COSC6364 Adv. Numerical Analysis

Introduction

P. Wu[1]

[1]Computer Science
University of Houston

Spring 2020

Table of Contents

What is this course about?

What is this course about?

- Basically two major parts:

What is this course about?

- Basically two major parts:
  - Numerical linear algebra (matrix computations)

What is this course about?

- Basically two major parts:
  - Numerical linear algebra (matrix computations)
  - (Convex) optimization

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

Tentative Topics for Numerical Linear Algebra

- Matrix multiplication, norms, orthogonality

Tentative Topics for Numerical Linear Algebra

- Matrix multiplication, norms, orthogonality
- Singular value decomposition (SVD)

Tentative Topics for Numerical Linear Algebra

- Matrix multiplication, norms, orthogonality
- Singular value decomposition (SVD)
- QR factorization and least square problems

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

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)

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

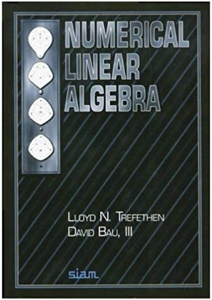
Tentative Topics for Numerical Linear Algebra

- 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

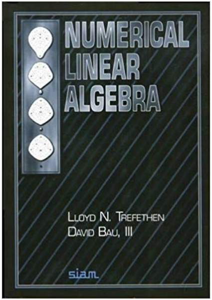
Books

- (Highly recommend; main reference)
  Numerical Linear Algebra (Lloyd N. Trefethen, David Bau)

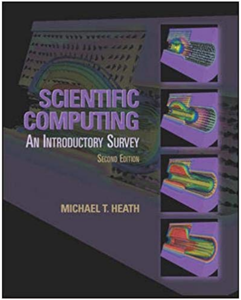


Books

- (Highly recommend; main reference)
  Numerical Linear Algebra (Lloyd N. Trefethen, David Bau)



- Scientific Computing: An introductory Survey (Michael T. Heath)



Tentative Topics for Convex Optimization

Tentative Topics for Convex Optimization

- Convexity

Tentative Topics for Convex Optimization

- Convexity
- Gradient/subgradient descent, proximal GD, stochastic GD

Tentative Topics for Convex Optimization

- Convexity
- Gradient/subgradient descent, proximal GD, stochastic GD
- Duality, KKT conditions

Tentative Topics for Convex Optimization

- Convexity
- Gradient/subgradient descent, proximal GD, stochastic GD
- Duality, KKT conditions
- (Quasi) Newton methods, Interior point methods

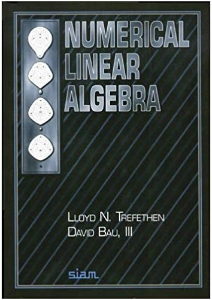
Tentative Topics for Convex Optimization

- 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...

Books

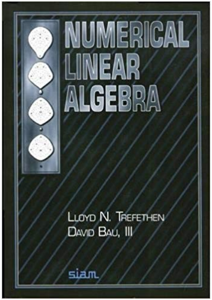
Books

- (Highly recommend; main reference)
  Numerical Linear Algebra
  (Lloyd N. Trefethen, David Bau)

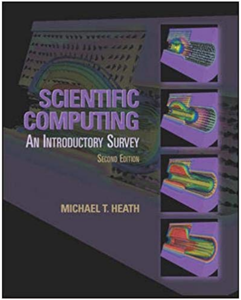


Books

- (Highly recommend; main reference)
  Numerical Linear Algebra (Lloyd N. Trefethen, David Bau)



- Scientific Computing: An introductory Survey (Michael T. Heath)



Objectives for Numerical Linear Algebra

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

Objectives for Numerical Linear Algebra

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

- Understand basic matrix computation concepts and algorithms;

Objectives for Numerical Linear Algebra

Upon successful 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, ...

Objectives for Numerical Linear Algebra

Upon successful 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, ...
- Analyze the computational cost of numerical algorithms

Objectives for Numerical Linear Algebra

Upon successful 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, ...
- Analyze the computational cost of numerical algorithms
- Apply different algorithms for different situations; able to diagnose numerical stability and errors

Objectives for Optimization

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

Objectives for Optimization

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

- Recognize convex optimizations

Objectives for Optimization

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

- Recognize convex optimizations
- Be able to formulate and transform convex optimization problem

Objectives for Optimization

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

- 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.

Objectives for Optimization

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

- 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

Objectives for Optimization

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

- 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?

Why study all these?

- Data science is largely about manipulating matrices/vectors.

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):

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.

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 $\mathcal{O}(n^3)$ floating point operations; on a large matrix where $n \approx 10^5$, $n^3 \approx 10^{15}$ operations $\approx$ 280 hours assuming 10^9 operations/s.

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 $\mathcal{O}(n^3)$ floating point operations; on a large matrix where $n \approx 10^5$, $n^3 \approx 10^{15}$ operations $\approx$ 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?

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 $\mathcal{O}(n^3)$ floating point operations; on a large matrix where $n \approx 10^5$, $n^3 \approx 10^{15}$ operations $\approx$ 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;

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 $\mathcal{O}(n^3)$ floating point operations; on a large matrix where $n \approx 10^5$, $n^3 \approx 10^{15}$ 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 $\mathcal{O}(n^2)$; again grows fast with size of matrix.

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 $\mathcal{O}(n^3)$ floating point operations; on a large matrix where $n \approx 10^5$, $n^3 \approx 10^{15}$ operations $\approx$ 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 $\mathcal{O}(n^2)$; again grows fast with size of matrix.
  - Scalability: you don't have enough memory space or computing power so you wantto use parallel computers. Does more nodes mean faster execution time?

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 $\mathcal{O}(n^3)$ floating point operations; on a large matrix where $n \approx 10^5$, $n^3 \approx 10^{15}$ 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 $\mathcal{O}(n^2)$; again grows fast with size of matrix.
  - Scalability: you don't have enough memory space or computing power so you wantto use parallel computers. Does more nodes mean faster execution time?
- Because the world is harsh, we'll frequently need to tradeoff between them: but first you need to know and understand them

Can't I just use a library?

...like Sci-kit learn / [put your favorite framework here]
Probably, but what if

Can't I just use a library?

. . . like Sci-kit learn / [put your favorite framework here]
Probably, but what if

- Too slow

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?

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 (☹)?

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?

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.