Sparsity in Linear Least Squares

Graph Theoretic Approaches to Sparse Factorization

Manmohan Krishna Chandraker

CSE 252C, Fall 2004, UCSD
Linear Least Squares Problem

• Find \( x \in \mathbb{R}^n \) that minimizes

\[
\min_x \|Ax - b\|, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m, \quad m \geq n.
\]

• Residual vector, \( r = b - Ax \).

• Sparse LLS: \( A \) is sparse.
Is Sparsity Useful?

- Electric grids
- Geodetic measurements
- Bundle adjustment
Characterization of LS solutions

• Normal Equations
  • \( x \) is a solution to the Least Squares problem if and only if
  \[
  A^\top Ax = A^\top b
  \]

• Solution method : Cholesky Decomposition

• QR decomposition
  • \( \min_x \|Ax - b\| = \min_x \|Q^\top (Ax - b)\| \) for \( Q \in SO(m) \).
Time Complexity of Direct Methods

- Structure of $A$ influences choice of algorithm.
  \[ Ax = b \]
  \[ A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \]

- Dense - Gaussian elimination: $\frac{2}{3}n^3 + \mathcal{O}(n^2)$ flops.
- Symmetric, positive definite - Cholesky decomposition: $\frac{1}{3}n^3 + \mathcal{O}(n^2)$ flops.
- Triangular - Simple substitution: $n^2 + \mathcal{O}(n)$. 

Sparse Linear Least Squares – p.5
What kind of sparsity is useful?

- When there are $\mathcal{O}(n)$ non-zero entries.
  - Sparse data structures include more storage overhead.
  - Arithmetic operations are slower (due to indirect addressing).
- When the sparsity has a pattern.
Sparse Data Structures

Static, compressed row storage

$$A = \begin{bmatrix}
a_{11} & 0 & a_{13} & 0 & 0 \\
a_{21} & a_{22} & 0 & 0 & 0 \\
0 & 0 & a_{33} & 0 & 0 \\
0 & a_{42} & 0 & a_{44} & 0 \\
0 & 0 & 0 & a_{54} & a_{55} \\
0 & 0 & 0 & 0 & a_{65}
\end{bmatrix}$$

$$AC = (a_{11}, a_{13} \mid a_{22}, a_{21} \mid a_{33} \mid a_{42}, a_{44} \mid a_{54}, a_{55} \mid a_{65})$$

$$IA = (1, 3, 5, 6, 8, 10, 11)$$

$$JA = (1, 3, 2, 1, 3, 2, 4, 4, 5, 5)$$
**Gaussian Elimination: Fill-in**

**Fill-in** : Non-zero elements created by Gaussian elimination.

\[
A = A^{(1)} = \begin{bmatrix}
a & r^T \\
c & \bar{A}
\end{bmatrix}
\quad \text{where} \quad a \in \mathbb{R}^{1 \times 1}, \quad \bar{A} \in \mathbb{R}^{(n-1) \times (n-1)}
\]

\[
A^{(1)} = \begin{bmatrix}
1 & 0 \\
c & a
\end{bmatrix} \begin{bmatrix}
a & r^T \\
0 & A^{(2)}
\end{bmatrix} \Rightarrow A^{(2)} = \bar{A} - \begin{pmatrix} cr^T \\
\frac{c}{a} \end{pmatrix}
\]

Repeat same for \( A^{(2)} \), say, \( A^{(2)} = L_2U_2 \).
Processing Order and Fill-in

- Column order greatly affects fill-in.

\[
\begin{bmatrix}
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\end{bmatrix}
\]

- Find the column order that minimizes fill-in:
Processing Order and Fill-in

- Column order greatly affects fill-in.

\[
\begin{bmatrix}
\times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \\
\times & \times & \\
\times & \\
\end{bmatrix}
\]

- Find the column order that minimizes fill-in:
  
  NP-complete!!
Steps in a Sparse LS Problem

Symbolic factorization

• Find the structure of $A^TA$.
• Determine a column order that reduces fill-in.
• Predict the sparsity structure of the decomposition and allocate storage.

Numerical solution

• Read numerical values into the data structure.
• Do a numerical factorization and solve.
Graph for Sparse Symmetric Matrix

- A node $v_i$ for each column $i$
- $(v_i, v_j) \in E \iff a_{ij} \neq 0$
Predicting Structure of $A^\top A$

- $A^\top A = \sum_{i=1}^{m} a_i a_i^\top$ where $a_i^\top = i$-th row of $A$.
- $G(A^\top A) = \text{direct sum of } G(a_i^\top a_i)$,
  \[ i = 1, \ldots, m. \]
- Non-zeros in $a_i^\top$ form a clique subgraph.
Predicting Structure of Cholesky Factor $R$:

Elimination Graphs: Represent fill-in during factorization.

\[ \begin{align*}
5 & \quad 4 \\
1 & \quad 2 \\
3 & \quad 6 \\
6 & \quad 7
\end{align*} \quad \begin{align*}
5 & \quad 4 \\
2 & \\
3 & \quad 6 \\
6 & \quad 7
\end{align*} \quad \begin{align*}
5 & \quad 4 \\
3 & \quad 6 \\
6 & \quad 7
\end{align*} \quad \begin{align*}
5 & \quad 4 \\
4 & \quad 6 \\
7
\end{align*} \]
Filled Graph

- $G_F(A)$: Direct sum of elimination graphs.
Structure of Cholesky Factor

- Filled graph bounds structure of Cholesky factor
  \[ G(R + R^\top) \subset G_F(A) \]

- Equality when no-cancellation holds.

\[
\begin{bmatrix}
  x & x & x & x & x \\
  x & x & x & x & x \\
  x & x & x & x & x \\
  x & x & x & x & x \\
  x & x & x & x & x \\
\end{bmatrix}
\quad \begin{bmatrix}
  x & x & x & x \\
  x & x & x & x \\
  x & x & x & x \\
  x & x & x & x \\
  x & x & x & x \\
\end{bmatrix}
\]
Efficient Computation of Filled Graph

- **Theorem**: Let $G(A) = (V, E)$ be an ordered graph of $A$. Then $(v_i, v_j)$ is an edge of the filled graph $G_F(A)$ if and only if $(v_i, v_j) \in E$ or there is a path in $G(A)$ from vertex $v_i$ to $v_j$ passing only through vertices with numbering less than $\min\{i, j\}$.

- Allows construction of filled graph in $O(n|E|)$ time.
Fill Minimizing Column Orderings

- Reordering rows of $A$ does not affect $A^\top A$.
- Column reordering in $A$ keeps number of non-zeros same in $A^\top A$.
- Greatly affects number of non-zeros in $R$.
- Heuristics to reduce fill-in:
  - Minimum degree ordering
  - Nested dissection orderings.
Minimum Degree Ordering

Let $G^{(0)} = G(A)$.

for $i = 1, \cdots, (n - 1)$ :

Select a node $v$ in $G^{(i-1)}$ of minimal degree.
Choose $v$ as next pivot.
Update elimination graph to get $G^{(i)}$.

end
Without Ordering

Order 1, 2, 3, 4, 5, 6, 7 → fill-in = 10.
With Minimum Degree Ordering

Order 4, 5, 6, 7, 1, 2, 3 → fill-in = 0 !!
Does not always work!

- Order 1, 2, ⋅ ⋅ ⋅ , 9 → fill-in = 0.
- Minimum degree node : 5 → fills in (4, 6).
Nested Dissection Ordering

- Reorder to obtain block angular sparse matrix.

Level 1 Dissection

\[ A_1 \quad B \quad A_2 \]

Level 2 Dissection

\[ \begin{align*} A_1 & \quad B \quad A_2 \\ A_3 & \quad D \quad A_4 \\ B & \quad C \end{align*} \]
Nested Dissection: Block Structure and Elimination Tree

\[ A = \begin{bmatrix} A_1 & B_1 \\ A_2 & B_2 \end{bmatrix} \]

\[ A = \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \end{bmatrix} \]

\[ A = \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \\ A_3 & A_4 \\ C_3 & C_4 \end{bmatrix} \]

\[ D \]

\[ B \]

\[ C \]

\[ A_1 \]

\[ A_2 \]

\[ A_3 \]

\[ A_4 \]
Numerical Factorization

• Mathematically, Cholesky factor of $A^\top A$ same as $R$ in QR-decomposition of $A$.

• Numerical issues govern choice of algorithm.

• Symbolic factorization same for both.
Numerical Cholesky Factorization

Symbolic Phase

1. Find symbolic structure of $A^\top A$.
2. Find column permutation $P_c$ such that $P_c^\top A^\top A P_c$ has sparse Cholesky factor $R$.
3. Perform symbolic factorization and generate storage structure for $R$.

Numerical Phase

1. Compute $B = P_c^\top A^\top A P_c$ and $c = P_c^\top A^\top b$ numerically.
2. Compute $R$ numerically and solve $R^\top z = c$, $Ry = z$, $x = P_c y$. 
Cholesky vs QR

- Symbolic computation: No pivoting for needed for Cholesky.
- Loss of information in $A^\top A$ and $A^\top b$.
- Condition number squared in $A^\top A$.
- Inefficient memory utilization: both rows and columns accessed in elimination.
Sparse QR algorithm

Symbolic Phase

1. Determine structure of $\mathbf{R}$ and allocate storage.
2. Determine row permutation $\mathbf{P}_r$ and reorder rows to get $\mathbf{P}_r\mathbf{A}\mathbf{P}_c$.

Numerical Phase

1. Apply orthogonal transformations to $(\mathbf{P}_r\mathbf{A}\mathbf{P}_c, \mathbf{P}_r\mathbf{b})$ to get $\mathbf{R}$.
2. Solve $\mathbf{Ry} = \mathbf{c}$ and $\mathbf{x} = \mathbf{P}_c\mathbf{y}$. 
QR-Decomposition

- Dense problems: Sequence of Householder reflections.
- Expensive computation and storage.
- Row sequential QR-decomposition.
Row Sequential QR Algorithm

- Row-oriented Givens rotations for orthogonalization avoid intermediate fill-in.

\[
\begin{bmatrix}
R_{k-1} \\
a_k^\top
\end{bmatrix} = \begin{bmatrix}
\times & 0 & \times & 0 & 0 & \times & 0 & 0 \\
\times & 0 & \oplus & \times & 0 & 0 & 0 \\
\times & 0 & \times & 0 & 0 & 0 \\
\times & \oplus & 0 & \times & 0 & 0 \\
\times & \oplus & 0 & 0 \\
0 & \times & 0 & \oplus & 0 & \oplus & \oplus \\
\end{bmatrix}
\]
Summary

• Exploiting sparsity: storage and time savings.

• A sparse LS problem can be subdivided into a symbolic phase and a numerical phase.

• The symbolic phase:
  • Determines a column ordering that makes the Cholesky factor sparse.
  • Determines the structure of the Cholesky factor.

• The numerical phase:
  • Uses specialized orthogonalization algorithms to determine the numerical factorization.