Basic Linear Algebra
Matrices and Vectors

- **Matrix**: A rectangular array of numbers, e.g., $A \in \mathbb{R}^{m \times n}$:

  $$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

- **Vector**: A matrix consisting of only one column (default) or one row, e.g., $x \in \mathbb{R}^n$

  $$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$
Matrix Multiplication

- If $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $C = AB$, then $C \in \mathbb{R}^{m \times p}$:

  $$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

- Special cases: Matrix-vector product, inner product of two vectors. e.g., with $x, y \in \mathbb{R}^n$:

  $$x^T y = \sum_{i=1}^{n} x_i y_i \in \mathbb{R}$$
Properties of Matrix Multiplication

- **Associative:** \((AB)C = A(BC)\)
- **Distributive:** \(A(B + C) = AB + AC\)
- **Non-commutative:** \(AB \neq BA\)
- **Block multiplication:** If \(A = [A_{ik}]\), \(B = [B_{kj}]\), where \(A_{ik}\)'s and \(B_{kj}\)'s are matrix blocks, and the number of columns in \(A_{ik}\) is equal to the number of rows in \(B_{kj}\), then \(C = AB = [C_{ij}]\)
  where \(C_{ij} = \sum_k A_{ik}B_{kj}\)

**Example:** If \(\vec{x} \in \mathbb{R}^n\) and \(A = [\vec{a}_1 | \vec{a}_2 | \ldots | \vec{a}_n] \in \mathbb{R}^{m \times n}\), \(B = [\vec{b}_1 | \vec{b}_2 | \ldots | \vec{b}_p] \in \mathbb{R}^{n \times p}\):

\[
A \vec{x} = \sum_{i=1}^{n} x_i \vec{a}_i
\]

\[
AB = [Ab_1 | Ab_2 | \ldots | Ab_p]
\]
Operators and properties

- **Transpose:** $A \in \mathbb{R}^{m \times n}$, then $A^T \in \mathbb{R}^{n \times m}$: $(A^T)_{ij} = A_{ji}$

- **Properties:**
  - $(A^T)^T = A$
  - $(AB)^T = B^T A^T$
  - $(A + B)^T = A^T + B^T$

- **Trace:** $A \in \mathbb{R}^{n \times n}$, then: $tr(A) = \sum_{i=1}^{n} A_{ii}$

- **Properties:**
  - $tr(A) = tr(A^T)$
  - $tr(A + B) = tr(A) + tr(B)$
  - $tr(\lambda A) = \lambda tr(A)$
  - If $AB$ is a square matrix, $tr(AB) = tr(BA)$
Special types of matrices

- **Identity matrix**: \( I = I_n \in \mathbb{R}^{n \times n} \):
  \[
  I_{ij} = \begin{cases} 
  1 & \text{if } i=j, \\
  0 & \text{otherwise}.
  \end{cases}
  \]

- For all \( A \in \mathbb{R}^{m \times n} \): \( AI_n = I_mA = A \)

- **Diagonal matrix**: \( D = \text{diag}(d_1, d_2, \ldots, d_n) \):
  \[
  D_{ij} = \begin{cases} 
  d_i & \text{if } j=i, \\
  0 & \text{otherwise}.
  \end{cases}
  \]

- **Symmetric matrices**: \( A \in \mathbb{R}^{n \times n} \) is symmetric if \( A = A^T \).

- **Orthogonal matrices**: \( U \in \mathbb{R}^{n \times n} \) is orthogonal if \( UU^T = I = U^TU \)
A set of vectors \( \{x_1, \ldots, x_n\} \) is linearly independent if
\[
\not\exists \{\alpha_1, \ldots, \alpha_n\}: \sum_{i=1}^{n} \alpha_i x_i = 0
\]

Rank: \( A \in \mathbb{R}^{m \times n} \), then \( \text{rank}(A) \) is the maximum number of linearly independent columns (or equivalently, rows)

Properties:
- \( \text{rank}(A) \leq \min\{m, n\} \)
- \( \text{rank}(A) = \text{rank}(A^T) \)
- \( \text{rank}(AB) \leq \min\{\text{rank}(A), \text{rank}(B)\} \)
- \( \text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B) \)
If $A \in \mathbb{R}^{n \times n}$, $\text{rank}(A) = n$, then the inverse of $A$, denoted $A^{-1}$, is the matrix that: $AA^{-1} = A^{-1}A = I$

- Properties:
  - $(A^{-1})^{-1} = A$
  - $(AB)^{-1} = B^{-1}A^{-1}$
  - $(A^{-1})^T = (A^T)^{-1}$

- The inverse of an orthogonal matrix is its transpose
Range and Nullspace of a Matrix

- **Span:** $\text{span}(\{x_1, \ldots, x_n\}) = \{\sum_{i=1}^{n} \alpha_i x_i | \alpha_i \in \mathbb{R}\}$

- **Projection:**
  \[
  \text{Proj}(y; \{x_i\}_{1 \leq i \leq n}) = \text{argmin}_{v \in \text{span}(\{x_i\}_{1 \leq i \leq n})} \{\|y - v\|_2\}
  \]

- **Range:** $A \in \mathbb{R}^{m \times n}$, then $\mathcal{R}(A) = \{Ax | x \in R^n\}$ is the span of the columns of $A$

- **Proj($y$, $A$)** = $A(A^T A)^{-1} A^T y$

- **Nullspace:** $\text{null}(A) = \{x \in \mathbb{R}^n | Ax = 0\}$
Determinant

- \( A \in \mathbb{R}^{n \times n}, a_1, \ldots, a_n \) the rows of \( A \), then \( \text{det}(A) \) is the volume of \( S = \{ \sum_{i=1}^{n} \alpha_i a_i \mid 0 \leq \alpha_i \leq 1 \} \).

- Properties:
  - \( \text{det}(I) = 1 \)
  - \( \text{det}(\lambda A) = \lambda \text{det}(A) \)
  - \( \text{det}(A^T) = \text{det}(A) \)
  - \( \text{det}(AB) = \text{det}(A)\text{det}(B) \)
  - \( \text{det}(A) \neq 0 \) if and only if \( A \) is invertible.
  - If \( A \) invertible, then \( \text{det}(A^{-1}) = \text{det}(A)^{-1} \)
Quadratic Forms and Positive Semidefinite Matrices

- \( A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^n \), \( x^T Ax \) is called a quadratic form:

\[
x^T Ax = \sum_{1 \leq i, j \leq n} A_{ij}x_i x_j
\]

- \( A \) is positive definite if \( \forall x \in \mathbb{R}^n : x^T Ax > 0 \)
- \( A \) is positive semidefinite if \( \forall x \in \mathbb{R}^n : x^T Ax \geq 0 \)
- \( A \) is negative definite if \( \forall x \in \mathbb{R}^n : x^T Ax < 0 \)
- \( A \) is negative semidefinite if \( \forall x \in \mathbb{R}^n : x^T Ax \leq 0 \)
Eigenvalues and Eigenvectors

- \( A \in \mathbb{R}^{n \times n}, \lambda \in \mathbb{C} \) is an eigenvalue of \( A \) with the corresponding eigenvector \( x \in \mathbb{C}^n (x \neq 0) \) if:

\[
Ax = \lambda x
\]

- eigenvalues: the \( n \) possibly complex roots of the polynomial equation \( det(A - \lambda I) = 0 \), and denoted as \( \lambda_1, \ldots, \lambda_n \)

- Properties:
  - \( tr(A) = \sum_{i=1}^{n} \lambda_i \)
  - \( det(A) = \prod_{i=1}^{n} \lambda_i \)
  - \( rank(A) = |\{1 \leq i \leq n | \lambda_i \neq 0\}| \)
Matrix Eigendecomposition

- $A \in \mathbb{R}^{n \times n}$, $\lambda_1, \ldots, \lambda_n$ the eigenvalues, and $x_1, \ldots, x_n$ the eigenvectors. $X = [x_1 | x_2 | \ldots | x_n]$, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, then $AX = X\Lambda$.

- $A$ called diagonalizable if $X$ invertible: $A = X\Lambda X^{-1}$

- If $A$ symmetric, then all eigenvalues real, and $X$ orthogonal (hence denoted by $U = [u_1 | u_2 | \ldots | u_n]$):

$$A = U\Lambda U^T = \sum_{i=1}^{n} \lambda_i u_i u_i^T$$

- A special case of Singular Value Decomposition
A set of points $S$ is convex if, for any $x, y \in S$ and for any $0 \leq \theta \leq 1$, 

$$\theta x + (1 - \theta)y \in S$$

A function $f : S \rightarrow \mathbb{R}$ is convex if its domain $S$ is a convex set and 

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$$

for all $x, y \in S$, $0 \leq \theta \leq 1$.

A function $f : S \rightarrow \mathbb{R}$ is submodular if for any subset $A \subseteq B$, 

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$$

Convex functions can easily be minimized. Submodular functions allow approximate discrete optimization.
Proofs

- Induction:
  1. Show result on base case, associated with $n = k_0$
  2. Assume result true for $n \leq i$. Prove result for $n = i + 1$
  3. Conclude result true for all $n \geq k_0$

Example: In a complete graph, $E = \frac{1}{2}N(N - 1)$

- Contradiction (reductio ad absurdum):
  1. Assume result is false
  2. Follow implications in a deductive manner, until a contradiction is reached
  3. Conclude initial assumption was wrong, hence result true

Example: Strongly connected components partition nodes
Graph theory

- Definitions: vertex/node, edge/link, loop/cycle, degree, path, neighbor, tree, clique,…
- Random graph (Erdos-Renyi): Each possible edge is present with some probability $p$
- (Strongly) connected component: subset of nodes that can all reach each other
- Diameter: longest minimum distance between two nodes
- Bridge: edge connecting two otherwise disjoint connected components
Basic algorithms

- BFS: explore by “layers”
- DFS: go as far as possible, then backtrack
- Greedy: maximize goal at each step
- Binary search: on ordered set, discard half of the elements at each step
Complexity

- Number of operations as a function of the problem parameters.
- Examples
  1. Find shortest path between two nodes:
     - DFS: very bad idea, could end up with the whole graph as a single path
     - BFS from origin: good idea
     - BFS from origin and destination: even better!
  2. Given a node, find its connected component
     - Loop over nodes: bad idea, needs $N$ path searches
     - BFS or DFS: good idea