Whirlwind Tour of LA
Part 1: Some Nitty-Gritty Stuff

David Bindel

2015-01-30
Logistics

- PS1/2 deferred to next Weds/Fri
  - Brandon has OH tonight 5-7
  - I will arrange extra OH early next week (TBA)
- Please keep up with the reading!
- Ask me questions!
Big picture: What’s a matrix?

An array of numbers, or a representation of

- A tabular data set?
- A graph?
- A linear function between vector spaces?
- A bilinear function on two vectors?
- A pure quadratic function?

It’s all the above, plus being an interesting object on its own!

Let’s start concrete.
Basics: Constructing matrices and vectors

\[ x = \begin{bmatrix} 1; & 2 \end{bmatrix}; \quad \text{% Column vector} \]
\[ y = \begin{bmatrix} 1, & 2 \end{bmatrix}; \quad \text{% Row vector} \]
\[ M = \begin{bmatrix} 1, & 2; & 3, & 4 \end{bmatrix}; \quad \text{% 2–by–2 matrix} \]
\[ M = \begin{bmatrix} I, & A \end{bmatrix}; \quad \text{% Horizontal matrix concatenation} \]
Basics: Constructing matrices and vectors

\[ l = \text{eye}(n); \quad \% \text{Build } n\times n \text{ identity} \]
\[ Z = \text{zeros}(n); \quad \% n\times n \text{ matrix of zeros} \]
\[ b = \text{rand}(n,1); \quad \% n\times 1 \text{ random matrix (uniform)} \]
\[ e = \text{ones}(n,1); \quad \% n\times 1 \text{ matrix of ones} \]
\[ D = \text{diag}(e); \quad \% \text{Construct a diagonal matrix} \]
\[ e2 = \text{diag}(D); \quad \% \text{Extract matrix diagonal} \]
Basics: Transpose, rearrangements

% Reshape A to a vector, then back to a matrix
% Note: MATLAB is column–major
avec = reshape(A, prod(size(A)));
A = reshape(avec, n, n);

A = A'; % Conjugate transpose
A = A.'; % Simple transpose

idx = randperm(n); % Random permutation of indices
Ac = A(:,idx); % Permute columns of A
Ar = A(idx,:); % Permute rows of A
Ap = A(idx,idx); % Permute rows and columns
Basics: Submatrices, diagonals, triangles

\[ A = \text{randn}(6,6); \quad \text{\% 6–by–6 random matrix} \]
\[ A(1:3,1:3) \quad \text{\% Leading 3–by–3 submatrix} \]
\[ A(1:2:\text{end},:) \quad \text{\% Rows 1, 3, 5} \]
\[ A(:,3:\text{end}) \quad \text{\% Columns 3–6} \]

\[ \text{Ad} = \text{diag}(A); \quad \text{\% Diagonal of A (as vector)} \]
\[ \text{A1} = \text{diag}(A,1); \quad \text{\% First superdiagonal} \]
\[ \text{Au} = \text{triu}(A); \quad \text{\% Upper triangle} \]
\[ \text{Al} = \text{tril}(A); \quad \text{\% Lower triangle} \]
Basics: Matrix and vector operations

\[ y = d.*x; \quad \% \text{Elementwise multiplication of vectors/matrices} \]

\[ y = x./d; \quad \% \text{Elementwise division} \]

\[ z = x + y; \quad \% \text{Add vectors/matrices} \]

\[ z = x + 1; \quad \% \text{Add scalar to every element of a vector/matrix} \]

\[ y = A*x; \quad \% \text{Matrix times vector} \]

\[ y = x'*A; \quad \% \text{Vector times matrix} \]

\[ C = A*B; \quad \% \text{Matrix times matrix} \]

\[ \% \text{Don’t use inv!} \]

\[ x = A\backslash b; \quad \% \text{Solve } Ax = b \text{ or least squares} \]

\[ y = b/A; \quad \% \text{Solve } yA = b \text{ or least squares} \]
Two basic operations

- Matrix-vector product (matvec): $O(n^2)$
- Matrix-matrix product (matmul): $O(n^3)$
Matvec

A matvec is a collection of dot products.

A matvec is a sum of scaled columns.

Can also think of block rows/columns.

Same (scalar) operations, different order!
Matvec: Diagonal

% Bad idea
D = diag(1:n); % O(n^2) setup
y = D*x; % O(n^2) matvec

% Good idea
d = (1:n)'; % O(n) setup
y = d.*x; % O(n) matvec

- Matrix-vector products are a basic op.
- Can you write the two nested loops?
- Obvious form: y = Ax
- Obvious isn’t always best!
Matvec: Low rank

\[ A = u^*v'; \quad % O(n^2) \]
\[ y = A*x; \]

\[ a = v'*x; \quad % O(n) \]
\[ y = u*a; \]

Don’t form low rank matrices explicitly!
Matvec: Low rank

Write an outer-product decomposition for

- A matrix of all ones
- A matrix of $\pm 1$ in a checkerboard
- A matrix of ones and zeros in a checkerboard
Matvec: Sparse

% Sparse ($O(n) = \text{number nonzeros to form / multiply}$)
\begin{align*}
e &= \text{ones}(n-1,1); \\
T &= \text{speye}(n) - \text{spdiags}(e,-1,n,n) - \text{spdiags}(e,1,n,n);
\end{align*}

% Dense ($O(n^2)$)
\begin{align*}
T &= \text{eye}(n) - \text{diag}(e,-1) - \text{diag}(e,1);
\end{align*}

Will talk about this in more detail – keep it in mind!
From matvec to matmul

Matrix-vector product is a key kernel in sparse NLA. Matrix-matrix product is a key kernel in dense NLA.

Surprisingly tricky to get fast – so let someone else write fast matmul, and use it to accelerate our codes!
Matmul: Inner product version

An entry in $C$ is a dot product of a row of $A$ and column of $B$. 
Matmul: Outer product version

$C$ is a sum of outer products of columns of $A$ and rows of $B$. 

\[
\begin{bmatrix}
\end{bmatrix}
\begin{bmatrix}
\end{bmatrix} = \begin{bmatrix}
\end{bmatrix} \times \begin{bmatrix}
\end{bmatrix}
\]
Matmul: Row-by-row

A row in $C$ is a row of $A$ multiplied by $B$. 

\[
\begin{array}{c}
\text{Row in } \mathbf{C} \\
\end{array} = \begin{array}{c}
\text{Row of } \mathbf{A} \\
\end{array} \times \begin{array}{c}
\text{Matmul} \\
\end{array}
\]
Matmul: Col-by-col

A column in $C$ is $A$ multiplied by a column of $B$. 
Reality intervenes

These arrangements of matmul are theoretically equivalent. What about in practice?

Answer: Big differences due to memory hierarchy.
One row in naive matmul

- Access $A$ and $C$ with stride of $8M$ bytes
- Access all $8M^2$ bytes of $B$ before first re-use
- Poor arithmetic intensity
Engineering strategy: blocking/tiling
Simple model

Consider two types of memory (fast and slow) over which we have complete control.

- \( m \) = words read from slow memory
- \( t_m \) = slow memory op time
- \( f \) = number of flops
- \( t_f \) = time per flop
- \( q = f / m \) = average flops / slow memory access

Time:

\[
ft_f + mt_m = ft_f \left( 1 + \frac{t_m}{t_f} \right)
\]

Larger \( q \) means better time.
How big can \( q \) be?

1. Dot product: \( n \) data, \( 2n \) flops
2. Matrix-vector multiply: \( n^2 \) data, \( 2n^2 \) flops
3. Matrix-matrix multiply: \( 2n^2 \) data, \( 2n^3 \) flops

These are examples of level 1, 2, and 3 routines in *Basic Linear Algebra Subroutines* (BLAS). We like building things on level 3 BLAS routines.
$q$ for naive matrix multiply

$q \approx 2$ (on board)
$q$ for blocked matrix multiply

$q \approx b$ (on board). If $M_f$ words of fast memory, $b \approx \sqrt{M_f/3}$.

Th: (Hong/Kung 1984, Irony/Tishkin/Toledo 2004): Any reorganization of this algorithm that uses only associativity and commutativity of addition is limited to $q = O(\sqrt{M_f})$

Note: Strassen uses distributivity...
Concluding thoughts

- Will not focus on performance *details* here (see CS 5220!)
- Knowing “big picture” issues makes a big difference
  - Order-of-magnitude improvements through blocking ideas
  - Even more possible through appropriate use of structure
- Next time: More theoretical stuff!