Linear Systems

Motivation

- "Matrices are bad, vector spaces are good"
  - Don't think of matrices as a collection of numbers
  - Instead, think of the columns as vectors in a high dimensional space
- We don't have great intuition going from R^1 to R^2 to R^3 to R^n (for large n)
- Thinking about vectors in high dimensional spaces is a good way of gaining intuition about what's going on
- Linear algebra contains a lot of machinery for dealing with, discussing, and gaining intuition about vectors in high dimensional spaces
- We will cover linear algebra from the viewpoint of <u>*understanding higher dimensional spaces*</u>

System of Linear Equations

- System of equations: $3c_1 + 2c_2 = 6$ and $-4c_1 + c_2 = 7$
- Matrix form: $\begin{pmatrix} 3 & 2 \\ -4 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 6 \\ 7 \end{pmatrix}$ or $Ac = b$
- Given A and b, determine c
- Theoretically, there is a unique solution, no solution, or infinite solutions
- Ideally, software would determine whether there was a unique solution, no solution, or infinite solutions; in the last case, it would list a parameterized family of solutions. Unfortunately, this is quite difficult to accomplish.

- Note: in this class, x is used for data, and c is used for unknowns (such as for the unknown parameters of a neural network)

"Zero"

- On the computer, defining "zero" is not straightforward
- When dealing with large numbers (e.g. Avogadro's number: $6.022e23$) zero can be quite large
  - E.g. $6.022e23 - 1e7 = 6.022e23$ in double precision, making $1e7$ behave like "zero"
- When dealing with small numbers (e.g. $1e-23$), "zero" is much smaller
  - In this case, on the order of $1e-39$ in double precision

- Mixing big and small numbers often wreaks havoc on algorithms
- So, we typically <u>non-dimensionalize</u> and <u>normalize</u> to make equations $O(1)$ as opposed to $O(\text{"}big\text{"})$ or $O(\text{"}small\text{"})$

Row/Column Scaling

- Consider: $$\begin{pmatrix} 3e6 & 2e10 \\ 1e-4 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 5e10 \\ 6 \end{pmatrix}$$
- <u>Row Scaling</u> - divide first row by 1e10 to obtain:
$$\begin{pmatrix} 3e-4 & 2 \\ 1e-4 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$
- <u>Column Scaling</u> - define a new variable $c_3 = (1e-4)c_1$ to obtain:
$$\begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c_3 \\ c_2 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$
- The final matrix is much easier to treat with finite precision arithmetic
- Solve for c_3 and c_2; then, $c_1 = (1e4)c_3$

Some Definitions…

- <u>Elements</u> of a matrix are often referred to by their row and column
- For example, a_{ik} is the element of matrix A in row i and column k

- <u>Transpose</u> swaps the row and column of every entry
- A^T moves element a_{ik} to row k column i (and vice versa)
- Non-square matrices change size: $\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

<u>Symmetric Matrices</u> have $A^T = A$ meaning that $a_{ik} = a_{ki}$ for all i and k

Square Matrices

- A <u>size</u> mxn matrix has m rows and n columns
- For now, let's just consider <u>square</u> nxn matrices
- We will consider non-square (rectangular) matrices with $m \neq n$ a bit later

Solvability

- Singular – A is singular when it is not invertible (does not have an inverse)
- Various ways of showing this:
  - At least one column is linearly dependent on others (as discussed in Unit 1)
  - The determinant is zero: $\det A = 0$
  - A has a nonempty null space, i.e. $\exists c \neq 0$ with $Ac = 0$
- Rank - maximum number of linearly independent columns
- Singular matrices have rank $< n$ (the # of columns), i.e. they are rank-deficient
  - So, they have either no solution or infinite solutions
- Nonsingular square matrices are invertible: $AA^{-1} = A^{-1}A = I$
  - So, $Ac = b$ can be solved for c via $c = A^{-1}b$
- *Note: we typically do not compute the inverse, but instead have a solution algorithm that exploits its existence*

Matrices as Vectors (an example)

- Recall $Ac = \sum_k c_k a_k$ where the a_k are the columns of A

- Consider $Ac = 0$ or $\sum_k c_k a_k = 0$
- If one column is a linear combination of others, then the linear combination weights can be used to obtain $Ac = 0$ with c nonzero
  - This nonzero c is in the null space of A, and A is singular
- Conversely: If the only solution to $Ac = 0$ is c identically 0, then no column is linearly dependent on the others
  - Thus, A is nonsingular

Diagonal Matrices

- All off-diagonal entries are 0
- Equations are decoupled, and easy to solve
- E.g. $\begin{pmatrix} 5 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 10 \\ -1 \end{pmatrix}$ has $5c_1 = 10$ and $2c_2 = -1$; so, $c_1 = 2$ and $c_2 = -.5$
- A zero on the diagonal indicates a singular system
  - Either no solution (e.g. $0c_1 = 10$) or infinite solutions (e.g. $0c_1 = 0$)
- The determinant of a diagonal matrix is obtained by multiplying all the diagonal elements together
  - Thus, a 0 on the diagonal implies a zero determinant and a singular matrix

Upper Triangular Matrices

- All entries below the diagonal are 0
- Nonsingular when the diagonal elements are all nonzero
  - Determinant is obtained by multiplying all the diagonal elements together
- Solve via <u>back substitution</u>
- E.g. consider $\begin{pmatrix} 2 & 3 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 5 \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 10 \\ 10 \end{pmatrix}$
  - Start at the bottom: $5c_3 = 10$; so, $c_3 = 2$
  - Move up one row: $c_2 - c_3 = 10$; so, $c_2 - 2 = 10$ and $c_2 = 12$
  - Move up one row: $2c_1 + 3c_2 + c_3 = 0$; so, $2c_1 + 36 + 2 = 0$ and $c_1 = -19$

Lower Triangular Matrices

- All entries above the diagonal are 0
- Nonsingular when the diagonal elements are all nonzero
  - Determinant is obtained by multiplying all the diagonal elements together
- Solve via forward substitution
- E.g. consider $\begin{pmatrix} 5 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 3 & 2 \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \\ 0 \end{pmatrix}$
  - Start at the top: $5c_1 = 10$, so, $c_1 = 2$
  - Move down one row: $-c_1 + c_2 = 10$; so, $-2 + \ c_2 = 10$ and $c_2 = 12$
  - Move down one row: $c_1 + 3c_2 + 2c_3 = 0$; so, $2 + 36 + 2c_3 = 0$ and $\mathrm{c}_3 = -19$

Elimination Matrix

- Given a column $\begin{pmatrix} a_{1k} \\ \vdots \\ a_{ik} \\ a_{i+1,k} \\ \vdots \\ a_{mk} \end{pmatrix}$, define $m_{ik} = \frac{1}{a_{ik}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_{i+1,k} \\ \vdots \\ a_{mk} \end{pmatrix}$
- Then, the size mxm elimination matrix $M_{ik} = I_{mxm} - m_{ik}\hat{e}_i^T$ subtracts multiples of row i from rows $> i$ in order to create zeroes in column k
- Standard basis vector $\hat{e}_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ has a 1 in the i-th row

Elimination Matrix

- Let $a_k = \begin{pmatrix} 2 \\ 4 \\ -8 \end{pmatrix}$

- $M_{1k} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 0 \\ 4 \\ -8 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}$ and $M_{1k}a_k = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}$

- $M_{2k} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 0 \\ 0 \\ -8 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}$ and $M_{2k}a_k = \begin{pmatrix} 2 \\ 4 \\ 0 \end{pmatrix}$

Elimination Matrix Inverse

- Inverse of an elimination matrix is $L_{ik} = M_{ik}^{-1} = I_{mxm} + m_{ik}\hat{e}_i^T$
- L_{ik} is a size mxm elimination matrix that adds multiples of row i to rows $> i$ in order to reverse the effect of M_{ik}

- $L_{1k} = M_{1k}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -4 & 0 & 1 \end{pmatrix}$

- $L_{2k} = M_{2k}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}$

Combining Elimination Matrices

- $M_{i_1k_1}M_{i_2k_2} = I - m_{i_1k_1}\hat{e}_{i_1}^T - m_{i_2k_2}\hat{e}_{i_2}^T$ when $i_1 < i_2$ (but not when $i_1 > i_2$)

$$M_{1k}M_{2k} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 2 & 1 \end{pmatrix}, \text{ but } M_{2k}M_{1k} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}$$

- $L_{i_1k_1}L_{i_2k_2} = I + m_{i_1k_1}\hat{e}_{i_1}^T + m_{i_2k_2}\hat{e}_{i_2}^T$ when $\mathrm{i}_1 < i_2$ (but not when $i_1 > i_2$)

$$L_{1k}L_{2k} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -4 & -2 & 1 \end{pmatrix}, \text{ but } L_{2k}L_{1k} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -8 & -2 & 1 \end{pmatrix}$$

Gaussian Elimination

- Consider $\begin{pmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \\ 10 \end{pmatrix}$

- $M_{11}A = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{pmatrix} = \begin{pmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{pmatrix}$ and $M_{11}b = \begin{pmatrix} 2 \\ 4 \\ 12 \end{pmatrix}$

- $M_{22}M_{11}A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{pmatrix} = \begin{pmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{pmatrix}$ and $M_{22}M_{11}b = \begin{pmatrix} 2 \\ 4 \\ 8 \end{pmatrix}$

- Then, solve the upper triangular $\begin{pmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 8 \end{pmatrix}$ via back substitution

LU Factorization

- Gaussian Elimination gives an upper triangular $U = M_{n-1,n-1} \cdots M_{22} M_{11} A$
- Using inverses, $A = L_{11} L_{22} \cdots L_{n-1,n-1} M_{n-1,n-1} \cdots M_{22} M_{11} A = L_{11} L_{22} \cdots L_{n-1,n-1} U$
- Since $L_{i_1 i_1} L_{i_2 i_2} = I + m_{i_1 i_1} \hat{e}_{i_1}^T + m_{i_2 i_2} \hat{e}_{i_2}^T$ when $i_1 < i_2$, $L = L_{11} L_{22} \cdots L_{n-1,n-1}$ is lower triangular and $A = LU$
- Here $L = L_{11} L_{22} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix}$

$$A = \begin{pmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{pmatrix} = LU$$

LU Factorization

- Factoring $A = LU$ helps to solve $Ac = b$

- In order to solve $LUc = b$, define an auxiliary variable $\hat{c} = Uc$
- First, solve $L\hat{c} = b$ for $\hat{c}$ via forward substitution
- Second, solve $Uc = \hat{c}$ for c via back substitution

- Note: the LU factorization is only computed once, and then can be used afterwards on many right hand side vectors (on many b vectors)

Pivoting

- $A = \begin{pmatrix} 0 & 4 \\ 4 & 9 \end{pmatrix}$ requires division by zero in order to create M_{11}

- (Partial) Pivoting - swap rows to use the largest (magnitude) element in the column under consideration
  - Don't forget to swap the right hand side b too
- Full Pivoting swap rows and columns to use the largest possible element
  - Don't forget to change the order of the unknowns c

- When considering column k, can only swap with rows/columns $\geq k$

Permutation Matrix

- Constructed by switching the 2 rows of I that one wants swapped
- E.g. $P_{13} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$, and $P_{13}A$ swaps the first and third rows of A
- Permutation matrices are their own inverses (swapping again restores the rows)
- Switching rows i_1 and i_2 moves a 1 from $a_{i_1 i_1}$ to $a_{i_2 i_1}$ as well as from $a_{i_2 i_2}$ to $a_{i_1 i_2}$, preserving symmetry (i.e. $P_{i_1 i_2}^T = P_{i_1 i_2}$)

- To swap the first and third unknowns: $Ac = AP_{13}P_{13}c = (AP_{13})(P_{13}c)$ where $P_{13}c$ swaps the unknowns and AP_{13} swaps the columns (to see this, consider $(AP_{13})^{TT} = (P_{13}A^T)^T$ which swaps the rows of A^T)

Full Pivoting

- Let P_{r_i} be the permutation matrix that (potentially) switches row i with a row $> i$
- Let P_{c_k} be the permutation matrix that (potentially) switches column k with a col $> k$
- Then full pivoting can be written as:

$$(M_{n-1,n-1}P_{r_{n-1}} \cdots M_{22}P_{r_2}M_{11}P_{r_1}AP_{c_1}P_{c_2} \cdots P_{c_{n-1}})(P_{c_{n-1}} \cdots P_{c_2}P_{c_1}c)$$

- Once known, $P_r = P_{r_{n-1}} \cdots P_{r_2}P_{r_1}$ and $P_c = P_{c_{n-1}} \cdots P_{c_2}P_{c_1}$ can be used to do all the permutations ahead of time (the resulting matrix doesn't require pivoting)
- $Ac = b$ becomes $(P_rAP_c^T)(P_cc) = P_rb$ or $A_Pc_P = b_P$; then, $A_P = L_PU_P$ can be computed without pivoting
- Subsequently, given any right hand side b, solve $L_PU_Pc_P = P_rb$ to find c_P using forward/back substitution; then, $c = P_c^Tc_p$

Permuting before Elimination

- Assume $i > j$,

$$P_{r_i} M_{jj} P_{r_i} = I_{mxm} - P_{r_i} m_{jj} \hat{e}_j^T P_{r_i} = I_{mxm} - \widehat{m}_{jj} \hat{e}_j^T = \widehat{M}_{jj}$$

$$P_{r_i} M_{jj} = P_{r_i} M_{jj} P_{r_i} P_{r_i} = \widehat{M}_{jj} P_{r_i}$$

- Thus, for some suitable definition of the hat notation (there are multiple premutation operators to consider for each M_{jj}, except $M_{n-2,n-2}$):

$$M_{n-1,n-1} P_{r_{n-1}} \cdots M_{22} P_{r_2} M_{11} P_{r_1} A = M_{n-1,n-1} \cdots \widehat{M}_{22} \widehat{M}_{11} P_r A$$

- This shows that you can permute first and do elimination afterwards

Sparsity

- Most large matrices (of interest) operate on variables that only interact with a sparse set of other variables
- This makes the matrix sparse (as opposed to dense), with most entries identically 0
- However, the inverse of a sparse matrix can contain an unwieldy amount of non-zero entries

- E.g. the 3D Poisson equation on a relatively small 100^3 Cartesian grid has an unknown for each of the 10^6 grid points
- For each unknown, the discretized Poisson equation depends on the unknown itself and its 6 immediate Cartesian grid neighbors
- Thus, the size $10^6 x 10^6$ matrix has only $7x10^6$ nonzero entries
- But, the inverse can have 10^{12} nonzero entries!

Computing the Inverse

- When A is relatively small (and dense), computing $A^{-1}$is fine

- Since $AA^{-1} = I$, the solution c_k to $Ac_k = \hat{e}_k$ is the k-th column of A^{-1}

- First, compute $A_P = L_P U_P$ as usual
- Then, solve $Ac_k = \hat{e}_k$ once for each column (n times)