CS-GY 6763: Lecture 10
Ellipsoid Method, Linear Programming, Singular Value Decomposition

NYU Tandon School of Engineering, Prof. Christopher Musco

DIMENSION DEPENDENT CONVEX OPTIMIZIATION

Consider a convex function $f(\mathrm{x})$ be bounded between $\underline{[-B, B]}$ on a constraint set $\mathcal{S}$.

Theorem (Dimension Dependent Convex Optimization)

The Center-of-Gravity Method finds $\hat{\mathrm{x}}$ satisfying $f(\hat{\mathrm{x}}) \leq \min_{\mathrm{x}\in\mathcal{S}} f(\mathrm{x}) + \underline{\epsilon}$ using $\underline{O(d \log(B/\epsilon))}$ calls to a function and gradient oracle for convex f.

ϵ

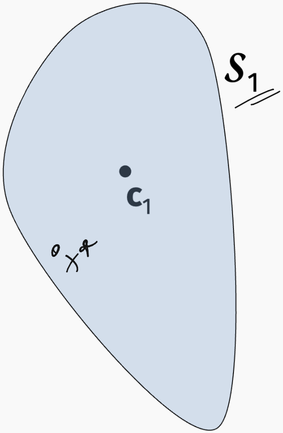
$1/\epsilon^2$

$f: B^d \rightarrow B$

CENTER OF GRAVITY METHOD

Natural "cutting plane" method.

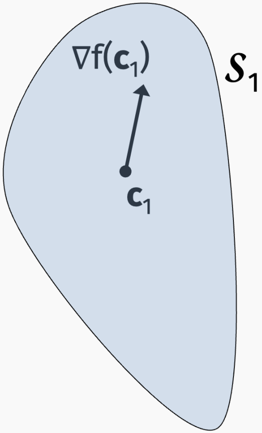
- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$:
    - $\mathbf{c}_t$ = center of gravity of $\mathcal{S}_t$.
    - Compute $\nabla f(\mathbf{c}_t)$.
    - $\mathcal{H} = \{\mathbf{x} \,|\, \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
    - $\mathcal{S}_{t+1} = \mathcal{S}_t \cap H$
- Return $\hat{\mathbf{x}} = \arg\min_t f(\mathbf{c}_t)$



CENTER OF GRAVITY METHOD

Natural "cutting plane" method.

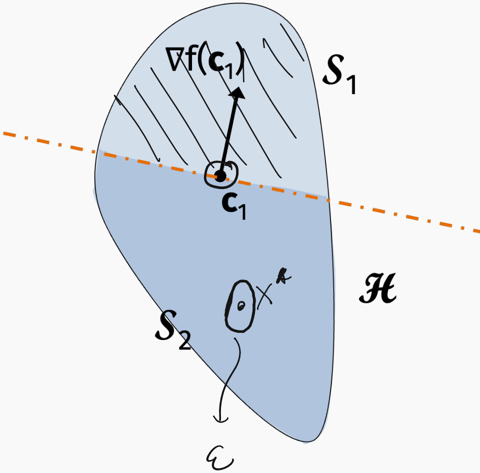
- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$:
    - $\mathbf{c}_t =$ center of gravity of $\mathcal{S}_t$.
    - Compute $\nabla f(\mathbf{c}_t)$.
    - $\mathcal{H} = \{\mathbf{x} \,|\, \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
    - $\mathcal{S}_{t+1} = \mathcal{S}_t \cap H$
- Return $\hat{\mathbf{x}} = \arg\min_t f(\mathbf{c}_t)$



CENTER OF GRAVITY METHOD

Natural “cutting plane” method.

- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$:
    - $\mathbf{c}_t$ = center of gravity of $\mathcal{S}_t$.
    - Compute $\nabla f(\mathbf{c}_t)$.
    - $\mathcal{H} = \{\mathbf{x} \mid \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
    - $\mathcal{S}_{t+1} = \mathcal{S}_t \cap H$
- Return $\hat{\mathbf{x}} = \arg\min_t f(\mathbf{c}_t)$

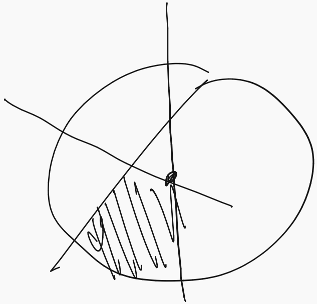


CENTER OF GRAVITY METHOD

Proof Reminder: V_ϵ

- By Grünbaum's Theorem, cut the volume of the search space by a constant every step.
- Need to reduce to a convex body whose volume is roughly ϵ^d smaller than the volume of $\mathcal{S}$.
- Final number of iterations scales with $\log(1/\epsilon^d)$

$= d \log(1/\epsilon)$



RUNTIME ISSUE

In general computing the centroid is hard. #P-hard even when when $\mathcal{S}$ is an intersection of half-spaces (a polytope).

Even if the problem isn't hard for your starting convex body $\mathcal{S}$, it likely will become hard for $\mathcal{S} \cap \mathcal{H}_1 \cap \mathcal{H}_2 \cap \mathcal{H}_3 \ldots$.

So while the oracle complexity of dimension-dependent optimization was settled in the 60s, a number of basic questions in terms of computational complexity.

We will see how to resolve this issue with an elegant cutting plane methods called the **Ellipsoid Method** that was introduced by Naum Shor in 1977.

PROBLEM ~~SIMPLIFICATION~~ Generalization

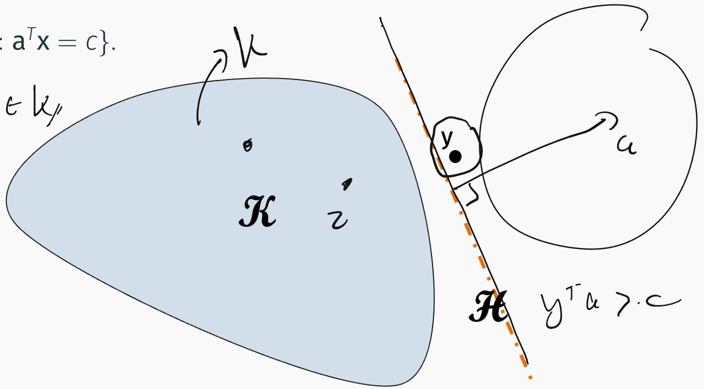
Slightly more general problem: Given a convex set $\mathcal{K}$ via access to separation oracle $S_{\mathcal{K}}$ for the set, (determine if $\mathcal{K}$ is empty, or otherwise return any point $\mathbf{x} \in \mathcal{K}$)

$$S_{\mathcal{K}}(\mathbf{y}) = \begin{cases} \cancel{\emptyset} & \text{if } \mathbf{y} \in \mathcal{K}. \\ \text{separating hyperplane } (\mathbf{a}, c) & \text{if } \mathbf{y} \notin \mathcal{K}. \end{cases}$$

Let $\mathcal{H} = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} = c\}$.

For any $z \in \mathcal{K}$, $z^T a < c$

Also $y^T a \geq c$



SEPARATION ORACLE

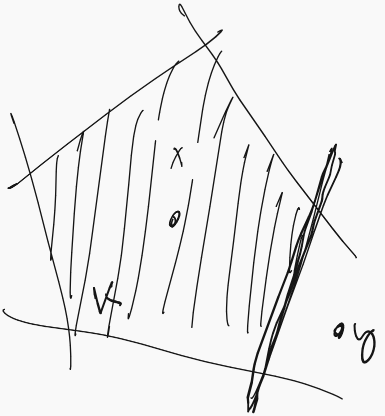
Example: How would you implement a separation oracle for a polytope $\{\mathbf{x} : \underline{\mathbf{A}\mathbf{x} \geq \mathbf{b}}\}$.

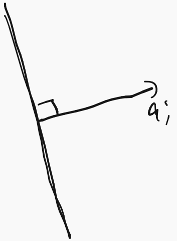
$x^T a_1 \geq b_1$

$x^T a_2 \geq b_2$

$\vdots$

$x^T a_n \geq b_n$





$y^T \underline{a_i} < \underline{b_i}$

but

$\underline{z^T a_i \geq b_i}$ for all $z \in K$

FROM MEMBERSHIP TO OPTIMIZATION

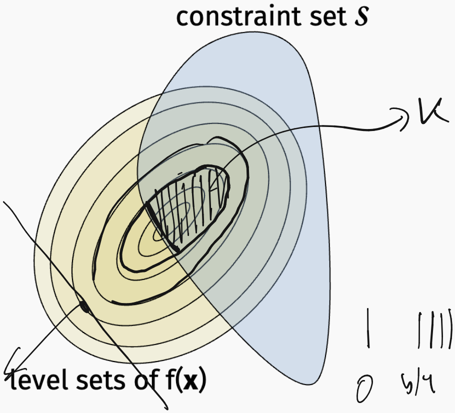
Original problem: $\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x})$.

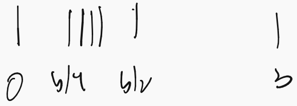
$\boxed{K = \mathcal{S} \cap \{f(x) \le c\}}$

How to reduce to determining if a convex set $\mathcal{K}$ is empty or not?

$\log(1/\epsilon)$

$\log(b/\epsilon)$

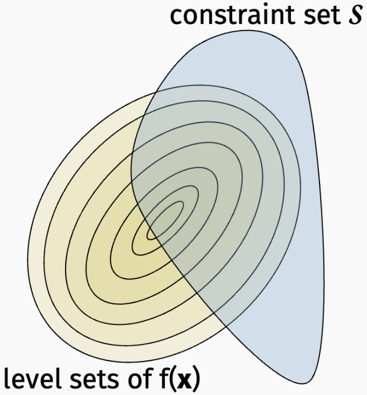




FROM MEMBERSHIP TO OPTIMIZATION

Original problem: $\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x})$.

How to reduce to determining if a convex set $\mathcal{K}$ is empty or not?

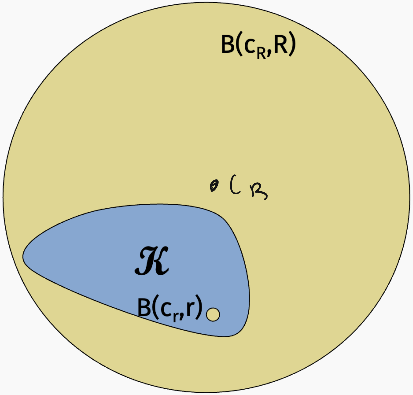


Binary search! For a convex function $f(\mathbf{x})$, $\{\mathbf{x} : f(\mathbf{x}) \leq c\}$ is convex, and you can get a separation oracle via the gradient.

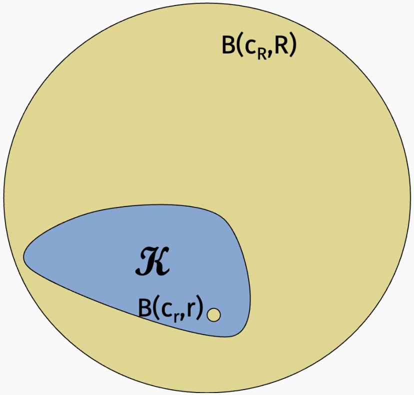
ELLIPSOID METHOD SKETCH

Goal of ellipsoid algorithm: Solve "Is $\mathcal{K}$ empty or not?" given a separation oracle for $\mathcal{K}$ under the assumptions that:

1. $\mathcal{K} \subset B(\underline{\mathbf{c}}_R, \underline{R})$.
2. If non-empty, $\mathcal{K}$ contains $\underline{B(\mathbf{c}_r, \boxed{r})}$ for some $r < R$.



ELLIPSOID METHOD SKETCH

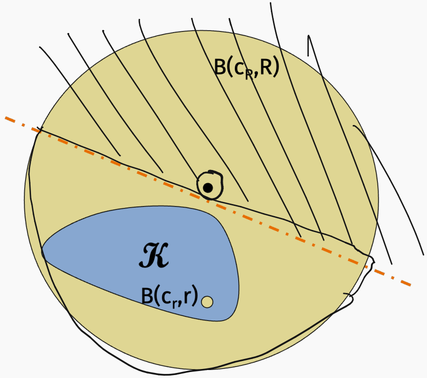


Application to original problem: For a convex function f such that $\|\nabla f(\mathbf{x})\|_2 \leq G$, it can be checked that the convex set $\{\mathbf{x} : f(\mathbf{x}) \leq \epsilon\}$ contains a ball of radius ϵ/G.

ELLIPSOID METHOD SKETCH

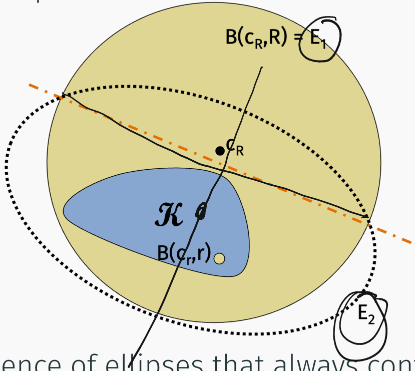
Iterative method similar to center-of-gravity:

1. Check if center $\mathbf{c}_R$ of $B(\mathbf{c}_R, R)$ is in $\mathcal{K}$.
2. If it is, we are done.
3. If not, cut search space in half, using separating hyperplane.



ELLIPSOID METHOD SKETCH

Key insight: Before moving on, approximate new search region by something that we can easily compute the centroid of. Specifically an ellipse!

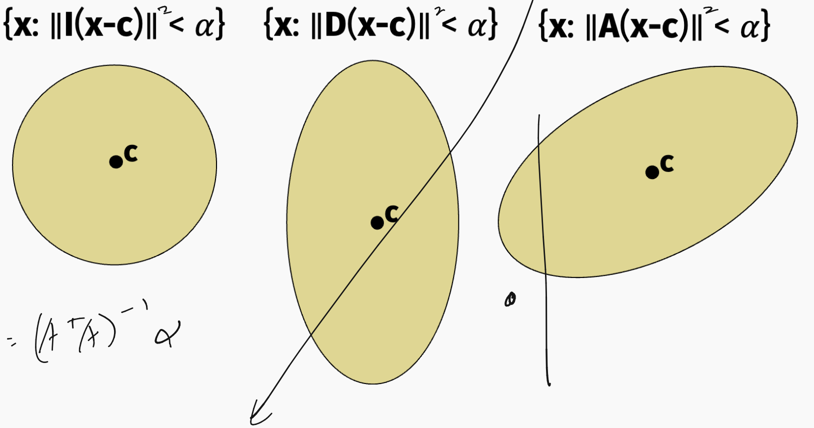


Produce a sequence of ellipses that always contain $\mathcal{K}$ and decrease in volume: $B(\mathbf{c}_R, R) = E_1, E_2, \ldots$. Once we get to an ellipse with volume $\leq B(\mathbf{c}_r, r)$, we know that $\mathcal{K}$ must be empty.

ELLIPSE

An ellipse is a convex set of the form: $\{\underline{\mathbf{x}} : \|\mathbf{A}(\mathbf{x} - \underline{\mathbf{c}})\|_2^2 \leq \alpha\}$ for some constant c and matrix $\mathbf{A}$. The center-of-mass is $\mathbf{c}$.

→ center



$\{x: \|I(x-c)\|^2 < \alpha\}$
$\{x: \|D(x-c)\|^2 < \alpha\}$
$\{x: \|A(x-c)\|^2 < \alpha\}$
c
c
c

$Q = (A^TA)^{-1}\alpha$

Often re-parameterized to say that the ellipse is all $\mathbf{x}$ with $\{\mathbf{x} : (\mathbf{x} - \mathbf{c})^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{c}) \leq 1\}$

ELLIPSOID UPDATE

There is a closed form solution for the equation of the smallest ellipse containing a given half-ellipse. I.e. let $\mathbf{E}_i$ have parameters $\mathbf{Q}_i, \mathbf{c}_i$ and consider the half-ellipse:

$$\mathbf{E}_i \cap \{\mathbf{x} : \mathbf{a}_i^T\mathbf{x} \leq \mathbf{a}_i^T\mathbf{c}_i\}.$$

Then $\mathbf{E}_{i+1}$ is the ellipse with parameters:

$$\mathbf{Q}_{i+1} = \frac{d^2}{d^2-1}\left(\mathbf{Q}_i - \frac{2}{d+1}\mathbf{h}\mathbf{h}^T\right) \qquad \mathbf{c}_{i+1} = \mathbf{c}_i - \frac{1}{n+1}\mathbf{h},$$

where $\mathbf{h} = \sqrt{\mathbf{a}_i^T\mathbf{Q}_i\mathbf{a}_i} \cdot \mathbf{a}_i$.

$a_i^T Q_i a_i$

$$\downarrow$$

$d \times d$

Computing the update takes $O(d^2)$ time.

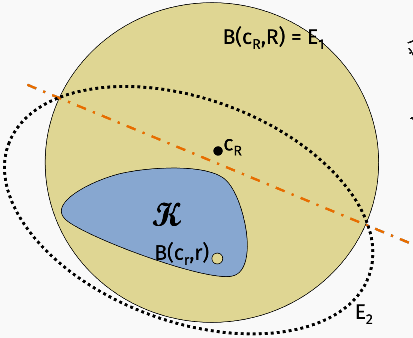
GEOMETRIC OBSERVATION

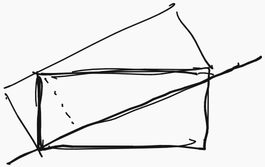
.64

Claim: $\underline{\text{vol}(E_{i+1})} \leq \underline{(1 - \frac{1}{2d})}\, \underline{\text{vol}(E_i)}$.

$(1 - \frac{1}{2d})^{\boxed{2d}} \approx \frac{1}{e}$

Proof: Via reduction to the "isotropic case". I will post a proof on the course website if you are interested.





Not as good as the $(1 - \frac{1}{e})$ constant-factor volume reduction we got from center-of-gravity, but still very good!

GEOMETRIC OBSERVATION

$d \log(1/\varepsilon)$

Claim: $\mathrm{vol}(E_{i+1}) \leq \underline{(1 - \frac{1}{2d})}\,\mathrm{vol}(E_i)$

If run for

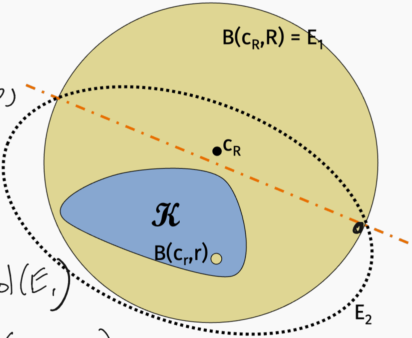
$T = d^2 \log(R/r)$ steps

then

$\mathrm{Vol}(E_T) \leq \left(\frac{r}{R}\right)^d \mathrm{Vol}(E_1)$

$\mathrm{Vol}(E_T) \leq \mathrm{Vol}(B(c_r, r))$

$(R/r)^d$



After $O(d)$ iterations, we reduce the volume by a constant.

In total require $\underline{O(d^2 \log(R/r))}$ iterations to solve the problem.

LINEAR PROGRAMMING

Linear programs (LPs) are one of the most basic convex constrained, convex optimization problems:

Let $\mathbf{c} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{n\times d}$ be fixed vectors that define the problem, and let $\mathbf{x}$ be our variable parameter.

$$\min \underline{f(\mathbf{x})} = \underline{\mathbf{c}^T\mathbf{x}}$$

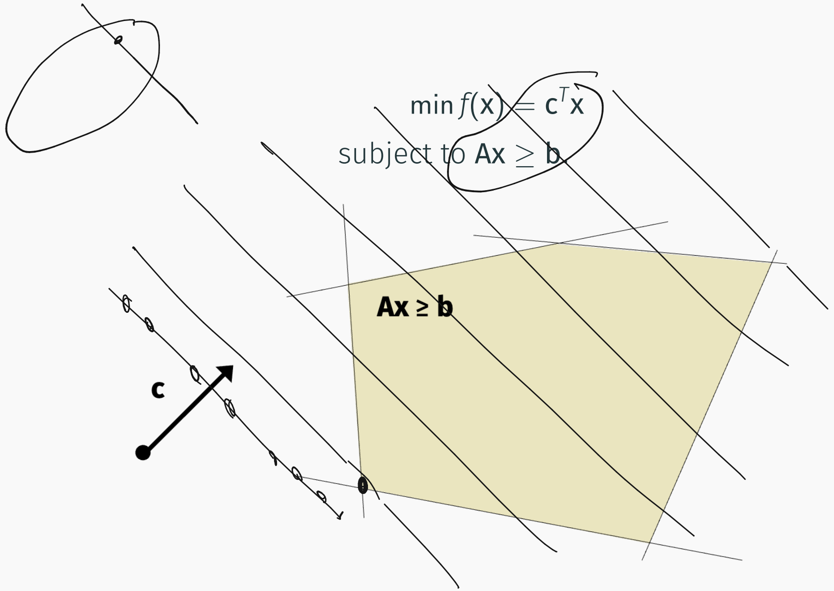
$$\text{subject to } \underbrace{\mathbf{A}\mathbf{x} \geq \mathbf{b}}_{\text{fixed}}. \xrightarrow{} \text{fixed}$$

Think about $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ as a union of half-space constraints:

$$\{\mathbf{x} : \mathbf{a}_1^T\mathbf{x} \geq b_1\}$$
$$\{\mathbf{x} : \mathbf{a}_2^T\mathbf{x} \geq b_2\}$$
$$\vdots$$
$$\{\mathbf{x} : \mathbf{a}_n^T\mathbf{x} \geq b_n\}$$

KILLER APPLICATION: LINEAR PROGRAMMING

$$\min f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$$
$$\text{subject to } \mathbf{A}\mathbf{x} \geq \mathbf{b}$$

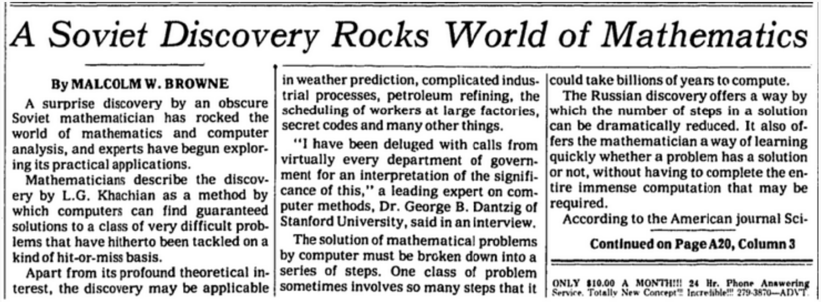


LINEAR PROGRAMMING APPLICATIONS

- Classic optimization applications: industrial resource optimization problems were killer app in the 70s.
- Robust regression: $\min_x \|Ax - b\|_1$.
- $L1$ constrained regression: $\boxed{\min_x \|x\|_1}$ subject to $\underline{Ax = b}$. Lots of applications in sparse recovery/compressed sensing.
- Solve $\min_x \|Ax - b\|_\infty$.
- Polynomial time algorithms for Markov Decision Processes.
- Many combinatorial optimization problems can be solved via LP relaxations.

Theorem (Khachiyan, 1979)

Assume $n = d$. The ellipsoid method solves any linear program with L-bit integer valued constraints exactly in $O(n^4 L)$ time.



A Soviet Discovery Rocks World of Mathematics

By MALCOLM W. BROWNE

A surprise discovery by an obscure Soviet mathematician has rocked the world of mathematics and computer analysis, and experts have begun exploring its practical applications.

Mathematicians describe the discovery by L.G. Khachian as a method by which computers can find guaranteed solutions to a class of very difficult problems that have hitherto been tackled on a kind of hit-or-miss basis.

Apart from its profound theoretical interest, the discovery may be applicable in weather prediction, complicated industrial processes, petroleum refining, the scheduling of workers at large factories, secret codes and many other things.

"I have been deluged with calls from virtually every department of government for an interpretation of the significance of this," a leading expert on computer methods, Dr. George B. Dantzig of Stanford University, said in an interview.

The solution of mathematical problems by computer must be broken down into a series of steps. One class of problem sometimes involves so many steps that it could take billions of years to compute.

The Russian discovery offers a way by which the number of steps in a solution can be dramatically reduced. It also offers the mathematician a way of learning quickly whether a problem has a solution or not, without having to complete the entire immense computation that may be required.

According to the American journal Sci-

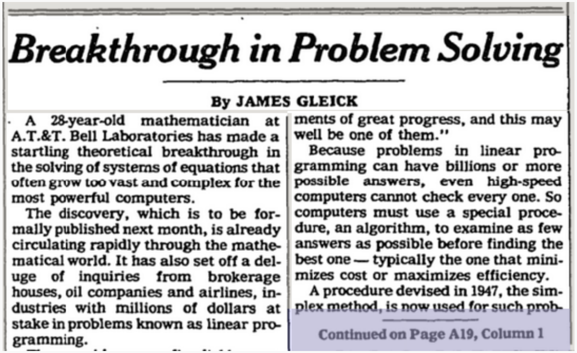
Continued on Page A20, Column 3

ONLY $10.00 A MONTH!!! 24 Hr. Phone Answering Service. Totally New Concept!! Incredible!!! 279-3870—ADVT.

Front page of New York Times, November 9, 1979.

Theorem (Karmarkar, 1984)

Assume $n = d$. The interior point method solves any linear program with L-bit integer valued constraints in $O(n^{3.5}L)$ time.



Breakthrough in Problem Solving

By JAMES GLEICK

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers.

The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming.

ments of great progress, and this may well be one of them.''

Because problems in linear programming can have billions or more possible answers, even high-speed computers cannot check every one. So computers must use a special procedure, an algorithm, to examine as few answers as possible before finding the best one — typically the one that minimizes cost or maximizes efficiency.

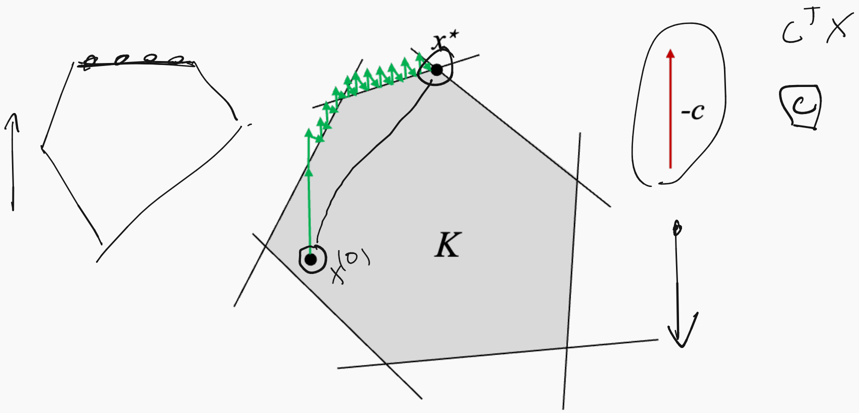
A procedure devised in 1947, the simplex method, is now used for such prob-

Continued on Page A19, Column 1

Front page of New York Times, November 19, 1984.

INTERIOR POINT METHODS

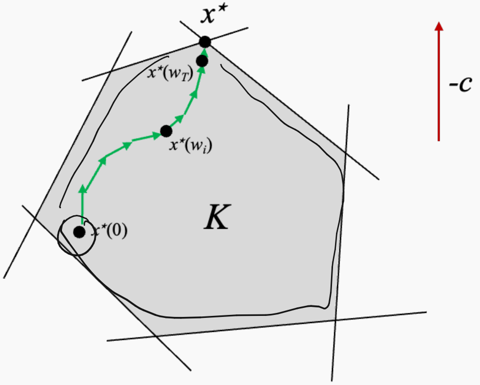
Lecture notes are posted on the website (optional reading).



Projected Gradient Descent Optimization Path

INTERIOR POINT METHODS

Lecture notes are posted on the website (optional reading).



Ideal Interior Point Optimization Path

POLYNOMIAL TIME LINEAR PROGRAMMING

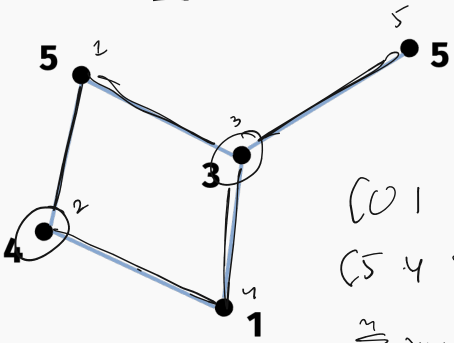
Both results had a huge impact on the theory of optimization, although at the time neither the ellipsoid method or interior point method were faster than a heuristic known at the Simplex Method.

These days, improved interior point methods compete with and often outperform simplex.

Polynomial time linear programming algorithms have also had a huge impact of combinatorial optimization. They are often the work-horse behind approximation algorithms for NP-hard problems.

EXAMPLE: VERTEX COVER

Given a graph G with n nodes and edge set E. Each node is assigned a weight $w_1, \ldots, w_n$.



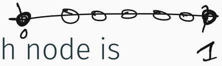
$[0\ 1\ 1\ 0\ 0] = x$

$[5\ 4\ 3\ 1\ 5] = w$

$\sum_{i=1}^{n} x_i w_i = 4+3 = 7$

Goal: Select subset of nodes with minimum total weight that covers all edges.

EXAMPLE: VERTEX COVER



Given a graph G with n nodes and edge set E. Each node is assigned a weight $w_1, \ldots, w_n$. $\boxed{0}, 1, 2$

Formally: Denote if node i is selected by assigning variable x_i to 0 or 1. Let $\mathbf{x} = [x_1, \ldots, x_n]$.

$$\underbrace{\min_{\mathbf{x}} \sum_{i=1}^{n} x_i w_i}_{f(x)} \xrightarrow{\text{constraints}} \quad \text{subject to} \quad \begin{array}{c} x_i \in \{0,1\} \text{ for all } i \\ x_i + x_j \geq 1 \text{ for all } (i,j) \in E \end{array}$$

NP-hard to solve exactly. We will use convex optimization give a 2-approximation in polynomial time. $x^T w$

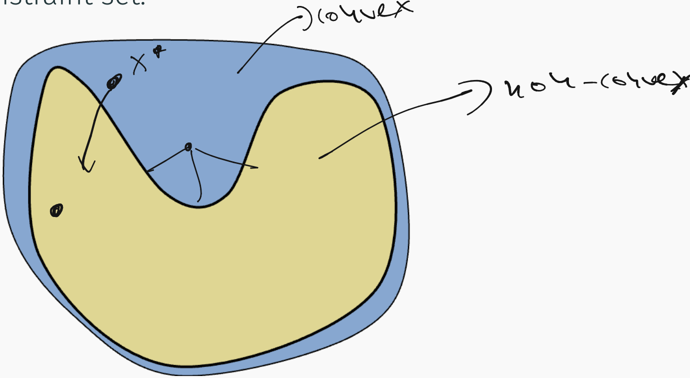
Function to minimize is linear (so convex) but constraint set is not convex. Why?

$\min_x f(x)$ where $f(x) = \sum_{i=1}^{n} x_i w_i$

RELAX-AND-ROUND

High level approach:

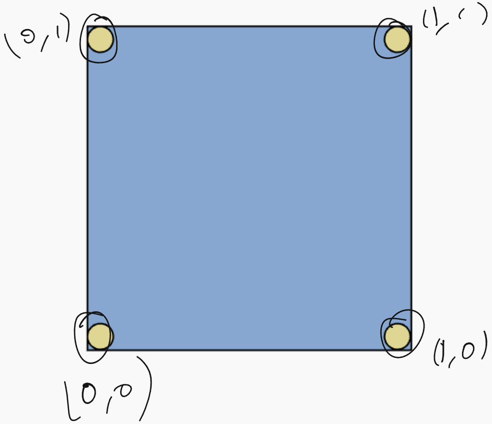
- Relax to a problem with convex constraints.
- Round optimal solution of convex problem back to original constraint set.



RELAX-AND-ROUND

High level approach:

- Relax to a problem with convex constraints.
- Round optimal solution of convex problem back to original constraint set.



High level approach: original constraints

relaxed constraints

- Relax to a problem with convex constraints.
- Round optimal solution of convex problem back to original constraint set.

Let $\bar{\mathcal{S}} \supseteq \mathcal{S}$ be the relaxed constraint set. Let $\mathbf{x}^* = \underline{\arg\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x})}$ and let $\underline{\bar{\mathbf{x}}^*} = \arg\min_{\mathbf{x}\in\bar{\mathcal{S}}} f(\mathbf{x})$. We always have that:

$$\underline{f(\bar{\mathbf{x}}^*)} \leq f(\mathbf{x}^*).$$

So typically the goal is to round $\bar{\mathbf{x}}^*$ to $\mathcal{S}$ in such a way that we don't increase the function value too much.

$$\bar{x}^* \longrightarrow y \in S \qquad f(y) \leq f(\underline{\bar{x}^*}) + \epsilon$$

RELAXING VERTEX COVER

Vertex Cover:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} x_i w_i \quad \text{subject to} \quad x_i \in \{0,1\} \text{ for all } i$$

$$x_i + x_j \geq 1 \text{ for all } (i,j) \in E$$

Relaxed Vertex Cover:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} x_i w_i \quad \text{subject to} \quad 0 \leq x_i \leq 1 \text{ for all } i$$

$$x_i + x_j \geq 1 \text{ for all } (i,j) \in E$$

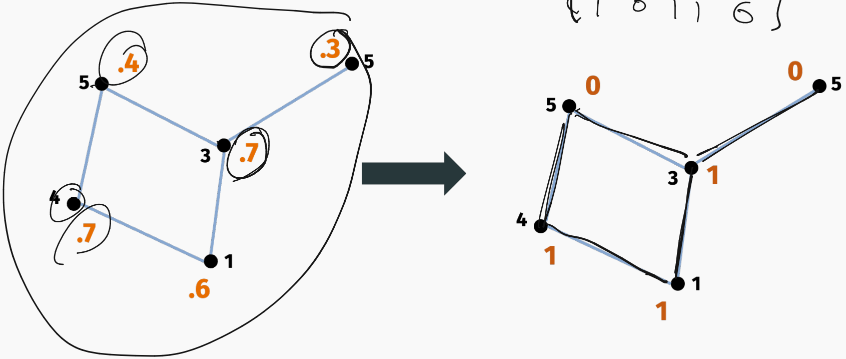
The second problem is a linear program! It can be solved in poly(n) time!

$$f(\bar{x}^*) \leq \min_{x \in S} f(x)$$

ROUNDING VERTEX COVER

Simply set all variable $x_i = 1$ of $\bar{x}_i^* \geq 1/2$ and $x_i = 0$ otherwise.

[1 0 1 1 0]



Observation 1: All edges remain covered. I.e., the constraint $x_i + x_j \geq 1$ for all $(i, j) \in E$ is not violated.

ROUNDING VERTEX COVER

Observation 2: Let x be the rounded version of $\underline{\bar{x}^*}$. We have $f(x) \leq 2 \cdot f(\bar{x}^*)$, and thus $f(x) \leq 2 \cdot f(x^*)$.

Proof:

$$f(x) = \sum_{i=1}^{n} w_i \, \mathbb{1}[\bar{x}_i^* \geq 1/2] \leq \sum_{i=1}^{n} w_i \cdot 2 \cdot \bar{x}_i^*$$

$$= 2 f(\bar{x}^*)$$

$$f(\bar{x}^*) = \sum_{i=1}^{n} w_i \, \bar{x}_i^*$$

$$.7 \rightarrow 1$$

$$f(\bar{x}^*) \leq f(x^*)$$

VERTEX COVER

So, a polynomial time algorithm for solving LPs immediately yields a 2-approximation algorithm for the NP-hard problem of vertex cover.

- Proven that it is NP-hard to do better than a 1.36 approximation in [Dinur, Safra, 2002].
- Recently improved to $\sqrt{2} \approx 1.41$ in [Khot, Minzer, Safra 2018], which proved the 2-to-2 games conjecture.
- Widely believed that doing better than $2 - \epsilon$ is NP-hard for any $\epsilon > 0$, and this is implied by Subhash Khot's Unique Games Conjecture.

There is a simpler greedy 2-approximation algorithm that doesn't use optimization at all. Try coming up with it on your own!

BREAK

SPECTRAL METHODS

Next section of course: Spectral methods and numerical linear algebra.

Spectral methods generally refer to methods based on the "spectrum" of a matrix. I.e. on it's eigenvectors/eigenvalues and singular vectors/singular values. We will look at applications in:

- Low-rank approximation and dimensionality reduction.
- Data clustering and related problems.
- Constructing data embeddings (e.g. Word2Vec).

SPECTRAL METHODS

Reminder: A vector $\mathbf{v} \in \mathbb{R}^d$ is an eigenvector of a matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$, if there exists a scalar λ such that

$$\mathbf{X}\mathbf{v} = \lambda \mathbf{v}$$

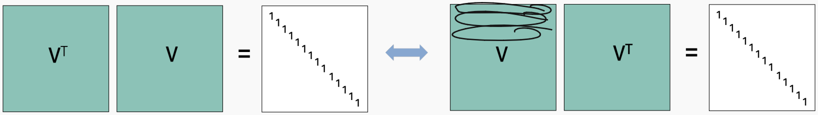
The scalar λ is called the eigenvalue associated with $\mathbf{v}$.

Matrices can often be written completely in terms of their eigenvectors and eigenvalues. This is called eigendecomposition.

We will actually focus on a related tool called (singular value decomposition).

LINEAR ALGEBRA REMINDER

If a square matrix has orthonormal rows, it also has orthonormal columns:



$$\boxed{V^T V} = \boxed{I} = \underline{VV^T}$$

$$\begin{bmatrix} -0.62 & 0.78 & -0.11 \\ -0.28 & -0.35 & -0.89 \\ -0.73 & -0.52 & 0.44 \end{bmatrix} \cdot \begin{bmatrix} -0.62 & -0.28 & -0.73 \\ 0.78 & -0.35 & -0.52 \\ -0.11 & -0.89 & 0.44 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

LINEAR ALGEBRA REMINDER

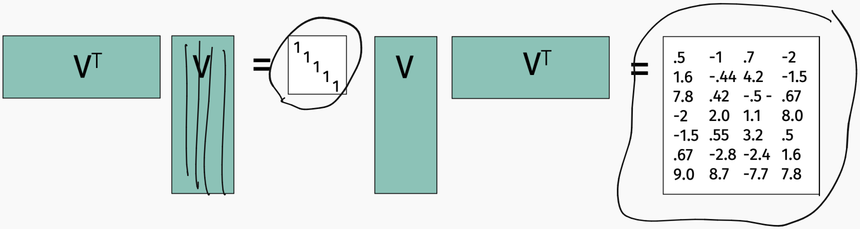
Implies that for any vector $\underline{\underline{\mathbf{x}}}$, $\underline{\|\mathbf{V}\mathbf{x}\|_2^2} = \|\mathbf{x}\|_2^2$ and $\|\mathbf{V}^T\mathbf{x}\|_2^2$.

Same thing goes for Frobenius norm: for any matrix $\mathbf{X}$,
$\|\mathbf{V}\mathbf{X}\|_F^2 = \|\mathbf{X}\|_F^2$ and $\|\mathbf{V}^T\mathbf{X}\|_F^2 = \|\mathbf{X}\|_F^2$.

$$\|Vx\|_2^2 = (Vx)^T Vx = x^T \underbrace{V^T V}_{I} x = x^T x = \|x\|_2^2$$

$$\|V^T x\|_2^2 = x \underbrace{V V^T}_{I} x = \|x\|_2$$

The same is not true for rectangular matrices:



$\mathbf{V}^T\mathbf{V} = \mathbf{I}$ but $\mathbf{V}\mathbf{V}^T \neq \mathbf{I}$

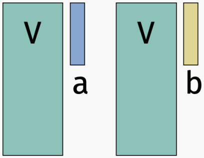
For any $\mathbf{x}$, $\|\mathbf{V}\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2$ but $\|\mathbf{V}^T\mathbf{x}\|_2^2 \neq \|\mathbf{x}\|_2^2$ in general.

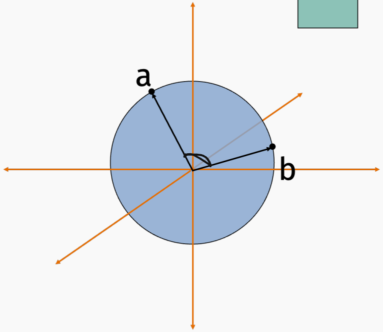
LINEAR ALGEBRA REMINDER

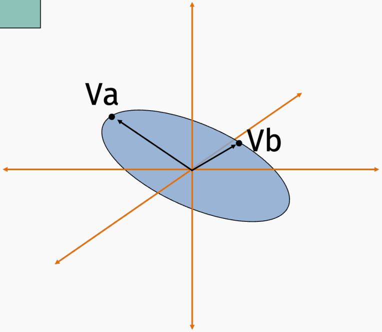
Multiplying a vector by $\mathbf{V}$ with orthonormal columns rotates and/or reflects the vector.

$$\langle a, b \rangle = a^T b$$

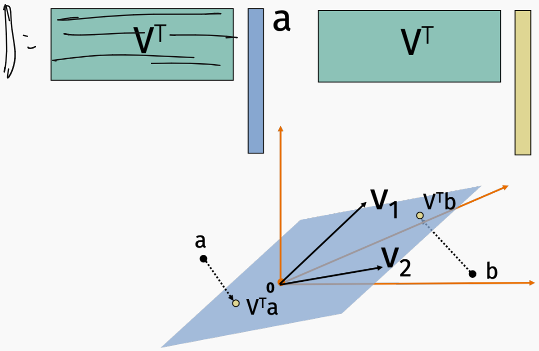
$$\langle Va, Vb \rangle = a^T V^T V b$$
$$= a^T b$$







Multiplying a vector by a rectangular matrix $\mathbf{V}^T$ with orthonormal rows projects the vector (representing it as coordinates in the lower dimensional space).

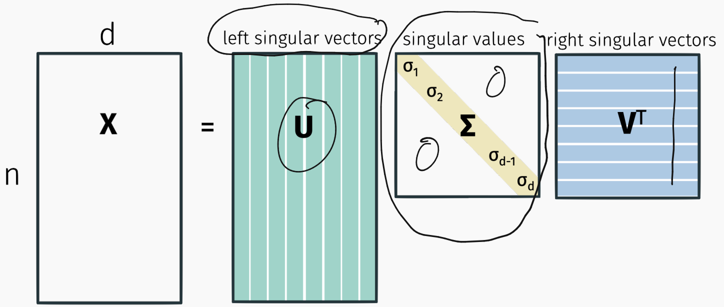


So we always have that $\|\mathbf{V}^T\mathbf{x}\|_2 \leq \|\mathbf{x}\|_2$.

SINGULAR VALUE DECOMPOSITION

One of the most fundamental results in linear algebra.

Any matrix $\mathbf{X}$ can be written:



Where $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, and $\underline{\sigma_1 \geq \sigma_2 \geq \ldots \sigma_d \geq 0}$.

Singular values are unique. Factors are not. E.g. would still get a valid SVD by multiplying both i^{th} column of $\mathbf{V}$ and $\mathbf{U}$ by -1.

SINGULAR VALUE DECOMPOSITION

Important take away from singular value decomposition.

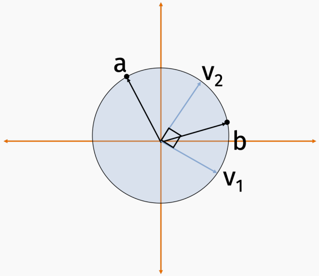
Multiplying any vector **a** by a matrix **X** to form **Xa** can be viewed as a composition of 3 operations:

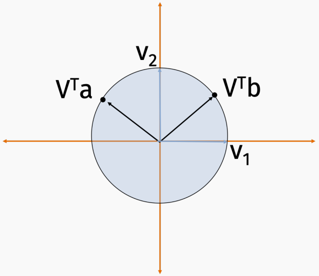
1. Rotate/reflect the vector (multiplication by to $\mathbf{V}^T$).
2. Scale the coordinates (multiplication by $\mathbf{\Sigma}$.
3. Rotate/reflect the vector again (multiplication by **U**).

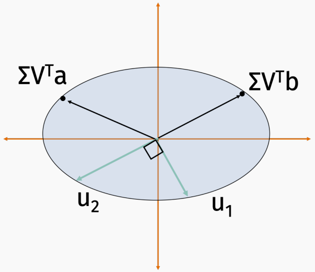
Xa

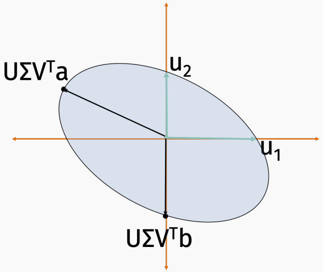
$U(\Sigma(V^T a))$

SINGULAR VALUE DECOMPOSITION: ROTATE/REFLECT









COMPARISON TO EIGENDECOMPOSITION

(Recall that an eigenvalue of a square matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$ is any vector $\mathbf{v}$ such that $\mathbf{X}\mathbf{v} = \lambda \mathbf{v}$.) A matrix has at most d linearly independent eigenvectors. If a matrix has a full set of d eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ with eigenvalues $\lambda_1, \ldots, \lambda_d$ it is called "diagonalizable" and can be written as:

$$X = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}.$$

$\mathbf{V}$'s columns are $\mathbf{v}_1, \ldots, \mathbf{v}_d$.

$$\begin{bmatrix} v_1 & \ldots & v_d \end{bmatrix} \qquad \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix}$$

COMPARISON TO EIGENDECOMPOSITION

Singluar value decomposition

- Exists for all matrices, square or rectangular. ✓
- Singular values are always positive. ✓
- Factors $\mathbf{U}$ and $\mathbf{V}$ are orthogonal. ✓

Eigendecomposition

- Exists for <u>some</u> square matrices. ✓
- Eigenvalues can be positive or negative. ✓
- Factor $\mathbf{V}$ is orthogonal if and only if $\mathbf{X}$ is symmetric.

CONNECTION TO EIGENDECOMPOSITION

- $\boxed{U}$ contains the orthogonal eigenvectors of $\boxed{XX^T}$.
- $\boxed{V}$ contains the orthogonal eigenvectors of $\underline{X^TX}$.
- $\boxed{\sigma_i^2} = \lambda_i(\underline{X}X^T) = \lambda_i(X^TX)$

$X = U\Sigma U^T$ $\boxed{XX^T} = U\Sigma \cancel{U^T U} \Sigma^T U^T$

$= \underline{U\Sigma^2 U^T}$

SVD APPLICATIONS

Lots of applications.

- Compute pseudoinverse $\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$.
- Read off condition number of $\mathbf{X}$, σ_1^2/σ_d^2.
- Compute matrix norms. E.g. $\boxed{\|\mathbf{X}\|_2 = \sigma_1}$, $\underline{\|\mathbf{X}\|_F} = \boxed{\sqrt{\sum_{i=1}^d \sigma_i^2}}$.
- Compute matrix square root – i.e. find a matrix $\mathbf{B}$ such that $\mathbf{B}\mathbf{B}^T = \boxed{\mathbf{X}}$. Used e.g. in sampling from Gaussian with covariance $\mathbf{X}$.

  $B = U \Sigma^{1/2} U^T$
- <u>Principal component analysis</u>.

Killer app: $\underbrace{\text{Read off optimal \underline{low-rank}}}$ approximations for $\mathbf{X}$.

RANK

The column span of a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the set of all vectors that can be written as $\mathbf{X}\mathbf{a}$ for some $\mathbf{a}$.

The dimension of the column span, D_c, is the maximum number of linear independent vectors in that set.

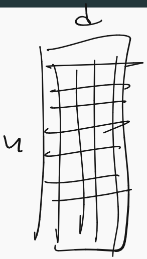
The row span of a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the set of all vectors that can be written as $\mathbf{b}^T\mathbf{X}$ for some $\mathbf{b}$.

The dimension of the row span, D_r, is the maximum number of linear independent vectors in that set.

RANK

For a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ we have:

$$\boxed{D_c \leq d}$$
$$D_r \leq n$$
$$\boxed{D_c = D_r.}$$

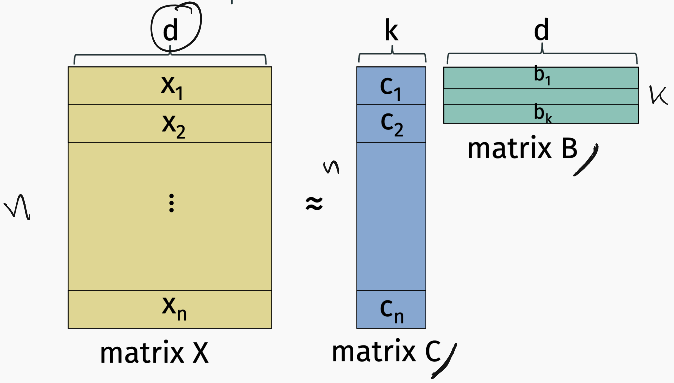


We call the value of $D_c = D_r$ the rank of $\mathbf{X}$.

$$\text{rank} \leq \min(d, n)$$

LOW-RANK APPROXIMATION

Approximate **X** as the product of two rank k matrices: $k << d$

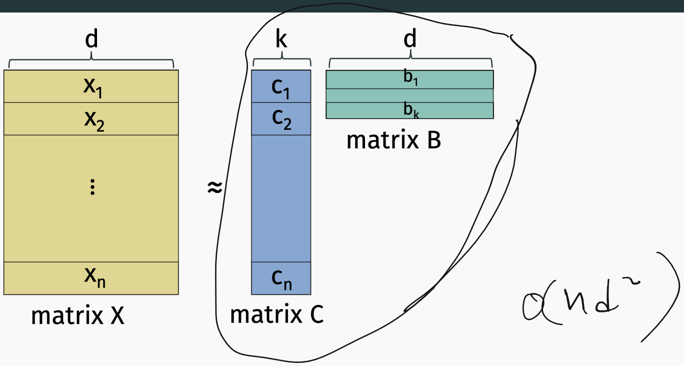


Typically choose **C** and **B** to minimize:

$$\min_{\mathbf{B},\mathbf{C}} \|\mathbf{X} - \mathbf{C}\mathbf{B}\|$$

for some matrix norm. Common choice is $\|\mathbf{X} - \mathbf{C}\mathbf{B}\|_F^2$.

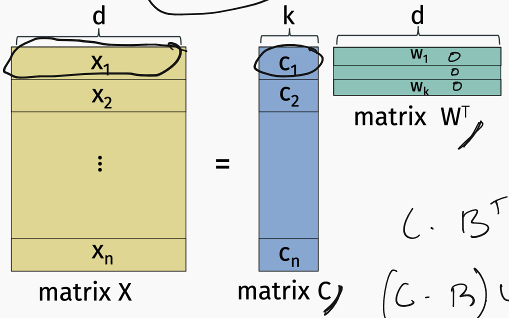
APPLICATIONS OF LOW-RANK APPROXIMATION



- CB takes $O(k(n+d))$ space to store instead of $O(nd)$.
- Regression problems involving CB can be solved in $\underline{O(nk^2)}$ instead of $O(nd^2)$ time.
- Will see a bunch more in a minute.

LOW-RANK APPROXIMATION

Without loss of generality can assume that the right matrix is orthogonal. I.e. W^T with $W^TW = I$



$C \cdot B^T$

$(C \cdot B) W^T$

Then we should choose $\overset{W}{C}$ to minimize:

$$\min_{C,W} \|X - CW^T\|_F^2$$

$\min \|x_1 - c_1^T W\|_2^2$

$= \arg\min_{c_1} \|W c_1 - x_1\|_2^2$

This is just n least squares regression problems!

LOW-RANK APPROXIMATION

$x_i \approx x_i W W^T$

$x_j \approx x_j W W^T$

$X - XGG^T$

$$\boxed{c_i = \arg\min_{c} \|Wc - x_i\|_2^2}$$

$c_i = \underbrace{(W^T W)}_{I}^{-1} W^T x_i = W^T x_i$

$\min \|X - XWW^T\|_F^2$

$= \sum_{i=1} \|x_i - x_i W W^T\|_F^2$

$$c_i = W^T x_i$$

$$\boxed{C = XW}$$

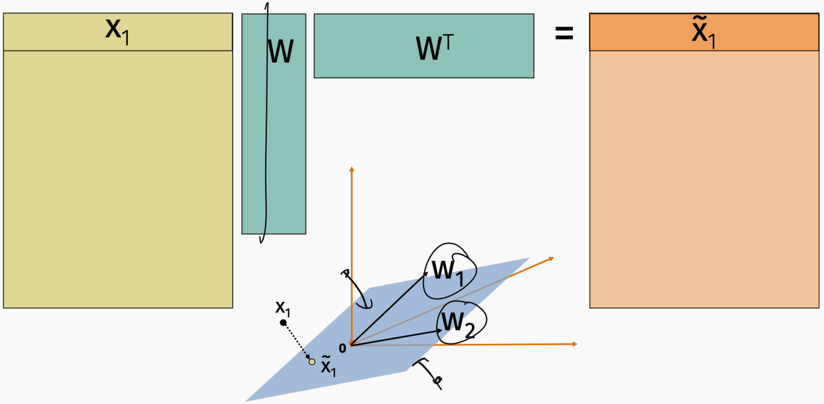
So our optimal low-rank approximation always has the form:

$$X \approx XWW^T$$

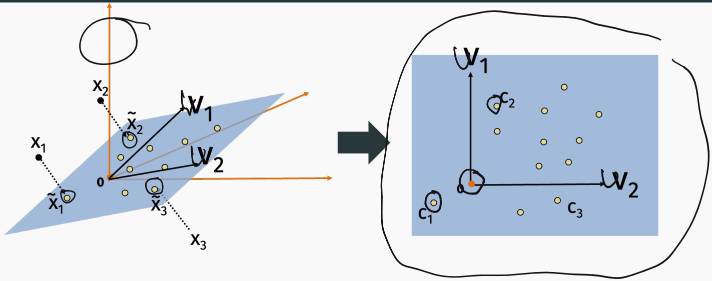
$\min_{C,B} \|X - CB\|_F^2$ $\min_{W} \|X - XWW^T\|_F^2$ where $W^T W = I$

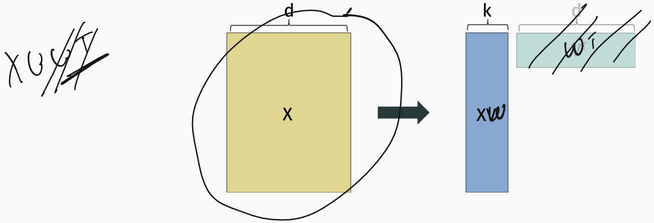
PROJECTION MATRICES

WW^T is a symmetric projection matrix.



LOW-RANK APPROXIMATION





$C = XW$ can be used as a compressed version of data matrix X.

DATA COMPRESSION

$c_i = x_i W \quad c_j = x_j W$

Let $\mathbf{C} = \mathbf{X}\mathbf{W}$. We have that:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 \approx \|\mathbf{x}_i^T \mathbf{W}\mathbf{W}^T - \mathbf{x}_j^T \mathbf{W}\mathbf{W}^T\|_2 = \|\mathbf{c}_i - \mathbf{c}_j\|_2$$

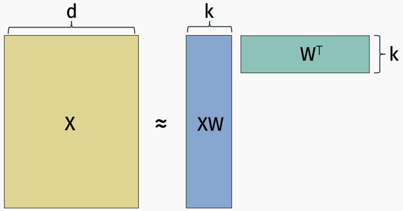
Similarly, we expect that:

- $\|\mathbf{x}_i\|_2 \approx \|\mathbf{c}_i\|_2$
- $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \approx \langle \mathbf{c}_i, \mathbf{c}_j \rangle$
- etc.

How does this compare to Johnson-Lindenstrauss projection?

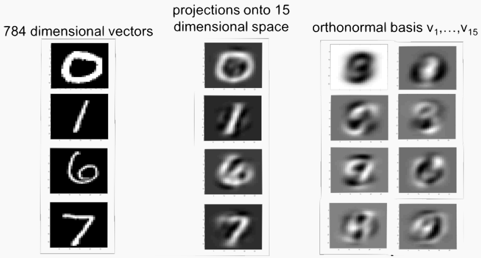
WHY IS DATA APPROXIMATELY LOW-RANK?

Rows of $\mathbf{X}$ (data points) are approximately spanned by k vectors. Columns of $\mathbf{X}$ (data features) are approximately spanned by k vectors.



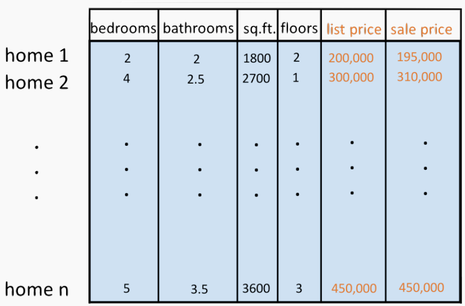
ROW REDUNDANCY

If a data set only had k unique data points, it would be exactly rank k. If it has k "clusters" of data points (e.g. the 10 digits) it's often very close to rank k.


784 dimensional vectors
projections onto 15 dimensional space
orthonormal basis $v_1, \ldots, v_{15}$

COLUMN REDUNDANCY

Colinearity/correlation of data features leads to a low-rank data matrix.

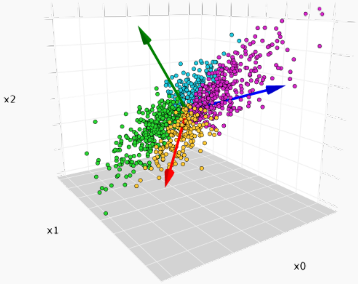


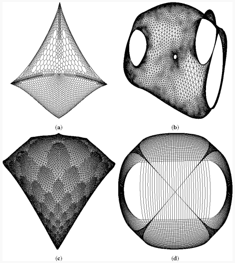
	bedrooms	bathrooms	sq.ft.	floors	list price	sale price
home 1	2	2	1800	2	200,000	195,000
home 2	4	2.5	2700	1	300,000	310,000
⋮	⋮	⋮	⋮	⋮	⋮	⋮
home n	5	3.5	3600	3	450,000	450,000

APPLICATIONS OF LOW-RANK APPROXIMATION

Fact that $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \approx \|\mathbf{x}_i^T\mathbf{W}\mathbf{W}^T - \mathbf{x}_j^T\mathbf{W}\mathbf{W}^T\|_2 = \|\mathbf{c}_i - \mathbf{c}_j\|_2$ leads to lots of applications.

- Data compression. E.g. used in state-of-the-art data dependent methods for nearest neighbor search.
- Data visualization when $k = 2$ or 3.

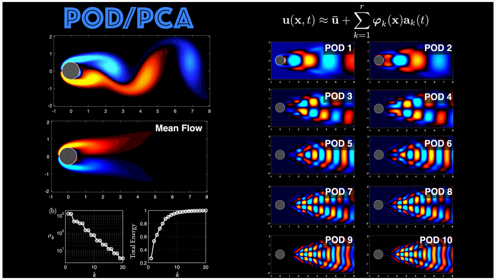




- Data embeddings (next lecture).

APPLICATIONS OF LOW-RANK APPROXIMATION

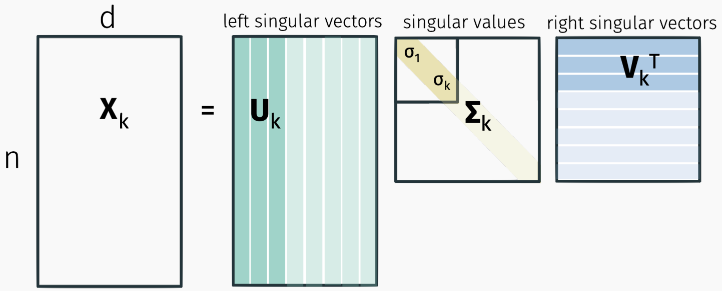
- Reduced order modeling for solving physical equations.



- Constructing preconditioners in optimization.
- Noisy triangulation (on problem set).

PARTIAL SVD

Can find the best projection from the singular value decomposition.



$$V_k = \underset{\text{orthogonal } W \in \mathbb{R}^{d \times k}}{\arg\min} \|X - XWW^T\|_F^2$$

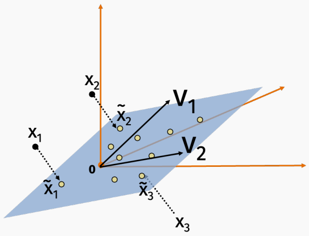
OPTIMAL LOW-RANK APPROXIMATION

Claim: $X_k = U_k \boldsymbol{\Sigma}_k V_k^T = X V_k V_k^T$.

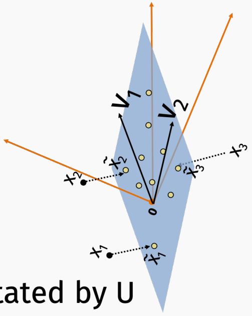
OPTIMALITY OF SVD

Claim 1:

$$\underset{\text{rank } k \ \mathbf{B}}{\arg\min} \|\mathbf{X} - \mathbf{B}\|_F^2 = \mathbf{U} \cdot \underset{\text{rank } k \ \mathbf{B}}{\arg\min} \|\mathbf{\Sigma V}^T - \mathbf{B}\|_F^2$$



ΣV^T



Rotated by U on the left

OPTIMALITY OF SVD

Claim 2:

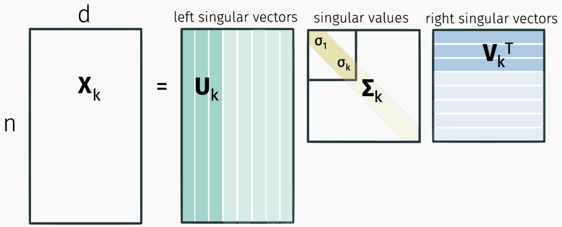
$$\underset{\text{rank } k \ \mathbf{B}}{\arg\min} \|\mathbf{\Sigma}\mathbf{V}^T - \mathbf{B}\|_F^2 = \underset{\text{rank } k \ \mathbf{B}}{\arg\min} \|\mathbf{V}\mathbf{\Sigma} - \mathbf{B}^T\|_F^2$$

Claim 3:

$$\underset{\text{rank } k \ \mathbf{B}}{\arg\min} \|\mathbf{V}\mathbf{\Sigma} - \mathbf{B}^T\|_F^2 = \underset{\text{rank } k \ \mathbf{B}}{\arg\min} \|\mathbf{\Sigma} - \mathbf{V}^T\mathbf{B}^T\|_F^2$$

Chose $\mathbf{B}^T$ so that $\mathbf{V}^T\mathbf{B}^T = \mathbf{\Sigma}_k$.

USEFUL OBSERVATIONS



Observation 1:

$$\underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\arg\min} \|\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T\|_F^2 = \underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\arg\max} \|\mathbf{X}\mathbf{W}\mathbf{W}^T\|_F^2$$

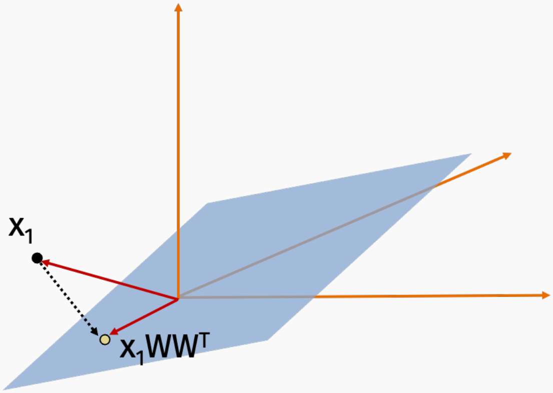
Follows from fact that for <u>all</u> orthogonal $\mathbf{W}$:

$$\|\mathbf{X}-\mathbf{X}\mathbf{W}\mathbf{W}^T\|_F^2 = \|\mathbf{X}\|_F^2 - \|\mathbf{X}\mathbf{W}\mathbf{W}^T\|_F^2$$

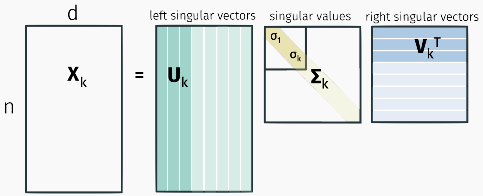
USEFUL OBSERVATIONS

Claim:

$$\|X - XWW^T\|_F^2 = \|X\|_F^2 - \|XWW^T\|_F^2$$



USEFUL OBSERVATIONS



Observation 2: The optimal low-rank approximation error $E_k = \|X - XV_kV_k^T\|_F^2 = \|X\|_F^2 - \|XV_kV_k^T\|_F^2$ can be written:

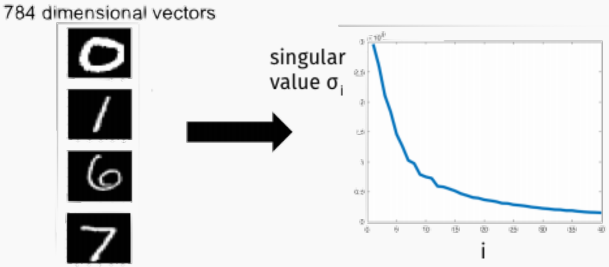
$$E_k = \sum_{i=k+1}^{d} \sigma_i^2.$$

SPECTRAL PLOTS

Observation 2: The optimal low-rank approximation error $E_k = \|\mathbf{X} - \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T\|_F^2 = \|\mathbf{X}\|_F^2 - \|\mathbf{X}\mathbf{V}_k\mathbf{V}_k^T\|_F^2$ can be written:

$$E_k = \sum_{i=k+1}^{d} \sigma_i^2.$$

Can immediately get a sense of "how low-rank" a matrix is from it's spectrum:

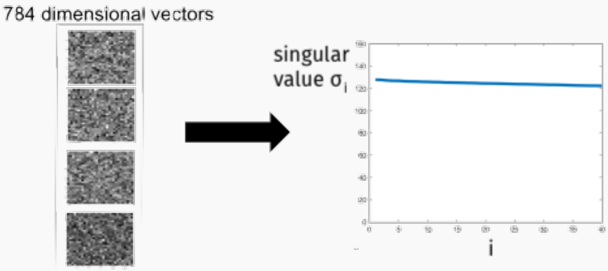


SPECTRAL PLOTS

Observation 2: The optimal low-rank approximation error $E_k = \|\mathbf{X} - \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T\|_F^2 = \|\mathbf{X}\|_F^2 - \|\mathbf{X}\mathbf{V}_k\mathbf{V}_k^T\|_F^2$ can be written:

$$E_k = \sum_{i=k+1}^{d} \sigma_i^2.$$

Can immediately get a sense of "how low-rank" a matrix is from it's spectrum:

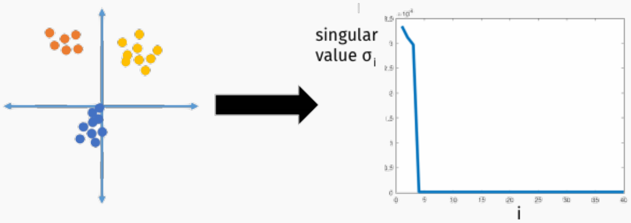


SPECTRAL PLOTS

Observation 2: The optimal low-rank approximation error $E_k = \|\mathbf{X} - \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T\|_F^2 = \|\mathbf{X}\|_F^2 - \|\mathbf{X}\mathbf{V}_k\mathbf{V}_k^T\|_F^2$ can be written:

$$E_k = \sum_{i=k+1}^{d} \sigma_i^2.$$

Can immediately get a sense of "how low-rank" a matrix is from it's spectrum:



COMPUTING THE SVD

Suffices to compute right singular vectors $\mathbf{V}$:

- Compute $\mathbf{X}^T\mathbf{X}$.
- Find eigendecomposition $\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T = \mathbf{X}^T\mathbf{X}$ using e.g. QR algorithm.
- Compute $\mathbf{L} = \mathbf{X}\mathbf{V}$. Set $\sigma_i = \|\mathbf{L}_i\|_2$ and $\mathbf{U}_i = \mathbf{L}_i / \|\mathbf{L}_i\|_2$.

Total runtime $\approx$

computing the svd (faster)

Next class:

- Compute approximate solution.
- Only compute top k singular vectors/values. Runtime will depend on k. When $k = d$ we can't do any better than classical algorithms based on eigendecomposition.
- Iterative algorithms achieve runtime $\approx O(ndk)$ vs. $O(nd^2)$ time.
  - **Krylov subspace methods** like the Lanczos method are most commonly used in practice.
  - **Power method** is the simplest Krylov subspace method, and still works very well.