Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Some Perspective

- We have so far (mostly) looked at what we can do with single numbers (and functions that return single numbers).
- Things can get *much* more interesting once we allow not just one, but *many* numbers together.
- It is natural to view an *array of numbers* as one object with its own rules.
  The simplest such set of rules is that of a vector.
- A 2D array of numbers can also be looked at as a matrix.
- So it's natural to use the tools of computational linear algebra.
- 'Vector' and 'matrix' are just *representations* that come to life in many (*many!*) applications. The purpose of this section is to explore some of those applications.

Vectors

What's a vector?

An array that defines *addition* and *scalar multiplication* with reasonable rules such as

$$\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$$

$$\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$$

$$\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \alpha\mathbf{v}$$

These axioms generally follow from properties of "$+$" and "$\cdot$" operators

Vectors from a CS Perspective

What would the concept of a vector look like in a programming language (e.g. Java)?

In a sense, 'vector' is an *abstract interface*, like this:

```
interface Vector
{
  Vector add(Vector x, Vector y);
  Vector scale(Number alpha, Vector x);
}
```

(Along with guarantees that `add` and `multiply` interact appropriately.)

Vectors in the 'Real World'

Demo: Images as Vectors
Demo: Sound as Vectors
Demo: Shapes as Vectors

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Matrices

What does a matrix do?

It represents a *linear function* between two vector spaces $f : U \to V$ in terms of bases $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ of U and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ of V. Let

$$\boldsymbol{u} = \alpha_1 \boldsymbol{u}_1 + \cdots + \alpha_n \boldsymbol{u}_n$$

and

$$\boldsymbol{v} = \beta_1 \boldsymbol{v}_1 + \cdots + \beta_m \boldsymbol{v}_m.$$

Then f can *always* be represented as a matrix that obtains the βs from the αs:

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_m \end{pmatrix}.$$

Example: The 'Frequency Shift' Matrix

Assume both $\boldsymbol{u}$ and **v** are linear combination of sounds of different frequencies:

$$\boldsymbol{u} = \alpha_1 \boldsymbol{u}_{\text{110 Hz}} + \alpha_2 \boldsymbol{u}_{\text{220 Hz}} + \cdots + \alpha_4 \boldsymbol{u}_{\text{880 Hz}}$$

(analogously for $\boldsymbol{v}$, but with βs). What matrix realizes a 'frequency doubling' of a signal represented this way?

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix}$$

Matrices in the 'Real World'

What are some examples of matrices in applications?

Demo: Matrices for Geometry Transformation
Demo: Matrices for Image Blurring
In-class activity: Computational Linear Algebra

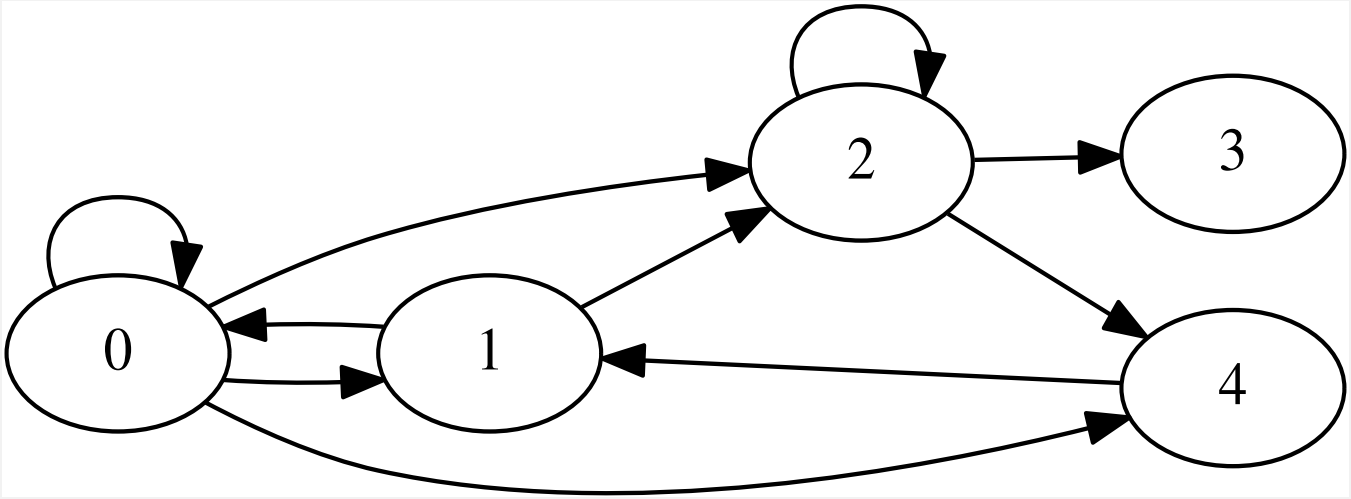
Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Graphs as Matrices

How could this (directed) graph be written as a matrix?



$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & & & & 1 \\ 1 & 1 & 1 & & \\ & & 1 & & \\ 1 & & 1 & & \end{pmatrix}$$

Matrices for Graph Traversal: Technicalities

What is the general rule for turning a graph into a matrix?

If there is an edge from node i to node j, then $A_{ji} = 1$.
(otherwise zero)

What does the matrix for an *undirected* graph look like?

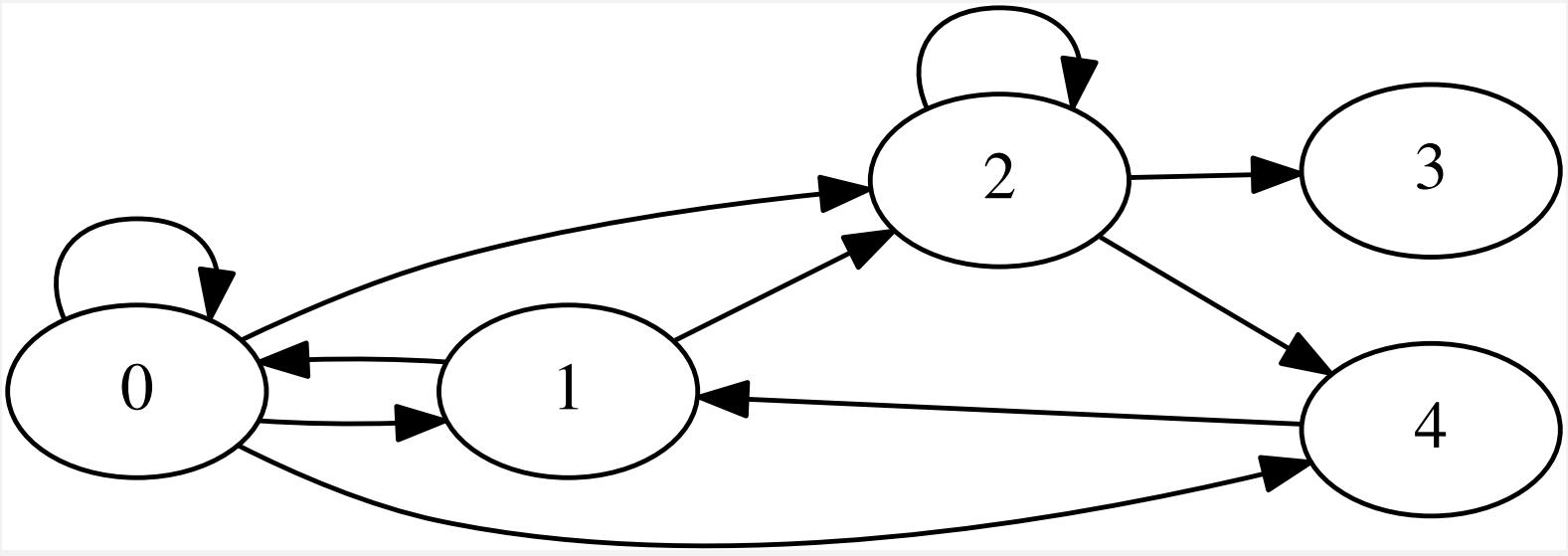
Symmetric.

How could we turn a *weighted graph* (i.e. one where the edges have weights–maybe 'pipe widths') into a matrix?

Allow values other than zero and one for the entries of the matrix.

Graph Matrices and Matrix-Vector Multiplication

If we multiply a graph matrix by the ith unit vector, what happens?



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

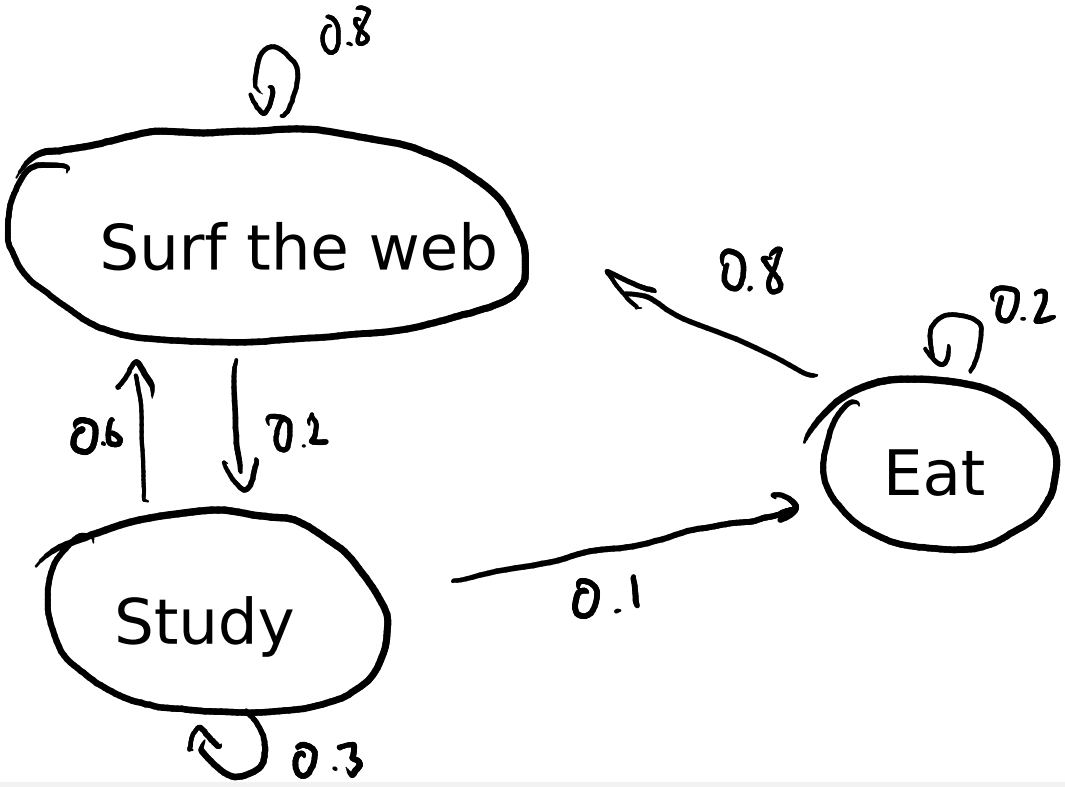
Graph Matrices and Matrix-Vector Multiplication (II)

We get a vector that indicates (with a 1) all the nodes that are reachable from node i.

Demo: Matrices for Graph Traversal

Markov chains

Consider the following graph of states:



Suppose this is an accurate model of the behavior of the average student. :) Can this be described using a matrix?

Markov chains (II)

Important assumption: Only the most recent state matters to determine probability of next state. This is called the Markov property, and the model is called a Markov chain.

Write transition probabilities into matrix as before:
(Order: surf, study, eat–'from' state along columns)

$$A = \begin{pmatrix} .8 & .6 & .8 \\ .2 & .3 & 0 \\ 0 & .1 & .2 \end{pmatrix}$$

Observe: Columns add up to 1, to give sensible probability distribution of next states. Given probabilities of states $\boldsymbol{p} = (p_{\text{surf}}, p_{\text{study}}, p_{\text{eat}})$, $A\boldsymbol{p}$ gives us the probabilities after one unit of time has passed.

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Storing Sparse Matrices

Some types of matrices (including graph matrices) contain many zeros.
Storing all those zero entries is wasteful.
How can we store them so that we avoid storing tons of zeros?

- Python dictionaries (easy, but not efficient)
- Using arrays...?

Storing Sparse Matrices Using Arrays

How can we store a sparse matrix using just arrays? For example:

$$\begin{pmatrix} 0 & 2 & 0 & 3 \\ 1 & 4 & & \\ & & 5 & \\ 6 & & & 7 \end{pmatrix}$$

Idea: 'Compressed Sparse Row' ('CSR') format

- Write all non-zero *values* from top-left to bottom-right
- Write down what *column* each value was in
- Write down the index where each *row started*

$$\begin{aligned} \text{RowStarts} &= \begin{pmatrix} 0 & 2 & 4 & 5 & 7 \end{pmatrix} && \text{(zero-based)} \\ \text{Columns} &= \begin{pmatrix} 1 & 3 & 0 & 1 & 2 & 0 & 3 \end{pmatrix} && \text{(zero-based)} \\ \text{Values} &= \begin{pmatrix} 2 & 3 & 1 & 4 & 5 & 6 & 7 \end{pmatrix} \end{aligned}$$

Demo: Sparse Matrices in CSR Format

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors

The 'Undo' Button for Linear Operations: LU

Repeating Linear Operations: Eigenvalues and Steady States

Eigenvalues: Applications

Approximate Undo: SVD and Least Squares

SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Norms

What's a norm?

- A generalization of 'absolute value' to vectors.
- $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}_0^+$, returns a 'magnitude' of the input vector
- In symbols: Often written $\|\boldsymbol{x}\|$.

Define norm.

A function $\|\boldsymbol{x}\| : \mathbb{R}^n \to \mathbb{R}_0^+$ is called a norm if and only if

1. $\|\boldsymbol{x}\| > 0 \Leftrightarrow \boldsymbol{x} \neq \boldsymbol{0}$.
2. $\|\gamma\boldsymbol{x}\| = |\gamma| \, \|\boldsymbol{x}\|$ for all scalars γ.
3. Obeys triangle inequality $\|\boldsymbol{x} + \boldsymbol{y}\| \leqslant \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$

Examples of Norms

What are some examples of norms?

The so-called p-norms:

$$\left\| \begin{pmatrix} x_1 \\ \\ x_n \end{pmatrix} \right\|_p = \sqrt[p]{|x_1|^p + \cdots + |x_n|^p} \quad (p \geqslant 1)$$

$p = 1, 2, \infty$ particularly important

Demo: Vector norms

Norms and Errors

If we're computing a vector result, the error is a vector.
That's not a very useful answer to 'how big is the error'.
What can we do?

Apply a norm!

How? Attempt 1:

$$\text{Magnitude of error} \neq \|\text{true value}\| - \|\text{approximate value}\| \quad \textbf{WRONG!}$$

Attempt 2:

$$\text{Magnitude of error} = \|\text{true value} - \text{approximate value}\|$$

Absolute and Relative Error

What are the absolute and relative errors in approximating the location of Siebel center $(40.114, -88.224)$ as $(40, -88)$ using the 2-norm?

$$\begin{pmatrix} 40.114 \\ -88.224 \end{pmatrix} - \begin{pmatrix} 40 \\ -88 \end{pmatrix} = \begin{pmatrix} 0.114 \\ -.224 \end{pmatrix}$$

Absolute magnitude;

$$\left\| \begin{pmatrix} 40.114 \\ -88.224 \end{pmatrix} \right\|_2 \approx 96.91$$

Absolute error:

$$\left\| \begin{pmatrix} 0.114 \\ -.224 \end{pmatrix} \right\|_2 \approx .2513$$

Relative error:

$$\frac{.2513}{96.91} \approx .00259.$$

But: Is the 2-norm really the right norm here?

Demo: Calculate geographic distances using `tripstance.com`

Matrix Norms

What norms would we apply to matrices?

- Easy answer: '*Flatten*' matrix as vector, use vector norm. This corresponds to an entrywise matrix norm called the Frobenius norm,

$$\|A\|_F := \sqrt{\sum_{i,j} a_{ij}^2}.$$

- However, interpreting matrices as linear functions, what we are really interested in is the maximum amplification of the norm of any vector multiplied by the matrix,

$$\|A\| := \max_{\|\boldsymbol{x}\|=1} \|A\boldsymbol{x}\|.$$

These are called induced matrix norms, as each is associated with a specific vector norm $\|\cdot\|$.

Matrix Norms (II)

- The following are equivalent:

$$\max_{\|\boldsymbol{x}\|\neq\mathbf{0}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\|\boldsymbol{x}\|\neq\mathbf{0}} \left\| A \underbrace{\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}}_{\boldsymbol{y}} \right\| \overset{\|\boldsymbol{y}\|=1}{=} \max_{\|\boldsymbol{y}\|=1} \|A\boldsymbol{y}\| = \|A\| .$$

- Logically, for each vector norm, we get a different matrix norm, so that, e.g. for the vector 2-norm $\|\boldsymbol{x}\|_2$ we get a matrix 2-norm $\|A\|_2$, and for the vector ∞-norm $\|\boldsymbol{x}\|_\infty$ we get a matrix ∞-norm $\|A\|_\infty$.

Demo: Matrix norms

In-class activity: Matrix norms

Properties of Matrix Norms

Matrix norms inherit the vector norm properties:

1. $\|A\| > 0 \Leftrightarrow A \neq \mathbf{0}$.
2. $\|\gamma A\| = |\gamma| \, \|A\|$ for all scalars γ.
3. Obeys triangle inequality $\|A + B\| \leqslant \|A\| + \|B\|$

But also some more properties that stem from our definition:

1. $\|A\boldsymbol{x}\| \leqslant \|A\| \, \|\boldsymbol{x}\|$
2. $\|AB\| \leqslant \|A\| \, \|B\|$ (easy consequence)

Both of these are called submultiplicativity of the matrix norm.

Example: Orthogonal Matrices

What is the 2-norm of an orthogonal matrix?

Linear Algebra recap: For an orthogonal matrix A, $A^{-1} = A^T$.

In other words: $AA^T = A^TA = I$.

Next:

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

where

$$\|A\boldsymbol{x}\|_2 = \sqrt{(A\boldsymbol{x})^T(A\boldsymbol{x})} = \sqrt{\boldsymbol{x}^T(A^TA)\boldsymbol{x}} = \sqrt{\boldsymbol{x}^T\boldsymbol{x}} = \|\boldsymbol{x}\|_2\,,$$

so $\|A\|_2 = 1$.

Conditioning

Now, let's study condition number of solving a linear system

$$A\boldsymbol{x} = \boldsymbol{b}.$$

Input: $\boldsymbol{b}$ with error $\Delta\boldsymbol{b}$,
Output: $\boldsymbol{x}$ with error $\Delta\boldsymbol{x}$.

Observe $A(\boldsymbol{x} + \Delta\boldsymbol{x}) = (\boldsymbol{b} + \Delta\boldsymbol{b})$, so $A\Delta\boldsymbol{x} = \Delta\boldsymbol{b}$.

$$\begin{aligned}
\frac{\text{rel err. in output}}{\text{rel err. in input}} &= \frac{\|\Delta\boldsymbol{x}\| \,/\, \|\boldsymbol{x}\|}{\|\Delta\boldsymbol{b}\| \,/\, \|\boldsymbol{b}\|} = \frac{\|\Delta\boldsymbol{x}\| \, \|\boldsymbol{b}\|}{\|\Delta\boldsymbol{b}\| \, \|\boldsymbol{x}\|} \\
&= \frac{\left\|A^{-1}\Delta\boldsymbol{b}\right\| \, \|A\boldsymbol{x}\|}{\|\Delta\boldsymbol{b}\| \, \|\boldsymbol{x}\|} \\
&\leqslant \left\|A^{-1}\right\| \, \|A\| \, \frac{\|\Delta\boldsymbol{b}\| \, \|\boldsymbol{x}\|}{\|\Delta\boldsymbol{b}\| \, \|\boldsymbol{x}\|} \\
&= \left\|A^{-1}\right\| \, \|A\| \, .
\end{aligned}$$

Conditioning (II)

So we've found an *upper bound* on the condition number. With a little bit of fiddling, it's not too hard to find examples that achieve this bound, i.e. that it is *tight*.

So we've found the condition number of linear system solving, also called the condition number of the matrix A:

$$\operatorname{cond}(A) = \kappa(A) = \|A\| \left\|A^{-1}\right\| .$$

- cond is relative to a given norm. So, to be precise, use

$$\operatorname{cond}_2 \quad \text{or} \quad \operatorname{cond}_\infty .$$

- If A^{-1} does not exist: $\operatorname{cond}(A) = \infty$ by convention.

Demo: Condition number visualized

Demo: Conditioning of 2×2 Matrices

More Properties of the Condition Number

What is $\mathrm{cond}(A^{-1})$?

$$\mathrm{cond}(A^{-1}) = \|A\| \cdot \left\|A^{-1}\right\| = \mathrm{cond}(A).$$

What is the condition number of applying the matrix-vector multiplication $A\boldsymbol{x} = \boldsymbol{b}$? (I.e. now $\boldsymbol{x}$ is the input and $\boldsymbol{b}$ is the output)

Let $B = A^{-1}$.
Then computing $\boldsymbol{b} = A\boldsymbol{x}$ is equivalent to *solving* $B\boldsymbol{b} = \boldsymbol{x}$.
Solving $B\boldsymbol{b} = \boldsymbol{x}$ has condition number
$\mathrm{cond}(B) = \mathrm{cond}(A^{-1}) = \mathrm{cond}(A)$.

So the operation 'multiply a vector by matrix A' has the same condition number as 'solve a linear system with matrix A'.

Matrices with Great Conditioning (Part 1)

Give an example of a matrix that is *very* well-conditioned.
(I.e. has a condition-number that's *good* for computation.)
What is the best possible condition number of a matrix?

Small condition numbers mean *not a lot of error amplification*.
Small condition numbers are good.

The identity matrix I should be well-conditioned:

$$\|I\| = \max_{\|\boldsymbol{x}\|=1} \|I\boldsymbol{x}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{x}\| = 1.$$

It turns out that this is the smallest possible condition number:

$$1 = \|I\| = \left\|A \cdot A^{-1}\right\| \leqslant \|A\| \cdot \left\|A^{-1}\right\| = \kappa(A).$$

Both of these are true for any norm $\|\cdot\|$.

Matrices with Great Conditioning (Part 2)

What is the 2-norm condition number of an orthogonal matrix A?

$$\kappa_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \|A\|_2 \left\|A^T\right\|_2 = 1.$$

That means *orthogonal matrices have optimal conditioning*.
They're very well-behaved in computation.

In-class activity: Matrix Conditioning

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors

The 'Undo' Button for Linear Operations: LU

Repeating Linear Operations: Eigenvalues and Steady States

Eigenvalues: Applications

Approximate Undo: SVD and Least Squares

SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Solving Systems of Equations

Want methods/algorithms to solve linear systems. Starting small, a kind of system that's easy to solve has a ... matrix.

'Triangular' $\rightarrow$ Easy to solve by hand, e.g. given

$$b_1 = a_{11}x_1 + a_{12}x_2$$
$$b_2 = a_{22}x_2$$

just substitute $x_2 = b_2/a_{22}$ into the first equation.
Generally written as upper/lower triangular matrices.

Triangular Matrices

Solve

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}.$$

- Solve for x_4 in $a_{44}x_4 = b_4$, so $x_4 = b_4/a_{44}$.
- Then solve (recurse)

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ & a_{22} & a_{23} \\ & & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 - a_{14}x_4 \\ b_2 - a_{24}x_4 \\ b_3 - a_{34}x_4 \end{pmatrix}.$$

- This process is called back-substitution.
- The analogous process for lower triangular matrices is called forward-substitution.

Demo: Back-substitution

In-class activity: Forward-substitution

General Matrices

What about non-triangular matrices?

Perform Gaussian Elimination, also known as LU factorization

Given $n \times n$ matrix A, obtain lower triangular matrix L and upper triangular matrix U such that $A = LU$.

Is there some redundancy in this representation?

Yes, the number of entries in a triangular matrix is $(n+1)\frac{n}{2} > \frac{n^2}{2}$. So, by convention we constrain L to have unit diagonal, so $L_{ii} = 1$ for all i. Then we have n^2 nontrivial values in L, U.

Using LU Decomposition to Solve Linear Systems

Given $A = LU$, how do we solve $A\boldsymbol{x} = \boldsymbol{b}$?

$$
\begin{aligned}
A\boldsymbol{x} &= \boldsymbol{b} \\
L\underbrace{U\boldsymbol{x}}_{\boldsymbol{y}} &= \boldsymbol{b} \\
L\boldsymbol{y} &= \boldsymbol{b} \quad \leftarrow \quad \text{solvable by fwd. subst.} \\
U\boldsymbol{x} &= \boldsymbol{y} \quad \leftarrow \quad \text{solvable by bwd. subst.}
\end{aligned}
$$

Now $\boldsymbol{x}$ is a solution to $A\boldsymbol{x} = \boldsymbol{b}$.

2-by-2 LU Factorization (Gaussian Elimination)

Lets consider an example for $n = 2$.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

First, we can observe

$$\begin{bmatrix} a_{11} & a_{12} \end{bmatrix} = 1 \cdot \begin{bmatrix} u_{11} & u_{12} \end{bmatrix},$$

so the first row of U is just the first row of A.

Second, we notice $a_{21} = l_{21} \cdot u_{11}$, so $l_{21} = a_{21}/u_{11}$.

Lastly, we just need to get u_{22}, which participates in the final equation,

$$a_{22} = l_{21} \cdot u_{12} + 1 \cdot u_{22}$$

thus we are left with $u_{22} = a_{22} - l_{21}u_{12}$.

General LU Factorization (Gaussian Elimination)

$$A = \begin{bmatrix} a_{11} & \mathbf{a_{12}} \\ \mathbf{a_{21}} & A_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \mathbf{l_{21}} & L_{22} \end{bmatrix} \cdot \begin{bmatrix} u_{11} & \mathbf{u_{12}} \\ 0 & U_{22} \end{bmatrix}$$

First, we can observe

$$\begin{bmatrix} a_{11} & \mathbf{a_{12}} \end{bmatrix} = 1 \cdot \begin{bmatrix} u_{11} & \mathbf{u_{12}} \end{bmatrix},$$

so the first row of U is just the first row of A.

Second, we notice $\mathbf{a_{21}} = \mathbf{l_{21}} \cdot u_{11}$, so $\mathbf{l_{21}} = \mathbf{a_{21}}/u_{11}$.

To get L_{22} and U_{22}, we use the equation,

$$A_{22} = \mathbf{l_{21}} \cdot \mathbf{u_{12}} + L_{22} \cdot U_{22}.$$

To solve, perform the Schur complement update and 'recurse',

$$[L_{22}, U_{22}] = \text{LU-decomposition}(A_{22} - \underbrace{\mathbf{l_{21}} \cdot \mathbf{u_{12}}}_{\text{Schur complement}})$$

Demo: Gaussian Elimination

LU: Failure Cases?

Is LU/Gaussian Elimination bulletproof?

No, the process can break, try performing LU on $A = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}$.

Q: Is this a problem with the process or with the entire *idea* of LU?

$$\begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \ell_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix}$$

We observe that

$$\begin{pmatrix} 1 & \\ \ell_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix} \rightarrow u_{11} = 0$$

and yet simultaneously $\underbrace{u_{11} \cdot \ell_{21}}_{0} + 1 \cdot 0 = 2$

It turns out to be that A doesn't *have* an LU factorization.

LU: Failure Cases? (II)

What can be done to get something *like* an LU factorization?

Idea: In Gaussian elimination: simply swap rows, equivalent linear system.

Approach:

- Good Idea: Swap rows if there's a zero in the way
- Even better Idea: Find the largest entry (by absolute value), swap it to the top row.

The entry we divide by is called the pivot.
Swapping rows to get a bigger pivot is called (partial) pivoting.

Partial Pivoting Example

Lets try to get a pivoted LU factorization of the matrix

$$A = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}.$$

Start by swapping the two rows

$$\bar{A} = PA = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}.$$

P is a permutation matrix,

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Now proceed as usual with the Gaussian elimination on $\bar{A}$,

$$\bar{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}}_{U}.$$

Partial Pivoting Example (II)

Thus, we obtained a pivoted LU factorization,

$$PA = LU.$$

Written differently, we have

$$A = P^T LU.$$

To solve a linear system $A\mathbf{x} = \mathbf{b}$, it suffices to compute

$$\mathbf{x} = \underbrace{U^{-1}}_{\text{bwd. subs.}} \cdot \underbrace{L^{-1}}_{\text{fwd. subs.}} \cdot \underbrace{P}_{\text{permute}} \cdot \mathbf{b}.$$

Permutation Matrices

How do we capture 'row swaps' in a factorization?

$$\underbrace{\begin{pmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{pmatrix}}_{P} \begin{pmatrix} A & A & A & A \\ B & B & B & B \\ C & C & C & C \\ D & D & D & D \end{pmatrix} = \begin{pmatrix} A & A & A & A \\ C & C & C & C \\ B & B & B & B \\ D & D & D & D \end{pmatrix}.$$

P is called a permutation matrix.

Q: What's P^{-1}?

General LU Partial Pivoting

What does the overall process look like?

1. pivot row with largest leading entry to top,

$$\bar{A} = P_1 A = \begin{bmatrix} \bar{a}_{11} & \bar{\mathbf{a}}_{\mathbf{12}} \\ \bar{\mathbf{a}}_{\mathbf{21}} & \bar{A}_{22} \end{bmatrix}$$

2. the top row of $\bar{A}$ is the top row of U
3. compute $\bar{\mathbf{l}}_{\mathbf{21}}$ by dividing $\bar{\mathbf{a}}_{\mathbf{21}}$ by $\bar{a}_{11}$
4. perform Schur complement update and recurse, get

$$\bar{P}(\bar{A}_{22} - \bar{\mathbf{l}}_{\mathbf{21}}\mathbf{u}_{\mathbf{12}}) = L_{22}U_{22}$$

5. permute the first column of L, $\mathbf{l}_{\mathbf{21}} = \bar{P}\bar{\mathbf{l}}_{\mathbf{21}}$
6. combine permutations $P = \begin{bmatrix} 1 & \\ & \bar{P} \end{bmatrix} P_1$, so $PA = LU$

Computational Cost

What is the computational cost of multiplying two $n \times n$ matrices?

$O(n^3)$

More precisely, we have n accumulated outer products with n^2 additions and multiplications, so to leading order the cost is $2n^3$.

What is the computational cost of carrying out LU factorization on an $n \times n$ matrix?

$O(n)$ cost to form $\mathbf{l_{21}}$

$O(n^2)$ to perform Schur complement update $\mathbf{l_{21}u_{12}}$

Overall $O(n^3)$ since we continue for n steps

More precisely, we have n outer products of decreasing size,

$$\sum_{i=1}^{n} 2i^2 \approx 2n^3/3.$$

More cost concerns

What's the cost of solving $A\boldsymbol{x} = \boldsymbol{b}$?

LU: $O(n^3)$
FW/BW Subst: $2 \times O(n^2) = O(n^2)$

What's the cost of solving $A\boldsymbol{x_1} = \boldsymbol{b_1}, \ldots, A\boldsymbol{x_n} = \boldsymbol{b_n}$?

LU: $O(n^3)$
FW/BW Subst: $2n \times O(n^2) = O(n^3)$

What's the cost of finding A^{-1}?

Same as solving

$$AX = I,$$

so still $O(n^3)$.

LU: Rectangular Matrices

Can we compute LU of an $m \times n$ rectangular matrix?

Yes, two cases:

- $m > n$ (tall and skinny): $L : m \times n$, $U : n \times n$
- $m < n$ (short and fat): $L : m \times m$, $U : m \times n$

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Eigenvalue Problems: Setup/Math Recap

A is an $n \times n$ matrix.

- $\boldsymbol{x} \neq \mathbf{0}$ is called an eigenvector of A if there exists a λ so that

$$A\boldsymbol{x} = \lambda\boldsymbol{x}.$$

- In that case, λ is called an eigenvalue.
- By this definition if $\boldsymbol{x}$ is an eigenvector then so is $\alpha\boldsymbol{x}$, therefore we will usually seek normalized eigenvectors, so $\|\boldsymbol{x}\|_2 = 1$.

Finding Eigenvalues

How do you find eigenvalues?

Linear Algebra approach:

$$\begin{aligned} & A\boldsymbol{x} = \lambda \boldsymbol{x} \\ \Leftrightarrow \quad & (A - \lambda I)\boldsymbol{x} = 0 \\ \Leftrightarrow \quad & A - \lambda I \text{ is singular} \\ \Leftrightarrow \quad & \det(A - \lambda I) = 0 \end{aligned}$$

$\det(A - \lambda I)$ is called the characteristic polynomial, which has degree n, and therefore n (potentially complex) roots.

Q: Does that help computationally?
A: Abel showed that for $n \geqslant 5$ there is no general formula for the roots of the polynomial. (i.e. no analog to the quadratic formula for $n = 5$)

Finding Eigenvalues (II)

Algorithmically, that means we will need to *approximate*. So far (e.g. for LU and QR), if it had not been for FP error, we would have obtained *exact* answers. For eigenvalue problems, that is no longer true–we can only hope for an *approximate* answer.

Distinguishing eigenvectors

Assume we have normalized eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ with eigenvalues $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Show that the eigenvectors are linearly-independent.

We'd like to show that if

$$\mathbf{0} = \alpha_1 \boldsymbol{x}_1 + \cdots + \alpha_n \boldsymbol{x}_n$$

each $\alpha_i = 0$. Intuitively, we can see that if we multiply the expression by A, the $\boldsymbol{x}_1$ component would grow faster than others:

$$\lim_{k \to \infty} \|(1/\lambda_1^k) A^k (\alpha_1 \boldsymbol{x}_1 + \cdots + \alpha_n \boldsymbol{x}_n)\| = \alpha_1 = 0.$$

We can then apply the same argument for α_2, etc.

Diagonalizability

If we have n eigenvectors with different eigenvalues, the matrix is diagonalizable.

Define a matrix whose columns are the eigenvectors

$$X = \begin{pmatrix} | & & | \\ \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_n \\ | & & | \end{pmatrix},$$

and observe

$$AX = \begin{pmatrix} | & & | \\ \lambda_1 \boldsymbol{x}_1 & \cdots & \lambda_n \boldsymbol{x}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_n \\ | & & | \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

This corresponds to a similarity transform

$$AX = XD \quad \Leftrightarrow \quad A = XDX^{-1},$$

Diagonalizability (II)

where D is a diagonal matrix with the eigenvalues.

In that sense: "Diagonalizable" $=$ "Similar to a diagonal matrix".

Are all Matrices Diagonalizable?

Give characteristic polynomial, eigenvalues, eigenvectors of

$$\begin{pmatrix} 1 & 1 \\ & 1 \end{pmatrix}.$$

CP: $(1-\lambda)^2$

Eigenvalues: 1 (with multiplicity 2)

Eigenvectors:

$$\begin{pmatrix} 1 & 1 \\ & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$$

$\Rightarrow x + y = x \Rightarrow y = 0$. So all eigenvectors must look like $\begin{pmatrix} x \\ 0 \end{pmatrix}$.

Eigenvector matrix X won't be invertible. $\rightarrow$ This matrix is *not diagonalizable*!

Power Iteration

We can use linear-independence to find the eigenvector with the largest eigenvalue. Consider the eigenvalues of A^{1000}.

Now, define for example $\boldsymbol{x} = \alpha\boldsymbol{x}_1 + \beta\boldsymbol{x}_2$, so

$$\boldsymbol{y} = A^{1000}(\alpha\boldsymbol{x}_1 + \beta\boldsymbol{x}_2) = \alpha\lambda_1^{1000}\boldsymbol{x}_1 + \beta\lambda_2^{1000}\boldsymbol{x}_2$$

and observe

$$\frac{\boldsymbol{y}}{\lambda_1^{1000}} = \alpha\boldsymbol{x}_1 + \beta \underbrace{\left(\underbrace{\frac{\lambda_2}{\lambda_1}}_{<1} \right)^{1000}}_{\ll 1} \boldsymbol{x}_2.$$

Idea: Use this as a computational procedure to find $\boldsymbol{x}_1$.
Called Power Iteration.

Power Iteration: Issues?

What could go wrong with Power Iteration?

- Starting vector has no component along $\boldsymbol{x}_1$
  Not a problem in practice: Rounding will introduce one.
- Overflow in computing λ_1^{1000}
  $\rightarrow$ Normalized Power Iteration
- $\lambda_1 = \lambda_2$
  Real problem.

What about Eigenvalues?

Power Iteration generates eigenvectors. What if we would like to know eigenvalues?

Estimate them:

$$\frac{\boldsymbol{x}^T A \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}$$

- $= \lambda$ if $\boldsymbol{x}$ is an eigenvector w/ eigenvalue λ
- Otherwise, an estimate of a 'nearby' eigenvalue

This is called the Rayleigh quotient.

Convergence of Power Iteration

What can you say about the convergence of the power method? Say $\boldsymbol{v}_1^{(k)}$ is the kth estimate of the eigenvector $\boldsymbol{x}_1$, and

$$e_k = \left\| \boldsymbol{x}_1 - \boldsymbol{v}_1^{(k)} \right\|.$$

Easy to see:

$$e_{k+1} \approx \frac{|\lambda_2|}{|\lambda_1|} e_k.$$

We will later learn that this is linear convergence. It's quite slow.

What does a shift do to this situation?

$$e_{k+1} \approx \frac{|\lambda_2 - \sigma|}{|\lambda_1 - \sigma|} e_k.$$

Picking $\sigma \approx \lambda_1$ does not help...

Idea: Invert *and* shift to bring $|\lambda_1 - \sigma|$ into numerator.

Transforming Eigenvalue Problems

Suppose we know that $A\boldsymbol{x} = \lambda\boldsymbol{x}$. What are the eigenvalues of these changed matrices?

Power. $A \to A^k$

$$A^k\boldsymbol{x} = \lambda^k\boldsymbol{x}$$

Shift. $A \to A - \sigma I$

$$(A - \sigma I)\boldsymbol{x} = (\lambda - \sigma)\boldsymbol{x}$$

Inversion. $A \to A^{-1}$

$$A^{-1}\boldsymbol{x} = \lambda^{-1}\boldsymbol{x}$$

Inverse Iteration / Rayleigh Quotient Iteration

Describe inverse iteration.

$$\boldsymbol{x}_{k+1} := (A - \sigma I)^{-1} \boldsymbol{x}_k$$

- Implemented by storing/solving with LU factorization
- Converges to eigenvector for eigenvalue closest to σ, with

$$e_{k+1} \approx \frac{|\lambda_{\text{closest}} - \sigma|}{|\lambda_{\text{second-closest}} - \sigma|} e_k.$$

Describe Rayleigh Quotient Iteration.

Compute $\sigma_k = \boldsymbol{x}_k^T A \boldsymbol{x}_k / \boldsymbol{x}_k^T \boldsymbol{x}_k$ to be the Rayleigh quotient for $\boldsymbol{x}_k$.

$$\boldsymbol{x}_{k+1} := (A - \sigma_k I)^{-1} \boldsymbol{x}_k$$

Demo: Power Iteration and its Variants

In-class activity: Eigenvalue Iterations

Computing Multiple Eigenvalues

All Power Iteration Methods compute one eigenvalue at a time. What if I want *all* eigenvalues?

Two ideas:

1. Deflation: Suppose $A\boldsymbol{v} = \lambda\boldsymbol{v}$ $(\boldsymbol{v} \neq \mathbf{0})$. Let $V = \operatorname{span}\{\boldsymbol{v}\}$. Then

$$\begin{aligned} A: \quad V &\to V \\ V^\perp &\to V \oplus V^\perp \end{aligned}$$

In matrix form

$$A = \underbrace{\begin{pmatrix} | & \\ \boldsymbol{v} & \text{Basis of } V^\perp \\ | & \end{pmatrix}}_{Q_1} \begin{pmatrix} \lambda & * & * & * \\ 0 & * & * & * \\ \vdots & & & \\ 0 & * & * & * \end{pmatrix} Q_1^T.$$

Computing Multiple Eigenvalues (II)

Now call B the shaded part of the resulting matrix

$$\text{eigenvalues of } A = \text{eigenvalues of } B \cup \{\lambda\}.$$

I.e. we've reduced the rest of the problem to finding the eigenvalues of B–which is smaller $\rightarrow$ We have shrunk the problem size, or 'deflated' the problem.

2. Iterate with multiple vectors simultaneously.

Simultaneous Iteration

What happens if we carry out power iteration on multiple vectors simultaneously?

Simultaneous Iteration:

1. Start with $X_0 \in \mathbb{R}^{n \times p}$ $(p \leqslant n)$ with (arbitrary) iteration vectors in columns
2. $X_{k+1} = AX_k$

Problems:

- Needs rescaling
- X increasingly ill-conditioned: all columns go towards $\boldsymbol{x}_1$

Fix: orthogonalize! (using, e.g. Gram-Schmidt)

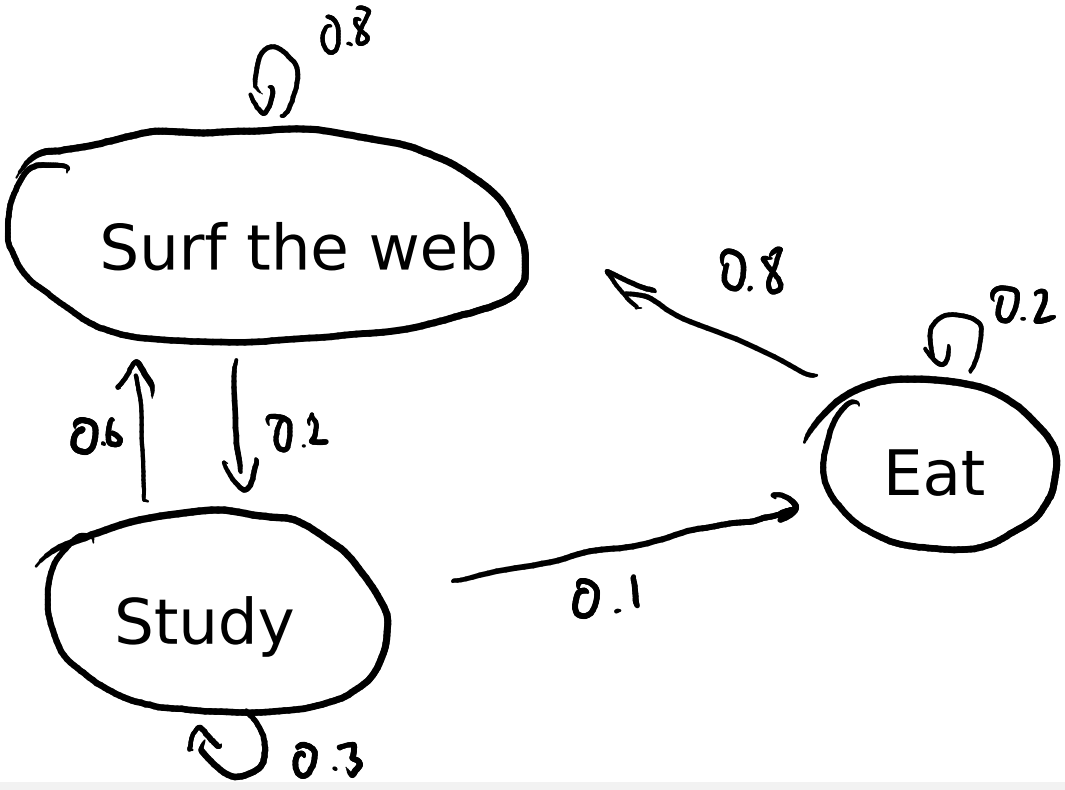
Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Markov chains and Eigenvalue Problems

Recall our example of a Markov chain:



Suppose this is an accurate model of the behavior of the average student. :) How likely are we to find the average student in each of these states?

Write transition probabilities into matrix as before:

Markov chains and Eigenvalue Problems (II)

(Order: surf, study, eat–'from' state along columns)

$$A = \begin{pmatrix} .8 & .6 & .8 \\ .2 & .3 & 0 \\ 0 & .1 & .2 \end{pmatrix}$$

Recall: Columns add up to 1. Given probabilities of states $\boldsymbol{p} = (p_{\text{surf}}, p_{\text{study}}, p_{\text{eat}})$, $A\boldsymbol{p}$ gives us the probabilities after one unit of time has passed.

Idea: Look for a steady state, i.e. $A\boldsymbol{p} = \boldsymbol{p}$.

Phrase as an eigenvalue problem: $A\boldsymbol{p} = \lambda\boldsymbol{p}$.

Demo: Finding an equilibrium distribution using the power method

Understanding Time Behavior

Many important systems in nature are modeled by describing the time rate of change of something.

- E.g. every bird will have 0.2 baby birds on average per year.
- But there are also foxes that eat birds. Every fox present decreases the bird population by 1 birds a year.
  Meanwhile, each fox has 0.3 fox babies a year. And for each bird present, the population of foxes grows by 0.9 foxes for every bird present.

Set this up as equations and see if eigenvalues can help us understand what's going on.

Equation just for birds:

$$\frac{d}{dt}b = 0.2b.$$

Understanding Time Behavior (II)

Equations for birds and foxes:

$$\begin{aligned} \frac{d}{dt}b &= 0.2b - 1f, \\ \frac{d}{dt}f &= 0.9b + .3f. \end{aligned}$$

Shorter, letting the population $\boldsymbol{p} = (b, f)$:

$$\frac{d}{dt}\boldsymbol{p} = \begin{pmatrix} 0.2 & -1 \\ 0.9 & .3 \end{pmatrix} \boldsymbol{p}.$$

Bold (but pretty good) assumption:

$$\boldsymbol{p}(t) = e^{\lambda t}\boldsymbol{p}_0.$$

Then:

$$\lambda \boldsymbol{p}_0 = \begin{pmatrix} 0.2 & -1 \\ 0.9 & .3 \end{pmatrix} \boldsymbol{p}_0.$$

Hey look, an eigenvalue problem! :)

Demo: Understanding the birds and the foxes with eigenvalues

In-class activity: Eigenvalues 2

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Singular Value Decomposition

What is the Singular Value Decomposition ('SVD')?

The SVD is a factorization of an $m \times n$ matrix into

$$A = U\Sigma V^T, \quad \text{where}$$

- U is an $m \times m$ orthogonal matrix
  (Its columns are called 'left singular vectors'.)
- Σ is an $m \times n$ diagonal matrix
  with the singular values on the diagonal

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & & 0 \end{pmatrix} \qquad \text{Convention: } \sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0.$$

- V^T is an $n \times n$ orthogonal matrix
  (V's columns are called 'right singular vectors'.)

Computing the SVD

How can I compute an SVD of a matrix A?

1. Compute the eigenvalues and eigenvectors of $A^T A$.

$$A^T A \boldsymbol{v}_1 = \lambda_1 \boldsymbol{v}_1 \quad \cdots \quad A^T A \boldsymbol{v}_n = \lambda_n \boldsymbol{v}_n$$

2. Make a matrix V from the vectors $\boldsymbol{v}_i$:

$$V = \begin{pmatrix} | & & | \\ \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_n \\ | & & | \end{pmatrix}.$$

($A^T A$ symmetric: V orthogonal if columns have norm 1.)

Computing the SVD (II)

3. Make a diagonal matrix Σ from the square roots of the eigenvalues:

$$\Sigma = \begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_n} \quad 0 \end{pmatrix}$$

4. Find U from

$$A = U\Sigma V^T \quad \Leftrightarrow \quad U\Sigma = AV.$$

(While being careful about non-squareness and zero singular values)
In the simplest case:

$$U = AV\Sigma^{-1}.$$

Computing the SVD (III)

Observe U is orthogonal: (Use: $V^T A^T A V = \Sigma^2$)

$$U^T U = \Sigma^{-1} \underbrace{V^T A^T A V}_{\Sigma^2} \Sigma^{-1} = \Sigma^{-1}\Sigma^2\Sigma^{-1} = I.$$

(Similar for UU^T.)

Demo: Computing the SVD

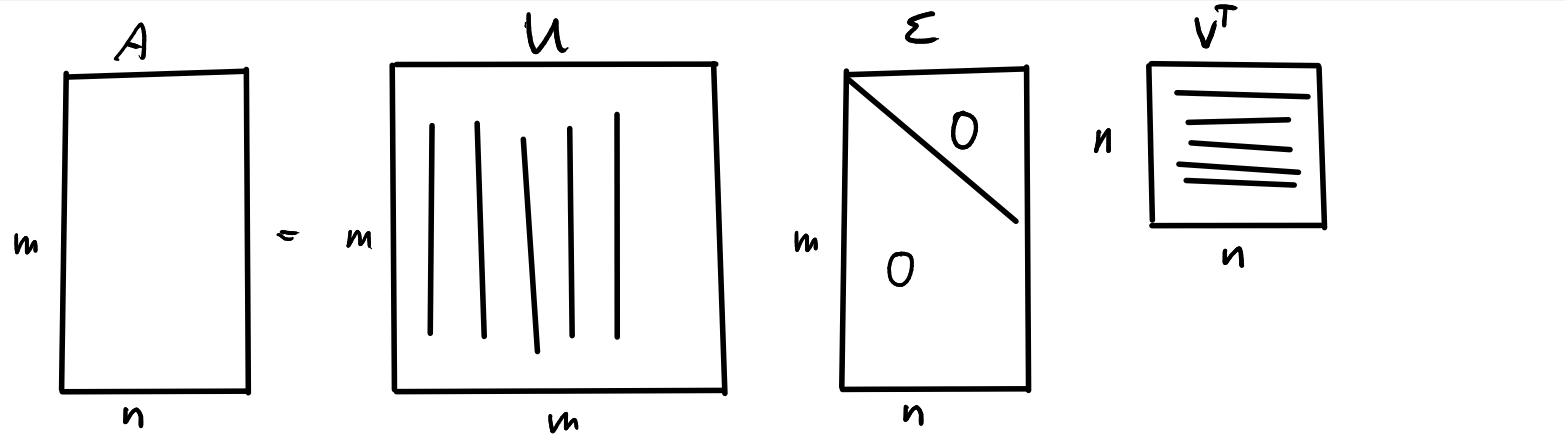
How Expensive is it to Compute the SVD?

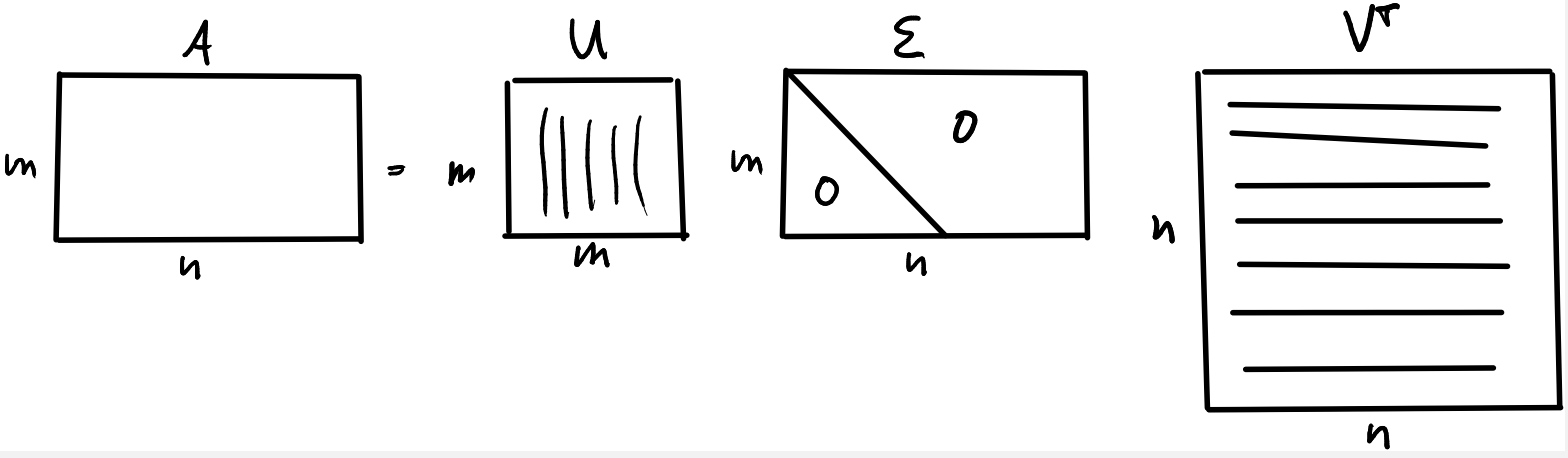
Demo: Relative Cost of Matrix Factorizations

‘Reduced’ SVD

‘Reduced’ SVD (II)

Is there a ‘reduced’ factorization for non-square matrices?





Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Solve Square Linear Systems

Can the SVD $A = U\Sigma V^T$ be used to solve *square* linear systems? At what cost (once the SVD is known)?

Yes, easy:

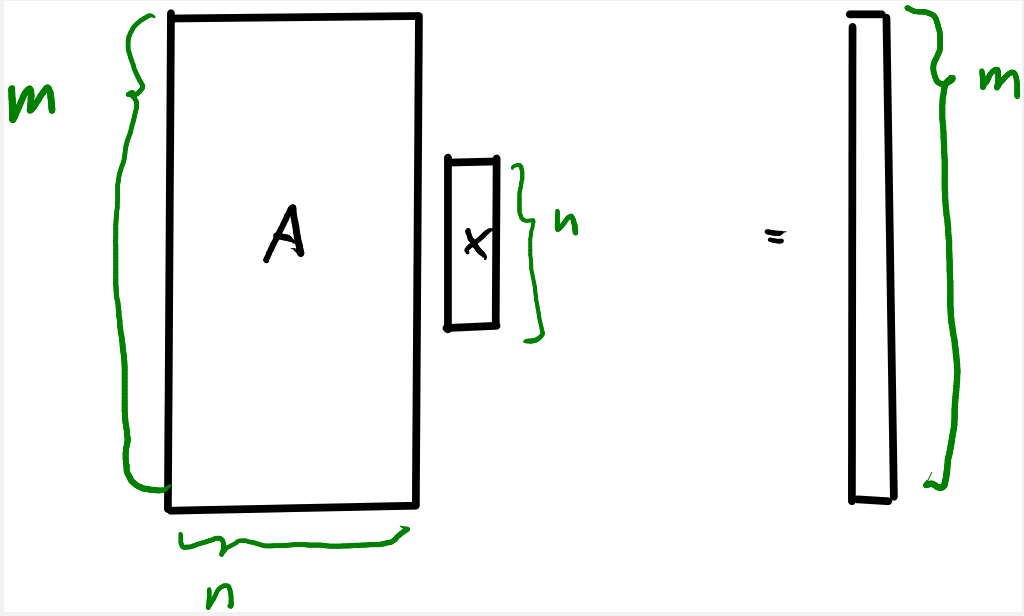
$$\begin{aligned} A\boldsymbol{x} &= \boldsymbol{b} \\ U\Sigma V^T\boldsymbol{x} &= \boldsymbol{b} \\ \Sigma \underbrace{V^T\boldsymbol{x}}_{\boldsymbol{y}:=} &= U^T\boldsymbol{b} \\ \text{(diagonal, easy to solve)} \quad \Sigma\boldsymbol{y} &= U^T\boldsymbol{b} \end{aligned}$$

Know $\boldsymbol{y}$, find $\boldsymbol{x} = V\boldsymbol{y}$.

Cost: $O(n^2)$–but overall much slower than using fw/bw subst. Even worse when including comparison of LU vs. SVD.

Tall and Skinny Systems

Consider a 'tall and skinny' linear system, i.e. one that has more equations than unknowns:



In the figure: $m > n$. How could we solve that?

First realization: A square linear system often only has a single solution. So applying *more* conditions to the solution will mean we have no exact solution.

$$A\boldsymbol{x} = \boldsymbol{b} \quad \leftarrow \quad \text{Not going to happen.}$$

Tall and Skinny Systems (II)

Instead: Find $\boldsymbol{x}$ so that $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ is as small as possible.

$\boldsymbol{r} = A\boldsymbol{x} - \boldsymbol{b}$ is called the residual of the problem.

$$\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 = r_1^2 + \cdots + r_m^2 \quad \leftarrow \quad \text{squares}$$

This is called a (linear) least-squares problem. Since

Find $\boldsymbol{x}$ so that $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ is as small as possible.

is too long to write every time, we introduce a shorter notation:

$$A\boldsymbol{x} \cong \boldsymbol{b}.$$

Solving Least-Squares

How can I actually *solve* a least-squares problem $A\boldsymbol{x} \cong \boldsymbol{b}$?

The job: Make $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ is as small as possible.
Equivalent: Make $\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$ is as small as possible.
Use: The SVD $A = U\Sigma V^T$.

Find $\boldsymbol{x}$ to minimize:

$$
\begin{aligned}
& \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\
= \; & \left\|U\Sigma V^T\boldsymbol{x} - \boldsymbol{b}\right\|_2^2 \\
= \; & \left\|U^T(U\Sigma V^T\boldsymbol{x} - \boldsymbol{b})\right\|_2^2 \qquad \text{(because } U \text{ is orthogonal)} \\
= \; & \left\|\Sigma \underbrace{V^T\boldsymbol{x}}_{\boldsymbol{y}} - U^T\boldsymbol{b}\right\|_2^2 \\
= \; & \left\|\Sigma\boldsymbol{y} - U^T\boldsymbol{b}\right\|_2^2
\end{aligned}
$$

Solving Least-Squares (II)

What $\boldsymbol{y}$ minimizes

$$\left\|\Sigma \boldsymbol{y} - U^T \boldsymbol{b}\right\|_2^2 = \left\| \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & 0 \end{pmatrix} \boldsymbol{y} - \boldsymbol{z} \right\|_2^2 ?$$

Pick

$$y_i = \begin{cases} z_i/\sigma_i & \text{if } \sigma_i \neq 0, \\ 0 & \text{if } \sigma_i = 0. \end{cases}$$

Find $\boldsymbol{x} = V\boldsymbol{y}$, done.

Slight technicality: There only *is* a choice if some of the σ_i are zero. (Otherwise $\boldsymbol{y}$ is uniquely determined.) If there *is* a choice, this $\boldsymbol{y}$ is the one with the smallest 2-norm that *also* minimizes the 2-norm of the residual. And since $\|\boldsymbol{x}\|_2 = \|\boldsymbol{y}\|_2$ (because V is

Solving Least-Squares (III)

orthogonal), $\boldsymbol{x}$ *also* has the smallest 2-norm of all $\boldsymbol{x'}$ for which $\|A\boldsymbol{x}' - \boldsymbol{b}\|_2$ is minimal.

In-class activity: SVD and Least Squares

The Pseudoinverse: A Shortcut for Least Squares

How could the solution process for $A\boldsymbol{x} \cong \boldsymbol{b}$ be with an SVD $A = U\Sigma V^T$ be 'packaged up'?

$$\begin{aligned} U\Sigma V^T\boldsymbol{x} &\approx \boldsymbol{b} \\ \Leftrightarrow \quad \boldsymbol{x} &\approx V\Sigma^{-1}U^T\boldsymbol{b} \end{aligned}$$

Problem: Σ may not be invertible.
Idea 1: Define a 'pseudo-inverse' Σ^+ of a diagonal matrix Σ as

$$\Sigma_i^+ = \begin{cases} \sigma_i^{-1} & \text{if } \sigma_i \neq 0, \\ 0 & \text{if } \sigma_i = 0. \end{cases}$$

Then $A\boldsymbol{x} \cong \boldsymbol{b}$ is solved by $V\Sigma^+U^T\boldsymbol{b}$.

Idea 2: Call $A^+ = V\Sigma^+U^T$ the pseudo-inverse of A.
Then $A\boldsymbol{x} \cong \boldsymbol{b}$ is solved by $A^+\boldsymbol{b}$.

The Normal Equations

You may have learned the 'normal equations' $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$ to solve $A\boldsymbol{x} \cong \boldsymbol{b}$.
Why not use those?

$$\mathrm{cond}(A^T A) \approx \mathrm{cond}(A)^2$$

I.e. if A is even somewhat poorly conditioned, then the conditioning of $A^T A$ will be a disaster.

The normal equations are not well-behaved numerically.

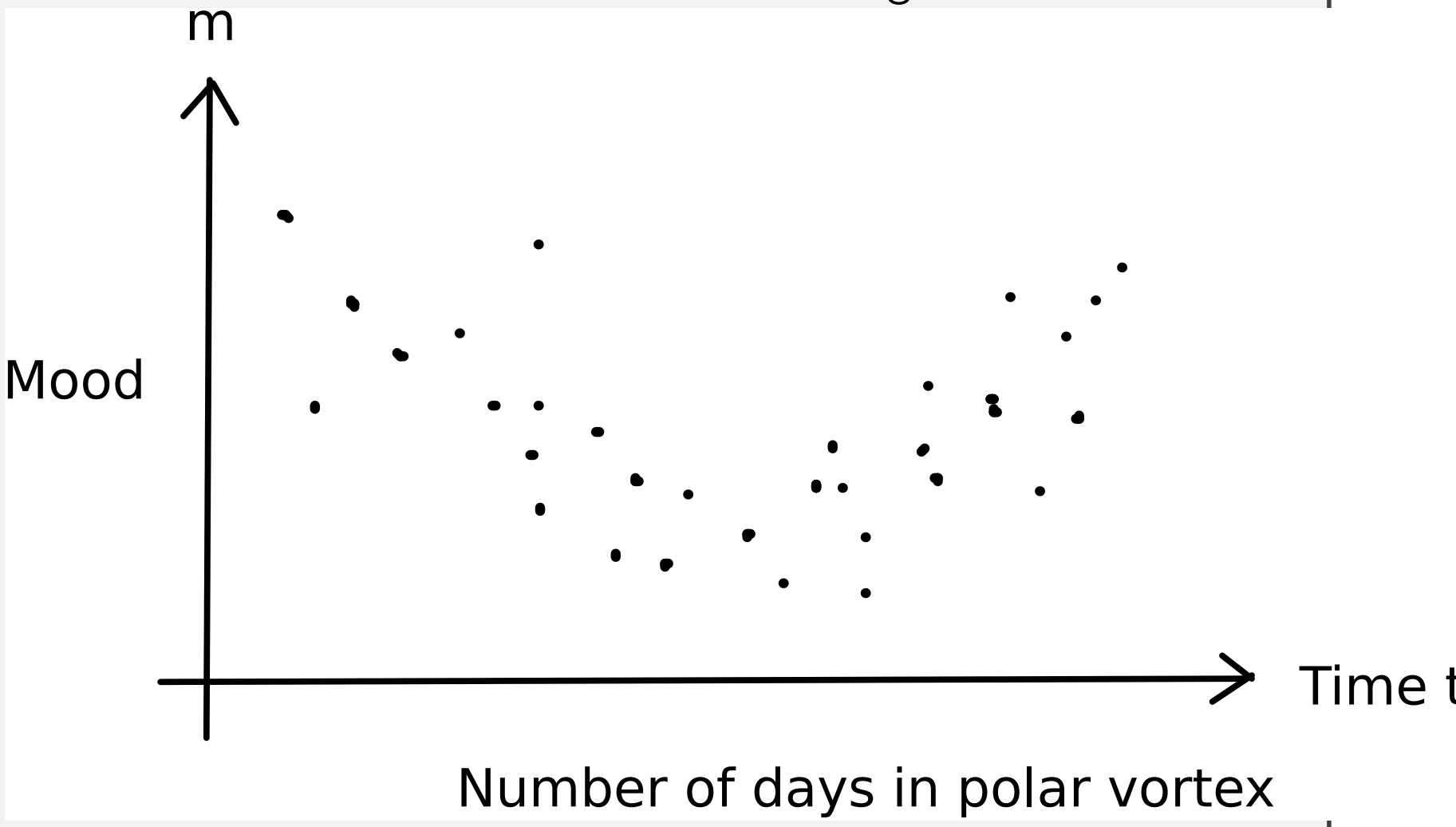
Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Fitting a Model to Data

How can I fit a model to measurements? E.g.:



Fitting a Model to Data (II)

Maybe:

$$\hat{m}(t) = \alpha + \beta t + \gamma t^2$$

Have: 300 data pooints: $(t_1, m_1), \ldots, (t_{300}, m_{300})$
Want: 3 unknowns α, β, γ
Write down equations:

$$\begin{array}{rcl} \alpha + \beta t_1 + \gamma t_1^2 & \approx & m_1 \\ \alpha + \beta t_2 + \gamma t_2^2 & \approx & m_2 \\ \vdots & \vdots & \vdots \\ \alpha + \beta t_{300} + \gamma t_{300}^2 & \approx & m_{300} \end{array} \rightarrow \begin{pmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_{300} & t_{300}^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \cong \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{300} \end{pmatrix}$$

So data fitting is just like interpolation, with a Vandermonde matrix:

$$V\boldsymbol{\alpha} = \boldsymbol{m}.$$

Only difference: More rows. Solvable using the SVD.

Demo: Data Fitting using Least Squares

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

Meaning of the Singular Values

What do the singular values mean? (in particular the first/largest one)

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

$$\|A\|_2 = \max_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2 = \max_{\|\boldsymbol{x}\|_2=1} \left\|U\Sigma V^T\boldsymbol{x}\right\|_2 \overset{U \text{ orth.}}{=} \max_{\|\boldsymbol{x}\|_2=1} \left\|\Sigma V^T\boldsymbol{x}\right\|_2$$

$$\overset{V \text{ orth.}}{=} \max_{\|V^T\boldsymbol{x}\|_2=1} \left\|\Sigma V^T\boldsymbol{x}\right\|_2 \overset{\text{Let } \boldsymbol{y} = V^T\boldsymbol{x}}{=} \max_{\|\boldsymbol{y}\|_2=1} \|\Sigma\boldsymbol{y}\|_2 \overset{\Sigma \text{ diag.}}{=} \sigma_1.$$

So the SVD (finally) provides a way to find the 2-norm.

Entertainingly, it does so by reducing the problem to finding the 2-norm of a diagonal matrix.

$$\|A\|_2 = \sigma_1.$$

Condition Numbers

How would you compute a 2-norm condition number?

$$\operatorname{cond}_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \sigma_1/\sigma_n.$$

Outline

Modeling the World with Arrays
- The World in a Vector
- What can Matrices Do?
- Graphs
- Sparsity

Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations: Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

SVD as Sum of Outer Products

What's another way of writing the SVD?

Starting from (assuming $m > n$ for simplicity)

$$A = U\Sigma V^T = \begin{pmatrix} | & & | \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_m \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \\ & & 0 \end{pmatrix} \begin{pmatrix} - & \boldsymbol{v}_1 & - \\ & \vdots & \\ - & \boldsymbol{v}_n & - \end{pmatrix}$$

we find that

$$\begin{aligned} A &= \begin{pmatrix} | & & | \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_m \\ | & & | \end{pmatrix} \begin{pmatrix} - & \sigma_1\boldsymbol{v}_1 & - \\ & \vdots & \\ - & \sigma_n\boldsymbol{v}_n & - \end{pmatrix} \\ &= \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T + \sigma_2\boldsymbol{u}_2\boldsymbol{v}_2^T + \cdots + \sigma_n\boldsymbol{u}_n\boldsymbol{v}_n^T. \end{aligned}$$

SVD as Sum of Outer Products (II)

That means: The SVD writes the matrix A as a sum of outer products (of left/right singular vectors). What could that be good for?

Low-Rank Approximation (I)

What is the *rank* of $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$?

1. (1 linearly independent column!)

What is the *rank* of $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T$?

2. (2 linearly independent–orthogonal–columns!)

Demo: Image Compression

Low-Rank Approximation

What can we say about the low-rank approximation

$$A_k = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \cdots + \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T$$

to

$$A = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T + \cdots + \sigma_1 \boldsymbol{u}_n \boldsymbol{v}_n^T ?$$

For simplicity, assume $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n > 0$.

Observe that A_k has *rank* k. (And A has rank n.)

Then $\|A - B\|_2$ among all rank-k (or lower) matrices B is *minimized* by A_k.
(Eckart-Young Theorem)

Even better:

$$\min_{\operatorname{rank} B \leqslant k} \|A - B\|_F = \|A - A_k\|_2 = \sigma_{k+1}.$$

Low-Rank Approximation (II)

A_k is called the best rank-k approximation to A.
(where k can be any number)

This best-approximation property is what makes the SVD extremely useful in applications and ultimately justifies its high cost.

It's also the rank-k best-approximation in the Frobenius norm:

$$\min_{\operatorname{rank} B \leqslant k} \|A - B\|_F = \|A - A_k\|_F = \sqrt{\sigma_{k+1}^2 + \cdots \sigma_n^2}.$$