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

NYU Tandon School of Engineering, Prof. Christopher Musco
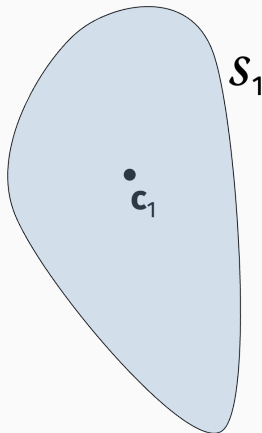
Consider a convex function $f(\mathbf{x})$ be bounded between $[-B, B]$ on a constraint set $\mathcal{S}$.

Theorem (Dimension Dependent Convex Optimization)

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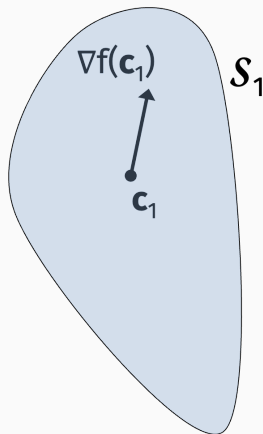
Natural "cutting plane" method.

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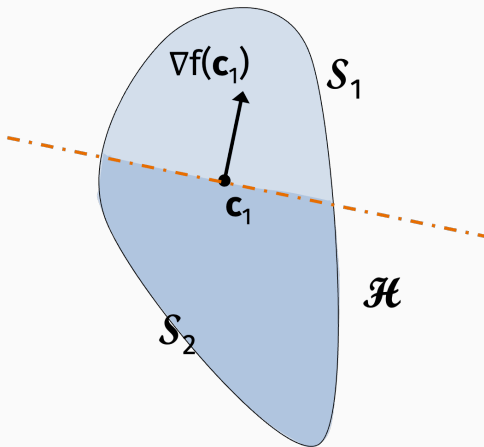- Return $\hat{\mathbf{x}} = \arg \min_t f(\mathbf{c}_t)$

Natural "cutting plane" method.

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

Natural "cutting plane" method.

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

Proof Reminder:

- 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 $\log(1/\epsilon^d)$

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 <u>oracle complexity</u> of dimension-dependent optimization was settled in the 60s, a number of basic questions in terms of <u>computational complexity.</u>
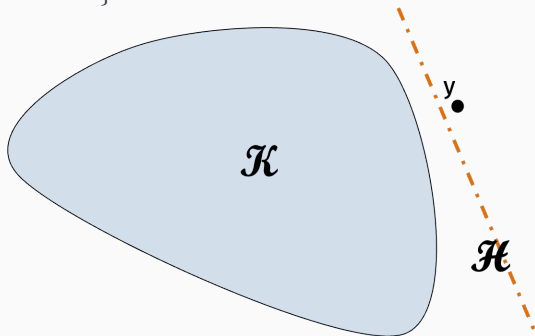
We will see how to resolve this issue with an elegant cutting plane methods called the **Ellipsoid Method** that was introduced by Naum Shor in 1977.

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

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

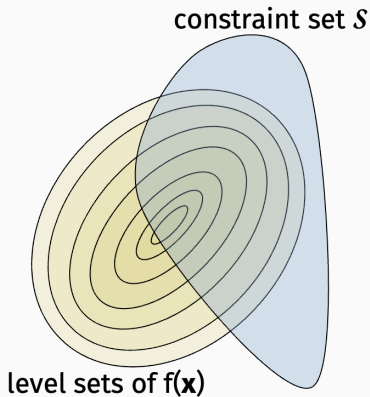Let $\mathcal{H} = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} = c\}$.

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

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

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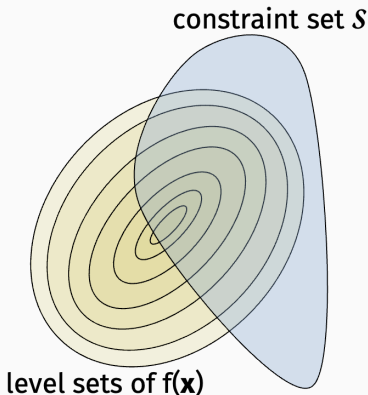

constraint set $\mathcal{S}$

level sets of f($\mathbf{x}$)

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

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

constraint set $\mathcal{S}$

level sets of f(x)

Binary search! For a convex function $f(x)$, $\{x : f(x) \leq c\}$ is convex, and you can get a separation oracle via the gradient.
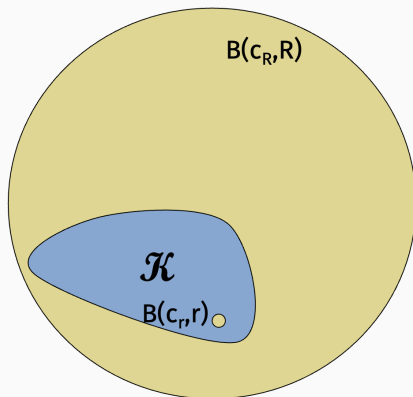
11

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

1. $\mathcal{K} \subset B(\mathbf{c}_R, R)$.
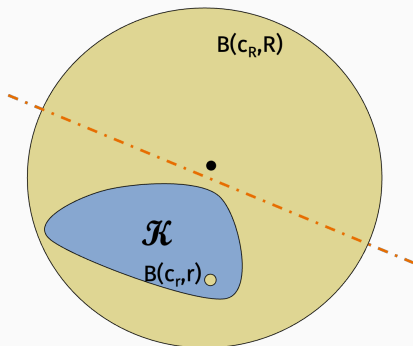2. If non-empty, $\mathcal{K}$ contains $B(\mathbf{c}_r, r)$ for some $r < R$.
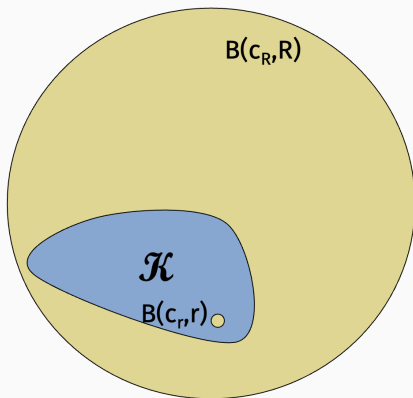
Application to original problem: For a convex function f such that $\|\nabla f(\mathbf{x})\|_2 \leq G$, it can be checked that the convex set $\{\mathbf{x} : f(\mathbf{x}) \leq \epsilon\}$ contains a ball of radius ϵ/G.
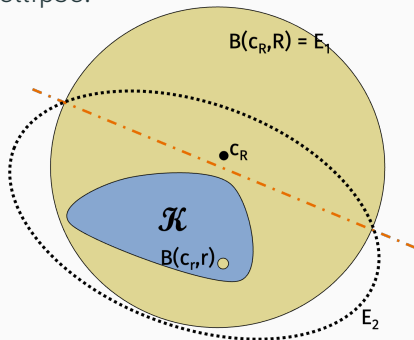
Iterative method similar to center-of-gravity:

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

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

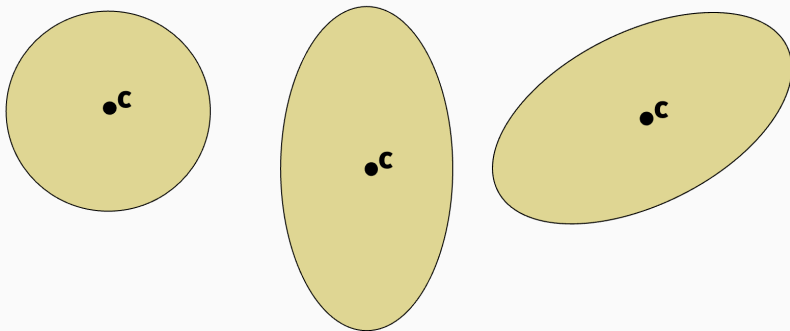

Produce a sequence of ellipses that <u>always contain</u> $\mathcal{K}$ and decrease in volume: $B(\mathbf{c}_R, R) = E_1, E_2, \ldots$. Once we get to an ellipse with volume $\leq B(\mathbf{c}_r, r)$, we know that $\mathcal{K}$ must be empty.

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

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

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

16

There is a closed form solution for the equation of the smallest ellipse containing a given half-ellipse. I.e. let E_i have parameters Q_i, c_i and consider the half-ellipse:

$$E_i \cap \{x : a_i^T x \leq a_i^T c_i\}.$$

Then E_{i+1} is the ellipse with parameters:

$$Q_{i+1} = \frac{d^2}{d^2 - 1} \left(Q_i - \frac{2}{d+1} hh^T \right) \qquad c_{i+1} = c_i - \frac{1}{n+1} h,$$
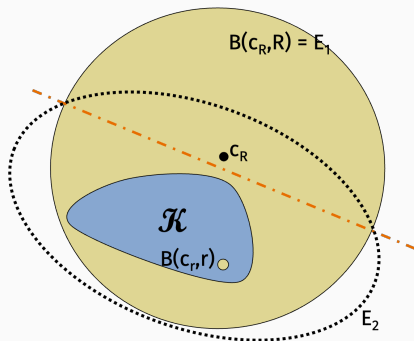
where $h = \sqrt{a_i^T Q_i a_i} \cdot a_i$.
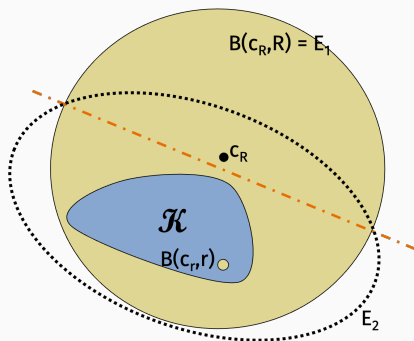
Computing the update takes $O(d^2)$ time.

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

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!

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

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

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

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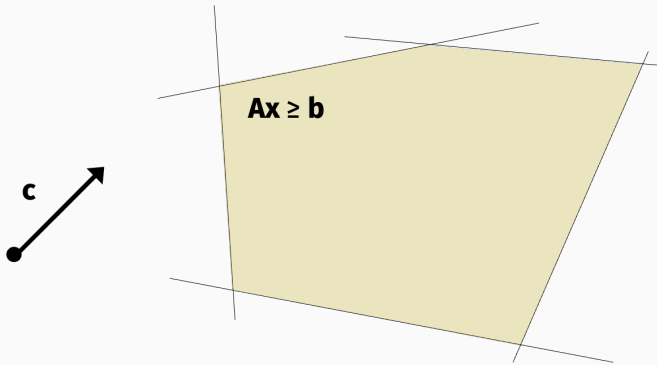
$$\min f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$$
$$\text{subject to } \mathbf{A}\mathbf{x} \geq \mathbf{b}.$$

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

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

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

subject to $\mathbf{Ax} \geq \mathbf{b}$.

- Classic optimization applications: industrial resource optimization problems were killer app in the 70s.
- Robust regression: $\min_x \|Ax - b\|_1$.
- $L1$ constrained regression: $\min_x \|x\|_1$ subject to $Ax = b$. 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 <u>ellipsoid method</u> solves any linear program with L-bit integer valued constraints <u>exactly</u> 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.
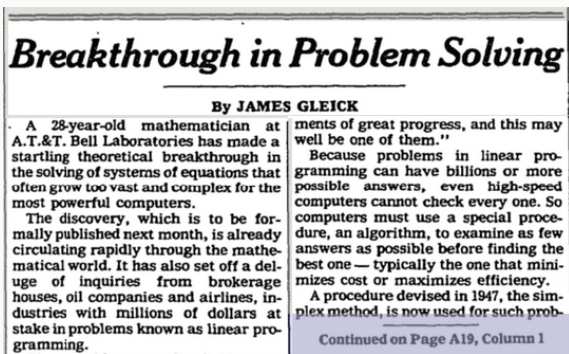
According to the American journal Sci-

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.

23

Theorem (Karmarkar, 1984)

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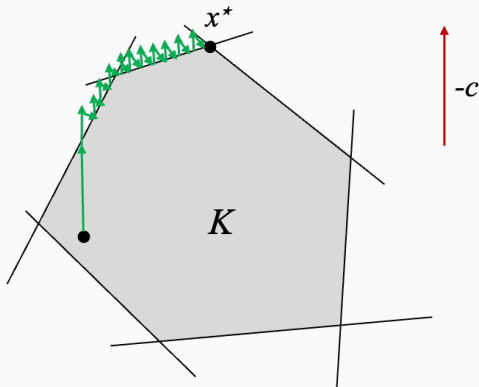
Breakthrough in Problem Solving

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-

Front page of New York Times, November 19, 1984.
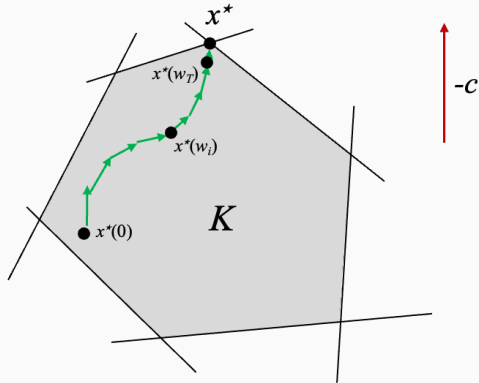
Lecture notes are posted on the website (optional reading).

Projected Gradient Descent Optimization Path

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


Ideal Interior Point Optimization Path

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.

Given a graph *G* with *n* nodes and edge set *E*. Each node is assigned a weight $w_1, \ldots, w_n$.
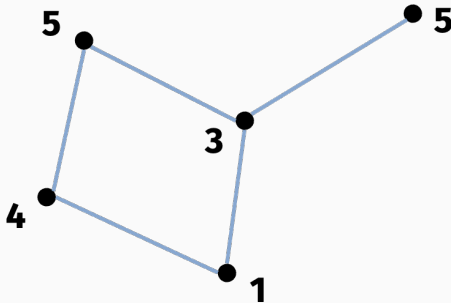
Goal: Select subset of nodes with minimum total weight that covers all edges.

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

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

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

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

**NP-hard to solve exactly.** We will use convex optimization give a 2-approximation in polynomial time.

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

High level approach:

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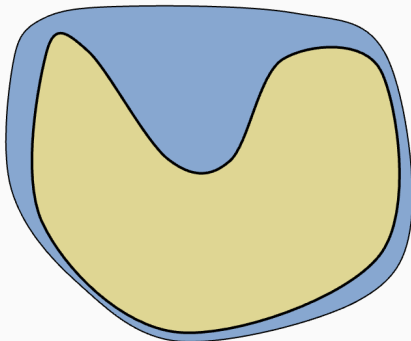
High level approach:

- <u>Relax</u> to a problem with convex constraints.
- <u>Round</u> optimal solution of convex problem back to original constraint set.
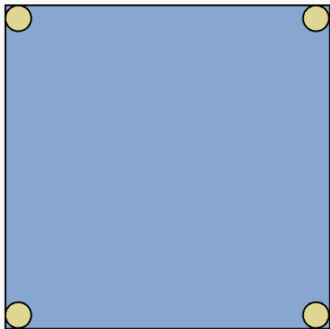
High level approach:

- 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 $x^* = \arg\min_{x \in \mathcal{S}} f(x)$ and let $\bar{x}^* = \arg\min_{x \in \bar{\mathcal{S}}} f(x)$. We always have that:

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

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

Vertex Cover:

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

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

Relaxed Vertex Cover:

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

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

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

Simply set all variable $x_i = 1$ of $\bar{x}_i^* \geq 1/2$ and $x_i = 0$ otherwise.
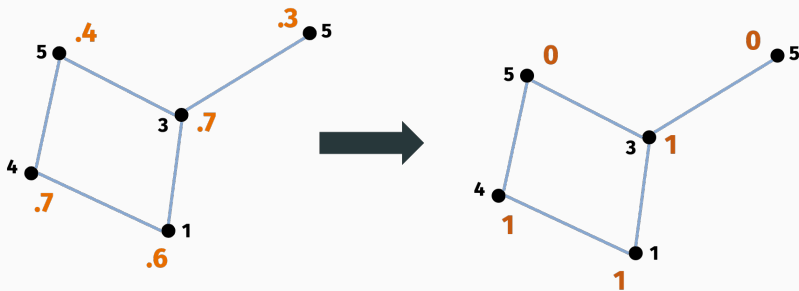
Observation 1: All edges remain covered. I.e., the constraint $x_i + x_j \geq 1$ for all $(i, j) \in E$ is not violated.

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

Proof:

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

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).

Reminder: A vector $\mathbf{v} \in \mathbb{R}^d$ is an <u>eigenvector</u> 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 <u>eigenvalue</u> 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 <u>singular value decomposition.</u>

If a <u>square</u> matrix has orthonormal rows, it also has orthonormal columns:

$$V^T V = I = V V^T$$

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

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

The same is <u>not true</u> for rectangular matrices:

| V^T | V | = | $\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$ | V | V^T | = | .5 -1 .7 -2
 1.6 -.44 4.2 -1.5
 7.8 .42 -.5 - .67
 -2 2.0 1.1 8.0
 -1.5 .55 3.2 .5
 .67 -2.8 -2.4 1.6
 9.0 8.7 -7.7 7.8 |

$$V^T V = I \qquad \text{but} \qquad VV^T \neq I$$

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

Multiplying a vector by **V** with orthonormal columns <u>rotates and/or reflects</u> the vector.
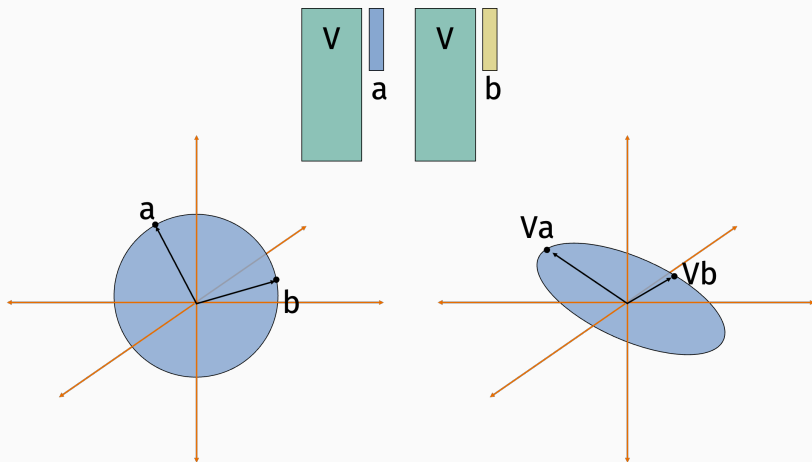
Multiplying a vector by a rectangular matrix V^T with orthonormal rows <u>projects</u> the vector (representing it as coordinates in the lower dimensional space).
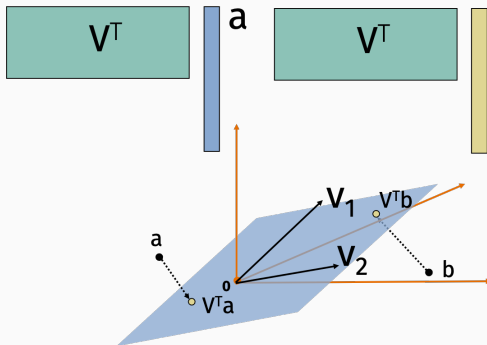
So we always have that $\|V^T\mathbf{x}\|_2 \leq \|\mathbf{x}\|_2$.

One of the most fundamental results in linear algebra.

<u>Any</u> matrix $\mathbf{X}$ can be written:
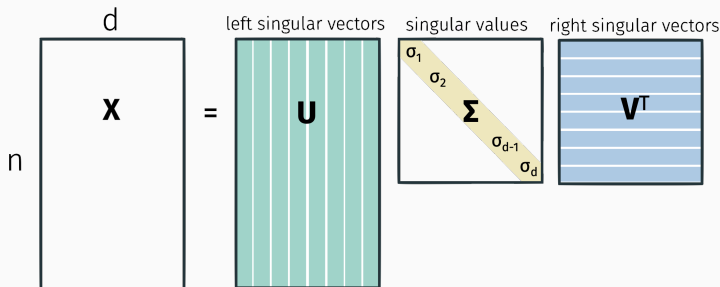
Where $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, and $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_d \geq 0$.

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.

44

Important <u>take away</u> 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 V^T).
2. Scale the coordinates (multiplication by **Σ**.
3. Rotate/reflect the vector again (multiplication by **U**).

Recall that an eigenvalue of a <u>square</u> matrix $X \in \mathbb{R}^{d \times d}$ is any vector v such that $Xv = \lambda v$. A matrix has at most d linearly independent eigenvectors. If a matrix has a full set of d eigenvectors $v_1, \ldots, v_d$ with eigenvalues $\lambda_1, \ldots, \lambda_d$ it is called "diagonalizable" and can be written as:

$$V \Lambda V^{-1}.$$

V's columns are $v_1, \ldots, v_d$.
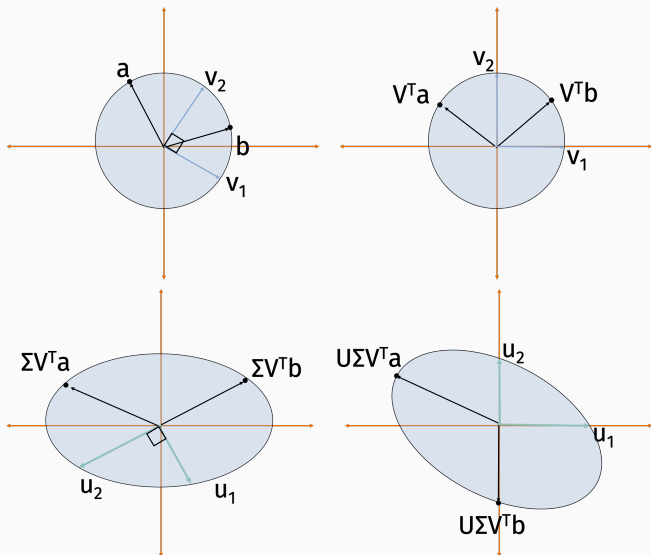
COMPARISON TO EIGENDECOMPOSITION

Singluar value decomposition

- Exists for all matrices, square or rectangular.
- Singular values are always positive.
- Factors U and V are orthogonal.

Eigendecomposition

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

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

Lots of applications.

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

Killer app: Read off optimal <u>low-rank</u> approximations for $\mathbf{X}$.

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

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

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

$$D_c \leq d$$
$$D_r \leq n$$
$$D_c = D_r.$$

We call the value of $D_c = D_r$ the rank of X.

Approximate X as the product of two rank k matrices:
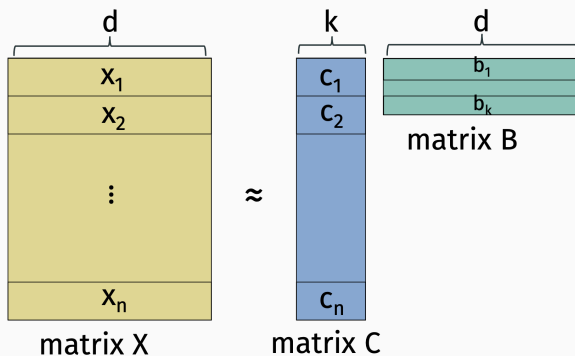
Typically choose C and B to minimize:

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

for some matrix norm. Common choice is $\|X - CB\|_F^2$.

53
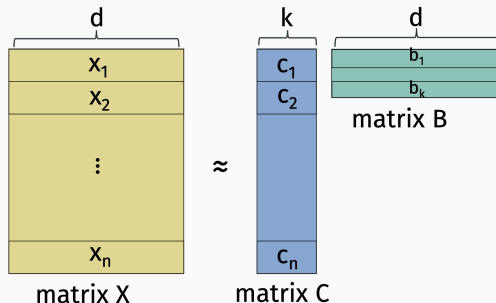
- **CB** takes $O(k(n + d))$ space to store instead of $O(nd)$.
- Regression problems involving **CB** can be solved in $O(nk^2)$ instead of $O(nd^2)$ time.
- Will see a bunch more in a minute.

Without loss of generality can assume that the right matrix is orthogonal. I.e. W^T with $W^T W = I$
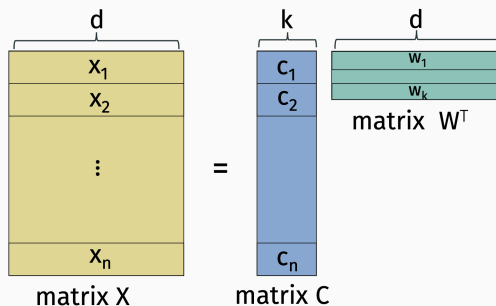
Then we should choose C to minimize:

$$\min_C \|X - CW^T\|_F^2$$

This is just n least squares regression problems!

55

$$c_i = \arg\min_c \|Wc - x_i\|_2^2$$

$$c_i = W^T x_i$$

$$C = XW$$

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

$$X \approx XWW^T$$

WW^T is a symmetric <u>projection matrix</u>.
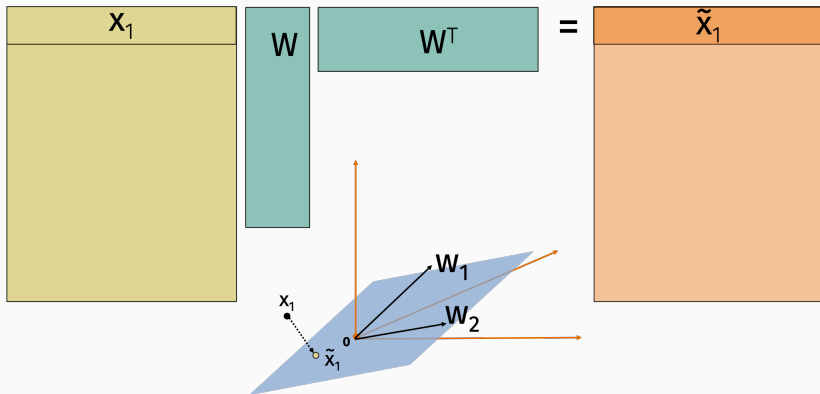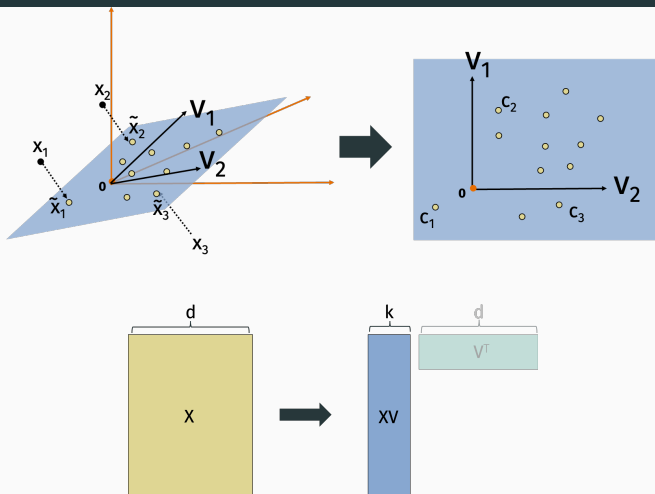
$C = XW$ can be used as a compressed version of data matrix X.

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

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 \approx \|\mathbf{x}_i^T \mathbf{WW}^T - \mathbf{x}_j^T \mathbf{WW}^T\|_2 = \|\mathbf{c}_i - \mathbf{c}_i\|_2$$
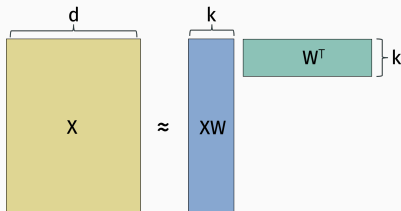
Similarly, we expect that:

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

How does this compare to Johnson-Lindenstrauss projection?

Rows of X (data points) are approximately spanned by k vectors. Columns of X (data features) are approximately spanned by k vectors.

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 $v_1,\ldots,v_{15}$
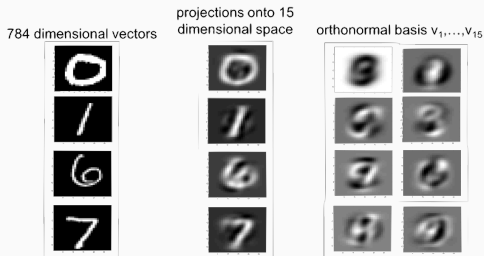
Colinearity/correlation of data features leads to a low-rank data matrix.

	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

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.
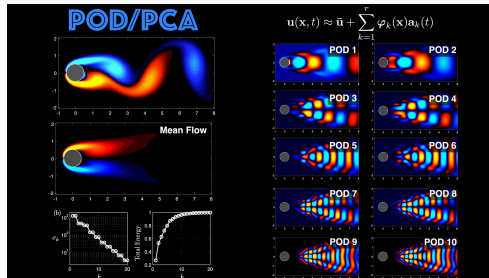- Data visualization when $k = 2$ or 3.
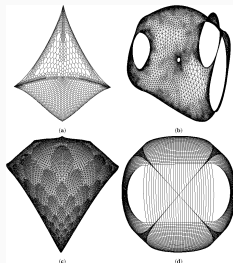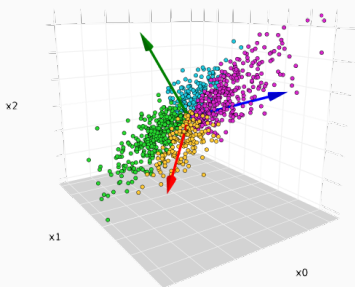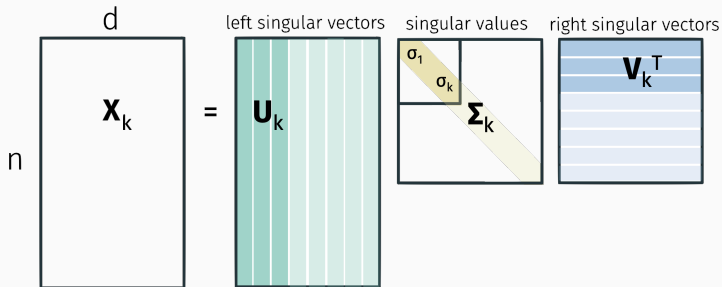

- Data embeddings (next lecture).

- Reduced order modeling for solving physical equations.

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

Can find the best projection from the singular value decomposition.

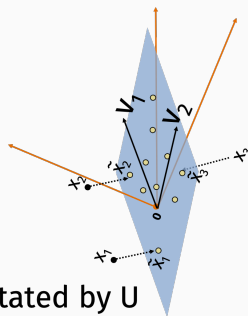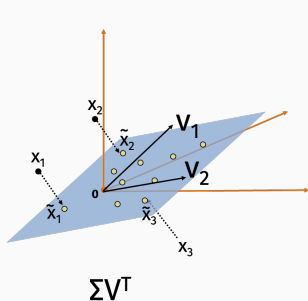$$V_k = \underset{\text{orthogonal } W \in \mathbb{R}^{d \times k}}{\arg\min} \|X - XWW^T\|_F^2$$

Claim: $X_k = U_k \Sigma_k V_k^T = X V_k V_k^T$.

Claim 1:

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

ΣV^T

Rotated by U
on the left

67

Claim 2:

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

Claim 3:

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

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

left singular vectors singular values right singular vectors
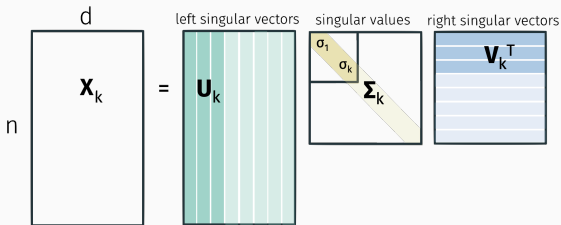
Observation 1:

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

Follows from fact that for <u>all</u> orthogonal W:

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

Claim:

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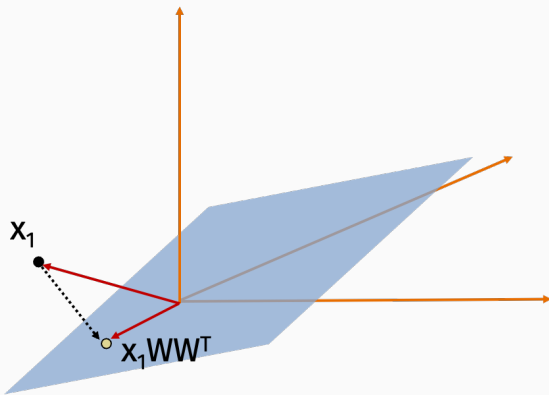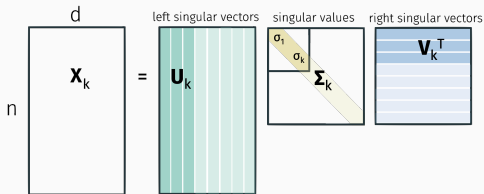
Observation 2: The optimal low-rank approximation error
$E_k = \|X - XV_k V_k^T\|_F^2 = \|X\|_F^2 - \|XV_k V_k^T\|_F^2$ can be written:

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

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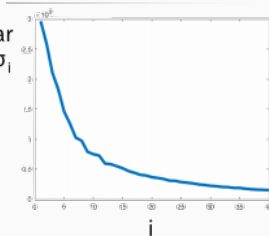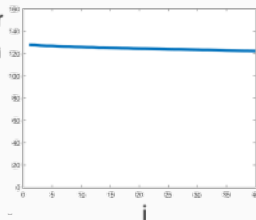
$$E_k = \sum_{i=k+1}^{d} \sigma_i^2.$$

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

784 dimensional vectors

singular value σ_i

i

72

Observation 2: The optimal low-rank approximation error
$E_k = \|X - XV_k V_k^T\|_F^2 = \|X\|_F^2 - \|XV_k V_k^T\|_F^2$ 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:

73

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:

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

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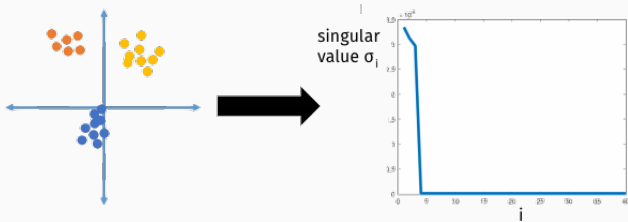from it's spectrum:

singular value σ_i

Suffices to compute right singular vectors V:

- Compute $X^T X$.
- Find eigendecomposition $V \Lambda V^T = X^T X$ using e.g. QR algorithm.
- Compute $L = XV$. Set $\sigma_i = \|L_i\|_2$ and $U_i = L_i / \|L_i\|_2$.

Total runtime $\approx$

computing the svd (faster)

Next class:

- Compute <u>approximate</u> solution.
- Only compute <u>top k singular vectors/values</u>. Runtime will depend on k. When $k = d$ we can't do any better than classical algorithms based on eigendecomposition.
- <u>Iterative algorithms</u> 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.