CS 598: Provably Efficient Algorithms for Numerical and Combinatorial Problems

Part 1: Background Numerical Analysis Tools

Edgar Solomonik

University of Illinois at Urbana-Champaign

Matrices and Tensors

- What is a matrix?

- What is a tensor?

Matrix and Tensor Decompositions

- What is a matrix factorization?

- What is a tensor decomposition?

Graphs and Hypergraphs

- What is a graph?

- What is a hypergraph?

Numerical and Combinatorial Problems

- What numerical problems will we look at?
- What combinatorial problems will we look at?
- What applications do these have?

Provably Efficient Algorithms

- What makes an algorithm provably efficient?

Provably Efficient Parallel Schedules

- What makes a parallel schedule good
- What makes a parallel schedule for a given algorithm optimal?

Error Analysis

- Forward Error:

- Backward Error:

Conditioning

- Conditioning measures the worst-case sensitivity of the output with respect to perturbations of the input
- The absolute condition number is a property of the problem, which measures its sensitivity to perturbations in input
- The relative condition number considers relative perturbations in input and output, so that

Rounding Error in Floating Point Operations

- Addition and Subtraction

Matrix Condition Number

- The matrix condition number $\kappa(\boldsymbol{A})$ is the ratio between the max and min distance from the surface to the center of the unit ball transformed by $\kappa(\boldsymbol{A})$:

- The matrix condition number bounds the worst-case amplification of error in a matrix-vector product:

Singular Value Decomposition

- The singular value decomposition (SVD)

- Condition number in terms of singular values

Linear Least Squares

- Find $\boldsymbol{x}^\star = \text{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2$ where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$:

- Given the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ we have $\boldsymbol{x}^\star = \underbrace{\boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T}_{\boldsymbol{A}^\dagger}\boldsymbol{b}$, where $\boldsymbol{\Sigma}^\dagger$ contains the reciprocal of all nonzeros in $\boldsymbol{\Sigma}$, and more generally $\dagger$ denotes pseudoinverse:

Normal Equations

Demo: *Normal equations vs Pseudoinverse*
Demo: *Issues with the normal equations*

- *Normal equations* are given by solving $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$:

- However, solving the normal equations is a more ill-conditioned problem then the original least squares algorithm

Solving the Normal Equations

- If $\boldsymbol{A}$ is full-rank, then $\boldsymbol{A}^T\boldsymbol{A}$ is symmetric positive definite (SPD):

- Since $\boldsymbol{A}^T\boldsymbol{A}$ is SPD we can use Cholesky factorization, to factorize it and solve linear systems:

QR Factorization

- If $\boldsymbol{A}$ is full-rank there exists an orthogonal matrix $\boldsymbol{Q}$ and a unique upper-triangular matrix $\boldsymbol{R}$ with a positive diagonal such that $\boldsymbol{A} = \boldsymbol{QR}$

- A reduced QR factorization (unique part of general QR) is defined so that $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$ has orthonormal columns and $\boldsymbol{R}$ is square and upper-triangular

- We can solve the normal equations (and consequently the linear least squares problem) via reduced QR as follows

Eigenvalue Decomposition

- If a matrix $\boldsymbol{A}$ is diagonalizable, it has an *eigenvalue decomposition*

- $\boldsymbol{A}$ and $\boldsymbol{B}$ are *similar*, if there exist $\boldsymbol{Z}$ such that $\boldsymbol{A} = \boldsymbol{Z}\boldsymbol{B}\boldsymbol{Z}^{-1}$

Similarity of Matrices

matrix	*similarity*	*reduced form*
SPD		
real symmetric		
Hermitian		
normal		
real		
diagonalizable		
arbitrary		

Rayleigh Quotient

- For any vector x that is close to an eigenvector, the *Rayleigh quotient* provides an estimate of the associated eigenvalue of A:

Introduction to Krylov Subspace Methods

- *Krylov subspace methods* work with information contained in the $n \times k$ matrix

$$\boldsymbol{K}_k = \begin{bmatrix} \boldsymbol{x_0} & \boldsymbol{A x_0} & \cdots & \boldsymbol{A}^{k-1}\boldsymbol{x_0} \end{bmatrix}$$

- $\boldsymbol{A}$ is similar to *companion matrix* $\boldsymbol{C} = \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{K}_n$:

Krylov Subspaces

- Given $\boldsymbol{Q}_k \boldsymbol{R}_k = \boldsymbol{K}_k$, we obtain an orthonormal basis for the Krylov subspace,

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{x}_0) = span(\boldsymbol{Q}_k) = \{p(\boldsymbol{A})\boldsymbol{x}_0 : deg(p) < k\},$$

  where p is any polynomial of degree less than k.
- The Krylov subspace includes the $k-1$ approximate dominant eigenvectors generated by $k-1$ steps of power iteration:

Krylov Subspace Methods

- The $k \times k$ matrix $\boldsymbol{H}_k = \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k$ minimizes $||\boldsymbol{A}\boldsymbol{Q}_k - \boldsymbol{Q}_k \boldsymbol{H}_k||_2$:

- $\boldsymbol{H}_k$ is Hessenberg, because the companion matrix $\boldsymbol{C}_k$ is Hessenberg:

Rayleigh-Ritz Procedure

- The eigenvalues/eigenvectors of $\boldsymbol{H}_k$ are the *Ritz values/vectors*:

- The Ritz vectors and values are the *ideal approximations* of the actual eigenvalues and eigenvectors based on only $\boldsymbol{H}_k$ and $\boldsymbol{Q}_k$:

General Multidimensional Optimization

- Steepest descent: minimize f in the direction of the negative gradient:

- Given quadratic optimization problem $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$ where $\boldsymbol{A}$ is symmetric positive definite, the error $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$ satisfies

  $$||\boldsymbol{e}_{k+1}||_{\boldsymbol{A}} =$$

  - When sufficiently close to a local minima, general nonlinear optimization problems are described by such an SPD quadratic problem.
  - Convergence rate depends on the conditioning of $\boldsymbol{A}$, since

Gradient Methods with Extrapolation

- We can improve the constant in the linear rate of convergence of steepest descent by leveraging *extrapolation methods*, which consider two previous iterates (maintain *momentum* in the direction $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$):

- The *heavy ball method*, which uses constant $\alpha_k = \alpha$ and $\beta_k = \beta$, achieves better convergence than steepest descent:

Conjugate Gradient Method

- The *conjugate gradient method* is capable of making the optimal (for a quadratic objective) choice of α_k and β_k at each iteration of an extrapolation method:

- *Parallel tangents* implementation of the method proceeds as follows

Krylov Optimization

- Conjugate Gradient finds the minimizer of $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$ (which satisfies optimality condition $\boldsymbol{A}\boldsymbol{x} = -\boldsymbol{c}$) within the Krylov subspace of $\boldsymbol{A}$:

Newton's Method

- Newton's method in n dimensions is given by finding minima of n-dimensional quadratic approximation using the gradient and Hessian of f:

Nonlinear Least Squares

- An important special case of multidimensional optimization is *nonlinear least squares*, the problem of fitting a nonlinear function $f_{\boldsymbol{x}}(t)$ so that $f_{\boldsymbol{x}}(t_i) \approx y_i$:

- We can cast nonlinear least squares as an optimization problem to minimize residual error and solve it by Newton's method:

Gauss-Newton Method

- The Hessian for nonlinear least squares problems has the form:

- The *Gauss-Newton* method is Newton iteration with an approximate Hessian:

Tensors

- A *tensor* $\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ has

- Order d tensors represent d-dimensional arrays

Reshaping Tensors

When using tensors, it is often necessary to transition between high-order and low-order representations of the same object

- Recall for a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ its *unfolding* is given by

$$\boldsymbol{v} = \operatorname{vec}(\boldsymbol{A}) \Rightarrow$$

- A tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ can be fully unfolded the same way

$$\boldsymbol{v} = \operatorname{vec}(\boldsymbol{\mathcal{T}}) \Rightarrow$$

- Often we also want to *fold* tensors into higher-order ones
- Generally, we can *reshape* (fold or unfold) any tensor

$$\boldsymbol{\mathcal{U}} = o_{n_1 \times \cdots \times n_d}(\boldsymbol{\mathcal{V}}) \Rightarrow$$

Canonical Polyadic (CP) Decomposition

- A rank R *CP decomposition* of an $s \times s \times s \times s$ tensor is

- We can represent the CP using the following *tensor diagram*:

- Finding an approximate tensor decomposition corresponds to a nonlinear least squares problem: