COSC6364
Adv. Numerical Analysis

Panruo Wu, PhD

Spring 2020

What is this course about?

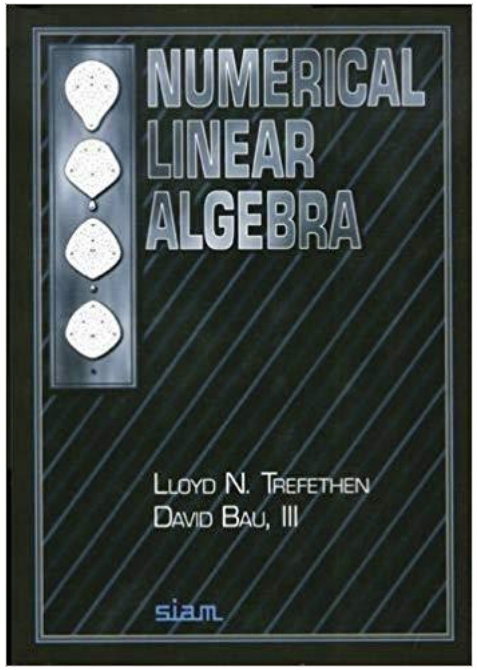
- Basically two major parts:
  - Numerical linear algebra (matrix computations)
  - (Convex) optimization
- The topics are primarily geared towards (large scale) statistical learning (**learning from data**!) and to scientific/engineering computing (**technical computing**).

Tentative Topics for Numerical Linear Algebra

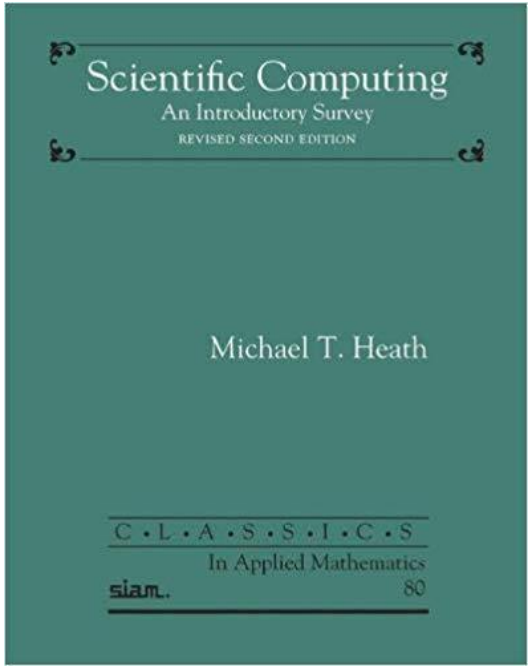
- Matrix multiplication, norms, orthogonality
- Singular value decomposition (SVD)
- QR factorization and least square problems
- LU/Cholesky factorization for linear systems
- QR algorithm and Eigenvalue Decomposition (EVD)
- Iterative methods for solving linear equation and eigen/singular decomposition
- Fast implementation, parallelization, randomization, etc…

Books

- (Highly recommend; main reference):
  Numerical Linear Algebra

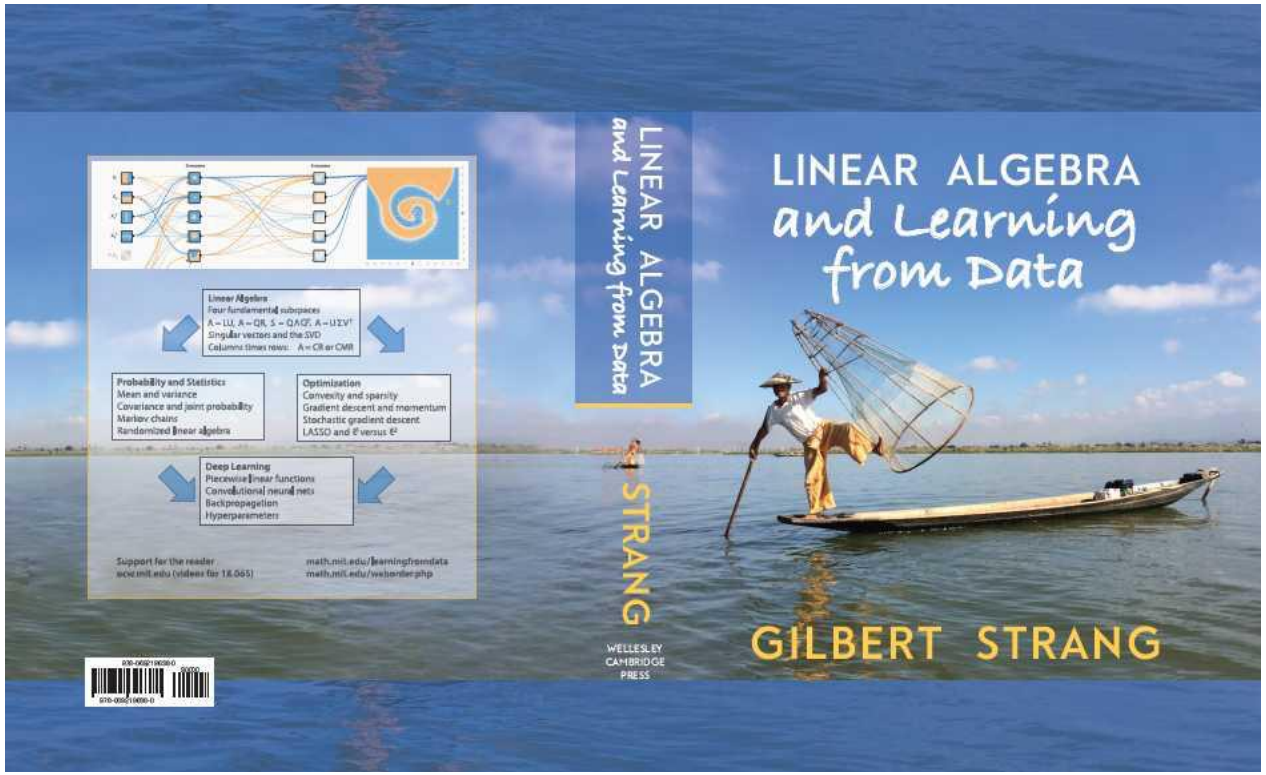


- References:
  Scientific Computing: An introductory Survey

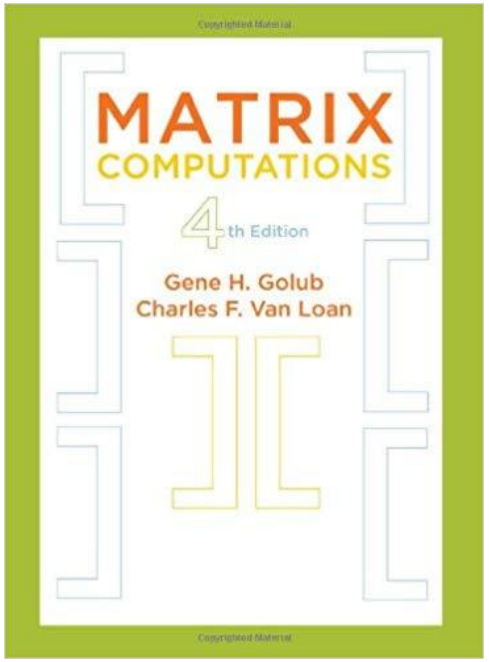


Books

- Linear Algebra and Learning from Data, Gilbert Strang 2019



- Matrix Computations 4th ed (This is kind of encyclopedia of anything related to matrix computations)

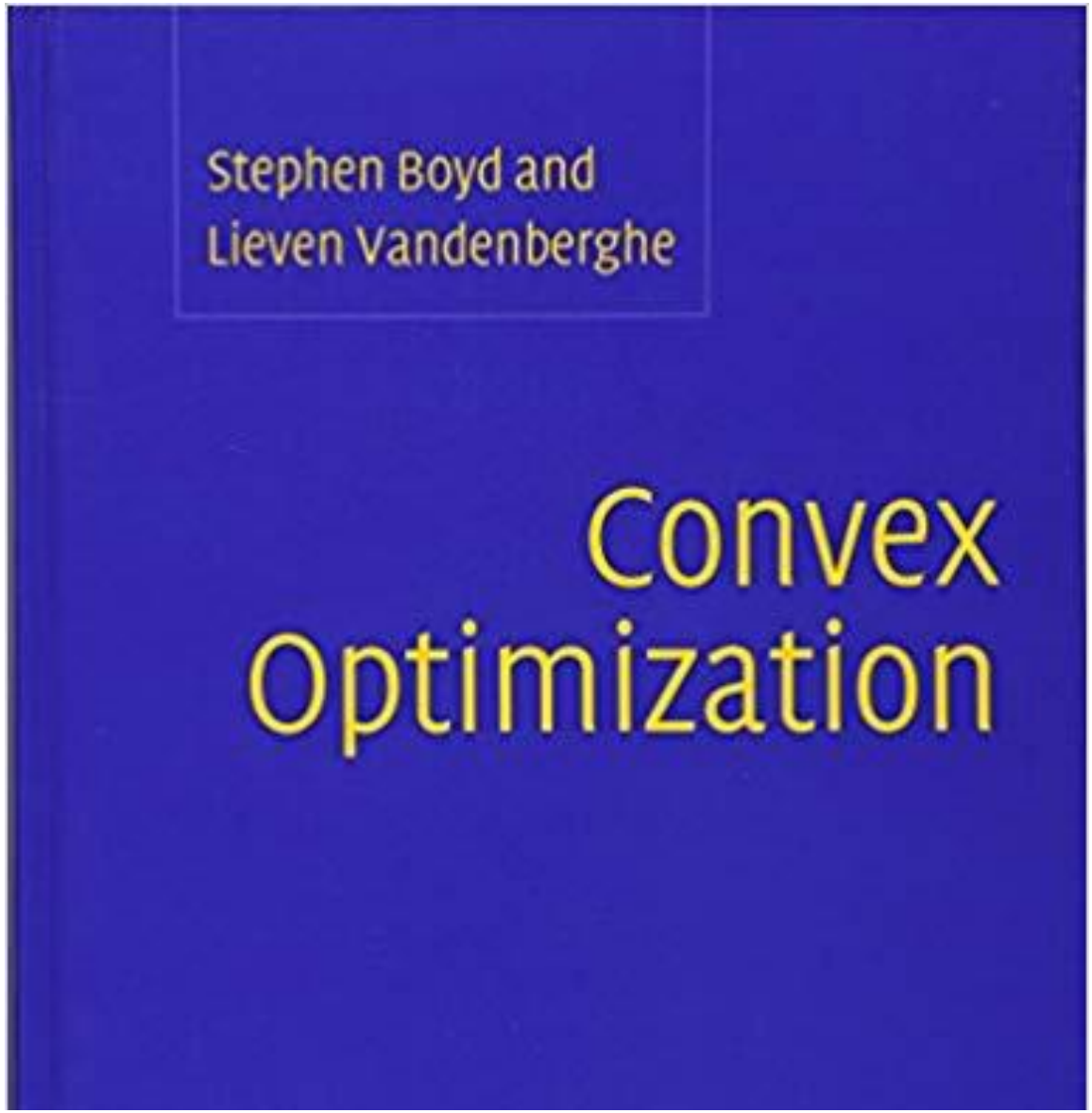


Tentative Topics for Convex Optimization

- Convexity
- Gradient/subgradient descent, proximal GD, stochastic GD
- Duality, KKT conditions
- (Quasi) Newton methods, Interior point methods
- Coordinate descent, dual ascent, Alternating direction method of multipliers (ADMM), etc…

Book

- Convex Optimization, Stephen Boyd & Lieven Vandenbergh
  Free ebook!
  https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf



Objectives: NLA

Upon successfully completion of this course, the students should be able to

- Understand basic matrix computation concepts and algorithms;
- Know how to compute matrix factorization/inversion, linear systems, least square problems, eigen/singular value decomposition, …
- Able to analyze the computational cost of numerical algorithms
- Able to apply different algorithms for different situations; able to diagnose numerical stability and errors

Objectives: Optimization

- Recognize convex optimizations
- Be able to formulate and transform convex optimization problem
- Understand how different optimization algorithms work; under what conditions do they work; the convergence rate; the cost of each iteration.
- Use KKT conditions to solve/characterize optimization problem
- Understand the role of optimization in statistical learning

Why study all these?

- Data science is largely about manipulating matrices/vectors.
- There are four aspects that are important for manipulating large matrices (which are not covered in typical college linear algebra courses):
  - **Speed**: big matrix can be very **slow** to compute.
    - Doing pretty much anything interesting on dense matrix cost about O(n^3) floating point operations; on a large matrix where n~10^5, n^3~10^15 operations ~ 280 hours assuming 10^9 operations/s.
  - **Accuracy**: digital computers do finite precision arithmetic, meaning that each arithmetic (usually) incurs some small error (rounding error). If you do billions of operations does the errors accumulate or amplify?
    - Some algorithms are more stable than the others;
  - **Memory**: the storage is usually O(n^2); again grows fast with size of matrix.
  - **Scalability**: you don't have enough memory space or computing power so you want to use parallel computers. Does more nodes mean faster execution time?
- Because the world is harsh, we'll frequently need to tradeoff between them: <u>but first you need to know and understand them</u>

Can’t I just use a library?

… like Sci-kit learn/<put your favorite framework here>

Probably, but what if

- Too slow
- Low quality result. What’s wrong?
- Variants not implemented (☹)?
- Doing cutting edge research/development?

It’s worth knowing what’s going on under the hood. Plus, you get to study and play with really cool stuff that I found fascinating.

Don’t believe me?

In putting together this issue of *Computing in Science & Engineering*, we knew three things: it would be difficult to list just 10 algorithms; it would be fun to assemble the authors and read their papers; and, whatever we came up with in the end, it would be controversial. We tried to assemble the 10 algorithms with the greatest influence on the development and practice of science and engineering in the 20th century. Following is our list (here, the list is in chronological order; however, the articles appear in no particular order):

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

Another list by Nick Higham

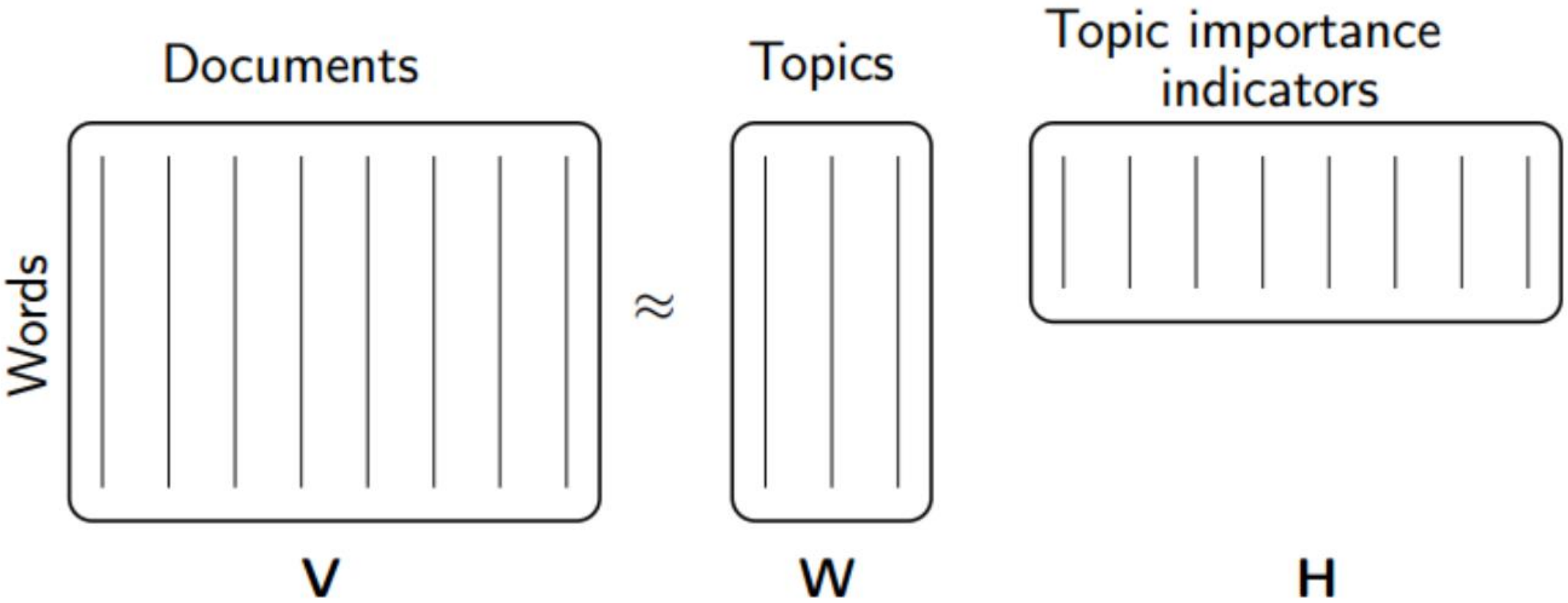
- Newton and quasi-Newton methods
- Matrix factorizations (LU, Cholesky, QR)
- Singular value decomposition, QR and QZ algorithms
- Monte-Carlo methods
- Fast Fourier transform
- Krylov subspace methods (conjugate gradients, Lanczos, GMRES, minres)
- JPEG
- PageRank
- ~~Simplex algo~~rithm
- Kalman filter

Topic Modeling

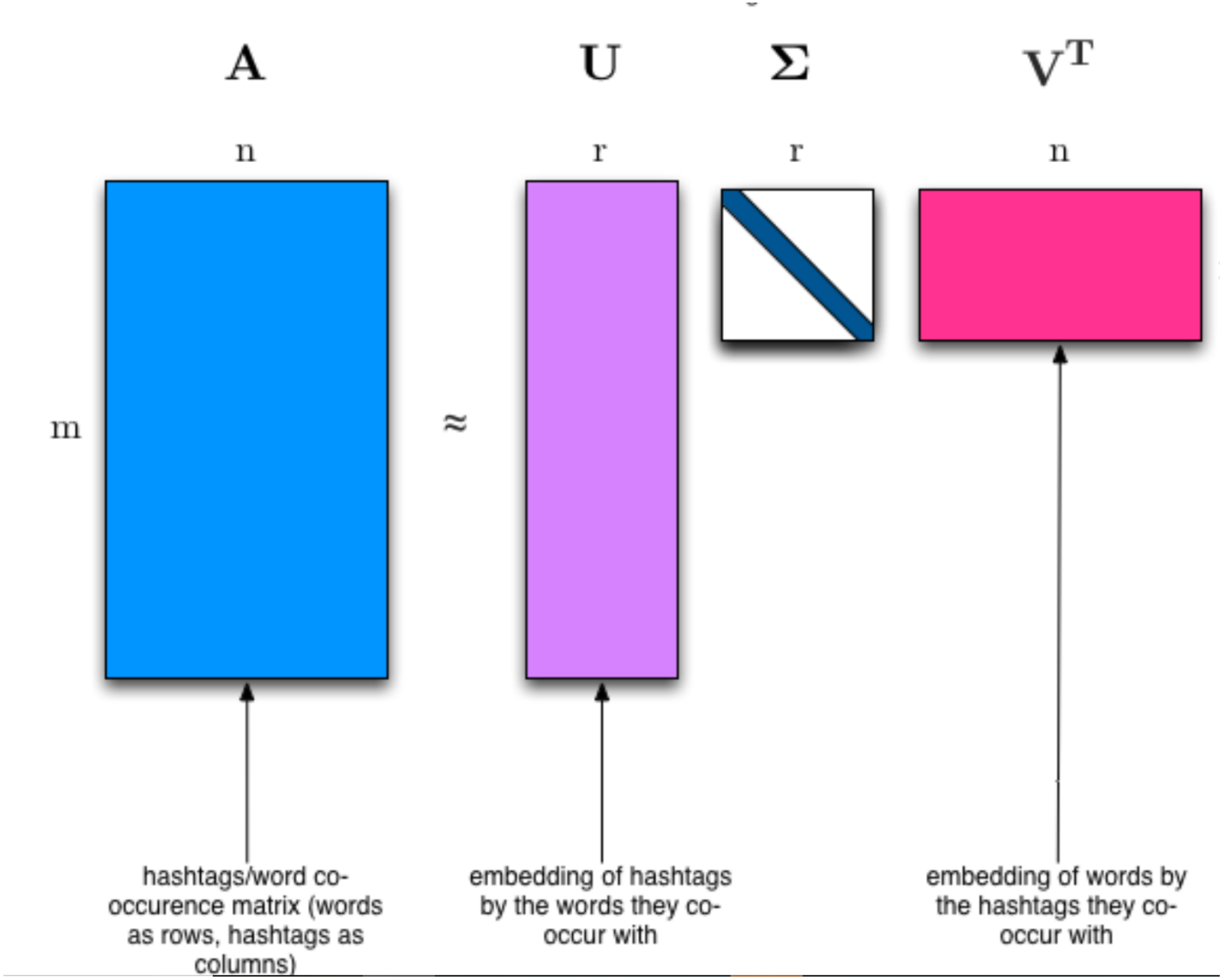
	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5

Bag of words model: term-document matrix.

Topic Modeling by Non-negative Matrix Multiplication



Topic Modeling by Singular Value Decomposition (SVD)



20 Newsgroups Datasets

- http://qwone.com/~jason/20Newsgroups/
- 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups

comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x

misc.forsale

rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey

talk.politics.misc
talk.politics.guns
talk.politics.mideast

sci.crypt
sci.electronics
sci.med
sci.space

talk.religion.misc
alt.atheism
soc.religion.christian

Right singular vectors (top 10 topics, top 8 frequent words)

- ['critus ditto propagandist surname galacticentric kindergarten surreal imaginative',
- 'jpeg gif file color quality image jfif format',
- 'graphics edu pub mail 128 3d ray ftp',
- 'jesus god matthew people atheists atheism does graphics',
- 'image data processing analysis software available tools display',
- 'god atheists atheism religious believe religion argument true',
- 'space nasa lunar mars probe moon missions probes',
- 'image probe surface lunar mars probes moon orbit',
- 'argument fallacy conclusion example true ad argumentum premises',
- 'space larson image theory universe physical nasa material']

Background removal with SVD

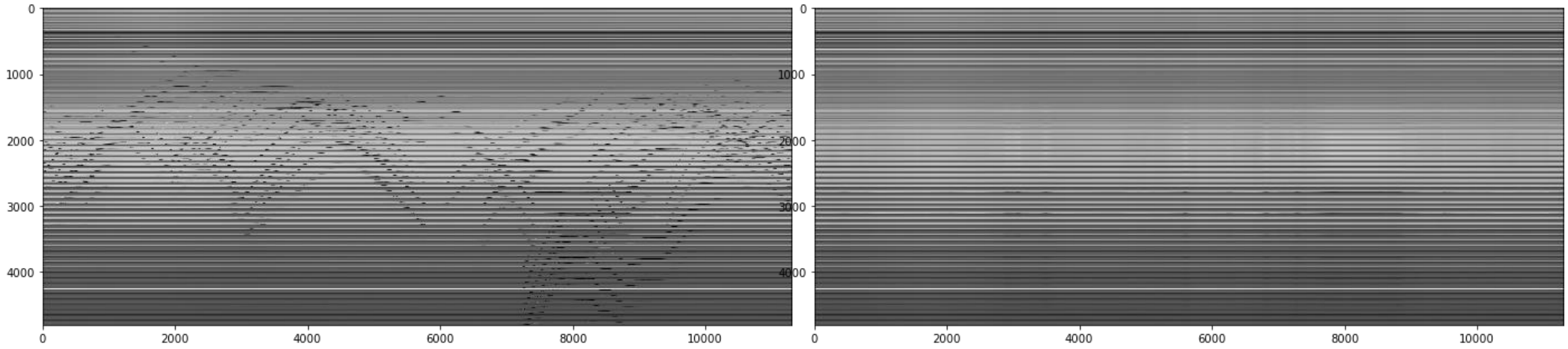
- Given a video recording, how to separate the foreground (moving people) from the background (unmoving stuff)

Representing video as a matrix

- BMC 2012 Dataset Video 003
- 60x80 size; 113s
- How to represent a video using a matrix?
  - Sample 1 frame every 1/10th second
  - Stack the frame into 1 column
  - We get a 4800x1130 matrix!

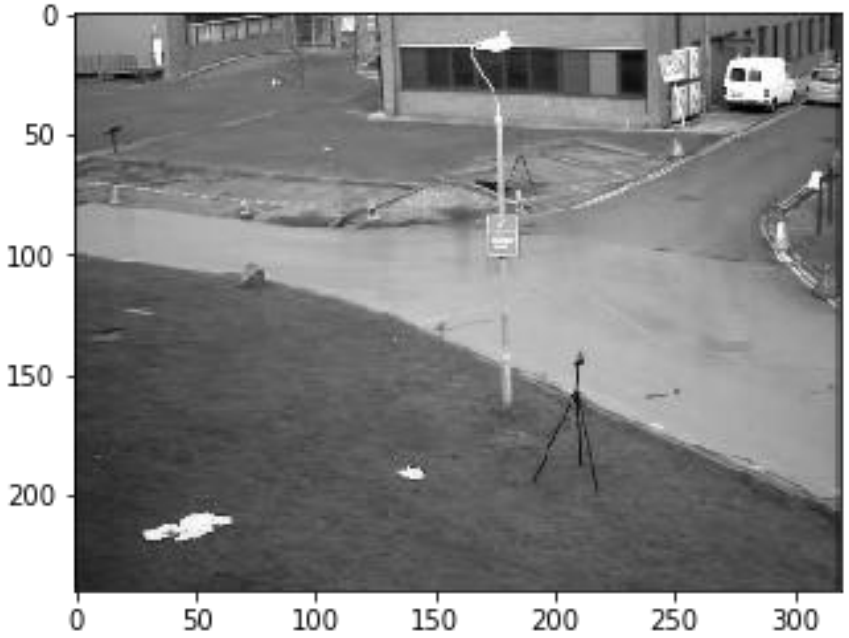
What are the curves?

- Original matrix visualization
- Rank-2 approximation by (randomized) SVD

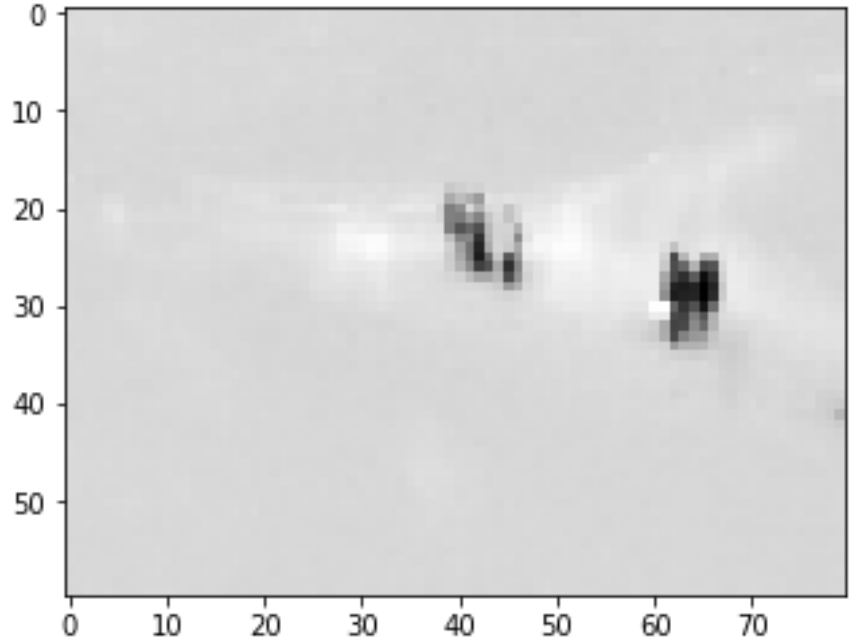


Background removed (one frame)

- Background



- Foreground



rank=10

rank=20

rank=40

rank=80

rank=160

rank=512

Now onto (convex) optimizations…

- Optimization underlies pretty much all (statistical) Machine Learning

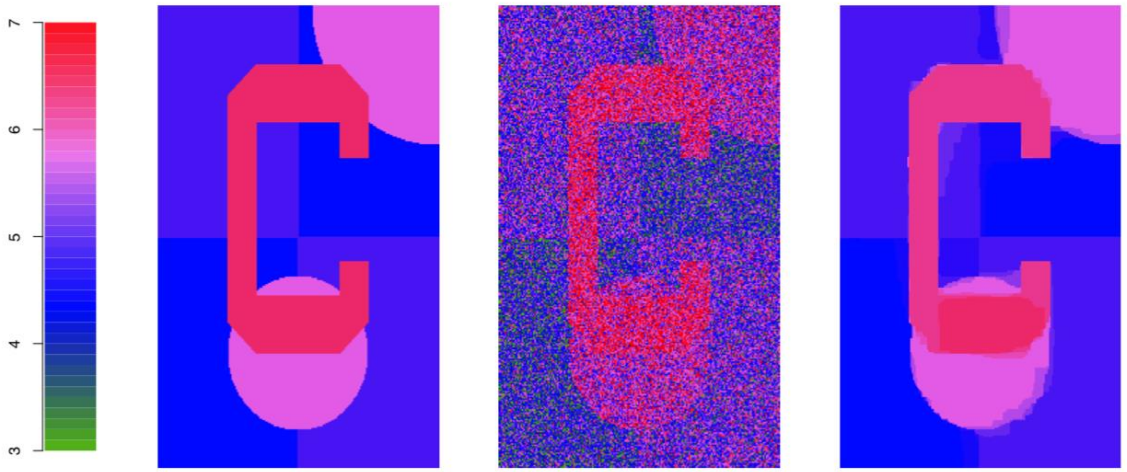
translate  into $P : \min_{x \in D} f(x)$

Conceptual idea *Optimization problem*

How to solve P?
Which algorithms to use?
Can we come up with new P?
…

Denoise: fused 2d lasso

- $\min_{\theta} \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta_i)^2 + \lambda \sum_{(i,j) \in E} |\theta_i - \theta_j|$



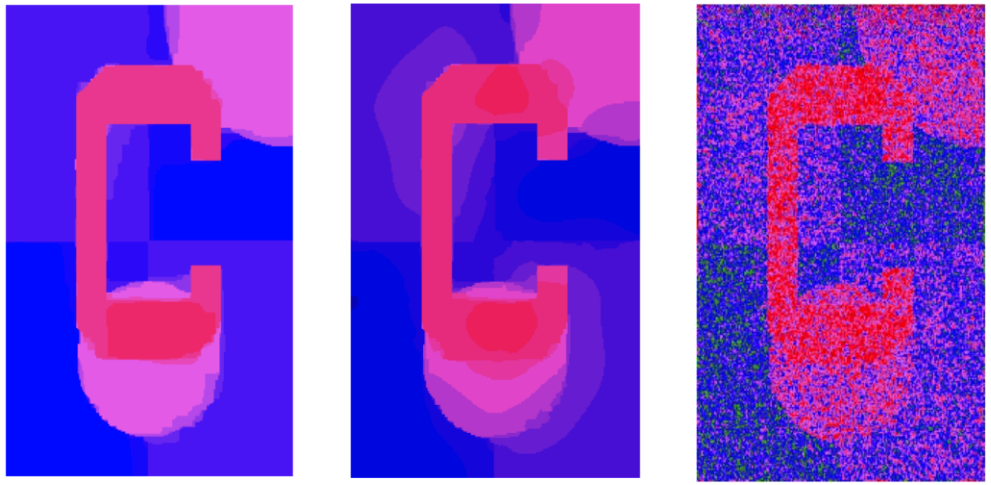


Fig 1: Specialized **ADMM**, 20 iterations
Fig 2: **Proximal gradient descent**, 1000 iterations
Fig 3: **Coordinate descent**, 10K cycles

Different algorithms converge in dramatically different speed. (hint: last two algorithms are from dual problem)

Relationships between NLA and CVX

- NLA serves as basic operations for CVX algorithms
  - matrix multiplication/inversion, matrix factorization
- Many NLA problems can be understood as optimization problems
  - (symmetric) linear equations ⇔ quadratic optimization
  - Many iterative solvers actually solves optimization problem in a low dimension space in every iteration.

Logistics, etc…

- 2 lectures (TuTh 4-5:30pm)
- Office hours of the instructor and TA TBA on Piazza
- 1 homework/2 weeks (40%); submit via Blackboard only
  - Typed documents preferred; photocopy of handwriting is OK too but please make it **legible** (otherwise you will not receive credits)
- Two quizzes, one before Spring break, one at the last lecture (50%)
- Attendance, participation (both online and offline) (10%)
  - Please join online discussion: https://piazza.com/uh/spring2020/cosc6364/home

Honesty

- Discussion is encouraged; but write your own answers **independently**.
- You will receive **negative** points from copied answers in homework/quiz and potential escalations.

Quiz#0

- Available today; please do it at home and compare with the solutions.
- This is to test your background for successful completing this course by your own.
- Review Resources:
  - Linear algebra review, videos by Zico Kolter
  - Real analysis, calculus, and more linear algebra, videos by Aaditya Ramdas
  - Convex optimization prequisites review by Nicole Rafidi

References:

- Numerical Linear Algebra, Lloyd Trefethen, David Bau III.
- Fast.ai, Computational Linear Algebra https://www.fast.ai/2017/07/17/num-lin-alg/
- CMU, Ryan Tibshirani, Convex Optimization 10-725 http://www.stat.cmu.edu/~ryantibs/convexopt/