02-712 Week 6

Biological Modeling and Simulation

Aidan Jan

September 26, 2025

Derivative Free Optimization

A lot of times, we will be working with models that have a lot of parameters. (e.g., protein folding).

- With problems like this, it is often impossible to take a derivative, or know the function.
- With our protein folding example, you would have to be able to calculate the free energy of a given energy state, and also the free energy states of every possible tiny change to be able to get the derivative. This is simply not feasible.

For these types of problems, we have a function $\Phi(\vec{x})$, where Φ is sort of a black box function that takes in our parameters.

- In our protein folding problem, Φ is a function that takes in the protein, $\vec{x}$, and returns the current free energy.
- We want to find $\min \Phi(\vec{x})$

Some derivative-free methods

Metropolis Method

We can use a heuristic to solve these problems. Simulated annealing:

- Set up some "moves", $\vec{x} \to \vec{x}'$
- Define likelihood, $L(\vec{x}) = e^{-\Phi(\vec{x})/T}$
- Define metropolis, $r = \frac{e^{-\Phi(\vec{x}')/T}}{e^{-\Phi(\vec{x})/T}}$
- If r > 1, we jump back to step 2 (define likelihood), and repeat until $r < 1$.

This is called the Metropolis method.

Genetic Algorithm

- mutate($\vec{x}$) $\to \vec{x}'$
- mating($\vec{x}_1, \vec{x}_2$) $\to \vec{x}'$
- repeat:
  - "breed" mutate(mate($\vec{x}_i, \vec{x}_j$))
  - select n best models, and continue until the model is good enough.

Response Surface Method

Suppose you want to optimize some unknown function, $f(x)$, and you have some data points for it. You can take the derivative of the function, since maybe the data is extremely expensive to compute, or perhaps the data is noisy.

- First, fit a curve to the data points. (This curve is known as the response surface.)
- Now, optimize based on the curve. A well-selected curve reasonably passes through all the points, and it is differentiable.
- Now, find the minimum of the response surface, and test the minimum on the real function.
- This method can be done with many dimensions too, except it gets harder with more parameters.

Trust Region Method

Used with the response surface method. Essentially asks, how good is the response surface for the data?

- We would generate a lot of random points using the response surface and estimate the minimum of the surface.
- Additionally, we test whether the points generated are similar to the original dataset using some likelihood calculation.

Kriging Methods

This is a class of different derivative-free methods.

Gaussian Mixture Model (GMM)

A way of taking a sample of points and coming up with a representation of what is happening to the rest of the points.

- Sample some points and get $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$. Calculate $\Phi(\vec{x}_1), \ldots, \Phi(\vec{x}_n)$.
- All of the other points are labelled with a probability to be part of each group. Essentially, for a given point $\vec{x}$, we are finding $P(\Phi(\vec{x}) = \Phi(\vec{x}_i)), \quad \forall i \leq n$.

Gaussian Process Model

Suppose we have a set of a lot of (possibly noisy) points. We want a best fit curve.

- We assume that every point is preturbed from the best fit line by some gaussian.
- We are trying to find the best fit line (or even region!) that fits the points the best.

Lipschitzian Methods (Lipschitz Continuity)

$$||\Phi(\vec{x}) - \Phi(\vec{y})|| \leq L||\vec{x} - \vec{y}||$$

This essentially means that our functions do not change too quickly. (L is a constant, and if the change of the function can be written as a constant times the change of x and y, it must not be exponential.)

- L is known as the Lipschitz Constant.
- The goal is to pick a point, $\vec{x}$, such that $\Phi(\vec{x})$ is within the radius of the $\Phi(\vec{x_d})$, where x_d are datapoints. Basically, we can set a minimum and maximum bound for a data point to be part of the group.
- We can pick many different points to be locations for local optimizers, and from there, you know one of them will be the global optimizer.
- To find a global optimizer, you can recursively sample near the positions of local optimizers, and pick one that works best.

AI Surrogate Models

Turn a problem you are trying to solve, and turn it into a pattern recognization test.

- Suppose we have a very large set of data consisting of (Input 1, output 1), (Input 2, output 2), . . . , (Input n, output n). . .
- We would throw these in as training data for a deep learning neural network and have it 'classify' our current problem and spit out an output.
  - This is a way that estimates the function. It is essentially a not well-defined optimization problem.

Alphafold

This is a deep learning method for folding proteins. In theory, it is optimizing minimum free energy of the protein, but alpha does not really do that. Rather, it 'learns' from past work.

It requires multiple parts:

- Lots of data. Input amino acid sequences, and output protein pairs. Alphafold got this from the Protein Databank, where researchers would publish protein structures.
- Representational of model (response surface), transformer NN models
- Algorithms to train model (Adam optimizer)
- Hardware (GPUs)

Case Study: Cancer

Cancer is a disease of evolution. A single mutation may cause a cell to grow out of control, and also pick up different mutations much faster.

Evolutionary Biology and the Cancer Treatment Timeline

1. Surveillance (Health): Are there patterns of somatic evolution predictive of risk? Can we steer that towards lower risk?
2. Intervention (Precancer): Are precancerous lesions likely to evolve towards cancer?
3. Treatment (Cancer): Should we treat?
4. Monitoring (Remission): Is recurrence likely? Can we make it less likely?
5. Treatment (Recurrence): How can we reverse emerging resistance? Are there other resistance mechanisms we can anticipate?
6. Palliative Care (Metastasis): Is metastatic disease progressing? How can we slow that down?

Phylogeny Problem

A tumor can be made up of multiple different cell populations, so it is challenging to figure out how the different types of cells interact with each other.

- We can take the bulk tumor data, and do a deconvolution problem to separate out the different cells.
  - Deconvolution isn't as easy as finding two matrices to create the input matrix, because the problem is inherently underconstrained. We therefore must add an objective function to optimize the resulting matrices towards.
- Afterwards, we can build a phylogenetic tree.
  - We can do this using the continuous optimization methods discussed earlier.

Solving Optimization Problems using ILP

- Integer Linear Programming: create a set of variables to optimize, a linear objective, and constraints, and use a ILP solver.
- Semidefinite (Convex) Programming: similar to ILPs, except the objective need not be linear. Also solvable, but not as efficiently.

ILP for Phylogeny Inference: Intuition

1. Create a graph of all possible nodes and edges
2. Create a "flow" to each input sequence from an arbitrary root
3. Minimize edges needed to accommodate all flows.

Usually we make the root a healthy cell since all cancer cells are descendants of healthy cells.

- Then, we can define a flow from the healthy cell through steiner nodes to cancer cells.
- The paths between the cells would be a tree, and the tree would somewhat represent how the cell evolved.