Optimization

(Lectures on Numerical Analysis for Economists III)

Jesús Fernández-Villaverde,¹ Pablo Guerrón,² and David Zarruk Valencia³
November 18, 2019

¹University of Pennsylvania
²Boston College
³ITAM
Optimization

- Optimization of functions is at the core of most economic models: fundamental behavioral assumption of agents (even when we consider cognitive biases).

- Also, key for most methods is classical econometrics.

- Nowadays: machine learning → large optimization problems that require efficient computation. Think about OLS with thousands of regressors.

- We rarely have closed-form solutions.

- Minimization vs. maximization.

- Why minimization in this class?
Figure 10.0.1. Extrema of a function in an interval. Points $A$, $C$, and $E$ are local, but not global maxima. Points $B$ and $F$ are local, but not global minima. The global maximum occurs at $G$, which is on the boundary of the interval so that the derivative of the function need not vanish there. The global minimum is at $D$. At point $E$, derivatives higher than the first vanish, a situation which can cause difficulty for some algorithms. The points $X$, $Y$, and $Z$ are said to "bracket" the minimum $F$, since $Y$ is less than both $X$ and $Z$. One other section, §10.9, also lies outside of our main thrust, but for a different reason: so-called "annealing methods" are relatively new, so we do not yet know where they will ultimately fit into the scheme of things. However, these methods have solved some problems previously thought to be practically insoluble; they address directly the problem of finding global extrema in the presence of large numbers of undesired local extrema.

The other sections in this chapter constitute a selection of the best established algorithms in unconstrained minimization. (For definiteness, we will henceforth regard the optimization problem as that of minimization.) These sections are connected, with later ones depending on earlier ones. If you are just looking for the one "perfect" algorithm to solve your particular application, you may feel that we are telling you more than you want to know. Unfortunately, there is no perfect optimization algorithm. This is a case where we strongly urge you to try more than one method in comparative fashion. Your initial choice of method can be based on the following considerations:

• You must choose between methods that need only evaluations of the function to be minimized and methods that also require evaluations of the derivative of that function. In the multidimensional case, this derivative is the gradient, a vector quantity. Algorithms using the derivative are somewhat more powerful than those using only the function, but not always enough so as to compensate for the additional calculations of derivatives. We can easily construct examples favoring one approach or favoring the other. However, if you can compute derivatives, be prepared to try using them.

• For one-dimensional minimization (minimize a function of one variable) without calculation of the derivative, bracket the minimum as described in §10.1, and then use Brent's method as described in §10.2. If your function has a discontinuous second (or lower) derivative, then the parabolic...
Some preliminaries I

• Particularly important to implement it well:

  1. Optimization is costly. In fact, we want to avoid it if possible.
  2. Nested inside some loop.
  3. Always possible to miss the exact solution.
  4. Errors might accumulate.
  5. Often, hard to parallelize.
Some preliminaries II

• Transformations of the objective function.

• Including constraints:
  1. Design algorithm: interior point, SQP, trust-region reflective.
  2. Penalty functions and Lagrangian methods.

• When possible, use state-of-the-art software.
  1. NLopt: https://nlopt.readthedocs.io.
  2. IPOPT: https://projects.coin-or.org/Ipopt.

• Test, test, and test.
• Four class of methods:
  1. Use derivatives.
  2. Do not use derivatives.
  4. Simulation methods.
The landscape II

• We will skip:

1. Linear programming (including simplex, interior point, and active-set).
2. Linear-quadratic programming.
3. Integer programming.
4. Multiobjective optimization (including minmax-type problems).
5. Global optimization (including multistart solvers, generalized pattern search (GPS), generating set search (GSS), and mesh adaptive search (MADS)).
Some references


- *Practical Methods of Optimization (2nd ed.)*, by R. Fletcher.

- *Linear and Nonlinear Programming (3rd ed.)*, by David G. Luenberger and Yinyu Ye.

- *Numerical Optimization: Theoretical and Practical Aspects (2nd ed.)* by Joseph-Frédéric Bonnans, Jean Charles Gilbert, Claude Lemarechal, and Claudia A. Sagastizábal.

- *Derivative-Free and Blackbox Optimization* by Charles Audet and Warren Hare.
Derivative-based methods
Newton-Raphson method

- Most common optimization method in economics (either basic implementation or, more likely, with modifications).
- Works with univariate and multivariate optimization problems, but requires twice-differentiability of function.
- Named after Isaac Newton and Joseph Raphson.
- Intimately related with the Newton method designed to solve for root to equation $f(x) = 0$.
- Optimizes $f(x)$ by using successive quadratic approximations to it.
Given an initial guess $x_0$, compute the second-order Taylor approximation of $f(x)$ around $x_0$:

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

The minimization of this approximation with respect to $x$ has first-order conditions

$$f'(x_0) + f''(x_0)(x^* - x_0) = 0$$

which gives:

$$x^* = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

This suggests the iteration

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

that ensures quadratic convergence.
Graphical view

- Move to minimum of quadratic fit at each point
- Quadratic convergence

Newton Raphson
Idea: multivariate case

- For a $N$-dimensional vector function $f(x), x \in \mathbb{R}^N$, we can follow the same steps.

- We get:

$$x_{n+1} = x_n - H_n^{-1}\nabla f(x_n)$$

where $\nabla f(x_n)$ is the gradient of $f(x) = 0$ and $H(\cdot)$ its Hessian.

- Problems:


  2. Local vs. global optima.

  3. Very sensitive with respect to initial guess. You can “cool down” the update (manually or with algorithms).

Heard in Minnesota Econ grad student lab

If you do not know where you are going, at least go slowly.
Quasi-Newton methods

- Evaluating the Hessian is numerically costly: scale $O(n^3)$.
- The Hessian captures the local variation in $\nabla f(x)$.
- First order Taylor approximation of gradient, from $x_n$ yields:
  \[
  \nabla f(x) \approx \nabla f(x_n) + H_n(n - x_n)
  \]
- We want to find a $H_n$ such that:
  1. $H_n$ is symmetric. (Strict concavity can guarantee positive-definiteness).
  2. $\nabla f(x_n)$ evaluated through the approximation should equal to the actual one (secant condition).
  3. $H_{n+1}$ should be as “close” to $H_n$ as possible.
- Different proposals to approximate $H_n$ generate different quasi-Newton.
- For example, we can make $H_n = I$. 
BFGS

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) developed an efficient algorithm to approximate the Hessian:

\[
H_{n+1} = H_n + \frac{yy^T}{y^Ts} - \frac{H_nss^TH_n^T}{s^TH_n s}
\]

\[s = x_{n+1} - x_n\]

\[y = \nabla f(x_{n+1}) - \nabla f(x_n)\]

- If we take into consideration taking inverse of the Hessian, the scale for computation now is \(O(n^2)\).

- Furthermore:

\[
H_{n+1}^{-1} = \left( I - \frac{sy^T}{y^Ts} \right) H_n^{-1} \left( I - \frac{ys^T}{y^Ts} \right) + \frac{ss^T}{y^Ts}
\]

- This is computationally efficient since taking inverse of matrices is very slow.
No-derivative-based methods
Grid search

• We define a grid \([x_1, x_2, \ldots, x_N]\) with \(N\) points.

• We check the function \(f(\cdot)\) at each point of the grid.

• We keep the lowest (highest) value.

• Slow (with strong curse of dimensionality) and it may fail if grid too coarse.

• But, under certain condition it can be quite useful:
  1. Discrete choices.
  2. Monotonicities that we can exploit.
  3. Bracket initial choices for other algorithms.
  4. Easy to parallelize.
Golden section search

- Find minimum $x$ of unimodal continuous $f : X \to R$ in an interval $[a, c]$.
- Searching for minimum vs. maximum.
- By Weierstrass theorem, the minimum exists on $[a, c]$.
- Assume $\exists x \in (a, c)$ and $f(x) < \min[f(a), f(c)]$.

Idea:

1. Select triplet $(a, b, c)$.
2. Update triplet to $(a', b', c')$ with narrower value range that includes maximum.
3. Stop when value range is narrow enough.

Questions:

1. How do we optimally pick triplet $(a, b, c)$?
2. How do we optimally update triplet $(a, b, c)$?
Algorithm

1. Set $b = a + \frac{3-\sqrt{5}}{2} \times (c - a)$.

2. Set $x = a + \frac{\sqrt{5}-1}{2} \times (c - a)$.

3. If $|x - b| < tol$, then exit the algorithm with return $\frac{x+b}{2}$. If not, go to step 4.

4. If $f(b) < f(x)$, update triplet to $(a, b, x)$ and go to step 1.
   
   else, update triplet to $(b, x, c)$ and go to step 1.
Computing the Golden Ratio

- The next $x$ lies either on current $(a, x)$ or on $(b, c)$.
- Minimize the worst by equating the size of the intervals:
  \[
  \frac{b - a}{c - a} = w
  \]

  and
  \[
  \frac{c - b}{c - a} = 1 - w
  \]

- Scale similarity: choose $w$ to minimize expected length of next interval $\rightarrow$ golden ratio $\approx 0.38197$.
- Then:
  \[
  b = a + \frac{3 - \sqrt{5}}{2} \cdot (c - a)
  \]
  \[
  x = a + \frac{\sqrt{5} - 1}{2} \cdot (c - a)
  \]
Tolerance

- $\epsilon$ is your computer’s floating-point precision.

- Taylor expansion: $f(x) \approx f(b) + \frac{1}{2} f''(b)(x - b)^2$.

- If $f(x)$ and $f(b)$ are indistinguishable for our machine, their difference should be of order $\epsilon$:

$$\frac{1}{2} |f''(b)|(x - b)^2 < \epsilon |f(b)| \iff |x - b| < \sqrt{\frac{2\epsilon |f(b)|}{|f''(b)|}}$$

- $|f(b)|/|f''(b)| \approx 1$ implies $|x - b| < \sqrt{e\epsilon}$ (of order $10^{-4}$ if single precision and of order $10^{-8}$ if double precision).
Graphical explanation I

- Consider interval of function where minimum is located.
- Reduce interval until it converges.
Graphical explanation II

- Set triplet \((a, b, c)\).
- Choose \(x\) such that red and blue lines are equal.
- **Golden section**: Relative size of both lines is a particular number.
- More concretely, 
  \[
  \gamma = \frac{x-a}{c-a} = \frac{c-b}{c-a} = \frac{\sqrt{5}-1}{2} \approx 0.618
  \]
Graphical explanation III

- Check whether $f(b)$ or $f(x)$ is lower
Graphical explanation IV

- Ignore part of interval to the left of $b$
- Reset interval $b$ becomes new $a$
Graphical explanation V

- Find new $b$
- Must satisfy same rule as before so: $b = a + \frac{3 - \sqrt{5}}{2} \times (c - a)$
• Check again whether $f(b)$ or $f(x)$ is lower
• Ignore part of interval to the right of $x$
• Reset interval $x$ becomes new $c$
• Find new $x = a + \gamma(c - a)$
• Repeat process until $f(b) \approx f(x)$
Parabolic interpolation

- If the function is parabolic near to the minimum, a parabola fitted through three points will take us to an $\epsilon$—neighborhood of the minimum in a single step.

- Find an abscissa through inverse parabolic interpolation:

$$x = b - \frac{1}{2} \frac{(b - a)^2[f(b) - f(c)] - (b - c)^2[f(b) - f(a)]}{(b - a)[f(b) - f(c)] - (b - c)[f(b) - f(a)]}$$

- This formula fails if the three points are collinear $\Rightarrow$ denominator equals zero
Choose three points of the function and draw a parabola through them.
Graphical explanation II

- Find the minimum of such parabola, evaluate the function at that point, and update points \( c \to b \) and \( b \to x \).
• Draw a second parabola and find its minimum, evaluate, and update points.
• Repeat until convergence.
Brent’s Method

- **Problem**: Formula for $x$ simply finds an extremum, could be a minimum or maximum.

- In practice, no minimization scheme that depends solely on it is likely to succeed.

- **Solution**: Find scheme that relies on a sure-but-slow technique $\Rightarrow$ Combination of golden section search and inverse parabolic interpolation.

- **Brent’s method**: switch between Golden ratio and parabolic interpolation.

- Advantages:
  1. Avoids unnecessary function evaluations in switching between the two methods.
  2. Adequate ending configuration.
  3. Robust scheme to decide when to use either parabolic step or golden sections.
Downhill simplex method

- In one-dimensional minimization, possible to bracket a minimum.

- No analogous procedure in multidimensional space.

- **Downhill Simplex Method** by Nelder and Mead (1965):
  - **Pros**: Requires only function evaluations, not derivatives.
  - **Cons**: Not very efficient.

- **Simplex**: Geometrical figure consisting, in $N$ dimensions, of $N + 1$ points (or vertices) and all their interconnecting line segments, polygonal faces, etc. ($N = 2 \rightarrow$ triangle, $N = 3 \rightarrow$ tetrahedron)
Algorithm

1. Start with $N + 1$ points $\rightarrow$ Initial simplex.

2. Take one of those points to be initial starting point $P_0$.

3. Take other $N$ points to be $P_i = P_0 + \Delta e_i$:
   - $\Delta$: Guess of problem’s characteristic length scale (possibly $\Delta'_i$s for each vector direction).
   - $e_i'$s: $N$ unit vectors, give direction of where to move.

4. Reflection step: Move point of simplex where function is largest through opposite face of simplex to a lower point.

5. Terminate when decrease in value function (or vector distance moved) in last step is fractionally smaller in magnitude than some tolerance.

6. Restart algorithm: Reinitialize $N$ of the $N + 1$ vertices of the simplex again w/ previous equation, w/ $P_0$ being one of the vertices of the claimed minimum.
Different transformations

Figure 10.4.1. Possible outcomes for a step in the downhill simplex method. The simplex at the
beginning of the step, here a tetrahedron, is shown, top. The simplex at the end of the step can be any one
of (a) a reflection away from the high point, (b) a reflection and expansion away from the high point, (c) a
contraction along one dimension from the high point, or (d) a contraction along all dimensions towards
the low point. An appropriate sequence of such steps will always converge to a minimum of the function.

Refections, and they are constructed to conserve the volume of the simplex (hence
maintain its nondegeneracy). When it can do so, the method expands the simplex
in one or another direction to take larger steps. When it reaches a "valley floor,"
the method contracts itself in the transverse direction and tries to ooze down the
valley. If there is a situation where the simplex is trying to "pass through the eye
of a needle," it contracts itself in all directions, pulling itself in around its lowest
(best) point. The routine name amoeba is intended to be descriptive of this kind of
behavior; the basic moves are summarized in Figure 10.4.1.

Termination criteria can be delicate in any multidimensional minimization
routine. Without bracketing, and with more than one independent variable, we
no longer have the option of requiring a certain tolerance for a single independent
Powell’s method

- If start at point $P$ in $N$-dimensional space, and proceed in vector direction $n$, then any function of $N$ variables $f(P)$ can be minimized along the line $n$ by one-dimensional methods.

- Efficiency depends on how the next direction $n$ is chosen.

- **Powell’s Method** provides set of $N$ mutually conjugate directions.
  - Two vectors $u$ and $v$ are *conjugate* with respect to $Q$ (or $Q$-orthogonal) if $u^T Q v = 0$.

- Use this set to efficiently perform line minimization (reach minimum after $N$ line minimizations if $f$ quadratic).
Graphical explanation
Original algorithm

Initialize the set of directions $\mathbf{u}_i$ to the basis vectors: $\mathbf{u}_i = \mathbf{e}_i$, $i = 0, \ldots, N - 1$.

Repeat following sequence of steps until function stops decreasing:

1. Save your starting position as $\mathbf{P}_0$.

2. For $i = 0, \ldots, N - 1$, move $\mathbf{P}_i$ to the minimum along direction $\mathbf{u}_i$ and call this point $\mathbf{P}_{i+1}$.

3. For $i = 0, \ldots, N - 2$, set $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$.

4. Set $\mathbf{u}_{N-1} \leftarrow \mathbf{P}_N - \mathbf{P}_0$.

5. Move $\mathbf{P}_N$ to the minimum along direction $\mathbf{u}_{N-1}$ and call this point $\mathbf{P}_0$. 
Corrected algorithm

**Problem:** throwing away, at each stage, $u_0$ in favor of $P_N - P_0$ tends to produce sets of directions that “fold up on each other” and become linearly dependent.

**Solutions:**

1. Reinitialize the set of directions $u_i$ to the basis vectors $e_i$ after every $N$ or $N + 1$ iterations of the basic procedure.

2. Reset the set of directions to the columns of any orthogonal matrix.

3. Still take $P_N - P_0$ as new direction discarding the old direction along which the function $f$ made its *largest decrease*. 
Mixed methods
Brent’s method with first derivatives

- Same goal as w/o derivative: Isolate minimum bracketed, but now use information from derivative.
- Not enough to simply search for a zero of the derivative $\rightarrow$ Maximum or minimum?
- Derivatives only useful in choosing new trial points within bracket.
  - If $f'(b) > 0 \rightarrow$ next test point from interval $(a, b)$.
  - If $f'(b) < 0 \rightarrow$ next test point from interval $(b, c)$. 
Suppose our function $f$ is roughly approximated as a quadratic form:

$$f(x) \approx \frac{1}{2}x^T Ax - b^T x + c$$

$A$ is a known, square, symmetric, positive-definite matrix.

Then $f(x)$ is minimized by the solution to

$$Ax = b$$

Remember now we can calculate $f(P)$ and $\nabla f(P)$ for a given $N$-dimensional point $P$.

How can we use this additional information?
An tempting (but not very good) possibility: steepest descent method.

Start at a point $P_0$. As many times as needed, move from point $P_i$ to the point $P_{i+1}$ by minimizing along the line from $P_i$ in the direction of the local downhill gradient $-\nabla f(P_i)$.

Performs many small steps $\rightarrow$ Not very efficient!
Steepest descent method
Conjugate gradient method

- A better way.
- In $\mathbb{R}^N$ take $N$ steps each of which attains the minimum in each direction, w/o undoing previous steps’ progress.
- In other words, proceed in a direction that is somehow constructed to be conjugate to the old gradient, and to all previous directions traversed.
Conjugate Gradient Method
1. Let $d_0 = r_0 = b - Ax_0$.

2. For $i = 0, 1, 2, ..., N - 1$ do:
   - $\alpha_i = \frac{r_i^T r_i}{d_i^T A d_i}$.
   - $x_{i+1} = x_i + \alpha_i d_i$.
   - $r_{i+1} = r_i - \alpha_i A d_i$.
   - $\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$.
   - $d_{i+1} = r_{i+1} + \beta_{i+1} d_i$.

3. Return $x_N$. 

Algorithm - linear
1. Let $d_0 = r_0 = -f'(x_0)$.

2. For $i = 0, 1, 2, \ldots, N - 1$ do:
   - Find $\alpha_i$ that minimizes $f'(x_i + \alpha_i d_i)$.
   - $x_{i+1} = x_i + \alpha_i d_i$.
   - $r_{i+1} = -f'(x_{i+1})$.
   - $\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$ or $\beta_{i+1} = \max \left\{ \frac{r_{i+1}^T (r_{i+1} - r_i)}{r_i^T r_i}, 0 \right\}$.
   - $d_{i+1} = r_{i+1} + \beta_{i+1} d_i$.

3. Return $x_N$. 
Simulation methods
• We explore a function \( f(\cdot) \) by randomly drawing from it.

• Algorithm:

1. Given a state of the chain \( x_{n-1} \), we generate a proposal:

\[
x^* = x_{n-1} + \lambda \varepsilon, \; \varepsilon \sim \mathcal{N}(0, 1)
\]

2. We compute:

\[
\alpha = \min \left\{ 1, \frac{f(x^*)}{f(x_{n-1})} \right\}
\]

3. We set:

\[
x_n = x^* \text{ w.p. } \alpha \\
x_n = x_{n-1} \text{ w.p. } 1 - \alpha
\]

4. Keep \( x_n \) which yields the highest \( f(\cdot) \).
• Why does it work? Harris recurrence.
• Particularly easy to implement.
• Transformations of $f(\cdot)$.
• More sophisticated proposals.
• Also, it is straightforward to incorporate complex constraints.
• Equivalent to simulated annealing: iteration-varying $\lambda$ ("cooling down").
Genetic algorithms

- Large class of methods.
- Fraser and Burnell (1970) and Holland (1975).
- Build on two basic ideas of evolution:
  1. Random mutation (sexual or asexual reproduction).
  2. Survival-of-the-fittest.
- Not very efficient set of methods...
- ...but it can handle even the most challenging problems.
- They can be mixed with traditional methods.
Genetic algorithm basic structure

Population

GA Operators

Mutation

Crossover

Reproduction

Evolution Environment

Evaluation

Fitness value