Notes on Optimization

Paul Dubois

Abstract

Introduction to Optimization

Optimization involves finding the best solution from a set of possible solutions. Specifically, it entails finding the value of x that minimizes or maximizes a function $f(x)$.

Maximization and Minimization

Maximization of $f(x)$ can be transformed into a minimization problem by considering $-f(x)$.

$$\text{maximize } f(x) \equiv \text{minimize } -f(x)$$

Continuous vs. Discrete Optimization

In this session, we will focus on continuous optimization, as it is predominantly used in data science. For your mathematical culture, it's good to know that both exist. Optimization algorithms on continuous cases are quite different to the ones used in discrete cases.

Continuous Optimization: The variables can take any value within a given range, typically $\mathbb{R}$. *Example:* Linear regression, where parameters can take any real value.

Discrete Optimization: The variables can only take on discrete values, typically $\mathbb{Z}$ or $\mathbb{N}$. *Example:* Integer programming, where solutions are restricted to integers.

Optimization Algorithms

What algorithms can you think of for solving optimization problems?
[Interactive session]
Here, are methods presented in class:

Grid Search

Grid Search is a brute-force method that evaluates the function at a grid of points covering the domain. It is simple but computationally expensive, especially in high dimensions. If your domain is unbounded (such as $\mathbb{R}$), you will need to bound your grid search.

Dichotomy (Bisection Method)

The Bisection Method is fitted for one-dimensional optimization (although it is possible to extend it to multi-dimension). It repeatedly bisects an interval and selects the sub-interval in which the function changes sign. It is effective for finding roots but can be extended to optimization.

Gradient Descent

Gradient Descent is an iterative method used for finding local minima of a function. It updates the parameters in the opposite direction of the gradient of the function at the current point.

The update rule is: $x_{k+1} = x_k - \alpha \nabla f(x_k)$, where α is the learning rate and $\nabla f(x_k)$ is the gradient of f at x_k.

In one dimension, $\nabla f(x_k)$ is simply the derivative of f.

Theory and Practice

Grid Search

- **Step 1:** Define the grid over the domain.
- **Step 2:** Evaluate the function at each grid point.
- **Step 3:** Select the point with the best function value.

Advantages: Simple and straightforward.
Disadvantages: Computationally expensive, especially in high dimensions. It also needs a bounded domain.

Bisection Method

- **Step 1:** Choose initial interval $[a, b]$ to explore.
- **Step 2:** Compute the midpoint $c = \frac{a+b}{2}$.
- **Step 3:** Determine the "best"[1] sub-interval $[a, c]$ or $[c, b]$.
- **Step 4:** Repeat until the interval is sufficiently small.

[1]"best" means the one most likely to contain the minimum.

Advantages: Fun to implement.
Disadvantages: Extending the technique to multi dimensional problem is complicated. It also needs a bounded domain.

Gradient Descent

- **Step 1:** Initialize x_0.
- **Step 2:** Compute the gradient $\nabla f(x_k)$.
- **Step 3:** Update $x_{k+1} = x_k - \alpha \nabla f(x_k)$.
- **Step 4:** Repeat until convergence (i.e., $\|\nabla f(x_k)\|$ is small).

Advantages: Efficient for high-dimensional problems, widely used in machine learning.
Disadvantages: May converge to local minima, requires tuning of the learning rate.

Conclusion

Continuous optimization is a crucial tool in data science. It's a good exercise to test various optimization techniques, and you are encourage to try other ones on you own (genetic algorithms are interesting as well, although only used in niche cases). In practice, gradient descent (and variations of it, such as Adam) are used in the vast majority of the cases.