KUTAISI
INTERNATIONAL
UNIVERSITY

Numerical Linear Algebra

Ramaz Botchorishvili

Kutaisi International University

September 20, 2023

Course Overview, Norms

- Course Overview
- Computational project 1
- Calculation errors
- Difficulties with theoretical linear algebra(postpone ?)
- RGB colors and vectors
- Vector norms
- K-means clustering
- Q & A

Numerical Programming, Course Team

- - Lectures: Ramaz Botchorishvili
  - TTF: Mariam Mamageishvili (SA), Nino Sharvashidze (SA)

Numerical Programming, Students

-
  - Mathematics - 9
  - Computer Science - 21
  - Management - 2
  - Retake - 8

Numerical Programming, Students

- - Mathematics - 9
  - Computer Science - 21
  - Management - 2
  - Retake - 8

Prerequsites, desired background

- prerequisite: basic linear algebra, also in parallel
- desired: basic programming skills

Numerical Linear Algebra, Course Overview 1

- Course format: 2L + 2TTF

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- Course format: 2L + 2TTF

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF

- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF

- Quizzes: 2-3 graded quizzes, 50 min, during CE or online

- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF

- Quizzes: 2-3 graded quizzes, 50 min, during CE or online
- Homework: weekly Computational projects every 3-6 weeks

- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF

- Quizzes: 2-3 graded quizzes, 50 min, during CE or online
- Homework: weekly Computational projects every 3-6 weeks
- No make-up quizzes or homework or computational projects!!!

- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF

- Quizzes: 2-3 graded quizzes, 50 min, during CE or online
- Homework: weekly Computational projects every 3-6 weeks
- No make-up quizzes or homework or computational projects!!!
- Final examination: Project, written exam or combination?
- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*

- Quizzes: 2-3 graded quizzes, 50 min, during CE or online
- Homework: weekly Computational projects every 3-6 weeks
- No make-up quizzes or homework or computational projects!!!
- Final examination: Project, written exam or combination?
- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*

- Interaction during lectures: based on your feedback checkpoints only, interaction will not be forced
- Quizzes: 2-3 graded quizzes, 50 min, during CE or online
- Homework: weekly Computational projects every 3-6 weeks
- No make-up quizzes or homework or computational projects!!!
- Final examination: Project, written exam or combination?
- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 1

- What is class material: syllabus,readers,lecture slides,homework sheets
- How often is class material uploaded and where can you find it
- Course format: 2L + 2TTF
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Course pace: your feedback counts ⇒ more efficient use of time = higher pace
- Interaction during lectures: based on your feedback checkpoints only, interaction will not be forced
- Quizzes: 2-3 graded quizzes, 50 min, during CE or online
- Homework: weekly Computational projects every 3-6 weeks
- No make-up quizzes or homework or computational projects!!!
- Final examination: Project, written exam or combination?
- Assessment components: 10% Homework + 12% Activity + 24% Quizzes + 24% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 2

- What is class material: syllabus,readers,lecture slides,homework sheets
- Course format: 2L + 1CE + 1TTF or 1E
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Course pace: your feedback counts $\Rightarrow$ more efficient use of time = higher pace
- Interaction during lectures: based on your feedback checkpoints only, interaction will not be forced
- Assessment components: 10% Homework + 30% Quizzes + 30% Computational projects + 30% Final Exam

Numerical Linear Algebra, Course Overview 2

- What is class material: syllabus,readers,lecture slides,homework sheets
- Course format: 2L + 1CE + 1TTF or 1E
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Course pace: your feedback counts $\Rightarrow$ more efficient use of time = higher pace
- Interaction during lectures: based on your feedback checkpoints only, interaction will not be forced
- Assessment components: 10% Homework + 30% Quizzes + 30% Computational projects + 30% Final Exam

- How to get help: Instructor's office hours, TA's office hours, student TA's, additional appointment upon request

Numerical Linear Algebra, Course Overview 2

- What is class material: syllabus,readers,lecture slides,homework sheets
- Course format: 2L + 1CE + 1TTF or 1E
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Course pace: your feedback counts $\Rightarrow$ more efficient use of time = higher pace
- Interaction during lectures: based on your feedback checkpoints only, interaction will not be forced
- Assessment components: 10% Homework + 30% Quizzes + 30% Computational projects + 30% Final Exam
- Learning objectives: studying course content from the viewpoint of numerical mathematics

- How to get help: Instructor's office hours, TA's office hours, student TA's, additional appointment upon request

Numerical Linear Algebra, Course Overview 2

- What is class material: syllabus,readers,lecture slides,homework sheets
- Course format: 2L + 1CE + 1TTF or 1E
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Course pace: your feedback counts $\Rightarrow$ more efficient use of time = higher pace
- Interaction during lectures: based on your feedback checkpoints only, interaction will not be forced
- Assessment components: 10% Homework + 30% Quizzes + 30% Computational projects + 30% Final Exam
- Learning objectives: studying course content from the viewpoint of numerical mathematics
- Learning outcomes: apply knowledge and skills
- How to get help: Instructor's office hours, TA's office hours, student TA's, additional appointment upon request

Numerical Linear Algebra, Course Overview 3

- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*

Numerical Linear Algebra, Course Overview 3

- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Learning objectives: Studying four fundamental prolems of linear algebra (solving linear systems, eigenvalue problem, least square problem, singular value decomposition) from the viewpoint of numerical mathematics

Numerical Linear Algebra, Course Overview 3

- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Learning objectives: Studying four fundamental prolems of linear algebra (solving linear systems, eigenvalue problem, least square problem, singular value decomposition) from the viewpoint of numerical mathematics
- Learning outcomes: after taking this course students are able to

Numerical Linear Algebra, Course Overview 3

- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Learning objectives: Studying four fundamental prolems of linear algebra (solving linear systems, eigenvalue problem, least square problem, singular value decomposition) from the viewpoint of numerical mathematics
- Learning outcomes: after taking this course students are able to
  - apply direct and iterative methods for finding numerical solution to four fundamental problems of linear algebra

Numerical Linear Algebra, Course Overview 3

- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Learning objectives: Studying four fundamental prolems of linear algebra (solving linear systems, eigenvalue problem, least square problem, singular value decomposition) from the viewpoint of numerical mathematics
- Learning outcomes: after taking this course students are able to
  - apply direct and iterative methods for finding numerical solution to four fundamental problems of linear algebra
  - aelect suitable method of solution, analyze and prove properties

Numerical Linear Algebra, Course Overview 3

- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Learning objectives: Studying four fundamental prolems of linear algebra (solving linear systems, eigenvalue problem, least square problem, singular value decomposition) from the viewpoint of numerical mathematics
- Learning outcomes: after taking this course students are able to
  - apply direct and iterative methods for finding numerical solution to four fundamental problems of linear algebra
  - aelect suitable method of solution, analyze and prove properties
  - apply methods of numerical linear algebra to selected practical problems

Numerical Linear Algebra, Course Overview 3

- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Learning objectives: Studying four fundamental prolems of linear algebra (solving linear systems, eigenvalue problem, least square problem, singular value decomposition) from the viewpoint of numerical mathematics
- Learning outcomes: after taking this course students are able to
  - apply direct and iterative methods for finding numerical solution to four fundamental problems of linear algebra
  - aelect suitable method of solution, analyze and prove properties
  - apply methods of numerical linear algebra to selected practical problems
- Professor's office hours: present and discuss best solutions

Numerical Linear Algebra, Course Overview 3

- All details are in syllabus available in class material
- Course content: *Vector and matrix norms, conditioning and stability, computer arithmetic and error analysis, direct and iterative methods for solving linear systems of equations, matrix factorization methods, least squares for linear systems, eigenvalue problem, some applications*
- Learning objectives: Studying four fundamental prolems of linear algebra (solving linear systems, eigenvalue problem, least square problem, singular value decomposition) from the viewpoint of numerical mathematics
- Learning outcomes: after taking this course students are able to
  - apply direct and iterative methods for finding numerical solution to four fundamental problems of linear algebra
  - aelect suitable method of solution, analyze and prove properties
  - apply methods of numerical linear algebra to selected practical problems
- Professor's office hours: present and discuss best solutions

Numerical Linear Algebra, Course Overview related

QUESTIONS ?

- which slide? please give slide number
- which theorem?
- which example?
- which ...?

Computational project 1

Computational project 1

- Controlling laboratory equipment using camera (?Math, ?CS)

Computational project 1

- Controlling laboratory equipment using camera (?Math, ?CS)
- Collecting of problems on modeling with clustering algorithms (?MGMT)

Computational project 1

- Controlling laboratory equipment using camera (?Math, ?CS)
- Collecting of problems on modeling with clustering algorithms (?MGMT)
- Implementation and application of clustering using suitably selected norms (?Math)

Computational project 1

- Controlling laboratory equipment using camera (?Math, ?CS)
- Collecting of problems on modeling with clustering algorithms (?MGMT)
- Implementation and application of clustering using suitably selected norms (?Math)

Different projects, same weight?

A. Can You Trust Your Computer?

A. Can You Trust Your Computer?

Example 1.1

Computing Euler's Number

$$e = \lim_{n\to\infty}(1+\frac{1}{n})^n$$

A. Can You Trust Your Computer?

Example 1.1

Computing Euler's Number

$$e = \lim_{n \to \infty} (1 + \frac{1}{n})^n$$

n	(1+1/n)^n	Error
10	2.5937424601000023	0.12453936835904278
100	2.7048138294215285	0.013467999037516661
1000	2.7169239322355936	0.0013578962234515046
10000	2.7181459268249255	0.00013590163411958 4
100000	2.7182682371922975	1.3591266747603 47e-05
1000000	2.7182804690957534	1.359363291708604e-06
10000000	2.7182816941320818	1.3432696333026684e-07
100000000	2.7182817983473577	3.011168736577474e-08
1000000000	2.7182820520115603	-2.2355251516614771e-07
10000000000	2.7182820532347876	-2.2477574246337895e-07
100000000000	2.71828205335711	-2.248980650598753e-07
1000000000000	2.7185234960372378	-0.00024166757819266138
10000000000000	2.716110034086901	0.002171794372144209
100000000000000	2.716110034087023	0.0021717943720220845
1000000000000000	3.035035206549262	-0.31675337809021675

B. Can You Trust Your Computer?

B. Can You Trust Your Computer?

Example 1.2

Do computers know addition is associative ?

$$S_1 = \sum_{i=1}^{n} \frac{1}{i}, \; S_2 = \sum_{i=n}^{1} \frac{1}{i}$$

B. Can You Trust Your Computer?

Example 1.2

Do computers know addition is associative ?

$$S_1 = \sum_{i=1}^{n} \frac{1}{i}, \; S_2 = \sum_{i=n}^{1} \frac{1}{i}$$

n	SUM from 1 to n	SUM from n to 1	Difference
10	2.8289682539682537	1.928968253968254	0.8999999999999997
100	5.177377517639621	4.1873775176396215	0.9899999999999993
1000	7.484470860550343	6.485470860550341	0.9990000000000023
10000	9.787506036044348	8.787606036044386	0.9998999999999629
100000	12.090136129863335	11.090146129863408	0.9999899999999275
1000000	14.39272572286499	13.392726722865772	0.9999989999992174
10000000	16.69531126585727	15.695311365859965	0.9999998999973059
100000000	18.997896403852554	17.997896413853447	0.9999999899991074

Figure: $A + B \neq B + A$

A. Computational Difficulties of Theoretical Linear Algebra

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.3

Solving linear system $Ax = b$ by Cramer's rule

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.3

Solving linear system $Ax = b$ by Cramer's rule

- Cramer's rule needs determinants

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.3

Solving linear system $Ax = b$ by Cramer's rule

- Cramer's rule needs determinants
- Computing determinant of $n \times n$ matrix costs approximately $n!$ FLOPS

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.3

Solving linear system $Ax = b$ by Cramer's rule

- Cramer's rule needs determinants
- Computing determinant of $n \times n$ matrix costs approximately $n!$ FLOPS
- FLOPS = floating point operations per second

A. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.3

Solving linear system $Ax = b$ by Cramer's rule

- Cramer's rule needs determinants
- Computing determinant of $n \times n$ matrix costs approximately $n!$ FLOPS
- FLOPS = floating point operations per second
- Solving linear system $Ax = b$ with twenty unknowns will take millions of years on today's fastest computer

B. Computational Difficulties of Theoretical Linear Algebra

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
-

  Computing the unique solution of a linear system by matrix inversion

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.4

Computing the unique solution of a linear system by matrix inversion

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.4

Computing the unique solution of a linear system by matrix inversion

- $Ax = b$, $x = A^{-1}b$

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.4

Computing the unique solution of a linear system by matrix inversion

- $Ax = b$, $x = A^{-1}b$
- Algorithm:
  1. compute matrix inverse A^{-1}
  2. compute solution $x = A^{-1}b$

B. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving a least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.4

Computing the unique solution of a linear system by matrix inversion

- $Ax = b$, $x = A^{-1}b$
- Algorithm:
  1. compute matrix inverse A^{-1}
  2. compute solution $x = A^{-1}b$
- Computing matrix inverse is not practical:
  1. using standard elimination method is approximately 2.5 times faster
  2. other methods are often more accurate

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.5

Solving a least squares problem by normal equations

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.5

Solving a least squares problem by normal equations

- The least squares problem: $\min_x \|Ax - b\|_2, A \in \mathbb{R}^{n \times m}$

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.5

Solving a least squares problem by normal equations

- The least squares problem: $\min_x \|Ax - b\|_2, A \in \mathbb{R}^{n \times m}$
- Algorithm:

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.5

Solving a least squares problem by normal equations

- The least squares problem: $\min_x \|Ax - b\|_2, A \in \mathbb{R}^{n \times m}$
- Algorithm:
  1. Compute gradient of $\|Ax - b\|_2$

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.5

Solving a least squares problem by normal equations

- The least squares problem: $\min_x \|Ax - b\|_2, A \in \mathbb{R}^{n \times m}$
- Algorithm:
    1. Compute gradient of $\|Ax - b\|_2$
    2. Obtain normal equation by setting gradient to zero: $A^T Ax = A^T b$

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.5

Solving a least squares problem by normal equations

- The least squares problem: $\min_x \|Ax - b\|_2, A \in \mathbb{R}^{n \times m}$
- Algorithm:
  1. Compute gradient of $\|Ax - b\|_2$
  2. Obtain normal equation by setting gradient to zero: $A^T Ax = A^T b$
  3. Solve normal equation and obtain solution to least squares problem

C. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Computing the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.5

Solving a least squares problem by normal equations

- The least squares problem: $\min_x \|Ax - b\|_2, A \in \mathbb{R}^{n \times m}$
- Algorithm:
  1. Compute gradient of $\|Ax - b\|_2$
  2. Obtain normal equation by setting gradient to zero: $A^T Ax = A^T b$
  3. Solve normal equation and obtain solution to least squares problem
- Solving normal equation is not practical:
  1. Explicit formation of $A^T A$ may cause errors(remember $a + b \neq b + a$)
  2. Normal equation is more sensitive to perturbations than $Ax = b$ and it can lead to solution with errors

D. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

D. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.6

Finding the eigenvalues of a matrix using its characteristic polynomial

D. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.6

Finding the eigenvalues of a matrix using its characteristic polynomial

- The eigenvalue problem:
  $Ax_i = \lambda_i x_i, A \in \mathbb{R}^{n \times n}, x_i \in \mathbb{R}^n, x_i \neq 0, \lambda_i \neq 0, i = 1, 2, ..n$

D. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.6

Finding the eigenvalues of a matrix using its characteristic polynomial

- The eigenvalue problem:
  $Ax_i = \lambda_i x_i, A \in \mathbb{R}^{n \times n}, x_i \in \mathbb{R}^n, x_i \neq 0, \lambda_i \neq 0, i = 1, 2, ..n$
- Algorithm:
  1. Define characteristic polynomial $|Ax - \lambda I|$
  2. Find zeros of characteristic polynomial, solve $|Ax - \lambda I| = 0$

D. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.6

Finding the eigenvalues of a matrix using its characteristic polynomial

- The eigenvalue problem:
  $Ax_i = \lambda_i x_i, A \in \mathbb{R}^{n \times n}, x_i \in \mathbb{R}^n, x_i \neq 0, \lambda_i \neq 0, i = 1, 2, ..n$
- Algorithm:
  1. Define characteristic polynomial $|Ax - \lambda I|$
  2. Find zeros of characteristic polynomial, solve $|Ax - \lambda I| = 0$
- Solving characteristic equation is not practical:
  1. perturbed coefficients are computed for characteristic polynomial
  2. Zeroes of certain polynomials are sensitive to perturbations, e.g. Wilkinson polynomial, $n = 20$

E. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

E. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.7

E. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.7

- Singular value decomposition

$$A = U\Sigma V^*, A \in \mathbb{R}^{m\times n}$$

- Σ is "diagonal" matrix with singular values $\sigma_i, i = 1, 2, ..n$ on the diagonal

E. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.7

- Singular value decomposition

$$A = U\Sigma V^*, A \in \mathbb{R}^{m\times n}$$

- Σ is "diagonal" matrix with singular values $\sigma_i, i = 1, 2, ..n$ on the diagonal
- Algorithm:
    1. Compute $A^T A$ and find it's eigenvalues $\lambda_i, i = 1, 2, ..n$
    2. Set $\sigma_i = \lambda_i, i = 1, 2, ..n$

E. Computational Difficulties of Theoretical Linear Algebra

- solving a linear system by Cramer's rule
- Computing the unique solution of a linear system by matrix inversion
- Solving least squares problem by normal equations
- Finding the eigenvalues of a matrix using characteristic polynomial
- Finding the singular values by computing the eigenvalues of $A^T A$

Example 1.7

- Singular value decomposition

$$A = U \Sigma V^*, A \in \mathbb{R}^{m \times n}$$

- Σ is "diagonal" matrix with singular values $\sigma_i, i = 1, 2, ..n$ on the diagonal
- Algorithm:
  1. Compute $A^T A$ and find it's eigenvalues $\lambda_i, i = 1, 2, ..n$
  2. Set $\sigma_i = \lambda_i, i = 1, 2, ..n$
- Algorithm not viable: explicit computing of $A^T A$ might introduce errors

RGB color coding and vectors, 1

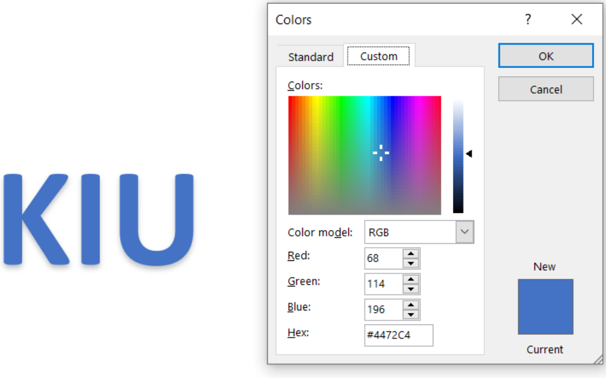

Figure: RGB code (68,114,196)

RGB color coding and vectors, 2

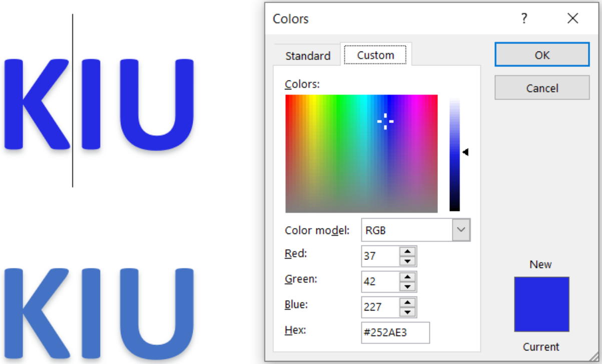

Figure: RGB code OLD=(68,114,196), NEW=(37,42,227)

RGB color coding and vectors, 3

Example 1.8

How colors can be compared? Which colors are most distinct or similar?

Figure: RGB codes: (37,42,227), (26,56,238), (68,114,196), (72,69,195)

Vector Norms

Definition 1.9

$\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$:

Vector Norms

Definition 1.9

$\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)

Vector Norms

Definition 1.9

$\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha| \|x\|$ (homogenity)

Vector Norms

Definition 1.9

$\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha| \|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

Vector Norms

Definition 1.9

$\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha| \|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Vector Norms

Definition 1.9

$\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha|\|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Example 1.10

Vector Norms

Definition 1.9

$\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha| \|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Example 1.10

- $\|x\|_1 = |x_1| + |x_2| + ... + |x_n|$ (the one-norm)

Vector Norms

Definition 1.9

$\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha|\|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Example 1.10

- $\|x\|_1 = |x_1| + |x_2| + ... + |x_n|$ (the one-norm)
- $\|x\|_2 = (x_1^2 + x_2^2 + ... + x_n^2)^{\frac{1}{2}}$ (the two-norm or Eucledian norm)

Vector Norms

Definition 1.9

$\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha|\|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Example 1.10

- $\|x\|_1 = |x_1| + |x_2| + ... + |x_n|$ (the one-norm)
- $\|x\|_2 = (x_1^2 + x_2^2 + ... + x_n^2)^{\frac{1}{2}}$ (the two-norm or Eucledian norm)
- $\|x\|_\infty = max\{|x_1|, |x_2|, ..., |x_n|\}$ (the max-norm or infinity norm)

Vector Norms

Definition 1.9

$\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha|\|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Example 1.10

- $\|x\|_1 = |x_1| + |x_2| + ... + |x_n|$ (the one-norm)
- $\|x\|_2 = (x_1^2 + x_2^2 + ... + x_n^2)^{\frac{1}{2}}$ (the two-norm or Eucledian norm)
- $\|x\|_\infty = max\{|x_1|, |x_2|, ..., |x_n|\}$ (the max-norm or infinity norm)
- $\|x\|_p = (|x_1|^p + |x_2|^p + ... + |x_n|^p)^{\frac{1}{p}}$ (the p-norm or Hoelder norm)

Vector Norms

Definition 1.9

$\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha| \|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Example 1.10

- $\|x\|_1 = |x_1| + |x_2| + ... + |x_n|$ (the one-norm)
- $\|x\|_2 = (x_1^2 + x_2^2 + ... + x_n^2)^{\frac{1}{2}}$ (the two-norm or Eucledian norm)
- $\|x\|_\infty = max\{|x_1|, |x_2|, ..., |x_n|\}$ (the max-norm or infinity norm)
- $\|x\|_p = (|x_1|^p + |x_2|^p + ... + |x_n|^p)^{\frac{1}{p}}$ (the p-norm or Hoelder norm)
- $\|x\| = |x_1| + |x_2 - x_1| + ... + |x_n - x_{n-1}|$

Vector Norms

Definition 1.9

$\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$:

1. $\|x\| \geq 0$, $\|x\| = 0$ if and only if $x = 0$ (positivity)
2. $\|\alpha x\| = |\alpha|\|x\|$ (homogenity)
3. $\|x + y\| \leq \|x\| + \|y\|$(triangle inequality or subadditivity)

holds true for all $x, y \in \mathbb{R}^n$ and for all scalars α

Example 1.10

- $\|x\|_1 = |x_1| + |x_2| + \dots + |x_n|$ (the one-norm)
- $\|x\|_2 = (x_1^2 + x_2^2 + \dots + x_n^2)^{\frac{1}{2}}$ (the two-norm or Eucledian norm)
- $\|x\|_\infty = max\{|x_1|, |x_2|, \dots, |x_n|\}$ (the max-norm or infinity norm)
- $\|x\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{\frac{1}{p}}$ (the p-norm or Hoelder norm)
- $\|x\| = |x_1| + |x_2 - x_1| + \dots + |x_n - x_{n-1}|$
- $\|x\|_A = \|A^{\frac{1}{2}} x\|_2, A \in \mathbb{R}^{n \times n}, A = A^T > 0$

Properties of Vector Norms

Properties of Vector Norms

- Inverse Triangle Inequality

$$\|x - y\| \geq \left|\, \|x\| - \|y\| \,\right|$$

Properties of Vector Norms

- Inverse Triangle Inequality

$$\|x - y\| \geq |\ \|x\| - \|y\|\ |$$

- Hoelder Inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \|x\|_p \|y\|_q, \frac{1}{p} + \frac{1}{q} = 1, p, q \geq 1$$

Properties of Vector Norms

- Inverse Triangle Inequality

$$\|x - y\| \geq |\ \|x\| - \|y\|\ |$$

- Hoelder Inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \|x\|_p \|y\|_q, \frac{1}{p} + \frac{1}{q} = 1, p, q \geq 1$$

- Cauchy-Schwartz Inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \|x\|_2 \|y\|_2$$

Properties of Vector Norms

- Inverse Triangle Inequality

$$\|x - y\| \geq | \; \|x\| - \|y\| \; |$$

- Hoelder Inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \|x\|_p \|y\|_q, \frac{1}{p} + \frac{1}{q} = 1, p, q \geq 1$$

- Cauchy-Schwartz Inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \|x\|_2 \|y\|_2$$

- Minkowsky Inequality

$$\|x + y\|_p \leq \|x\|_p + \|y\|_p$$

Properties of Vector Norms

- Inverse Triangle Inequality

$$\|x - y\| \geq |\ \|x\| - \|y\|\ |$$

- Hoelder Inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \|x\|_p \|y\|_q, \frac{1}{p} + \frac{1}{q} = 1, p, q \geq 1$$

- Cauchy-Schwartz Inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \|x\|_2 \|y\|_2$$

- Minkowsky Inequality

$$\|x + y\|_p \leq \|x\|_p + \|y\|_p$$

-

$$\lim_{p \to \infty} \|x\|_p = \|x\|_\infty$$

Equivalence of Vector Norms

Definition 1.11

Equivalence of Vector Norms

Definition 1.11

1. $C_m \|x\|_\alpha \leq \|x\|_\beta \leq C_M \|x\|_\alpha$

Equivalence of Vector Norms

Definition 1.11

1. $C_m\|x\|_\alpha \leq \|x\|_\beta \leq C_M\|x\|_\alpha$
2. C_m, C_M finite positive constants

Equivalence of Vector Norms

Definition 1.11

1. $C_m \|x\|_\alpha \leq \|x\|_\beta \leq C_M \|x\|_\alpha$
2. C_m, C_M finite positive constants
3. holds true for all vectors x

Example 1.12

Equivalence of Vector Norms

Definition 1.11

1. $C_m\|x\|_\alpha \le \|x\|_\beta \le C_M\|x\|_\alpha$
2. C_m, C_M finite positive constants
3. holds true for all vectors x

Example 1.12

- $\|x\|_2 \le \|x\|_1 \le \sqrt{n}\|x\|_2$

Equivalence of Vector Norms

Definition 1.11

1. $C_m\|x\|_\alpha \leq \|x\|_\beta \leq C_M\|x\|_\alpha$
2. C_m, C_M finite positive constants
3. holds true for all vectors x

Example 1.12

- $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$
- $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$

Equivalence of Vector Norms

Definition 1.11

1. $C_m \|x\|_\alpha \leq \|x\|_\beta \leq C_M \|x\|_\alpha$
2. C_m, C_M finite positive constants
3. holds true for all vectors x

Example 1.12

- $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n} \|x\|_2$
- $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty$
- $\|x\|_\infty \leq \|x\|_1 \leq n \|x\|_\infty$

Equivalence of Vector Norms

Definition 1.11

1. $C_m\|x\|_\alpha \leq \|x\|_\beta \leq C_M\|x\|_\alpha$
2. C_m, C_M finite positive constants
3. holds true for all vectors x

Example 1.12

- $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$
- $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$
- $\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty$
- $\|x\|_\infty \leq \|x\|_p \leq n^{\frac{1}{p}}\|x\|_\infty, p \geq 1$

Equivalence of Vector Norms

Definition 1.11

1. $C_m\|x\|_\alpha \leq \|x\|_\beta \leq C_M\|x\|_\alpha$
2. C_m, C_M finite positive constants
3. holds true for all vectors x

Example 1.12

- $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$
- $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$
- $\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty$
- $\|x\|_\infty \leq \|x\|_p \leq n^{\frac{1}{p}}\|x\|_\infty, p \geq 1$

Theorem 1.13

* *In $\mathbb{R}^n$ all vector norms are equivalent.*

Vector Norms, Functions, Convexity

Vector Norms, Functions, Convexity

Definition 1.14

Convex function

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), 0 \leq \alpha \leq 1$$

Vector Norms, Functions, Convexity

Definition 1.14

Convex function

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y), 0 \leq \alpha \leq 1$$

Definition 1.15

Function: $f(x) = \|x\|, x \in \mathbb{R}^n$

Vector Norms, Functions, Convexity

Definition 1.14

Convex function

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), 0 \leq \alpha \leq 1$$

Definition 1.15

Function: $f(x) = \|x\|, x \in \mathbb{R}^n$

- Any vector norm ($\mathbb{R}^n \to \mathbb{R}$) is uniformly continuous function

Vector Norms, Functions, Convexity

Definition 1.14

Convex function

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), 0 \leq \alpha \leq 1$$

Definition 1.15

Function: $f(x) = \|x\|, x \in \mathbb{R}^n$

- Any vector norm $(\mathbb{R}^n \to \mathbb{R})$ is uniformly continuous function
- Any vector norm $(\mathbb{R}^n \to \mathbb{R})$ is convex function

Vector Norms, Functions, Convexity

Definition 1.14

Convex function

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), 0 \leq \alpha \leq 1$$

Definition 1.15

Function: $f(x) = \|x\|, x \in \mathbb{R}^n$

- Any vector norm ($\mathbb{R}^n \to \mathbb{R}$) is uniformly continuous function
- Any vector norm ($\mathbb{R}^n \to \mathbb{R}$) is convex function
- $f(x) = \|x\|_p^p, p > 1, x \in \mathbb{R}^n$ is strictly convex function

Vector Norms, Functions, Convexity

Definition 1.14

Convex function

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), 0 \leq \alpha \leq 1$$

Definition 1.15

Function: $f(x) = \|x\|, x \in \mathbb{R}^n$

- Any vector norm $(\mathbb{R}^n \to \mathbb{R})$ is uniformly continuous function
- Any vector norm $(\mathbb{R}^n \to \mathbb{R})$ is convex function
- $f(x) = \|x\|_p^p, p > 1, x \in \mathbb{R}^n$ is strictly convex function
- Balls $\{\|x\| \leq 1\}$ are convex for any vector norm $(\mathbb{R}^n \to \mathbb{R})$

Vector Norms, Functions, Convexity

Definition 1.14

Convex function

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y), 0 \leq \alpha \leq 1$$

Definition 1.15

Function: $f(x) = \|x\|, x \in \mathbb{R}^n$

- Any vector norm $(\mathbb{R}^n \to \mathbb{R})$ is uniformly continuous function
- Any vector norm $(\mathbb{R}^n \to \mathbb{R})$ is convex function
- $f(x) = \|x\|_p^p, p > 1, x \in \mathbb{R}^n$ is strictly convex function
- Balls $\{\|x\| \leq 1\}$ are convex for any vector norm $(\mathbb{R}^n \to \mathbb{R})$

Example 1.16

1. Ball in Eucledian norm is round: $\{x \in \mathbb{R}^2, (x_1^2 + x_2^2)^{\frac{1}{2}} \leq 1\}$.
2. Ball in one-norm is not round: $\{x \in \mathbb{R}^2, |x_1| + |x_2| \leq 1\}$.

Vectors, k-means Clustering and Norms, 1

- Vectors can be used for ...
- Each entry of a vector may have meaning, different labels..

Vectors, k-means Clustering and Norms, 1

- Vectors can be used for ...
- Each entry of a vector may have meaning, different labels..

Example 1.17

Vectors, k-means Clustering and Norms, 1

- Vectors can be used for ...
- Each entry of a vector may have meaning, different labels..

Example 1.17

- Location in $3D$ space: (x_1, x_2, x_3), e.g.longitude, lattitude, elevation
- Color in digital images: Red, Green, Blue - RGB color coding $(i_1, i_2, i3), 0 \leq i_1, i_2, i_3 \leq 255$

Vectors, k-means Clustering and Norms, 1

- Vectors can be used for ...
- Each entry of a vector may have meaning, different labels..

Example 1.17

- Location in $3D$ space: (x_1, x_2, x_3), e.g.longitude, lattitude, elevation
- Color in digital images: Red, Green, Blue - RGB color coding $(i_1, i_2, i3), 0 \leq i_1, i_2, i_3 \leq 255$
- Portfolio:
  - Currency: labels (USD, EUR, GEL), amount (a_1, a_2, a_3)
  - Stocks: labels $(AMZN, AAPL, GOOGL, MSFT)$, vector: (a_1, a_2, a_3, a_4)

Vectors, k-means Clustering and Norms, 1

- Vectors can be used for ...
- Each entry of a vector may have meaning, different labels..

Example 1.17

- Location in $3D$ space: (x_1, x_2, x_3), e.g.longitude, lattitude, elevation
- Color in digital images: Red, Green, Blue - RGB color coding $(i_1, i_2, i3), 0 \leq i_1, i_2, i_3 \leq 255$
- Portfolio:
  - Currency: labels (USD, EUR, GEL), amount (a_1, a_2, a_3)
  - Stocks: labels $(AMZN, AAPL, GOOGL, MSFT)$, vector: (a_1, a_2, a_3, a_4)
- Customer purchase: $(e_1, e_2, ..., e_n)$, n-number of goods, e_i - Euros spent on the goods in i-th category

Vectors, k-means Clustering and Norms, 1

- Vectors can be used for ...
- Each entry of a vector may have meaning, different labels..

Example 1.17

- Location in $3D$ space: (x_1, x_2, x_3), e.g.longitude, lattitude, elevation
- Color in digital images: Red, Green, Blue - RGB color coding $(i_1, i_2, i3), 0 \leq i_1, i_2, i_3 \leq 255$
- Portfolio:
  - Currency: labels (USD, EUR, GEL), amount (a_1, a_2, a_3)
  - Stocks: labels $(AMZN, AAPL, GOOGL, MSFT)$, vector: (a_1, a_2, a_3, a_4)
- Customer purchase: $(e_1, e_2, ..., e_n)$, n-number of goods, e_i - Euros spent on the goods in i-th category
- Word count in documents:
  - Dictionary of words: $(w_1, w_2, ..., w_n)$, n-number of words, w_i - word in the dictionary
  - Occurrences vector: $(o_1, o_2, ..., a_n)$, o_i - occurrence of the word w_i in a document

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

- Streaming service offering n songs to m customers

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

- Streaming service offering n songs to m customers
- customer data is available for the past period:

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

- Streaming service offering n songs to m customers
- customer data is available for the past period:
  - data for i-th customer: $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

- Streaming service offering n songs to m customers
- customer data is available for the past period:
  - data for i-th customer: $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$
  - s_j-number, j-th song was streamed in the period under consideration

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

- Streaming service offering n songs to m customers
- customer data is available for the past period:
  - data for i-th customer: $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$
  - s_j-number, j-th song was streamed in the period under consideration
- **Task:** recommend songs to each customer based on past listening experience

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

- Streaming service offering n songs to m customers
- customer data is available for the past period:
  - data for i-th customer: $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$
  - s_j-number, j-th song was streamed in the period under consideration
- **Task:** recommend songs to each customer based on past listening experience
- **Solution:** cluster customers in different groups using available data $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$ and recommend songs from the cluster

Vectors, k-means Clustering and Norms, 2

Example 1.18

Reccomendation engine for streaming service

- Streaming service offering n songs to m customers
- customer data is available for the past period:
  - data for i-th customer: $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$
  - s_j-number, j-th song was streamed in the period under consideration
- **Task:** recommend songs to each customer based on past listening experience
- **Solution:** cluster customers in different groups using available data $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$ and recommend songs from the cluster
- **Algorithm:** k-means clustering

Vectors, k-means Clustering and Norms, 3

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently
- Name k-means has been used since the 1960s.

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently
- Name *k*-means has been used since the 1960s.
- *k*-means algorithm input data

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently
- Name *k*-means has been used since the 1960s.
- *k*-means algorithm input data
    1. list of m vectors $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, to be clustered

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently
- Name *k*-means has been used since the 1960s.
- *k*-means algorithm input data
  1. list of m vectors $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, to be clustered
  2. list of k vectors $z_i = (s_1, s_2, ..., s_n), i = 1, 2, .., k$, representing k different groups

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently
- Name k-means has been used since the 1960s.
- k-means algorithm input data
  1. list of m vectors $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, to be clustered
  2. list of k vectors $z_i = (s_1, s_2, ..., s_n), i = 1, 2, .., k$, representing k different groups
- k-means algorithm loop:

  Repeat until convergence

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently
- Name k-means has been used since the 1960s.
- k-means algorithm input data
  1. list of m vectors $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, to be clustered
  2. list of k vectors $z_i = (s_1, s_2, ..., s_n), i = 1, 2, .., k$, representing k different groups
- k-means algorithm loop:

  Repeat until convergence

  1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$

Vectors, k-means Clustering and Norms, 3

- Year- 1957, Authors - Stuart Lloyd and Hugo Steinhaus, discovered independently
- Name k-means has been used since the 1960s.
- k-means algorithm input data
  1. list of m vectors $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, to be clustered
  2. list of k vectors $z_i = (s_1, s_2, ..., s_n), i = 1, 2, .., k$, representing k different groups
- k-means algorithm loop:

  Repeat until convergence

  1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
  2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

Vectors, k-means Clustering and Norms, 4

- k-means algorithm loop:

  Repeat until convergence

  1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
  2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

Vectors, k-means Clustering and Norms, 4

- k-means algorithm loop:

Repeat until convergence

1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

- Related questions:

Vectors, k-means Clustering and Norms, 4

- k-means algorithm loop:

  Repeat until convergence

  1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
  2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

- Related questions:

  1. Q: besides the standard k-means (naive k-means) are other variants of the algorithm available?
     A: yes, various approaches for updating representatives

Vectors, k-means Clustering and Norms, 4

- *k*-means algorithm loop:

Repeat until convergence

1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

- Related questions:

1. Q: besides the standard k-means (naive k-means) are other variants of the algorithm available?
   A: yes, various approaches for updating representatives
2. Q: what is "nearest" representative vector to c_i?
   A: $\min_k \|c_i - z_k\|$

Vectors, k-means Clustering and Norms, 4

- *k*-means algorithm loop:

Repeat until convergence

1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

- Related questions:

1. Q: besides the standard k-means (naive k-means) are other variants of the algorithm available?
   A: yes, various approaches for updating representatives
2. Q: what is "nearest" representative vector to c_i?
   A: $\min_k \|c_i - z_k\|$
3. Q: does k-means algorithm always converge?
   A: no, the algorithm is heuristic

Vectors, k-means Clustering and Norms, 4

- k-means algorithm loop:

Repeat until convergence

1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

- Related questions:

1. Q: besides the standard k-means (naive k-means) are other variants of the algorithm available?
   A: yes, various approaches for updating representatives
2. Q: what is "nearest" representative vector to c_i?
   A: $\min_k \|c_i - z_k\|$
3. Q: does k-means algorithm always converge?
   A: no, the algorithm is heuristic
4. Q: Which parameters of the algorithm affect final partition?
   A: initial representatives, norm

Vectors, k-means Clustering and Norms, 4

- k-means algorithm loop:

Repeat until convergence

1. Partition: for each vector $c_i = (s_1, s_2, ..., s_n), i = 1, 2, .., m$, assign nearest representative $z_j = (s_1, s_2, ..., s_n), j = 1, 2, .., k$
2. Update: set z_j to the mean in the group $j = 1, 2, .., k$

- Related questions:

1. Q: besides the standard k-means (naive k-means) are other variants of the algorithm available?
   A: yes, various approaches for updating representatives
2. Q: what is "nearest" representative vector to c_i?
   A: $\min_k \|c_i - z_k\|$
3. Q: does k-means algorithm always converge?
   A: no, the algorithm is heuristic
4. Q: Which parameters of the algorithm affect final partition?
   A: initial representatives, norm
5. Q: How to compare different partitions of the same list of vectors?
   A:
$$\min_{j=1,..,k} \|c_1 - z_j\| + \min_{j=1,..,k} \|c_2 - z_j\| + .. + \min_{j=1,..,k} \|c_n - z_j\|$$

Vectors, k-means Clustering and Norms, 5

Example 1.19

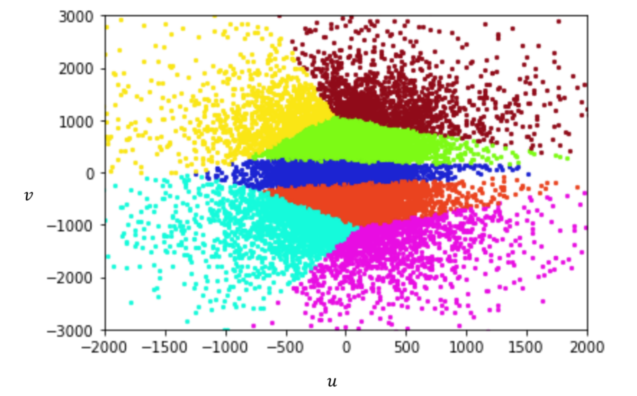

Figure: Optical flow clustering, M.Jananshvili

Vectors, k-means Clustering and Norms, 6

Example 1.20

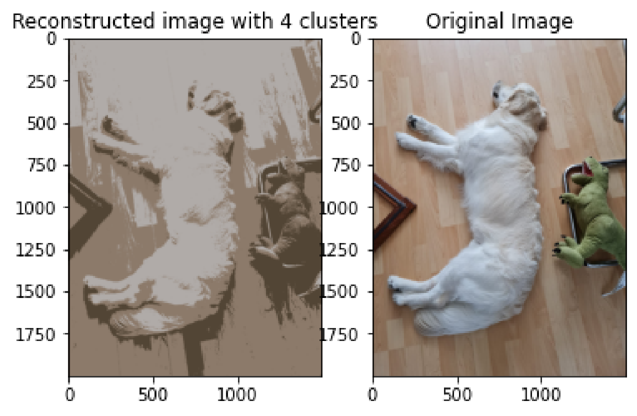

Figure: Image with reduced number of colors

Vectors, k-means Clustering and Norms, 7

Example 1.21

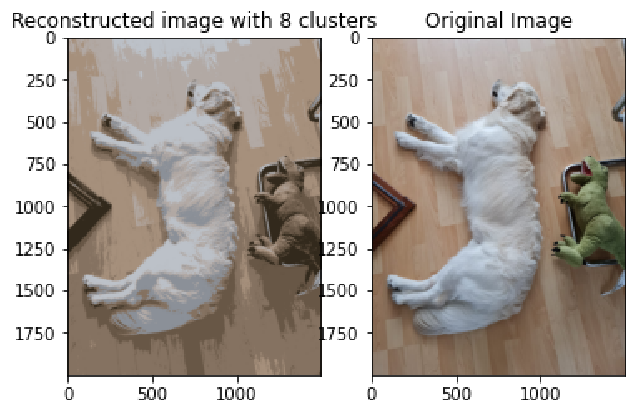

Figure: Image with reduced number of colors

Vectors, k-means Clustering and Norms, 8

Example 1.22

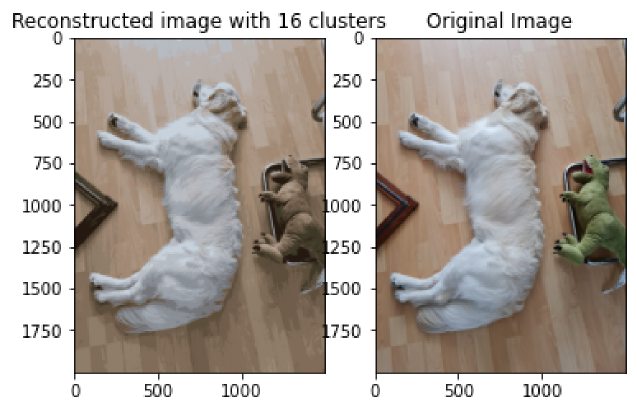

Figure: Image with reduced number of colors

Q & A