EECS 127: Optimization Models in Engineering

Tyler Zhu

February 14, 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

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

- $z^{(i)} \in \mathbb{R}^n$, $i = 1, \ldots, n$ 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

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

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.

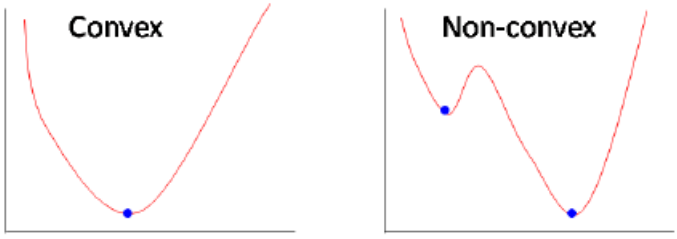

Figure 1: Convex and Non-convex functions

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

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 $x = [x(1)\ x(2)\ \dots\ x(T)]^\top \in \mathbb{R}^T$ 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 $S = \{x^{(1)}, \dots, x^{(m)}\}$ 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)

$$\|x\|_\infty = \max_{k=1,\ldots,n} |x_k|.$$

- 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 $|\cos\theta| \leq 1$, we easily get the following inequality.

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

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

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 $p, q \geq 1$ such that $1/p + 1/q = 1$, it holds that

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

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

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

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

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

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

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

For $p = 1$, we simply set

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

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

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.

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 $\mathcal{S} = \text{span}(x^{(1)}, \ldots, x^{(d)}) \subseteq \mathcal{X}$. 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 $\text{rank}(A) = \text{rank}(A^\top)$.

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, and I might put in when I find time.

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, ...

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 $e_1, \ldots, e_n$ 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 $\text{diag}(c_1, \ldots, c_n)$ in between this product, we can break it down as

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

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!

4 Thursday, January 28th

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

- 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

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 $whttps://www.reddit.com/r/berkeley/comments/lgmlnz/top_20_sc$ 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

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 $z_1, \ldots, z_m \in \mathbb{R}$, 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 $x_1, \ldots, x_m \in \mathbb{R}^n$, 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 10 (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. Also talked about Rayleigh quotients and matrix gain.

5.2 Positive-Semidefinite Matrices

Definition 11 (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

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

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

We then talked about Matrix square-root, Cholesky decomp, and PSD's with ellipsoids. Schur complements are on the homework.

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

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

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

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 12 (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 $(\sigma_1, \dots, \sigma_r)$ positive and decreasing in magnitude, and all other entries zero:

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

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

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

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 $\bar{x}$:

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

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

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

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

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

where $r = \operatorname{rank}(A), U_r = [u_1 \ \ldots \ u_r]$ is such that $U_r^\top U_r = I_r$, $V_r = [v_1 \ \ldots \ v_r]$ 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

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

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

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

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

Similarly,

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

where the inside matrix is $m \times m$. This is the spectral decomposition applied to the symmetric matrix $AA^\top$.

In fact, $AA^\top \succeq 0$ and $A^\top A \succeq 0$.

6.2 Matrix properties via SVD

6.3 Low-rank matrix approximation

6.4 Link with PCA

6.5 Generalized low-rank models

7 Tuesday, February 9th

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 12), 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 Appendix

List of Definitions and Theorems