02-712 Week 1

Biological Modeling and Simulation

Aidan Jan

September 2, 2025

Course Topics

Models for Optimization

- Ex. Sequence Similarity: Given a DNA or a protein, find other DNA or proteins that are similar.
  - We can model this problem as a sequence alignment problem, which can be optimized.
  - We can also model this problem as an alignment free problem, for example, dividing the DNA or protein into K-mers and comparing them.
- This class focuses not on how to align the DNA or protein, but instead focuses on why we would use one representation of the problem over another.
  - Why do we pick one representation over another?
  - What are the parameters for the problem?

Simulation and Sampling

Suppose A, B, and C are chemicals, such that $A + B \to C$, with rate k.

- How can you simulate the concentrations of A, B, or C over time?
- Maybe use a graph, or perhaps differential equations (mass action model).
- Or, we can do a discrete system. Start with some amount of A, and B, and roll a dice, and if it is greater than a certain number, make a C.
- Why would we pick one representation of the problem over another?

Model Inference and AI Modeling

Suppose we have chemicals A, B, and C, and $A + B \rightleftharpoons C$. Forward reaction with rate k_1, and backwards reaction with rate k_2. We did the reaction and recorded the concentrations over time (discrete), and plotted them on a graph.

- How do we pick parameters of the model to match the data?
- Maybe find the values of k_1 and k_2 that best fit the system?
- How do we even know the correct equation is $A + B \rightleftharpoons C$? It could be $A + 2B \rightleftharpoons C$, etc. (Structural Inference)

Model Building

Consider the evolutionary tree building problem. You have species human, gorilla, cow, chicken, tuna, and fly. How do we model this, or how do we design a program to model this?

Questions to consider:

1. How are we going to answer this?
2. What information do we have to work with? (e.g., phenotypes, genetic information, etc.)
3. Which properties matter a lot, and which do not? (e.g., in this case, if we want to build a species tree, we don't have to do too deep of an analysis into the genome, for example. we're doing a species tree, not a gene tree.)
4. What assumptions can we make? (e.g., all the organisms we are mapping are still alive, therefore only leaf nodes are labelled. we don't want to worry about edge weights since we are not mapping time, etc.)

From these questions, we gather some properties of what we want the output to be. For example, from the above,

- We want a leaf-labelled tree, with non-weighted edges, that connects all of the input organisms. More formally, the output should be a tree, $T = (V, E)$. The leaf set, $L \subseteq V$, such that $t : V \to \Sigma^n$. Σ is the alphabet (e.g., organisms).
- The input is simply the alphabet Σ, which is a set of all the organisms we want to map, along with the properties, such as a list of features per organism and genomic data.

Finally, we want to list our assumptions:

- Since evolution is very complex, there will be things we want to cut out
- We assume only point mutations occur. This isn't exactly true, but it is accurate enough.
- We assume mutations are rare. For example, two mutations won't occur at the same place.
- We also assume all mutations are equally likely.
- Quality of the evolutionary tree would be defined by the number of mutations. The less the DNA has to mutate to produce the tree, the better.
  - In this case, we can use $\min_{T,t} \sum_{(u,v)} d(u, v)$, where d is the edit distance between two neighboring nodes (u, v) as the objective function of the tree.

We can now write a formal statement:

- Input: Set $S = \{s_1, \ldots, s_n\}$, $s \in \{A, C, G, T\}$
- Output: Tree $T = (V, E)$, leafset $L \subseteq V$, $|L| = n$, $t : V \to \Sigma^n$, $\forall s \in S, \exists l \in L$, $t(l) = s$.
- Objective: $\min_{T,t} \sum_{(u,v)\in E} d(u, v)$

The main challenge in the research is making the model realistic enough that it captures the details (e.g., not making too many assumptions), but also solvable. Too realistic models become very complex and hard to solve. Additionally, the model should be learnable, or solvable by iteration, and applied to other datasets. Realism, solvability, and learnability tend to be trade-offs of each other.

Tractable vs. Intractable Problems

- **Tractable** are solvable problems, most often in polynomial time.
- **Intractable** are hard-to-solve, or impossible-to-solve problems. Those that are NP-hard or exponential time.

Graph Algorithms

Graphs are denoted as $G = (V, E)$, or that, a graph G consists of a set of vertices V and a set of edges E.

- A weighted graph has weights on the edges. An unweighted graph does not.
- A directed graph means edges only go one direction. An undirected graph means edges go both ways.

Some graphs include:

- Tree (graph with no cycles)
- Forest (disjoint trees)
- Bipartite (graphs where nodes can be divided into two groups, and every edge between nodes go between the two groups)
- Planar (graph can be drawn on a 2D plane without edges crossing)
- etc.

We categorize graphs like this because some problems are tractable on certain types of graphs, and intractable on others.

Minimum Spanning Tree Problem

Here is an example of a formal statement of a graph problem, specifically the Minimum Spanning Tree (MST) problem:

- Input: Weighted, undirected, connected $G = (V, E)$, $W : V \times V \to \mathbb{R}$
- Output: Subgraph $G' = (V, E'), E' \subseteq E$
- Objective: $\min_{(u,v) \in E'} w(u, v)$

Basically, create a tree that is a subgraph that contains all the nodes, but the minimum number edges, and minimize the total edge weight.
Some known solutions to this problem include Prim's Algorithm, and Kruskal's Algorithm, both of which run in O(n^2), where n is the number of nodes.

This graph problem may be a simplification of the phylogenetic problem.

Minimum Steiner Tree Problem

- Input: $G = (V, E)$, $w : E \to \mathbb{R}$, $S \subseteq V$
- Output: Tree $T = (V', E')$ such that $S \subseteq V' \subseteq V$
- Objective: $\min \sum_{e \in E'} w(e)$

This problem is similar to the minimum spanning tree problem, but the output tree only needs to include a given list of nodes, instead of all the nodes.
In the phylogenetic tree problem, this may be a better model than the standard MST problem since there may be some organisms that we don't care about the positions of.

- It turns out that the Minimum Steiner Tree problem is intractible. It is NP-hard.
- Because of this, it tends to be more like a decision problem (output yes/no for if an efficient answer is possible) rather than an optimization problem.

Shortest Path Problems

The simplest shortest path problem is given two nodes s, t, on the graph $G = (V, E)$, find the minimum weight path from s to t

- s is known as the source node, and t is known as the sink node

Formal problem:

- Input: Graph $G = (V, E)$, $w : E \to \mathbb{R}$, $s, t \in V$
- Output: Path $s, v_1, \ldots, v_k, t$, such that $(s, v_1) \in E, (v_i, v_i + 1) \in E, (v_k, t) \in E$
- Objective: $\min w(s, v_1) + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) + w(v_k, t)$

This problem is solvable using Dijkstra's algorithm, or Bellman Ford if there are negatively-weighted edges (but no negative cycles).

Other derivatives of this problem include:

- Single-source shortest path (shortest path to every node from a single source)
- Single-destination shortest path (shortest path between every node to one destination)
- All-pairs shortest path (shortest path between every pair of nodes on the graph)

Hamiltonian Path Problem

- Input: $G = (V, E)$, weighted, directed
- Output: True/False $Q : \exists \prod \pi_1, \pi_n$ such that $(\pi_1, \pi_n) \in V$

Essentially, is there a path that passes through every vertex exactly once? This problem is intractible.

Longest Path Problem

Same as the shortest path problem, but instead you want to find the longest path that doesn't repeat nodes. This problem is also intractible.

Traveling Salesperson Problem (TSP)

- Input: Directed $G = (V, E)$, $|v| = n$, $w : V \times V \to \mathbb{R}$, $B \in \mathbb{R}$
- Question: $\exists \text{permutation} \Pi = \pi_1, \ldots, \pi_n$ such that $w(\pi_n, \pi_1) + \sum_{i=1}^{n-1} w(\pi_i, \pi_{i+1}) \leq B$?

Basically, does there exist a path passing through all the vertices and loops back to the starting node? And the optimization version, minimize the edge weights.

Max Flow and Min Cut

Max flow:

- Input: weighted, directed $G = (V, E)$, $c : E \to \mathbb{R}$ such that $s, t \in V$ (s is source, t is sink)
- Output: flow: $f : E \to \mathbb{R} \forall v \in V = \{s, t\}$. $\sum_{(u,v) \in e} f(u, v) = \sum_{(u,v) \in E} f(v, u)$, $f(u, s) = 0 \forall u$, $f(t, u) = 0 \forall u$, $f(u, v) = c(u, v) \forall u, v$
- Objective: $\max \sum_{s,u \in E}, f(s, u)$

Min cut:

- Input: $G = (V, E)$, $c : E \rightarrow \mathbb{R}$, $s, t \in V$
- Output: cut $E' \subseteq E$ such that any path from s to t goes through some $e \in E'$.
- Objective: $\min \sum_{e \in E} c(e)$

These two problems are actually equivalent, since the min cut problem can represent the bottleneck for the max flow problem.

Max Cut Problem

Not exactly the opposite of the min cut problem, but it goes as follows:

- Input: Graph $G = (V, E), w : E \rightarrow \mathbb{R}, B \in \mathbb{R}$,
- Objective: $\exists V_1 \in V$, $V_2 = V/V_1$ such that $\sum_{v_1 \in V_1, v_2 \in V_2} w((v_1, v_2)) \geq B$.

This problem is intractible. Basically, if you cut the graph into two subgraphs, what is the maximum weight of the cut?

Additionally, Min 3-cut and Max 3-cut are both intractible. (Same as Max cut and Min cut except instead of there being one cut, there are two cuts, dividing the graph into three subgraphs.)