential point in a discrete probability distribution, i.e. the x_i's are nonnegative and they sum up to one.

The n vertices of this simplex in $\mathbb{R}^n$ are the standard orthonormal basis vectors for $\mathbb{R}^n$, i.e.

$$P = \text{co}\{e_1, \ldots, e_n\}$$

where $\text{co}\{x_1, \ldots, x_n\}$ contains all linear combinations $\sum \lambda_i x_i$ where $\sum \lambda_i = 1$ and $\lambda_i \geq 0$.

Example 11.2 (The ℓ_1-norm ball). The ℓ_1-norm ball is the set $\{x \in \mathbb{R}^n : \|x\|_1 \leq 1\}$, that is the set where $\sum_{i=1}^n |x_i| \leq 1$. This is indeed a polytope, since the previous inequality is equivalen to 2^n linear inequalities. We can see this by considering the sign variables $s_i \in \{-1, 1\}$, $i = 1, \ldots, n$. Then the condition can be written as

$$\|x\|_1 = \sum_{i=1}^n |x_i| = \max_{s_i \in \{-1,1\}} \sum_{i=1}^n s_i x_i \leq 1.$$

But this is equivalent to simply requiring all of the 2^n choices for $s_1, \ldots, s_n$ to be less than 1, i.e.

$$\sum_{i=1}^n s_i x_i \leq 1, \quad \text{for all } s_i \in \{-1, 1\}, i = 1, \ldots, m.$$

11.2 Linear Programs

Definition 19 (Linear Program). A linear optimization problem, or linear program (LP), is one of standard form where every function $f_0, f_1, \ldots, f_m$ is affine. Thus, the feasible set of an LP is a polyhedron.

Linear optimization problems admits several standard forms, such as

$$p^* = \min_x \ c^\top x + d \quad \text{s.t. } A_{\text{eq}} x = b_{\text{eq}};\ Ax \leq b,$$

where the inequalities are understood componentwise. We shall denote this form as the *inequality form* of the LP.

Of course, the constant d in the objective function has no influence on the minimizer. It is simply there to offset the value of the objective.

The set of points that satisfy the constraints of an LP (i.e. the feasible sete) is a polyhedron (or polytope when bounded): $\mathcal{X} = \{x \in \mathbb{R}^n : A_{\text{eq}}x = b_{\text{eq}}, Ax \leq b\}$.

Let $x_f \in \mathcal{X}$ be a feasible point. To such a point is associated the objective level $c^\top x_f$; from now one, we assume without loss of generality that $d = 0$.

A point $x_f \in \mathcal{X}$ is an optimal point, hence a solution of our LP, if and only if there is no other point $x \in \mathcal{X}$ with lower objective, that is:

$$x_f \in \mathcal{X} \text{ is optimal for LP} \iff c^\top x \geq c^\top x_f,\ \forall x \in \mathcal{X}.$$

Example 11.3 (A toy LP)**.** Take the following LP:

$$\min_{x\in\mathbb{R}^2} 3x_1 + 1.5x_2 \text{ subject to: } -1 \leq x_1 \leq 2,\ 0 \leq x_2 \leq 3.$$

We can put it in standard inequality form with the constraints

$$\min_x : \min_{x\in\mathbb{R}^2} 3x_1 + 1.5x_2 \text{ subject to: } -x_1 \leq 1, x_1 \leq 2, -x_2 \leq 0, x_2 \leq 3.$$

Using matrix notation, we can write this as $\min_x c^\top x$ subject to $Ax \leq b$, with

$$c^\top = \begin{bmatrix} 3 & 1.5 \end{bmatrix}, \quad A = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 3 \end{bmatrix}.$$

The level curves (curves of constant value) of the objective function are straight lines orthogonal to the objective vector, $c^\top = \begin{bmatrix} 3 & 1.5 \end{bmatrix}$. The problem amounts to finding the smallest value of p such that $p = c^\top x$ for some feasible x. The optimal point is $x^* = \begin{bmatrix} -1 & 0 \end{bmatrix}^\top$, which gives an optimal objective value of $p^* = -3$.

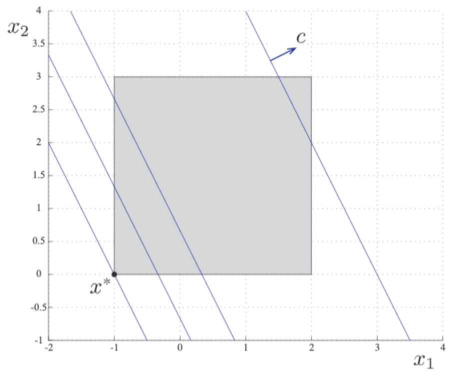


Figure 7: Visualization of the above LP, shamelessly borrowed from El Ghaoui's slides.

Definition 20 (Polyhedral Functions). We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is *polyhedral* if its epigraph is a polyhedron, that is if $\text{epi } f = \{(x,t) \in \mathbb{R}^{n+1} : f(x) \le t\}$ can be represented as

$$\text{epi } f = \left\{(x,t) \in \mathbb{R}^{n+1} : C \begin{bmatrix} x \\ t \end{bmatrix} \le d \right\},$$

for some matrix $C \in \mathbb{R}^{m,n+1}$ and vector $d \in \mathbb{R}^m$.

Polyhedral functions include in particular functions that can be expressed as a maximum of a finite number of affine functions

$$f(x) = \max_{i=1,\ldots,m} a_i^\top x + b_i,$$

where $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ for $i = 1, \ldots, m$. Indeed, the epigraph of such functions f

$$\text{epi } f = \{(x,t) \in \mathbb{R}^{n+1} : \max_{i=1,\ldots,m} a_i^\top x + b_i \le t\}$$

can indeed be expressed as the polyhedron

$$\text{epi } f = \{(x,t) \in \mathbb{R}^{n+1} : a_i^\top x + b_i \le t,\ i = 1, \ldots, m\}.$$

Here are some more examples of polyhedral functions:

- The ℓ_∞-norm function $f(x) = \|x\|_\infty$ is polyhedral since it can be written as the maximum of $2n$ affine functions
$$f(x) = \max_{i=1\ldots,n} \max(x_i, -x_i).$$
- The ℓ_1-norm function $f(x) = \|x\|_1$ is polyhedral since it can be written as the sum of maxima of affine functions:
$$f(x) = \sum_{i=1}^n \max(x_i, -x_i).$$
- Functions that are sums of maximums of affine functions (as discussed above) are polyhedral:
$$f(x) = \sum_{j=1}^q \max_{i=1,\ldots,m} a_{ij}^\top x + b_{ij}.$$
- Take f as in the previous part again. The condition $(x,t) \in \text{epi } f$ is equivalent to $\exists u \in \mathbb{R}^q$ such that
$$\sum_{j=1}^q u_j \le t, \quad a_{ij}^\top x + b_{ij} \le u_j,\ i = 1, \ldots, m;\ j = 1, \ldots, q. \tag{1}$$
The epigraph epi f is the projection (on the space of (x,t)-variables) of a polyhedron, so the epigraph itself is a polyhedron.

Polyhedral functions are convenient since minimizing them is equivalent to solving an LP. If f is polyhedral, then

$$\min_x f(x) \quad \text{s.t.: } Ax \le b,$$

can be written as

$$\min_{x,t} t \quad \text{s.t.: } Ax \le b, (x,t) \in \text{epi } f.$$

Since epi f is a polyhedron, we can express it as in Equation 1, hence the problem above is an LP. However, note that the explicit representation of an LP in standard form may require the introduction of additional slack variables, which are needed for representation of the epigraph.

11.3 Examples

Example 11.4 (ℓ_∞ regression problems). Take the following ℓ_∞ optimization problem:

$$\min_x \|Ax - b\|_\infty, \quad A \in \mathbb{R}^{m,n}, b \in \mathbb{R}^m.$$

The problem can first be rewritten in epigraphic form, adding a slack scalar variable t

$$\min_{x,t} \ t \quad \text{s.t.: } \|Ax - b\|_\infty \leq t.$$

Then simply using the definition of the ℓ_∞ norm, observe that

$$\|Ax - b\|_\infty \leq t \iff \max_{i=1,\dots,m} |a_i^\top x - b_i| \leq t \iff |a_i^\top x - b_i| \leq t,\ i = 1, \dots, m.$$

Hence, the problem is equivalent to the following LP in variables $x \in \mathbb{R}^n$ and $t \in \mathbb{R}$:

$$\begin{aligned} \min_{x,t} \quad & t \\ \text{s.t.} \quad & a_i^\top x - b_i \leq t, \quad i = 1, \dots, m \\ & a_i^\top x - b_i \geq -t, \quad i = 1, \dots, m. \end{aligned}$$

Example 11.5 (ℓ_1 regression problems). Here's another optimization problem using an ℓ_1 norm instead:

$$\min_x \|Ax - b\|_1, \quad A \in \mathbb{R}^{m,n}, b \in \mathbb{R}^m.$$

To convert this, we utilize a vector $u \in \mathbb{R}^m$ of slack variables to get

$$\min_{x,u} \sum_{i=1}^m u_i \quad \text{s.t. } |a_i^\top x - b_i| \leq u_i,\ i = 1, \dots, m.$$

This in turn is easily turned into a standard LP as follows:

$$\begin{aligned} \min_{x,u} \quad & \mathbb{1}^\top u \\ \text{s.t.} \quad & a_i^\top x - b_i \leq u_i, \quad i = 1, \dots, m \\ & a_i^\top x - b_i \geq -u_i, \quad i = 1, \dots, m. \end{aligned}$$

12 Thursday, February 25th: Convex Quadratic Programs

12.1 Convex Quadratic Functions and the QP Model

A quadratic function in a vector of variables $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ can be written generically as

$$f_0(x) = \frac{1}{2}x^\top Hx + c^\top x + d$$

where $d \in \mathbb{R}, c \in \mathbb{R}^n$, and $H \in \mathbb{R}^{n,n}$ is a symmetric matrix.

This last condition can always be achieved, since if I had some arbitrary $n \times n$ matrix A in my form, $x^\top Ax = x^\top A^\top x$, so

$$x^\top Ax = x^\top \left(\frac{A + A^\top}{2}\right) x,$$

where the inside term is symmetric. So any quadratic form $x^\top Hx$ can always be written as one where H is symmetric; we refer to H as the *Hessian* of the quadratic function.

A quadratic function is said to be *convex* if its Hessian is positive semi-definite, that is: every eigenvalue of the (symmetric) matrix H is non-negative. Convex quadratic functions typically look bowl-shaped, while non-convex ones can look saddle-like, etc.

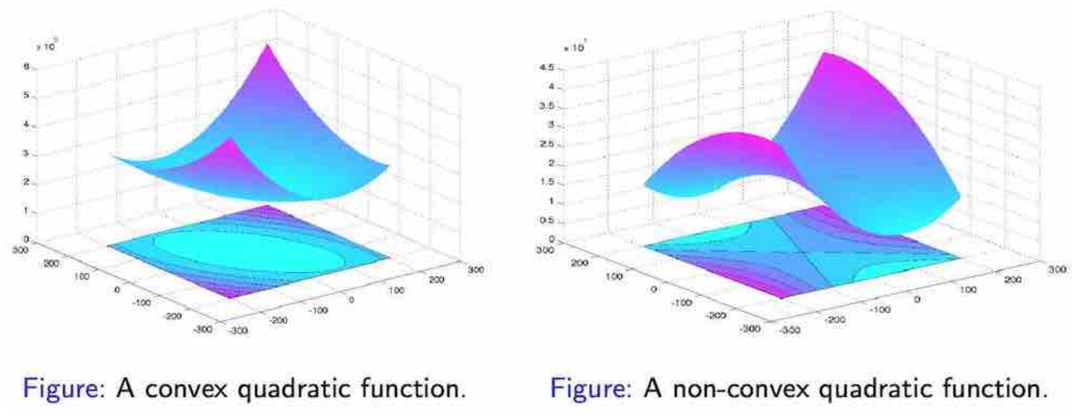


Figure 8: Taken from El Ghaoui's Slides.

Definition 21 (Convex Quadratic Programs)**.** Let $A \in \mathbb{R}^{m,n}, b \in \mathbb{R}^m$. The model

$$\begin{aligned} \min \quad & f_0(x) \\ \text{subject to:} \quad & Ax \le b, Cx = d, \end{aligned}$$

is called a *quadratic program* (QP for short) if f_0 is a convex quadratic function, that is:

$$f_0(x) = \frac{1}{2}x^\top Hx + c^\top x,$$

where $H = H^\top$ is positive semi-definite (PSD).

Note that f_0 is still quadratic if H is not PSD, but the problem above would not be referred as a QP. An artifact of historical naming conventions, supposedly.

12.1.1 Unconstrained QP Problems

In the unconstrained setting, we can still employ our usual linear algebra techniques to minimize quadratic functions. Consider the convex quadratic function

$$p^* = \min_{x\in\mathbb{R}^n} f(x) := \frac{1}{2}x^\top Hx + c^\top x + d.$$

If $H \succ 0$ (PD), we can write

$$f(x) = \frac{1}{2}(x-x_0)^\top H(x-x_0) + \alpha \geq \alpha,$$

where $x_0 = -H^{-1}c, \alpha = d - \frac{1}{2}x_0^\top Hx_0$. I think of this as completing the square, which you can see by expanding the above as

$$\begin{aligned} f(x) &= \frac{1}{2}(x^\top Hx - 2x_0^\top Hx + x_0^\top Hx_0) + \alpha \\ &= \frac{1}{2}x^\top Hx + (c^\top H^{-1})Hx + \frac{1}{2}x_0^\top Hx_0 + (d - \frac{1}{2}x_0^\top Hx_0) && \text{(since } H = H^\top\text{)} \\ &= \frac{1}{2}x^\top Hx + c^\top x + d \end{aligned}$$

Then you know the quadratic form will always be greater than 0 for nonzero $x - x_0$ by virtue of H being positive-definite, so the best minimizer is to pick $x^* = x_0$.

If $H \succeq 0$ is only positive semi-definite, and $c \in \mathcal{R}(H)$, then any x_0 such that $Hx_0 + c = 0$ is optimal by the same argument as before. Hencec, the set of solutions is given by

$$\{-H^\dagger c + \zeta,\ \zeta \in \mathcal{N}(H)\}.$$

Finally, if $H \succeq 0$ is PSD but $c \notin \mathcal{R}(H)$, then the function is unbounded below. By the fundamental theorem, we can write c as $c = -Hx_0 + r$, where $Hr = 0$ and for some $x_0, r \neq 0 \in \mathbb{R}^n$. Set $x(t) = x_0 - tr$ where $t \in \mathbb{R}$, and note that

$$\begin{aligned} f(x(t)) &= \frac{1}{2}(x_0 - tr)^\top H(x_0 - tr) + (-Hx_0 + r)^\top(x_0 - tr) + d \\ &= \frac{1}{2}x_0^\top Hx_0 - x_0^\top Hx_0 + r^\top x_0 - tr^\top r && \text{(since } Hr = 0\text{)} \\ &= \text{constant} - t(r^\top r), \end{aligned}$$

which tends to $-\infty$ as $t \to \infty$, as we stated.

Example 12.1 (Least-Squares)**.** Least-squares is actually a special case of an unconstrained QP model. The problem ammounts to minimizing $f_0(x) = \|Ax - y\|_2^2$, where

$$f_0(x) = (Ax-y)^\top(Ax-y) = x^\top A^\top Ax - 2y^\top Ax + y^\top y,$$

which is a quadratic function in standard form, with

$$H = 2(A^\top A), \quad c = -2A^\top y, \quad d = y^\top y.$$

Note that f_0 is convex since $A^\top A \succeq 0$. Since this is unconstrained, and $c = -HA^\dagger y \in \mathcal{R}(H)$, we have (from above) that the set of solutions is of the form

$$\mathcal{X}^{\text{opt}} = \{x^* = A^\dagger y + Nz : z \in \mathbb{R}^r\},$$

wherer r is the rank of H (and column rank of A), and N spans the nullspace of A.

If A is full column rank, then $A^\top A \succ 0$, so the solution is unique and given by the well-known formula

$$x^* = -H^{-1}c = (A^\top A)^{-1}A^\top y.$$

12.1.2 Linear Equality Constrained QP Problems

If we have only linear equality constraints, then we can convert it into unconstrained form by eliminating the equality constraints (similar to how we did for linear systems). Suppose we have the problem

$$\begin{aligned} \min \quad & f_0(x) := \frac{1}{2}x^\top Hx + c^\top x + d \\ \text{subject to:} \quad & Ax \leq b, \end{aligned}$$

Parameterize all of the solutions x to $Ax = b$ as $x = \bar{x} + Nz$, where $\bar{x}$ is a specific solution to $Ax = b$, N's columns form a basis for $\mathcal{N}(A)$, and z is a vector of free variables. Then we can substitute this in for x in f_0 to get the unconstrained problem

$$\min_z \varphi_0(z) = \frac{1}{2}z^\top \bar{H}z + \bar{c}^\top z + \bar{d},$$

where

$$\bar{H} = N^\top HN,\ \bar{c} = N^\top(c + H\bar{x}),\ \bar{d} = d + c^\top \bar{x} + \frac{1}{2}\bar{x}^T H\bar{x}.$$

12.2 Minimal cardinality solutions and ℓ_1-norm problems

Many engineering problems will concern themselves with finding solutions which are *sparse*, i.e. contain few non-zero entries (low-cardinality solutions). This is a very natural requirement, since minimal representations tend to be the best explanations. However, finding minimum cardinality solutions (i.e. solutions with small l_0 norm) is hard in general, from a computational point of view. Thus, often several *heuristics* are used in order to devise tractable numerical schemes which provide low cardinality solutions, such as using ℓ_1 norm in place of the ℓ_0 norm.

For example, using the Cauchy-Schwartz inequality (Theorem 4), we can obtain a close relation between $\text{card}\,(x)$ and its ℓ_1 norm. Let $\text{nz}\,(x)$ be the vector whose ith entry is one whenever $x_i \neq 0$, and is zero otherwise, and $|x|$ is the vector of absolute values of all of the entries of x. Note that $\|\text{nz}\,(x)\|_2 = \text{card}\,(x)$. Then

$$\|x\|_1 = \text{nz}\,(x)^\top |x| \leq \|\text{nz}\,(x)\|_2 \cdot \|x\|_2 = \|x\|_2\sqrt{\text{card}\,(x)},$$

hence

$$\text{card}\,(x) \leq k \implies \|x\|_1^2 \leq k\|x\|_2^2.$$

Similarly, using Holder's (Theorem 5) for $p = 1, q = \infty$,

$$\|x\|_1 = \text{nz}\,(x)^\top |x| \leq \|\text{nz}\,(x)\|_1 \|x\|_\infty = \|x\|_\infty \,\text{card}\,(x),$$

hence

$$\text{card}\,(x) \leq k \implies \|x\|_1 \leq k\|x\|_\infty.$$

The following example illustrates exactly how we can replace a cardinality constraint with one relying on the ℓ_1 norm instead.

Example 12.2 (Piece-wise constant fitting)**.** Suppose one observes a noisy time-series which is almost piece-wise constant. The goal would be to find out what the constant levels are, which could correspond to states or levels of a system in applications.

Let $x \in \mathbb{R}^n$ denote the signal vector (unknown), and $y \in \mathbb{R}^n$ denote the vector of noisy signal observations (i.e. the true signal x plus noise). Given y, we want an estimate $\hat{x}$ of the original signal x which has as few changes in consecutive time steps as possible. This requirement can be modeled by minimizing the cardinality of the different vector $D\hat{x}$, where $D \in \mathbb{R}^{n-1,n}$ is the difference matrix

$$D = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & & & \ddots & \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix},$$

so that $D\hat{x} = [\hat{x}_2 - \hat{x}_1, \hat{x}_3 - \hat{x}_2, \dots, \hat{x}_n - \hat{x}_{n-1}]^\top$.

Thus, we have the problem

$$p^* = \min_{\hat{x}} f(\hat{x}) := \|y - \hat{x}\|_2^2 \text{ s.t. } \operatorname{card}(D\hat{x}) \le k,$$

where k is an estimate on the number of jumps in the signal, and our objective function is measuring the error in our estimate.

We can get a *lower bound* on the problem by noticing that we have the obvious solution $\hat{x} = 0$ which satisfies our cardinality constraint, and thus our optimal value must be lower than this. Hence, $\|\hat{x} - y\|_2^2 \le f(0) = \|y\|_2^2$ and so by triangle inequality,

$$\|\hat{x}\|_2 = \|(\hat{x} - y) + y\|_2 \le \|\hat{x} - y\|_2 + \|y\|_2 \le 2\|y\|_2.$$

Thus, using the cardinality results we obtained above, the constraint $\operatorname{card}(D\hat{x}) \le k$ implies

$$\|D\hat{x}\|_1 \le \sqrt{k}\|D\hat{x}\|_2 \le \sqrt{k}\|D\|_F\|x\|_2 \le 2\sqrt{k}\|D\|_F\|y\|_2 =: \alpha.$$

The relaxed problem is a QP of the form

$$\min_{\hat{x}} \|y - \hat{x}\|_2^2 \quad \text{s.t.: } \|D\hat{x}\|_1 \le \alpha.$$

Alternatively, we could have instead chosen a problem with a weighted objective

$$\min_{\hat{x}} \|y - \hat{x}\|_2^2 + \gamma\|D\hat{x}\|_1,$$

for some suitable parameter $\gamma \ge 0$.

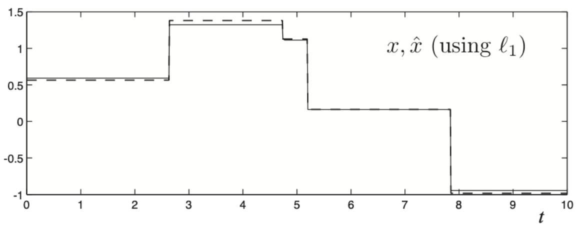


Figure 9: Example of the piece-wise constant fitting problem and result achieved with ℓ_1 objective. Taken from El Ghaoui's slides.

This general theme of using ℓ_1 norms in our optimization problems leads to another important problem. Solving least-squares problems that, instead of having an ℓ_2 regularization term as we've seen in the past, have an ℓ_1 norm regularizer.

Example 12.3 (LASSO). Consider the following problem, known as the *basis pursuit denoising problem* (BDPN), or as the *least absolute shrinkage and selection operator* (LASSO) problem:

$$\min_{x\in\mathbb{R}^n} \|Ax-y\|_2^2 + \lambda\|x\|_1, \quad \lambda \geq 0,$$

where $\|x\|_1 = |x_1| + \cdots + |x_n|$. This problem has received much attention in recent years due to its relevance in the field of *compressed sensing.*

The basic idea is that the ℓ_1 norm is used as a proxy for the cardinality of x. It formalizes a tradeoff between the accuracy with which Ax approximates y, and the *complexity* of the solution, i.e. the number of nonzero entries in x (which can be interpreted as the number of relevant features, for example). The larger λ is, the more biased our problem is towards finding low-complexity solutions.

We can rewrite the above problem in standard QP form by introducing slack variables $u \in \mathbb{R}^n$:

$$\begin{aligned} \min_{x,u\in\mathbb{R}^n} \quad & \|Ax-y\|_2^2 + \lambda\sum_{i=1}^n u_i \\ \text{s.t.} : \quad & |x_i| \leq u_i,\ i = 1,\dots,n. \end{aligned}$$

There are cool applications of LASSO to image compression in different bases, for example the wavelet basis, which is what modern JPEG compression schemes utilize. We also talked about Markowitz's mean-variance portfolio optimization model, which is central to modern portfolio theory.

13 Tuesday, March 2nd: Second-Order Cone Models

13.1 Second-order cone programs

Today, we'll be talking about *second-order cone programming* (SOCP), which is a generalization of linear and quadratic programming which allows for affine combinations of variables to be constrained in a special convex set, called a *second-order cone.* LPs and convex quadratic optimization fall under special cases of SOCP models, and they're quite useful in geometry problems, approximation problems, and probabilistic approaches to linear optimization.

Definition 22 (Second-Order Cone)**.** The *second-order cone* (SOC) in $\mathbb{R}^3$ is the set of vectors (x_1, x_2, t) such that $\sqrt{x_1^2 + x_2^2} \leq t$ (i.e. the ice cream cone). Horizontal sections of this set at level $\alpha \geq 0$ are discs of radius α.

In arbitrary dimensions, an $(n+1)$-dimensional SOC is the following set:

$$\mathcal{K}_n = \{(x,t), x \in \mathbb{R}^n, t \in \mathbb{R} : \|x\|_2 \leq t\}.$$

All of these cones point upwards however, so to capture the general notion of a second-order cone, we need to discuss what rotated second-order cones are.

Definition 23 (Rotated Second-Order Cone)**.** The rotated second-order cone in $\mathbb{R}^{n+2}$ is defined by

$$\mathcal{K}_n^r = \{(x,y,z), x \in \mathbb{R}^n, y \in \mathbb{R}^n, z \in \mathbb{R}^n : x^\top x \leq 2yz, y \geq 0, z \geq 0\}.$$

We can express this as a linear transformation (i.e. rotation) of our usual second order cones in $\mathbb{R}^{n+2}$, since

$$\|x\|_2^2 \leq 2yz, y \geq 0, z \geq 0 \iff \left\| \begin{bmatrix} x \\ \frac{1}{\sqrt{2}}(y-z) \end{bmatrix} \right\|_2 \leq \frac{1}{\sqrt{2}}(y+z),$$

where the right hand side is a form in our original constraint from above.

It's exactly as it sounds like: they're just second-order cones which are rotated, which is the purpose of the y and z parameters. Constraints of the form $\|x\|_2^2 \leq 2yz$ are usually referred to as *hyperbolic constraints.* To see the equivalence between the two conditions, simply expand the ℓ_2-norm condition as

$$\begin{bmatrix} x & \frac{1}{\sqrt{2}}(y-z) \end{bmatrix} \begin{bmatrix} x \\ \frac{1}{\sqrt{2}}(y-z) \end{bmatrix} = x^\top x + \frac{1}{2}(y-z)^2 \leq \frac{1}{2}(y+z)^2 \iff x^\top x \leq 2yz$$

as expected.

The standard format of a second-order cone constraint on a variable $x \in \mathbb{R}^n$ expresses the condition that $(y,t) \in \mathcal{K}_m$, with $y \in \mathbb{R}^m, t \in \mathbb{R}$ where y, t are some affine functions of x. We can express these affine functions as $y = Ax + b$ and $t = c^\top x + d$, so the condition $(y,t) \in \mathcal{K}_m$ becomes

$$\|Ax + b\|_2 \leq c^\top x + d, \tag{2}$$

where $A \in \mathbb{R}^{m,n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$, and $d \in \mathbb{R}$.

For example, our usual quadratic constraint

$$x^\top Q x + c^\top x \leq t, \quad Q \succeq 0$$

can be expressed in conic form using the square root $Q^{1/2}$ (since Q is PSD) as

$$\left\| \begin{bmatrix} \sqrt{2}Q^{1/2}x \\ t - c^\top x - 1/2 \end{bmatrix} \right\|_2 \leq t - c^\top x + 1/2. \tag{3}$$

This can be achieved from our above transformation for rotated second-order cones, by writing our constraint as $\|\sqrt{2}Q^{1/2}x\|_2^2 \leq 2(t - c^\top x)$ and setting

$$\tilde{x} := \sqrt{2}Q^{1/2}x,\ \tilde{y} := \sqrt{2}(t - c^\top x),\ \tilde{z} := \tfrac{1}{\sqrt{2}}.$$

Then

$$\frac{1}{\sqrt{2}}(\tilde{y} - \tilde{z}) = t - c^\top x - 1/2, \quad \frac{1}{\sqrt{2}}(\tilde{y} + \tilde{z}) = t - c^\top x + 1/2$$

as we have seen.

Now we can define what an SOCP is.

Definition 24 (Second-Order Cone Programs)**.** A *second-order cone program* is a convex optimization problem having **linear** objective and **SOC** constraints. When the SOC constraints have the standard form in (2), we have a SOCP in *standard inequality form*

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.:} \quad & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i, \quad i = 1, \dots, m, \end{aligned}$$

where $A_i \in \mathbb{R}^{m_i,n}$ are given matrices, $b_i \in \mathbb{R}^{m_i}$, $c_i \in \mathbb{R}^n$ are vectors, and d_i are given scalars.

SOCPs are quite a large class of convex optimization problems; in fact, they generalize all of the convex optimization problems we've seen thus far, including LPs, convex QPs, and convex QCQPs.

13.2 Examples

To demonstrate just how vast SOCPs really are, we'll show that basically every problem we've come across so far (and more!) can be written as an SOCP.

Example 13.1 (LPs are SOCPs)**.** Any standard linear program in standard inequality form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.:} \quad & a_i^\top x \leq b_i, \quad i = 1, \dots, m, \end{aligned}$$

can be easily written in SOCP form by setting the ℓ_2-norm condition to zero and using just the affine part, as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.:} \quad & \|C_i x + d_i\|_2 \leq b_i - a_i^\top x, \quad i = 1, \dots, m, \end{aligned}$$

where $C_i = 0, d_i = 0$ for $i = 1, \dots, m$.

Example 13.2 (QPs are SOCPs). Any standard quadratic program in standard form

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad & x^\top Qx + c^\top x \\ \text{s.t.:} \quad & a_i^\top x \le b_i, \quad i = 1,\dots,m,\end{aligned}$$

where $Q = Q^\top \succeq 0$ can be written in SOCP form as

$$\begin{aligned}\min_{x,y} \quad & c^\top x + y \\ \text{s.t.:} \quad & \left\| \begin{bmatrix} 2Q^{1/2}x \\ y-1 \end{bmatrix} \right\|_2 \le y+1, \\ & a_i^\top x \le b_i, \quad i = 1,\dots,m.\end{aligned}$$

The trick is to convert the quadratic form constraint into an inequality. First, we set $y = x^\top Qx$. Then use our earlier form for quadratic constraints (3) to write $x^\top Qx \le y$ in the form presented here by setting $c^\top x = 0$ and adjusting constants.

Directly expanding the constraint here gives us

$$4x^\top Qx + (y-1)^2 \le (y+1)^2 \iff x^\top Qx \le y,$$

which corresponds to our quadratic form condition.

Example 13.3 (QCQPs are SOCPs). Any convex quadratic-constrained quadratic programs of the form

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad & x^\top Q_0 x + a_0^\top x \\ \text{s.t.:} \quad & x^\top Q_i x + a_i^\top x \le b_i, \quad i = 1,\dots,m,\end{aligned}$$

where $Q_i = Q_i^\top \succeq 0$ can be written in SOCP form as

$$\begin{aligned}\min_{x,t} \quad & a_0^\top x + t \\ \text{s.t.:} \quad & \left\| \begin{bmatrix} 2Q_0^{1/2}x \\ t-1 \end{bmatrix} \right\|_2 \le t+1, \\ & \left\| \begin{bmatrix} 2Q_i^{1/2}x \\ b_i - a_i^\top x - 1 \end{bmatrix} \right\|_2 \le b_i - a_i^\top x + 1, \; i = 1,\dots,m\end{aligned}$$

using the same idea as before.

For a new example, consider the problem of sums of norms, i.e.

$$\min_x \sum_{i=1}^p \|A_i x - b_i\|_2,$$

where $A_i \in \mathbb{R}^{m,n}, b_i \in \mathbb{R}^m$ are given data. This can be easily written as an SOCP by introducing auxiliary scalar variables $y_1,\dots,y_p$ to make our objective linear, so

$$\begin{aligned}\min_{x,y} \quad & \sum_{i=1}^p y_i \\ \text{s.t.:} \quad & \|A_i x - b_i\|_2 \le y_i, \quad i = 1,\dots,p\end{aligned}$$

Similarly, the maxima of norms problem

$$\min_x \max_{i=1,\ldots,p} \|A_i x - b_i\|_2,$$

can be cast in SOCP form as

$$\begin{aligned} \min_{x,y} \quad & y \\ \text{s.t.:} \quad & \|A_i x - b_i\|_2 \leq y, \quad i = 1, \ldots, p \end{aligned}$$

It's important to note that while the constraint $\|z\|_2 \leq y$ is convex and can be optimized over, $\|z\|_2^2 \leq y^2, y \geq 0$ is *not convex*, so we can't optimize over it.

Example 13.4 (Square-root LASSO)**.** Consider the optimization problem

$$p^* := \min_w \|X^\top w - y\|_2 + \lambda \|w\|_1$$

where X is our data matrix, with $\lambda > 0$ a sparsity-inducing parameter. This is square-root LASSO, as we're missing the usual squared term on the ℓ_2-norm for our loss.

This turns out to be an SOCP as well, of the form

$$\min_{w,v,t} t + \lambda \sum v_i \quad \text{s.t.: } t \geq \|X^\top w - y\|_2,\ v_i \geq w_i \geq -v_i\ \forall i.$$

Example 13.5 (Examplar Selection)**.** In examplar selection, we are given a data matrix $X = [x_1, \ldots, x_m] \in \mathbb{R}^{m,n}$, where columns x_i are the data points. We wish to find a small subset of data points which are approximately representative of the whole dataset. Precisely, we seek to find a sparse subset of data points $\{x_j\}_{j \in \mathcal{J}}$, with $\mathcal{J} \subseteq \{1, \ldots, m\}$ some indexing set, such that

$$\forall i \in \{1, \ldots, n\},\ x_i \approx \sum_{j \in \mathcal{J}} x_j w_{ij}.$$

In other words, every data point can be accurately represented as a linear combination of the same few data points. We can write the above condition as $X \approx XW^\top$, where W has a lot of its columns entirely zero.

We can model this as an SOCP:

$$\min_{W=[w_1,\ldots,w_m]} \|X - XW^\top\|_F : \quad \sum_{j=1}^m \|w_j\|_2 \leq \kappa,$$

where a small value of parameter $\kappa > 0$ encourages many columns in W to be entirely zero.

Note that the constraint here is essentially an ℓ_1-norm on the w_i column norms, so we're doing a LASSO-like feature selection to find the sparse set $\mathcal{J}$. Also, we can easily write the Frobenius norm as a sum of ℓ_2 constraints on the column norms of the inside matrix. So this is indeed an SOCP.

Finally, SOCP can also model problems involving powers of variables. For example, consider the following variant of least-squares:

$$p^* := \min_w \|X^\top w - y\|_2 + \lambda \sum_{i=1}^n |w_i|^\alpha,$$

where $\alpha > 1$ is the ratio of two integers. Notice that $\alpha = 1$ gives us LASSO, while $\alpha = 2$ gives us Ridge regression. A typical choice is $\alpha \in (1, 2)$ to achieve a good trade-off between encouraging sparsity and robustness to noise.

For example, take $\alpha = 3/2$. The condition $t \geq u^{3/2}$ for $u \geq 0$ is equivalent to the existence of $v \geq 0$ such that

$$vt \geq u^2, u \geq v^2.$$

Hence we can rewrite this as an SOCP with rotated cone constraints

$$p^* = \min_{w,v,u,t} \|X^\top w - y\|_2 + \lambda \sum_{i=1}^n t_i : \quad t \geq 0, u \geq |w|,$$

$$v_i t_i \geq u_i^2, u_i \geq v_i^2, i = 1, \ldots, n.$$

We ended on an interesting discussion of PCA vs LASSO, i.e. how are they different in dimensionality reduction and sparsity. The summary is that PCA gives you the directions of most variance (in an unsupervised manner), but provides no sparsity nor interpretability of the resulting data. LASSO provides intepretability and sparsity, but requires labels in order to do so (and hence is supervised).

There is something called sparse PCA which can give you directions with sparser projections though, which could be considered somewhere in between the two.

14 Thursday, March 4th: Robust Optimization Models

14.1 Robust optimization framework

Consider a "nominal" optimization problem (meaning the original, idealistic no-noise model)

$$\min_x f_0(x) \ : \ f_i(x) \leq 0, i = 1, \dots, m.$$

In practice, our problem data is *uncertain*, i.e.

- *Estimation* errors affect problem parameters
- *Implementation* errors affect the decision taken.

Uncertainties often lead to highly unstable solutions or much degraded realized performance. These problems are even compounded in problems with multiple decision periods!

We call this the *curse of uncertainty*, and the goal of today's lecture is to formalize a framework for fixing nominal problems to be more robust.

Consider the nominal optimization problem from above

$$\min_x f_0(x) \ : \ f_i(x) \leq 0, i = 1, \dots, m.$$

It is possible now that each of these values in our functions is uncertain and has unknown parts. For example, if $f_0(x)$ is modeled as $c^\top x$ where c is known, it's possible that there is an unknown u (noise or uncertainty) for which our function is actually $f_0(x, u) = (c + u)^\top x$, for $\|u\|_\infty \leq .01$.

Our functions f_i then depend on a second variable u, the "uncertainty", which is constrained to lie in a given set $\mathcal{U}$. This creates the *robust counterpart* to our optimization problem:

$$\min_x \max_{u \in \mathcal{U}} f_0(x, u) \ : \ \forall u \in \mathcal{U}, \ f_i(x, u) \leq 0, i = 1, \dots, m.$$

You could think of this as playing a game, where the uncertainty is trying to make your life as hard as possible, and you need to minimize your objective in light of that.

This inherits convexity from the nominal, and is very tractable in some practically relevant cases. In general though, the robust counterpart has high complexity, but there are systematic ways to get relaxations.

You could also consider other cost functions instead of a max over all u's (which is a worst-case estimate); other options are to integrate over all values of f_0, or averaging.

Let's look at robustifying linear programs in particular. Take the nominal problem

$$\text{(nominal)} \ \min_x c^\top x : a_i^\top \leq b_i, i = 1, \dots, m. \tag{4}$$

Assume for now that a_i is only know to belong to a given set $\mathcal{U}_i \subseteq \mathbb{R}^n$ (we will soon generalize to errors everywhere). For example, we could have $a_i \in \mathcal{U}_i = \{\hat{a}_i + u_i : \|u_i\|_\infty \leq \epsilon\}$. Then our robust counterpart to (4) would be

$$\text{(robust)} \ \min_x c^\top x : \forall a_i \in \mathcal{U}_i, a_i^\top x \leq b_i, i = 1, \dots, m.$$

If we assume instead that the cost vector c in (4) is only known to belong to a given set $\mathcal{U} \subseteq \mathbb{R}^n$, then our robust counterpart would be

$$\text{(robust)} \ \min_x \max_{c \in \mathcal{U}} c^\top x : a_i^\top x \leq b_i, i = 1, \dots, m.$$

In general we can extend this robust approach to uncertainties affecting the cost vectors, coefficient matrix, and the right-hand side vector b. The solution may be hard, but it becomes easy for special uncertainty sets, which we'll examine now.

14.2 Special Uncertainty Sets for Robust LPs

Let's examine the robust counterpart to a single inequality constraint

$$\forall a \in \mathcal{U},\ a^\top x \le b$$

where $\mathcal{U}$ takes the following forms:

- scenario uncertainty: $\mathcal{U}$ is a finite set "scenarios";
- $\mathcal{U}$ is a box.
- $\mathcal{U}$ is a sphere, or more generally an ellipsoid;

The above constraint can be written as

$$b \ge \max_{a \in \mathcal{U}} a^\top x.$$

- **Scenario uncertainty**: This model assumes that the coefficient vector a is only known to lie in a finite sets in $\mathbb{R}^n$, say

  $$\mathcal{U} = \{a^{(1)}, \dots, a^{(K)}\}$$

  with $a^{(k)}$ a "scenario", $k = 1, \dots, K$. We then have

  $$\max_{a \in \mathcal{U}} a^\top x = \max_{1 \le k \le K} (a^{(k)})^\top x.$$

  For example, when $\mathcal{U}$ is a finite set of three scenarios, our feasibly set $\{x : a^\top x \le b \forall a \in \mathcal{U}\}$ will be a polyhedron made up of three half-spaces.

- **Box uncertainty**: This model assumes that the coefficient vector a_i is only known to lie in a "box" (or hyper-rectangle in $\mathbb{R}^n$). In its simplest case, this uncertainty model has the form

  $$\mathcal{U} = \{a : \|a - \hat{a}\|_\infty \le \rho\} = \{a + \rho u : \|u\|_\infty \le 1\},$$

  where $\rho \ge 0$ is a measure of the size of the uncertainty, and $\hat{a}$ is the nominal value of the coefficient vector. We have

  $$\max_{a \in \mathcal{U}} a^\top x = \hat{a}^\top x + \rho \cdot \left(\max_{u : \|u\|_\infty \le 1} u^\top x \right) = \hat{a}^\top x + \rho \|x\|_1.$$

  When $\mathcal{U}$ is a box, the set $\{x : a^\top x \le b \ \forall a \in \mathcal{U}\}$ is a polyhedron with at most 2^n vertices.

- **Spherical uncertainty**: This model has similar assumptions to the box uncertainty, except that the coefficient vector a_i is only known to lie in a sphere, i.e.

  $$\mathcal{U} = \{a : \|a - \hat{a}\|_2 \le \rho\} = \{a + \rho u : \|u\|_2 \le 1\},$$

  where $\rho \ge 0$ is a measure of the size of the uncertainty, and $\hat{a}$ is the nominal value of the coefficient vector. We have

  $$\max_{a \in \mathcal{U}} a^\top x = \hat{a}^\top x + \rho \cdot \left(\max_{u : \|u\|_2 \le 1} u^\top x \right) = \hat{a}^\top x + \rho \|x\|_2 \le b$$

  by Cauchy-Schwarz (Theorem 4).

  Notice that this condition is an SOCP constraint, which alludes to why they're so well studied currently. So when $\mathcal{U}$ is a sphere, the feasible set is defined by a single SOCP constraint.

- **Ellipsoidal uncertainty**: The uncertainty set in this model has form

$$\mathcal{U} = \{a : (a - \hat{a})^\top P^{-1}(a - \hat{a}) \leq 1\},$$

where $\hat{a}$ is the nominal coeffficient vector as usual, and $P = P^\top \succ 0$ determines the shape and size of the ellipse. Since $P \succ 0$, we can write $P = R^\top R$ for some matrix R and get

$$\mathcal{U} = \{a = \hat{a} + Ru : \|u\|_2 \leq 1\},$$

as $u = R^{-1}(a - \hat{a})$ gives back our original condition. So

$$\max_{a \in \mathcal{U}} a^\top x = \hat{a}^\top x + \max_{u : \|u\|_2 \leq 1} (Ru)^\top x = \hat{a}^\top x + \|R^\top x\|_2.$$

Next we can discuss what robust LP looks like with each of these uncertainty constraints, and look at the geometry of the feasible sets.

The robust LP with box uncertainty

$$\min_x \; c^\top x \quad \text{s.t.: } \forall a_i \in \mathcal{B}_i,\; a_i^\top x \leq b_i \quad i = 1, \ldots, m,$$

where $\mathcal{B}_i = \{\hat{a}_i + \rho_i u : \|u\|_\infty \leq 1\}$ is

$$\min_x \; c^\top x \quad \text{s.t.: } \hat{a}_i^\top x + \rho_i \|x\|_1 \leq b_i \quad i = 1, \ldots, m.$$

This can be expressed in standard LP form with our usual tricks as

$$\begin{aligned} \min_x \; c^\top x \quad \text{s.t.: } & \hat{a}_i^\top x + \rho_i \sum_{j=1}^n x \leq b_i \quad i = 1, \ldots, m, \\ & -u_j \leq x_j \leq u_j, \quad j = 1, \ldots, n. \end{aligned}$$

Then discuss ellipsoidal uncertainty, add in spherical geometry.

insert picture of geometry for box uncertain robust LP

14.3 Chance-constrained LP

Chance constrained linear programs arise naturally from standard LPs when some of the data could be uncertain and random. Then we want to model some of our variables as random variables, which generally would make the problem too hard to solve, but under certain assumptions we can arrive at some results.

As always, consider the standard inequality form LP

$$\min_x \; c^\top x \quad \text{s.t.: } a_i^\top x \leq b_i,\; i = 1, \ldots, m.$$

Suppose that the problem data vectors a_i are not knonw precisely, but rather are random vectors modeled by multivariate Gaussians with mean value $\mathbb{E}\{a_i\} = \bar{a}_i$ and covariance matrix $\text{var}\{a_i\} = \Sigma_i \succ 0$. In this case, the scalar $a_i^\top x$ is also a random variable, specifically a normal random variable with

$$\mathbb{E}\{a_i^\top x\} = \bar{a}_i^\top x, \quad \text{var}\{a_i^\top x\} = x^\top \Sigma_i x.$$

However, this means it makes no sense to impose a constraint of the form $a_i^\top x \leq b_i$, since the LHS is a normal random variable and can assume any value. So this constraint can always be violated by some outcomes of our random data a_i. Instead, we only ask for this constraint to be satisfied up to a given level of probability $p_i \in (0, 1)$ chosen *a priori* by the user. This represents the probabilistic *reliability* at which the constraint will be satisfied despite random fluctuations in data.

In summary, we can define chance-constrained LPs as follows:

Definition 25 (Chance-constrained LPs). The *chance-constrained* (or probability constrained) counterpart of the standard nominal LP is

$$\min_x \ c^\top x \quad \text{s.t.: } \Pr\{a_i^\top x \le b_i\} \ge p_i, \quad i = 1, \dots, m, \tag{5}$$

where p_i are the assigned reliability levels.

We have the following result that demonstrates this chance-constrained LP can be modeled by an SOCP.

Theorem 26 (Chance-constrained LPs are SOCP). Consider the problem in (5), under the assumptions that $p_i > 0.5$ for $i = 1, \dots, m$, and that a_i are independent, normal random vectors with means $\bar{a}_i$ and covariance matrices $\Sigma_i \succ 0$. Then the chance-constrained LP presented in (5) is equivalent to the SOCP

$$\min_x \ c^\top x \quad \text{s.t.: } \bar{a}_i^\top x \le b_i - \Phi^{-1}(p_i)\|\Sigma_i^{1/2} x\|_2, \quad i = 1, \dots, m,$$

where $\Phi^{-1}(p)$ is the inverse cumulative probability distribution of a standard normal variable.

Proof. □

14.4 Robust Least-Squares

15 March 9th: Review

We just did review.

16 March 11th: Midterm 1

We had our midterm.

17 Matrix Cheat Sheet

Useful matrix tips I've learned throughout this class.

18 Appendix

List of Definitions and Theorems

Todo list