OPTIMIZATION

Quantitative Economics 2024

Piotr Żoch

November 13, 2024

OPTIMIZATION IN ECONOMICS

- Economics: agents optimize subject to constraints
  - Maximize utility subject to budget constraints
  - Minimize expenditure to attain a given level of utility
  - Maximize profits subject to consumer demand
  - Minimize costs of production subject to available technology
- Optimization also used in estimation – find parameters that minimize some loss function / maximize likelihood.

OPTIMIZATION

The most general minimization problem is

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

where $f : \mathbb{R}^n \to R$ is the objective function, $g : \mathbb{R}^n \to \mathbb{R}^m$ is the vector of m equality constraints, and $h : \mathbb{R}^n \to \mathbb{R}^l$ is the vector of l inequality constraints.

Note: maximization problem can be converted to a minimization problem by multiplying the objective by -1.

WORDS OF WISDOM

- Optimization is difficult!
- *Never* reinvet the wheel! Understand some basics about the main algorithms, but use existing packages to solve your problems.
- No free lunch theorem by Wolpert and Macready (1997) – no optimization algorithm is better than any other algorithm on all problems.
- Be careful: try to understand what you are doing and what the algorithm is doing.
- Never be happy: always check if the algorithm converged, if the solution makes sense, if the solution is robust to changes in starting points, etc.

ALGORITHMS

- All numerical optimization algorithms – same idea: search through the space of feasible choices to find the best one. The difference is in how they search.
  - Comparison methods: compute the objective function at a finite number of points and choose the best one. No need for derivatives etc.
  - Gradient methods: use information about the gradient of the objective function and constraints to guide the search.
  - Curvature methods: use also information about the curvature of the objective function and constraints.

  Comparison methods do not require strong conditions, but they are slow. Curvature methods are fast, but require strong conditions.

UNIDIMENSIONAL OPTIMIZATION

BRACKETING METHODS

- A derivative-free method for univariate f.
- Works only on unimodal function f – otherwise it can converge to a local minimum.
- We can find a bracket in which the global minimum exists if we can find points $a < z < b$ such that

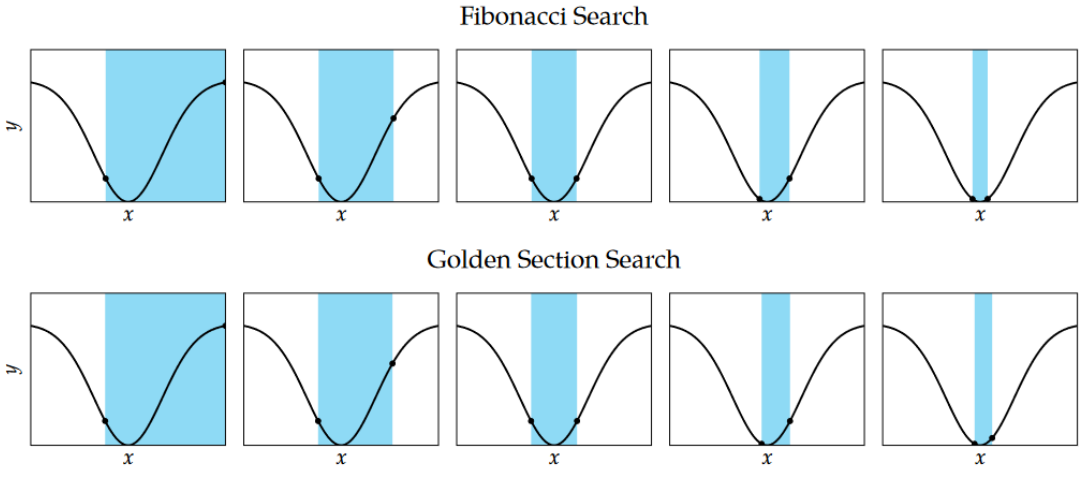
$$f(a) \geq f(z) < f(b) \text{ or } f(a) > f(z) \leq f(b)$$

- Bracketing methods differ in how:
  1. they choose the initial bracket;
  2. they shrink the bracket.

BRACKETING METHODS

- Golden section search: two function evaluations are made at the first iteration and then only one function evaluation is made for each subsequent iteration. Bracket shirnks by a constant factor $\phi \approx 1.618$ (golden ratio).
- Fibonacci search: bracket shrinks by a factor $\frac{F_{n-1}}{F_n}$, where F_n is the n-th Fibonacci number.
- For a large number of iterations, the two methods are equivalent.

BRACKETING METHODS



Source: Kochenderfer and Wheeler (2019)

GOLDEN SECTION SEARCH

- Focus on the bracket $[a, b]$.
- Constant reduction factor c:

$$b_{n+1} - a_{n+1} = c\,(b_n - a_n)$$

- We start with a bracket $[a_0, b_0]$.
- We initialize by evaluating f at two points: $x_1 < x_2$ inside the bracket.
- If $f(x_1) < f(x_2)$, we set $b_1 = x_2$:

$$x_2 = (1 - c)\,a_1 + cb_1$$

- Otherwise, we set $a_1 = x_1$:

$$x_1 = b_1 + c(b_1 - a_1)$$

GOLDEN SECTION SEARCH

- If we know c we know how to update brackets.
- How to choose c?
- Assume for simplicity that the length of the bracket is 1 and $f(x_1) < f(x_2)$.
- We then have a new bracket $[0, x_2]$, where $x_2 = c$ and $x_1 = 1 - c$.
- In the next iteration we reuse $f(x_1)$ and evaluate f at a new point x_2.
- Is $x_1 < x_2$ or $x_1 > x_2$? Let us consider both cases.

GOLDEN SECTION SEARCH

- Suppose $x_1 < x_2$.
- x_1 satisfies two conditions:

$$x_1 = 1 - c$$
$$x_1 = c(1 - c)$$

- This implies $(1 - c)^2 = 0$ with solution $c = 1$.
- But this is useless – $c = 1$ means that we do not shrink the bracket at all.

GOLDEN SECTION SEARCH

- Suppose $x_2 < x_1$.
- x_1 satisfies two conditions:

$$x_1 = 1 - c$$
$$x_1 = c^2$$

- This implies $c^2 + c - 1 = 0$ with a positive solution $c = \frac{-1+\sqrt{(5)}}{2}$. We take it as c.
- $1 + c = \frac{1+\sqrt{5}}{2} = \phi$ – the golden ratio.

NEWTON'S METHOD

- Let x_0 be a point. We can approximate the objective function f by a quadratic function $p(x)$ around x_0:

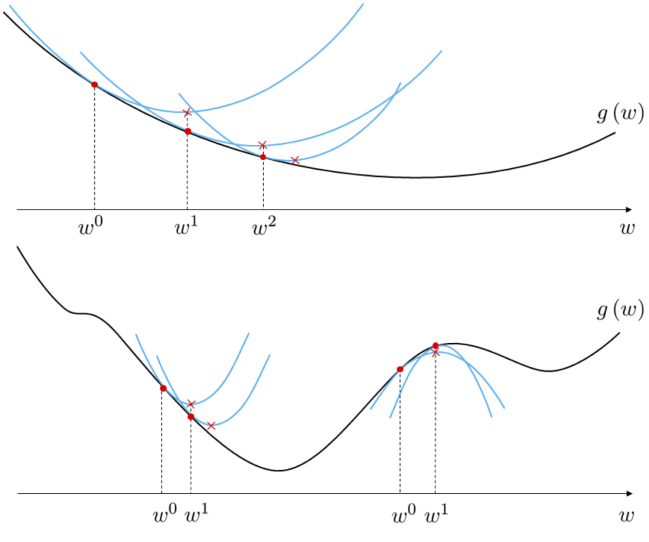
$$p(x) \equiv f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2$$

- The minimum of $p(x)$ is at $x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)}$.
- We can repeat this procedure to get $x_2, x_3, \ldots$, hoping that we will get closer to the extremum of f.
- To summarize:
  1. Start at x_0
  2. Compute $x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$
  3. Stop when $f'(x_n) \approx 0$

NEWTON'S METHOD

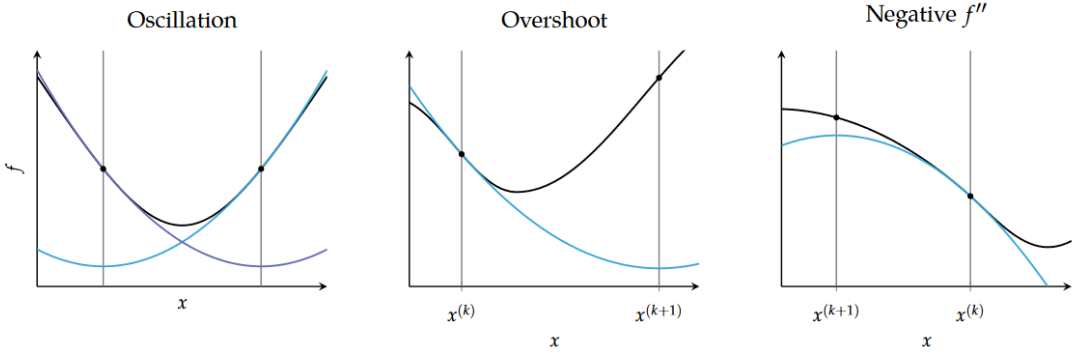
- Newton's method is a curvature method – it uses information about the curvature of the objective function.
- Requires the objective function to have both a continuous first derivative and a continuous second derivative (C^2).
- Fast – quadratic convergence in the neighborhood of the optimum.
- Not robust – it can fail to converge if the starting point is not close enough to the solution.
- Local – it can converge to a local minimum.

NEWTON'S METHOD



Source: here

NEWTON'S METHOD



Source: Kochenderfer and Wheeler (2019)

QUADRATIC CONVERGENCE

Theorem

Suppose $f(x)$ is minimized at x^*, C^3 in the neighborhood of x^*, and that $f''(x^*) \neq 0$. Then there exists $\varepsilon > 0$ such that if $|x_0 - x^*| < \varepsilon$, the sequence

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

converges to x^*.

Moreover

$$\lim_{n \to \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \frac{1}{2} \left| \frac{f'''(x^*)}{f''(x^*)} \right|$$

is the quadratic order of convergence.

STOPPING CRITERION

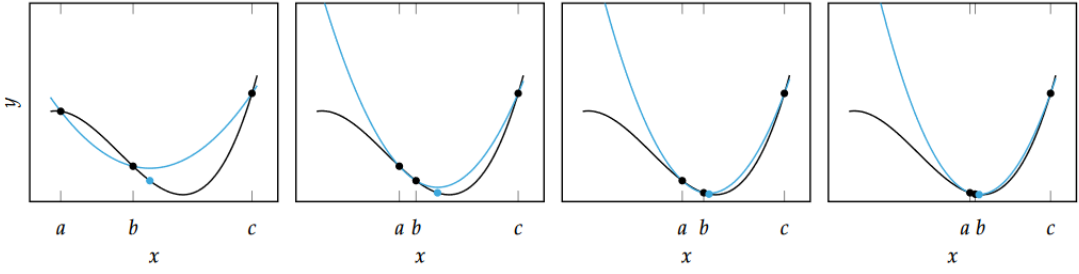
- We want to stop when $f'(x_n) \approx 0$.
- In practice, we stop when $\left|f'(x_n)\right| < \delta$ for some small δ. But this is not enough!
- If $\left|f''(x_n)\right|$ is close to zero, the function is flat – many points with similar f'.
- Stop if both $\left|f'(x_n)\right| < \delta$ and x_n and x_{n+1} are close.
- Be careful: $|x_{n+1} - x_n| = 0.0000001$ means something different if $x_n = 100$ and if $x_n = 0.0000001$!
- Judd (1998) suggest stopping when

$$\left|f'(x_n)\right| < \delta \quad \text{and} \quad |x_{n+1} - x_n| < \varepsilon\left(1 + |x_n|\right).$$

BRENT'S METHOD

- Combine a slow (but robust) bracketing method with a fast (but not robust) method.
- Start with a bracket $[x_0, x_1]$ and find x_2 with GSS.
- Now you have three points – find coefficents of the quadratic function that goes through them.
- You know how find the minimum of a quadratic function – this is our x_3.
- If points are colinear (so that the quadratic function is flat), make a GSS step and try again.
- Pick a new bracket (discard one point) and repeat.

BRENT'S METHOD



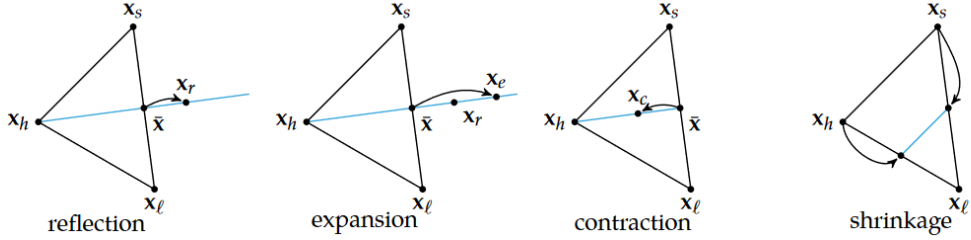
Source: Kochenderfer and Wheeler (2019)

MULTIDIMENSIONAL OPTIMIZATION

POLYTOPE METHOD (NELDER-MEAD)

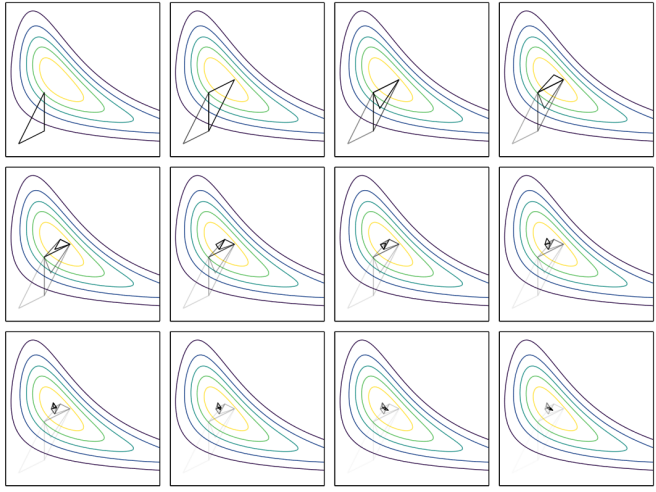
- Amoeba crawling downhill.
- Start with a simplex (triangle in 2D, tetrahedron in 3D, etc.) and move it downhill.
- Find the worst point and:
    1. Reflect it through the centroid of the other points.
    2. If the reflected point is better than the second worst point, try to expand the simplex in that direction.
    3. If the reflected point is worse than the second worst point, try to contract the simplex.
    4. If the reflected point is worse than the worst point, try to contract the simplex towards the best point.

POLYTOPE METHOD (NELDER-MEAD)



Source: Kochenderfer and Wheeler (2019)

POLYTOPE METHOD (NELDER-MEAD)



Source: Kochenderfer and Wheeler (2019)

POLYTOPE METHOD (NELDER-MEAD)

- *"The amoeba is the ideal role model for King Ludd disciples. It crawls to its destination. It has a simple structure. You can watch every move. It has no nervous system." – Ken Judd* (must read!)
- There are known cases where the algorithm fails to converge.
- Slow. For smooth problems it is outperformed by other methods.
- Easy to implement – no derivatives needed.

NEWTON'S METHOD

Let x be a column vector. Denote

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \quad \ldots, \quad \frac{\partial f}{\partial x_n}(x) \right), \quad H(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{i,j}^{n}$$

so $\nabla f(x)$ is the gradient of f and $H(x)$ is the Hessian of f.
Newton's method is the iterative scheme:

$$x^{k+1} = x^k - H\left(x^k\right)^{-1} \left(\nabla f\left(x^k\right)\right)^T.$$

- The idea is that x^{k+1} is the minimum of a quadratic approximation $f(x) \doteq f\left(x^k\right) + \nabla f\left(x^k\right)\left(x - x^k\right) + \frac{1}{2}\left(x - x^k\right)^T H\left(x^k\right)\left(x - x^k\right)$ if $H\left(x^k\right)$ is positive definite.

NEWTON'S METHOD

- Similar remarks as in the univariate case apply.
- It is fast – quadratic convergence in the neighborhood of the optimum.
- It is not robust – it can fail to converge if the starting point is not close enough to the solution.
- It is costly – it requires computing and storing the Hessian.
- New problem: matrix inversion (or solving a system of linear equations) – bad if H is "close to zero.

DIRECTION SET METHODS

- Idea: find the direction in which to move and then find the step size.
- Insight: finding the step size is a univariate optimization problem.
- Generic algorithm:
  1. Choose a search direction s^k.
  2. Find the step size λ^k that (approximately) minimizes $f\left(x^k + \lambda^k s^k\right)$.
  3. Set $s^{k+1} = x^k + \lambda^k s^k$.
  4. Go back to step 1. if not converged.
- Objective: find a direction that forces the function to decrease.

COORDINATE DIRECTION

- Let e^j be the j-th unit vector (vector of zeros with one on the j-th position).
- In iteration $mn + j + 1$ find λ to minimize

$$f\left(x^{mn+j} + \lambda e^j\right)$$

and set

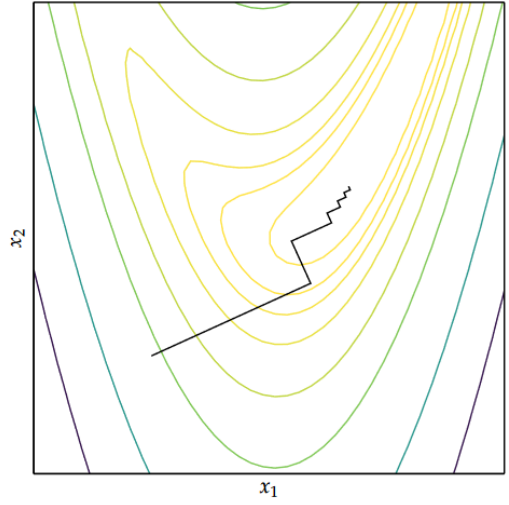
$$x^{mn+j+1} = x^{mn+j} + \lambda e^j.$$

- This changes an multivariate optimization problem to a series of univariate optimization problems.
- Slow.

STEEPEST DESCENT

- In each iteration find direction that reduces the objective function the most per unit length.
- This direction is $s^k = -\nabla f\left(x^k\right)$, the negative gradient.
- Slow. Tends to zig-zag.

STEEPEST DESCENT



Source: Kochenderfer and Wheeler (2019)

NEWTON'S METHOD WITH LINE SEARCH

- Recall that Newton's method is the iterative scheme:

$$x^{k+1} = x^k - H\left(x^k\right)^{-1}\left(\nabla f\left(x^k\right)\right)^T.$$

- This is a special case of the direction set method with

$$s^k = -H\left(x^k\right)^{-1}\left(\nabla f\left(x^k\right)\right)^T \quad \text{and} \quad \lambda = 1.$$

- But we can choose λ to minimize $f\left(x^k + \lambda s^k\right)$.
- Guaranteed to descend, but converges slower that Newton's method.

QUASI-NEWTON METHODS

- Instead of using the exact Hessian, use some other positive definite matrix – easier to compute.
- One popular choice is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method.
- Given some initial positive definite Hessian approximation H_0, the iterative scheme generates a sequence of positive definite matrices.
- One possibility: set H_0 to the identity matrix.
- Another possibility: set H_0 to the Hessian of the objective function at the initial point.
- No need to compute the true Hessian in each iteration.

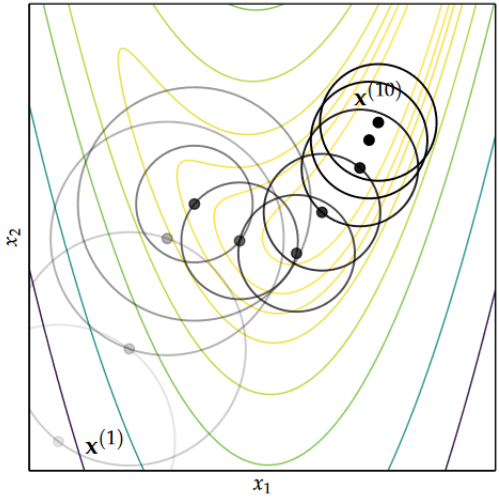
TRUST REGION

- First we set the maximum step size: trust region Δ_k.
- We construct a quadratic approximation m_k of the objective function around x^k: we "trust" this approximation is good in the trust region around x^k:

$$m_k(p) = f\left(x^k\right) + \nabla f\left(x^k\right)^T p + \frac{1}{2} p^T H_k p$$

  where H_k is a positive definite matrix:
- Find the minimum of m_k in the trust region: p^k.
- Set $x^{k+1} = x^k + p^k$ if $f\left(x^{k+1}\right) < f\left(x^k\right)$.
- Trust region increased if the approximation was good, decreased otherwise.

TRUST REGION



Source: Kochenderfer and Wheeler (2019)

CONSTRAINED OPTIMIZATION

OVERVIEW

- Constrained optimization much more difficult than unconstrained optimization.
  - Sometimes we can solve the problem by converting it to an unconstrained problem.
- Penalty methods: convert constrained problem to an unconstrained problem (by adding a penalty term to the objective function).
- Sequential quadratic programming: approximate the objective function by a quadratic and constraints by linear functions.
- Active set methods: consider easier subproblems with only a subset of constraints binding.
- Key takeaway: do not try to implement these methods yourself. Use existing packages.