Structure and matrix sparsity: Part 1
The simplex method: Exploiting sparsity

Julian Hall

School of Mathematics
University of Edinburgh

jajhall@ed.ac.uk

6 June 2018

Aim

- What should a 2-hour PhD lecture on structure and matrix sparsity achieve?
- Audience members may not have numerical linear algebra (NLA) background
- Discuss key features of NLA techniques for large scale optimization
  - Insight into NLA for those new to it
  - Application of NLA for those who've seen it before
  - Approach a couple of research frontiers

Structure and matrix sparsity: Overview

Part 1

- Exploiting matrix sparsity
  - What is sparsity?
  - Efficient $\boldsymbol{y} = A\boldsymbol{x}$ for sparse A
- Introduction to structure and matrix sparsity in convex optimization
- The simplex method: Exploiting sparsity
  - Inverting B
  - Exploiting hyper-sparsity
  - Forming $\boldsymbol{z} = N^T\boldsymbol{\pi}$ efficiently

Structure and matrix sparsity: Introduction

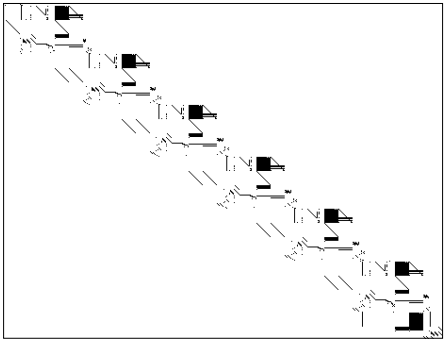
- Discuss structure and matrix sparsity for general LP problem

$$\text{minimize } \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to } A\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0}$$

  Problem has n **variables** and m **constraints**
- May be be modelled directly—or formed as a sub-problem when solving
  - Discrete optimization problems
  - Nonlinear optimization problems
- Modelling process frequently means that most of the entries in A are zero
  - Distribution of the nonzeros is generally non-random so the matrix A has **structure**
  - When it is worth exploiting the existence of these zeros in a computational process the matrix A is said to be **sparse**

Structure and matrix sparsity: Example

There are many LP test problems, one of which is called STAIR



Constraint matrix A for STAIR

- $n = 467$ columns (variables)
- $m = 363$ rows (constraints)
- $\tau = 3856$ nonzeros
- $\frac{\tau}{n} = \frac{3856}{467} = 8.3$ nonzeros/column
- Density $100 \times \frac{3856}{467 \times 363} = 2.3\%$

Matrix is **sparse** and has some repeated **structure**

Structure and matrix sparsity: Computational cost and efficiency

Crucial concept so worth establishing the basics

- Consider evaluating a matrix-vector product $\boldsymbol{y} = A\boldsymbol{x}$ as

$$y_i = \sum_{j=1}^{n} a_{ij} x_j \quad \text{for} \quad i = 1, \ldots, m$$

- For each entry a_{ij} in A there is exactly one multiplication and one addition as the terms $a_{ij}x_j$ are formed and added: total computational cost is $O(mn)$
- If most of the entries in A are zero then most multiplications involve zero so lead to zero being added to the summation
- Cost of calculations with zero will dominate
- Avoiding these makes the calculation much more efficient

Structure and matrix sparsity: Exploiting sparsity

- Store only the (τ) nonzeros values in the matrix
- Can do this row-by-row, with each nonzero value paired with the index of the column in which it is located
- For example

$$A = \begin{bmatrix} 11 & 12 & & 14 & \\ & 22 & 23 & & \\ 31 & & 33 & & 35 \end{bmatrix}$$

may be represented using arrays of

Values $\boldsymbol{v}$	11	12	14	22	23	31	33	35
Indices $\boldsymbol{j}$	1	2	4	2	3	1	3	5
Pointers $\boldsymbol{p}$	1	4	6	9				

Structure and matrix sparsity: Exploiting sparsity

In general

- Evaluate $\boldsymbol{y} = A\boldsymbol{x}$ as

$$y_i = \sum_{k=p_i}^{p_{i+1}-1} v_k x_{j_k} \quad \text{for} \quad i = 1, \ldots, m$$

- For each nonzero entry in A there is exactly one multiplication and one addition: total computational cost is $O(\tau)$
- Computation is more efficient since the sparsity of A has been exploited

Structure and matrix sparsity: In convex optimization

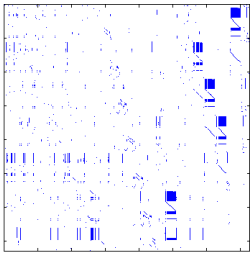
- Solving systems of equations is the major computational cost of simplex and IPM
  - Revised simplex method requires
    - Solution of $B\widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$
    - Solution of $B^T\boldsymbol{\pi} = \boldsymbol{c}_B$
  - Interior point methods require
    - Solution of $(A\Theta A^T)\boldsymbol{x} = \boldsymbol{b}$
- Essential to exploit matrix sparsity
- Special solution methods exist to exploit matrix structure

Simplex method: Exploiting sparsity

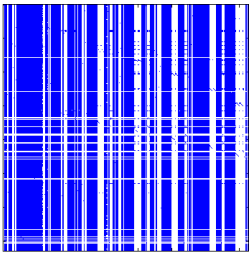
- **Recall:** Revised simplex method requires
  - Solution of $B\widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$
  - Solution of $B^T\boldsymbol{\pi} = \boldsymbol{c}_B$
- Why not form B^{-1} and evaluate $B^{-1}\boldsymbol{a}_q$ and $B^{-T}\boldsymbol{c}_B$?
  - For a *full* matrix B, forming B^{-1} and $B = LU$ both cost $O(m^3)$
  - For a *sparse* matrix B, forming B^{-1} is computationally inefficient

Inverting the matrix B: Introduction

Example: A basis matrix B for the LP test problem STAIR



- Matrix $B \in \mathbb{R}^{363\times363}$
- 3279 nonzeros
- Density 2.5%



- Matrix B^{-1}
- 80284 nonzeros
- Density 61%
- Fill factor $\dfrac{80284}{3279} = 24.5$

Inverting the matrix B: Why not form B^{-1}?

- For a typical sparse matrix B, the matrix B^{-1} is dense
  - Forming B^{-1} takes $O(m^3)$ time
  - Storing B^{-1} requires $O(m^2)$ memory
- Given B^{-1}, forming $B^{-1}\boldsymbol{b}$ takes $O(m^2)$ time
- This is all *very* expensive if m is large
- Don't form B^{-1} explicitly and then compute $[B^{-1}]\boldsymbol{b}$
- Find a way to solve $B\boldsymbol{x} = \boldsymbol{b}$ **directly**

Inverting the matrix B: Forming $B = LU$

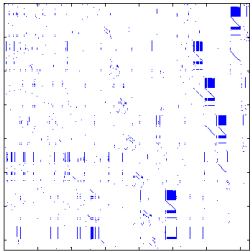
- Decompose B into product LU of
  - Lower triangular matrix L with unit diagonal entries
  - Upper triangular matrix U
- Then $B\boldsymbol{x} = \boldsymbol{b}$ is solved as

$$L\boldsymbol{y} = \boldsymbol{b} \quad \text{then} \quad U\boldsymbol{x} = \boldsymbol{y}$$

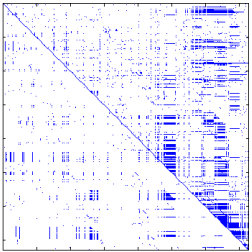
  where triangular systems are solved by substitution
- Can form L and U directly, but generally use **Gaussian elimination** (GE)

Inverting the matrix B: Forming $B = LU$ using GE

Cost of forming $B = LU$ and resulting fill-in is vastly less than for B^{-1}



- Matrix $B \in \mathbb{R}^{363\times363}$
- 3279 nonzeros
- Density 2.5%



- Matrix $\begin{bmatrix} L & U \end{bmatrix}$
- 7850 nonzeros
- Density 6.0%
- Fill factor $\dfrac{7850}{3279} = 2.4$

Inverting the matrix B: Forming $B = LU$ using GE

Use Gaussian elimination (GE) to form $B = LU$

- Stage 1 of GE for general sparse B: can expect
  - Few nonzeros in column 1
  - Few nonzeros in row 1
- GE is cheap:
  - Few nonzeros to eliminate in column 1
  - Do this by adding a multiple of the few nonzeros in row 1
- $B^{(2)}$ will generally have nonzeros in places where $B^{(1)}$ was zero
- Does it matter?

Inverting the matrix B: Forming $B = LU$ using GE

Standard Gaussian elimination can destroy sparsity

For $B = \begin{bmatrix} -1 & -1 & -1 & \dots & -1 & -1 \\ -1 & 1 & & & & \\ -1 & & 1 & & & \\ \vdots & & & \ddots & & \\ -1 & & & & 1 & \\ -1 & & & & & 1 \end{bmatrix}$

- After one stage of standard GE, sparsity is lost: $B^{(2)} = \begin{bmatrix} -1 & -1 & -1 & \dots & -1 & -1 \\ & 2 & 1 & \dots & 1 & 1 \\ & 1 & 2 & \dots & 1 & 1 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 1 & 1 & \dots & 2 & 1 \\ & 1 & 1 & \dots & 1 & 2 \end{bmatrix}$
- Total **fill-in** has occurred
- GE will cost $O(m^3)$

Inverting the matrix B: Forming $B = LU$ using GE

Sparse Gaussian elimination chooses pivots to preserve sparsity

For $B = \begin{bmatrix} -1 & -1 & -1 & \dots & -1 & -1 \\ -1 & 1 & & & & \\ -1 & & 1 & & & \\ \vdots & & & \ddots & & \\ -1 & & & & 1 & \\ -1 & & & & & 1 \end{bmatrix}$

- After one stage of GE pivoting on $b_{nn}^{(1)}$: $B^{(2)} = \begin{bmatrix} 1 & & & & & -1 \\ & 1 & & & & -1 \\ & & 1 & & & -1 \\ & & & \ddots & & \vdots \\ & & & & 1 & -1 \\ & -1 & -1 & \dots & -1 & -2 \end{bmatrix}$
- No **fill-in** has occurred
- Pivoting similarly, GE will cost $O(m)$

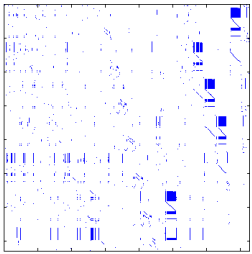
Inverting the matrix B: Forming $B = LU$ using GE

Gaussian elimination with Markowitz pivoting

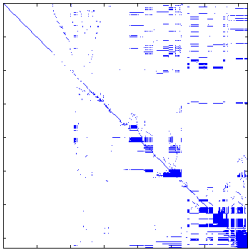
- In stage k of Gaussian elimination $B^{(k)} = \begin{bmatrix} b_{11}^{k} & \dots & b_{1k}^{(k)} & \dots & b_{1m}^{(k)} \\ & \ddots & \vdots & & \vdots \\ & & b_{kk}^{(k)} & \dots & b_{km}^{(k)} \\ & & \vdots & & \vdots \\ & & b_{mk}^{(k)} & \dots & b_{mm}^{(k)} \end{bmatrix}$
- In the trailing square **active** submatrix
  - Let r_i be the number of nonzeros in row i
  - Let c_j be the number of nonzeros in column j
- Max fill-in if $b_{ij}^{(k)}$ is the pivot is the **Markowitz** merit (1957)
$$(r_i - 1)(c_j - 1)$$
- Choose $b_{ij}^{(k)}$ to minimize this

Inverting the matrix B: Forming $B = LU$ using GE

Using sparse (Markowitz) GE to form $B = LU$ is even better



- Matrix $B \in \mathbb{R}^{363\times363}$
- 3279 nonzeros
- Density 2.5%



- Matrix $\begin{bmatrix} L & U \end{bmatrix}$
- 5043 nonzeros
- Density 3.8%
- Fill factor $\dfrac{5043}{3279} = 1.5$

Inverting the matrix B: Using $B = LU$

- GE with Markowitz pivoting identifies **permutation** matrices P and Q such that

$$PBQ = LU$$

  - P and Q are pemutations of the columns of I
  - P and Q are orthogonal so $P^TP = Q^TQ = I$
- Number of nonzeros in L and U is typically only a few more than in B
- Can solve $B\boldsymbol{x} = \boldsymbol{b}$ by permutation and forward/backward substitution as

$$L\boldsymbol{z} = P\boldsymbol{b}; \qquad U\boldsymbol{y} = \boldsymbol{z}; \qquad \boldsymbol{x} = Q\boldsymbol{y}$$

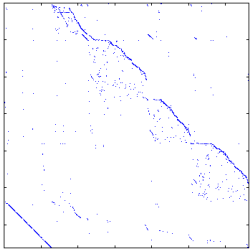
Inverting the matrix B: Using $B = LU$

- Consider L and U column-wise
- In general, each column consists of
  - A pivot value η_i
  - A vector $\boldsymbol{\eta}_i$ with zero i^{th} entry
- Triangular substitution is applied to $\boldsymbol{b}$ as

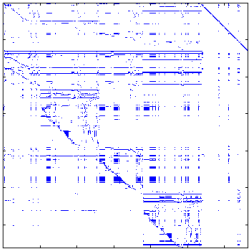
$$
\begin{aligned}
&\texttt{For } i = 1, \ldots, N \\
&\qquad b_i \;&:=\;& \frac{b_i}{\eta_i} \\
&\qquad \boldsymbol{b} \;&:=\;& \boldsymbol{b} - b_i \boldsymbol{\eta}_i \\
&\texttt{End}
\end{aligned}
$$

Hyper-sparsity: Motivation

- B^{-1} is not always dense
- **Example:** From test LP SHELL



- Matrix $B \in \mathbb{R}^{663\times663}$
- 1712 nonzeros
- Density 0.4%



- Matrix B^{-1}
- 10765 nonzeros
- Density 2.4%

Hyper-sparsity: Consequences

- If B^{-1} is sparse then
  - $\boldsymbol{\pi}_p = B^{-T}\boldsymbol{e}_p$ is sparse
  - $\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ is typically sparse
- Sparse B^{-1} is is a property of **hyper-sparse** LP problems
- Still form $B = LU$ using sparse GE rather than form B^{-1}

Hyper-sparsity: Using LU decomposition

- If $B^{-1}\boldsymbol{b}$ is sparse then few columns of L and U will be used
- Must improve efficiency of triangular substitution

```
For i = 1, ..., N
    If b_i ≠ 0 then
        b_i := b_i / η_i
        b   := b − b_i η_i
    End
End
```

- Cost of testing $b_i \neq 0$ can dominate!
- Motivates more sophisticated substitution techniques

Hyper-sparsity: Summary

Hyper-sparsity

- Vectors $\boldsymbol{e}_p$ and $\boldsymbol{a}_q$ are always sparse
- B may be highly reducible; B^{-1} may be sparse
- Vectors $\boldsymbol{\pi}_p$, $\hat{\boldsymbol{a}}_p$ and $\hat{\boldsymbol{a}}_q$ may be sparse
- Efficient implementations must exploit these features

H and McKinnon (1998–2005), Bixby (1999)
Clp, Koberstein and Suhl (2005–2008)

The simplex method: Forming $z = N^T\pi$ efficiently

- Each revised simplex iteration requires $z = N^T\pi$ to compute the reduced costs
- N is the submatrix of $\begin{bmatrix} A & I \end{bmatrix}$ given by $\mathcal{N}$ so is sparse if A is sparse
- Naïvely form $z = N^T\pi$ as

$$z_j = \sum_{i=1}^{m} a_{ij}\pi_i \quad \text{for} \quad j \in \mathcal{N}$$

  Costs $O(mn)$ so is inefficient when A is sparse
- Store A column-wise so N^T is known row-wise, then

$$z_j = \sum_{k=p_j}^{p_{j+1}-1} v_k \pi_{i_k} \quad \text{for} \quad j \in \mathcal{N}$$

  Costs $O(\tau)$ so is much more efficient when A is sparse
- What if the LP is hyper-sparse—so π is sparse?

The simplex method: Forming $z = N^T\pi$ efficiently

$$\text{Forming } z_j = \sum_{k=p_j}^{p_{j+1}-1} v_k \pi_{i_k} \text{ for } j \in \mathcal{N} \text{ costs } O(\tau) \text{ when } A \text{ is sparse}$$

- If π is also sparse, most terms $v_k\pi_{i_k}$ are zero $\Rightarrow$ inefficiency
- Consider forming $\boldsymbol{z}^T = \boldsymbol{\pi}^T N$ with $N = \begin{bmatrix} \boldsymbol{a}_1^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix}$ stored *row-wise*
- Combine rows of N corresponding to nonzeros in $\boldsymbol{\pi}^T$ so

$$\boldsymbol{z}^T = \sum_{\pi_i \neq 0} \pi_i \boldsymbol{a}_i^T$$

- No operations with zeros so much greater efficiency

Structure and matrix sparsity: Part 1: Summary

- Identified what it means to exploit sparsity
- Seen why forming the explicit inverse is inefficient
- Seen how to pivot for sparsity in Gaussian elimination
- Observed the phenomenon of hyper-sparsity
- Gained an insight into some tricks of efficient computational optimization