Multi-Objective and Model-Based Optimization

Lecture 1, Sep 18 2017
Introduction
Prashant Singh, TDB
Course Outline

• Lecture 1: Introduction to black-box and evolutionary optimization.

• Lecture 2: Bayesian and surrogate model-based optimization.

• Lecture 3: Multi-objective and constrained optimization.

• Lecture 4: Assignment discussion
  • Optimize a problem of your choice (or mine!).
  • Present and explain your solution.
Lecture Outline

- Motivation, a bit of history
- Black box optimization
- Simulation-driven optimization
- Parameter-tuning
- Evolutionary optimization
Mathematically…

- find \textbf{min} or \textbf{max} of an unknown objective function \( f \),
  \[ x^* = \arg\max_{x\in\mathcal{X}} f(x), \]
  where \( \mathcal{X} \) is some space of interest.

- \( f \) is \textbf{black-box}, we only have access to (possibly noisy) point-wise observations \( y = f(x) \).

- \( f \) is \textbf{real-valued}. 
Some terminology

- Optimization, mathematical programming, energy minimization (in Physics, Computer Vision).

- $f$: objective function, loss function, cost function (min), utility function, fitness function (max), energy function.

- $\mathcal{X}$: search space, choice set, design space, input space, variable space.

- Solution: optima, minima, maxima, candidate/feasible solution(s).
  - Convexity of $f$ (and feasible region) decides if there exist local optima.

- Global optimization.
Brief history

• 17th century:
  • Fermat explores calculus based formulae.
  • Newton and Gauss propose iterative methods.

• 1947: Dantzig and the simplex method.

• 1951: (Karush ’39) Kuhn and Tucker give optimality conditions.

• 1980s: Heuristics and the use of computers.
Black-box optimization

- Computer scientists are lazy!
- They prefer reusable and general methods.
- Sometimes, nothing is known about the objective function.
- Or, the objective is too complex!
Simulation-driven optimization

- Simulations are at the heart of computer-aided design.
- Aerospace, trains, automobiles, electronics, civil engineering, academia, etc…
Parameter tuning

- Algorithms often have several parameters to be tuned.
- Either tuned by a domain expert.
- Or a brute force search.
- Above a handful of parameters, brute force search can be really slow.

Applicable in:
- Simulators, software libraries, scientific applications, finding failure cases, …
- You tell me?
Parameter tuning

Want to build neural network
How many hidden layers/nodes to use?

Deep learning
Why you no work?!

Hyperparameter optimization

The response times are too damn high

We use the score function
Evolutionary optimization

• Inspired from biological processes.
  • Mutation, combination, selection, reproduction.

• Candidate solutions are generated randomly.

• Evaluated and ranked using a fitness/loss function.

• Population evolves so as to minimise loss.
Evolutionary optimization

- **Terminology**
  - **Gene**: encoded form of an input parameter (e.g., binary bit strings).
  - **Chromosome**: a complete set of genes taken together describing a candidate solution.
  - **Fitness**: objective function value.
Evolutionary optimization

- **Step 1: Initialization**
  - Choose a population size \( n \) (candidate solutions)
    - These solutions are ‘improved’ over time.
  - Generate \( n \) individuals (e.g., randomly).
  - Encode individuals.
    - E.g., two parameters with generated values 25 and 14.
    - In binary: 1100101110.
Evolutionary optimization

• **Step 2: Evaluation**
  - Evaluate the objective function value of \( n \) candidate solutions.
  - Each chromosome/individual can be evaluated in isolation of others.
    - Embarrassingly parallelizable.
  - The fitness function will be potentially called 1000s of times, helps if its cheap to evaluate.
Evolutionary optimization

• **Step 3: Selection**
  • Use selection rules and random behaviour to improve the population.
  
  • Select parents based on their fitness value.
    • Individuals with lower fitness (for minimization) should be more probable to have offsprings.
  
  • Many ways to do this.
    • E.g., tournament selection.
      • Select 4 individuals randomly.
      • Eliminate 2 based on fitness value.
      • The other 2 become parents.
      • This continues till we get a new complete population.
Evolutionary optimization

- **Step 4: Recombination**
  - Use chromosomes of the parents to create offspring.

- **Crossover**
  - E.g., Uniform crossover (random), crossover points (predefined).
  - Elitism: always save the best individuals from random changes.

- **Mutation**
  - The genetic representation can be mutated randomly with varying frequency.
  - E.g., swapping genes, or just ‘bits’.
Evolutionary optimization

- We can choose a number of crossover points

<table>
<thead>
<tr>
<th>Param. 1 (eyes)</th>
<th>Parent 1</th>
<th>Parent 2</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>A a</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B b</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C c</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D d</td>
<td>D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Param. 2 (nose)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H h</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this case, the parameters remained intact, and the child inherited the same eyes as parent1 and the same nose as parent2.
Evolutionary optimization

- Repeat steps 2-4
Evolutionary optimization

• Notes
  • Have we found the global optima?
    • Random restarts.

  • How to select and recombine?
    • 1000s of choices.

  • When to stop? (How many iterations)
    • Computational budget.
Evolutionary optimization
THANK YOU!

Prashant Singh <prashant.singh@it.uu.se>