Matrix Computations in Machine Learning

Inderjit S. Dhillon
University of Texas at Austin

International Conference on Machine Learning (ICML)
June 18, 2009

Outline

- Overview of Matrix Computations
- Matrix Computations in Machine Learning
  - Traditional — Regression, PCA, Spectral Graph Partitioning, etc.
  - Specialized — Lasso, NMF, Co-clustering, Kernel Learning, etc.
- Resources
- Discussion

Matrix Computations

- Linear Systems of Equations
  - Linear Least Squares
  - Dense versus Sparse — Gaussian Elimination versus Iterative Methods
  - Further Structure (Toeplitz, Hankel, Low Displacement Rank.....)
- Eigenvalue Problems
  - SVD
  - Dense versus Sparse — "Direct" Methods versus Iterative Methods
  - Symmetric versus Non-Symmetric
  - Further Structure (Tridiagonal, Banded)

Computational/Accuracy Issues

- Mathematician versus Numerical Analyst
  - Mathematician: To solve $Ax = b$, compute A^{-1}, and set $x = A^{-1}b$
  - Numerical Analyst: Use $A = PLU$ (Gaussian Elimination with Partial Pivoting) to solve for x
  - Mathematician: To solve for eigenvalues of A, form characteristic polynomial $det(\lambda I - A)$ and solve for its roots
  - Numerical Analyst: Use orthogonal transformations to transform A to either tridiagonal form or Hessenberg form, and then solve for eigenvalues of reduced form

Key Issues in Matrix Computations

- Computation Time
  - Factors of Two Matter! GEPP preferred over GECP.
  - Operation Counts
  - Memory Hierarchy (algorithms based on BLAS3 perform better)
  - LAPACK versus LINPACK
- Accuracy
  - Algorithms in Floating Point can have surprising behavior
  - Examples – Eigenvalues through Characteristic Polynomial, Gram-Schmidt Orthogonalization, Loss of Orthogonality in Lanczos Algorithms
  - Forward Stability usually not possible
  - Backward Error Analysis – pioneered by Wilkinson in 1950's, 1960's
  - Need to understand Floating Point Arithmetic
- Structure
  - Dense versus Sparse, Tridiagonal versus Banded, Toeplitz, Hankel.....
  - Has big impact on computation time and accuracy issues

Backward Error Analysis

- Consider $Ax = b$
- Suppose $\hat{x}$ is the computed answer
- Question: How close is $\hat{x}$ to x?
- Correct but Pessimistic Answer: $\hat{x}$ can be wrong in all digits for even the best algorithm
- Backward Error Analysis to the Rescue:
  - Show that the computed answer is EXACT for the perturbed problem:

$$(A + \delta A)\hat{x} = b + \delta b$$

  - Conditioning: How does the solution change when problem is perturbed?

$$\frac{\|\hat{x} - x\|}{\|x\|} \approx cond(A) \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right)$$

  - Separates Properties of Algorithm from Properties of Problem

Matrix Computations in Machine Learning

Regression

- Given: Training data $(\mathbf{x}_i, y_i)$, $i = 1, 2, \ldots, N$, where $\mathbf{x}_i \in R^d$, $y_i \in R$
- Goal: Find Linear Predictor: $\beta_0 + \beta_1 x_1 + \cdots \beta_d x_d$, where $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$ is an unseen data point
- Formulation: Linear Least Squares Problem:

$$\text{Solve} \quad \min_{\beta} \|X\beta - \mathbf{y}\|_2.$$

- Ridge Regression:

$$\text{Solve} \quad \min_{\beta} \|X\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_2^2.$$

- Algorithms (in order of increasing cost and increasing accuracy):
  - Normal Equations
  - QR decomposition
  - SVD

PCA

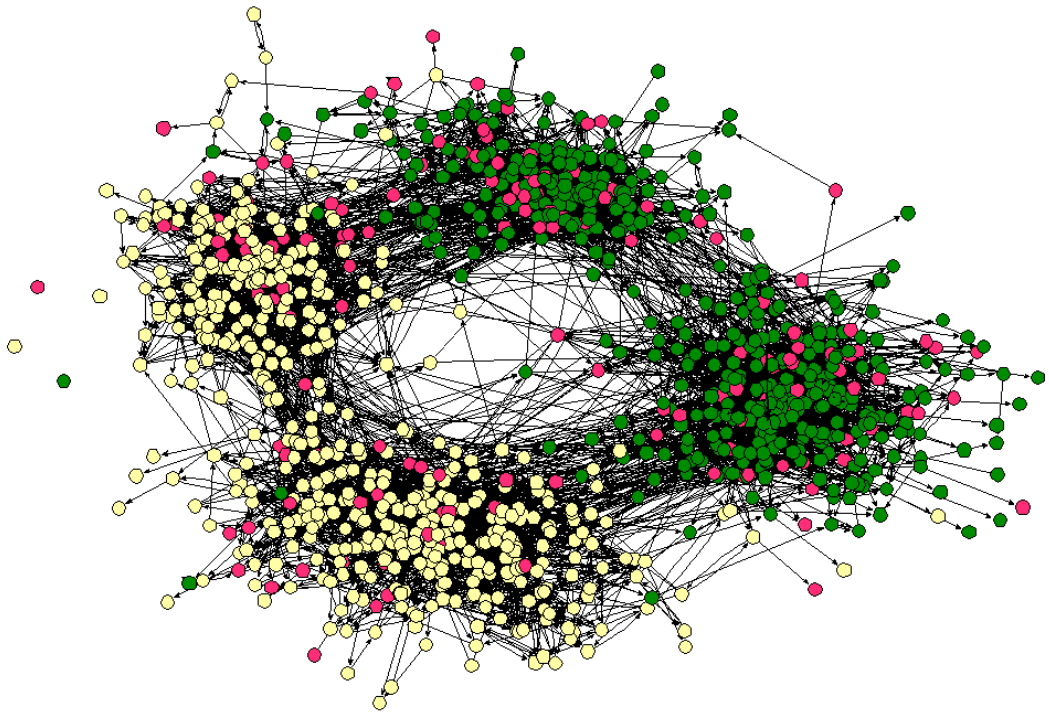
- Given: $\mathbf{x}_i$, $i = 1, 2, \ldots, N$, where $\mathbf{x}_i \in R^d$
- Goal: Find Direction(s) of Largest Variance
- Formulation:

$$\text{Solve} \quad \max_{Y^T Y = I_k} \text{trace}(Y^T (X - me^T)(X - me^T)^T Y).$$

- Solution: Leading Eigenvectors of $(X - me^T)(X - me^T)^T$ (Equivalently, leading left singular vectors of $X - me^T$)
- Algorithms when X is dense:
  - QR Algorithm (`eig` in MATLAB)
  - Divide & Conquer Algorithm
  - **MR^3** Algorithm (fastest)
- When X is sparse
  - Lanczos and its Variants
- Applications: Dimensionality Reduction, Visualization, LSI, Eigenfaces, Eigengenes, Eigenducks, Eigenlips.....

Graph Partitioning/Clustering

- In many applications, goal is to partition nodes of a graph:



High School Friendship Network

[James Moody. American Journal of Sociology, 2001]

Spectral Graph Partitioning

- Partitioning Objective: Minimize Cut subject to Equal Partition Sizes
- Applications in Circuit Layout and Parallel Computing
- Take a real relaxation of the Objective
- Globally optimal solution of the relaxed problem is given by eigenvectors of Graph Laplacian
- Post-process eigenvectors to obtain a discrete partitioning

- Long History
  - [Hall, 1970] *An r-dimensional quadratic placement algorithm*, Manag Sci
  - [Donath & Hoffman, 1972] *Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices*, IBM Techn. Disclosure Bull.
  - [Fiedler, 1973] *Algebraic connectivity of graphs*, Czech Math Journal
  - [Pothen, Simon & Liou, 1990] *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl.

"Specialized" Matrix Computations

Matrix Computations in Machine Learning

Traditional Problems

- Regression, PCA, Spectral Graph Partitioning

"Specialized" Problems that arise in Machine Learning

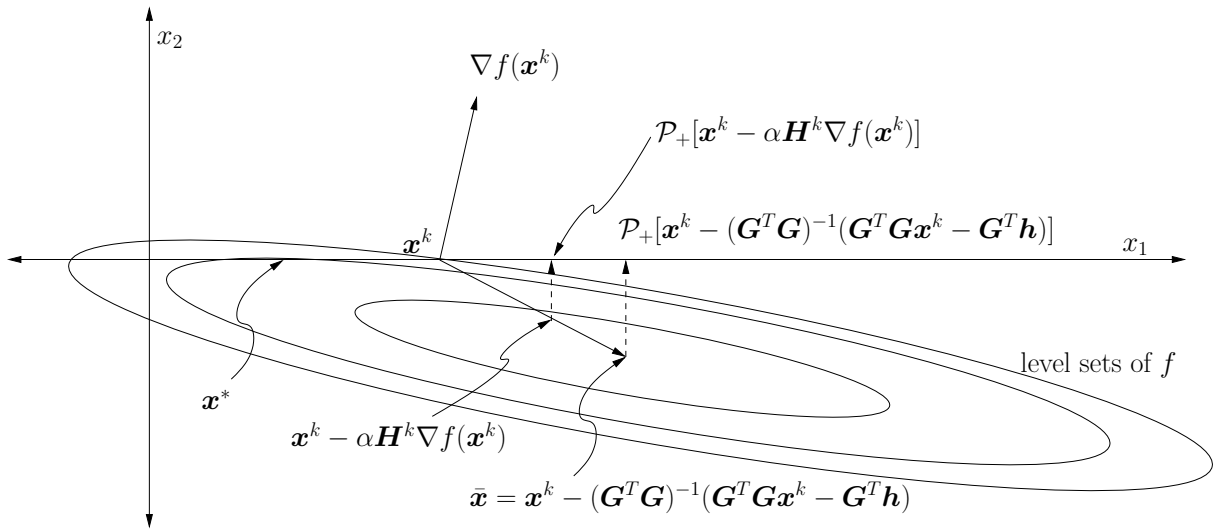
- Intersection of Numerical Optimization & Matrix Computations
  - Lasso – L1-regularized least squares
  - **Non-negative Matrix Factorization (NMF)**
  - Sparse PCA
  - Matrix Completion
  - Co-clustering
  - **Metric/Kernel Learning**
- Machine Learning to the Rescue
  - **Multilevel Graph Clustering**

Non-Negative Matrix Approximation

- Suppose data matrix $A \geq 0$
- NMF problem: Given A and k, solve

$$\min_{B,C \geq 0} \|A - BC\|_F^2$$

- Constrained Optimization Problem



- [Kim, Sra, Dhillon] *Fast Newton-type Methods for the Least Squares Nonnegative Matrix Approximation Problem*, SDM, 2007.

Graph Clustering

- How do we measure the quality of a graph clustering?
- Could simply minimize the *edge-cut* in the graph
  - Can lead to clusters that are highly unbalanced in size
- Could minimize the *edge-cut* in the graph while constraining the clusters to be equal in size
  - Not a natural restriction in data analysis

- Popular objectives include normalized cut and ratio cut:

$$\text{Normalized Cut:} \qquad \text{minimize} \quad \sum_{i=1}^{c} \frac{\text{links}(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{\text{degree}(\mathcal{V}_i)}$$

$$\text{Ratio Cut:} \qquad \text{minimize} \quad \sum_{i=1}^{c} \frac{\text{links}(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{|\mathcal{V}_i|}$$

[Shi & Malik, IEEE Pattern Analysis & Machine Intelligence, 2000]

[Chan, Schlag & Zien, IEEE Integrated Circuits & Systems, 1994]

Machine Learning to the Rescue

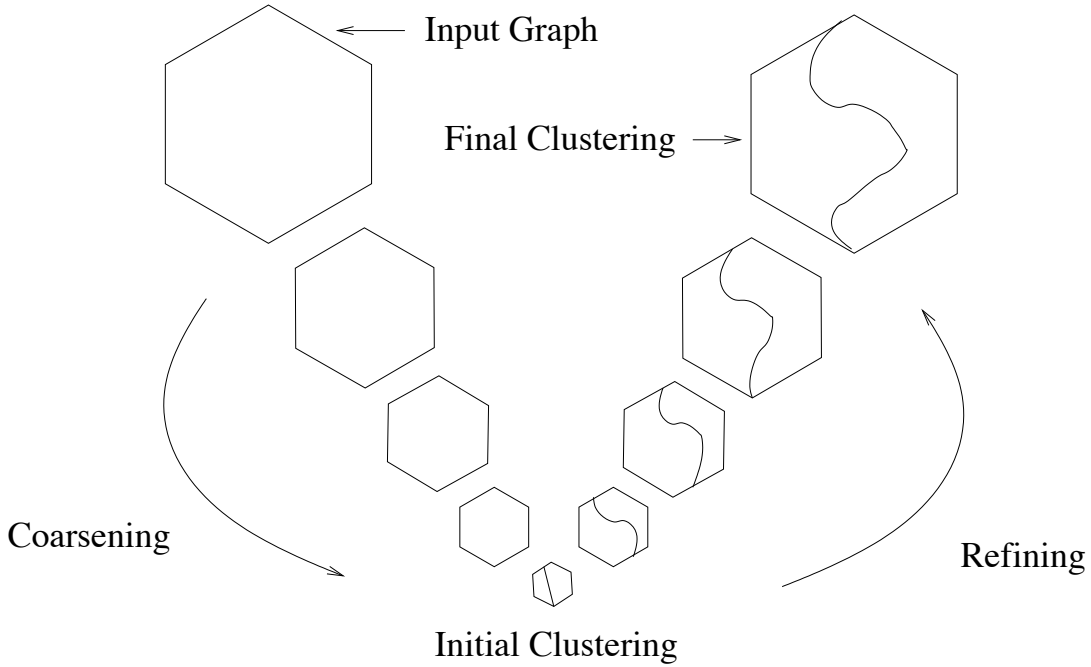
- Traditional Approach to minimize Normalized Cut: Compute eigenvectors of normalized Graph Laplacian
- *Exact Mathematical Equivalence*: Normalized Cut Objective is special case of Weighted Kernel-Kmeans Objective

Our Multilevel Approach:

- Phase I: Coarsening
  - Coarsen graph by merging nodes to form smaller and smaller graphs
  - Use simple greedy heuristic specialized to each graph cut objective
- Phase II: Base Clustering
  - Once the graph is small enough, perform a base clustering
  - Variety of techniques possible for this step
- Phase III: Refining
  - Uncoarsen the graph, level by level
  - Use weighted kernel k-means to refine the clusterings at each level
  - Input to weighted kernel k-means is clustering from previous level

Multilevel Graph Clustering

- Overview of our approach:



- [Dhillon, Guan & Kulis, 2007] *Weighted Graph Cuts without Eigenvectors: A Multilevel Approach*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2007.
- **Graclus** software: freely downloadable from the web

Metric Learning: Example

Similarity by Person(identity) or by Pose

Metric Learning: Formulation

- **Goal:**

$$\begin{aligned}
&\min_{\Sigma} \text{loss}(\Sigma, \Sigma_0) \\
&(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \leq u \quad \text{if } (i,j) \in S \text{ [similarity constraints]} \\
&(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \geq \ell \quad \text{if } (i,j) \in D \text{ [dissimilarity constraints]}
\end{aligned}$$

- Learn spd matrix Σ that is "close" to the baseline spd matrix Σ_0
- Other linear constraints on Σ are possible
- Constraints can arise from various scenarios
  - Unsupervised: Click-through feedback
  - Semi-supervised: must-link and cannot-link constraints
  - Supervised: points in the same class have "small" distance, etc.
- QUESTION: What should "loss" be?

Bregman Matrix Divergences

- Define

$$D_\varphi(X, Y) = \varphi(X) - \varphi(Y) - \text{trace}((\nabla\varphi(Y))^T(X - Y))$$

- Squared Frobenius norm: $\varphi(X) = \|X\|_F^2$. Then

$$D_F(X, Y) = \frac{1}{2}\|X - Y\|_F^2$$

- von Neumann Divergence: For $X \succeq 0$, $\varphi(X) = \text{trace}(X \log X)$. Then

$$D_{vN}(X, Y) = \text{trace}(X \log X - X \log Y - X + Y)$$

  - also called quantum relative entropy

- LogDet divergence: For $X \succ 0$, $\varphi(X) = -\log\det X$. Then

$$D_{\ell d}(X, Y) = \text{trace}(XY^{-1}) - \log\det(XY^{-1}) - d$$

Algorithm: Bregman Projections for LogDet

- **Goal:**

$$\begin{aligned}
&\min_{\boldsymbol{\Sigma}} D_{\ell d}(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}_0) \\
&(\mathbf{x}_i - \mathbf{x}_j)^T \boldsymbol{\Sigma} (\mathbf{x}_i - \mathbf{x}_j) \leq u \quad \text{if } (i,j) \in S \text{ [similarity constraints]} \\
&(\mathbf{x}_i - \mathbf{x}_j)^T \boldsymbol{\Sigma} (\mathbf{x}_i - \mathbf{x}_j) \geq \ell \quad \text{if } (i,j) \in D \text{ [dissimilarity constraints]}
\end{aligned}$$

- **Algorithm:** Cyclic Bregman Projections (successively onto each linear constraint) — converges to globally optimal solution

- Use Bregman projections to update the Mahalanobis matrix:

$$\begin{aligned}
\min_{\boldsymbol{\Sigma}} \quad & D_{\ell d}(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}_t) \\
\text{s.t.} \quad & (\mathbf{x}_i - \mathbf{x}_j)^T \boldsymbol{\Sigma} (\mathbf{x}_i - \mathbf{x}_j) \leq u
\end{aligned}$$

- Can be solved by rank-one update:

$$\boldsymbol{\Sigma}_{t+1} = \boldsymbol{\Sigma}_t + \beta_t \boldsymbol{\Sigma}_t (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \boldsymbol{\Sigma}_t$$

Algorithm: Details

- Exploits properties of LogDet and Sherman-Morrison-Woodbury formula
- Advantages:
  - Simple, closed-form projections
  - Automatic enforcement of positive semidefiniteness
  - No eigenvector calculation or need for SDP
  - Easy to incorporate slack for each constraint
- Question: Can the Algorithm be Kernelized?

Equivalence with Kernel Learning

LogDet Formulation (1)

$$\begin{aligned} \min_{\boldsymbol{\Sigma}} \quad & D_{\ell d}(\boldsymbol{\Sigma}, I) \\ \text{s.t.} \quad & \operatorname{tr}(\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \leq u \\ & \operatorname{tr}(\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \geq \ell \\ & \boldsymbol{\Sigma} \succeq 0 \end{aligned}$$

Kernel Formulation (2)

$$\begin{aligned} \min_{K} \quad & D_{\ell d}(K, K_0) \\ \text{s.t.} \quad & \operatorname{tr}(K(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T) \leq u \\ & \operatorname{tr}(K(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T) \geq \ell \\ & K \succeq 0 \end{aligned}$$

- (1) optimizes w.r.t. the $d \times d$ Mahalanobis matrix $\boldsymbol{\Sigma}$
- (2) optimizes w.r.t. the $N \times N$ kernel matrix K
- Let $K_0 = X^T X$, where X is the input data
- Let $\boldsymbol{\Sigma}^*$ be optimal solution to (1) and K^* be optimal solution to (2)
- **Theorem:** $K^* = X^T \boldsymbol{\Sigma}^* X$, and $\boldsymbol{\Sigma}^* = I + XMX^T$
- Consequence: Kernel Learning generalizes to unseen data

- [Davis, Kulis, Jain, Sra, Dhillon, 2007], *Information-Theoretic Metric Learning*, ICML.

Resources and Discussion

Books

- [Trefethen and Bau, 1997], *Numerical Linear Algebra*
- [G. Golub and C. Van Loan, 1996], *Matrix Computations* — Encylopedic reference
- [Watkins, 2002], *Fundamentals of Matrix Computations*
- [Demmel, 1997],*Applied Numerical Linear Algebra*
- [Parlett, 1998], *Symmetric Eigenvalue Problem*
- [Pete Stewart] – 6 Books on Matrix Computations
- [Gil Strang] – Excellent textbooks on linear algebra (esp., undergraduate), applied mathematics, computational science — video lectures available on MIT course page
- [Horn & Johnson] – Matrix Analysis
- [Barrett et al, 1994], *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*
- [Bai et al, 2000], *Templates for the Solution of Eigenvalue Problems: A Practical Guide*

Software

Dense Matrix Computations:

- Very well-developed public-domain software
- LAPACK, ScaLAPACK
- FLAME, PLAPACK
- Very efficient BLAS3 Code available
- LINPACK
- EISPACK

Sparse Matrix Computations:

- Less well-developed public-domain software, but still available
- Netlib
- Iterative Linear Solvers
- ARPACK — Sparse Eigenvalue Solver
- Sparse SVD – SVDPACKC, PROPACK

Discussion

Matrix Computations

- Well-established area (from 1950s)
- Problems have clear mathematical goals that can be directly optimized
- Theoretical analysis of computation time & accuracy
- Success often based on choosing right representation of problem
- Industry-strength public-domain software

Machine Learning

- Relatively newer area
- Problems have harder objectives that can't always be directly optimized
- Little analysis of accuracy of algorithms
- Little work on choice of representation
- Some public-domain software