Financial Modeling and Information Technology

Budapest Technology Centre

Morgan Stanley

Multidimensional Optimization

Zsolt Pándi

Morgan Stanley Hungary Analytics Ltd.

Session Outline

- Problem Definition
- One-dimensional Optimization
- Multidimensional Optimization
  - Methods without Using Derivatives
  - Methods with Using Derivatives
  - Heuristics Using Random Numbers
- Tips&Tricks, Suggestions
- References

Problem Definition

- We have a (possibly) non-linear, but deterministic function f(**x**) that depends on one or more independent variables.
- f(**x**) is probably expensive to evaluate, and we may or may not be able to efficiently and/or accurately (!) compute its gradient or Hessian.
- Problem: find a minimum of f(**x**) with as few evaluations as possible.
- Examples of real-life problems
  - MLE, model fitting, cost minimization, performance maximization
- How do we recognize what we are looking for?
  - local vs. global optima
  - trying to find the longest straw in the haystack

One-dimensional Optimization: Golden Section Search

- Analogy with bisection root search.
- Given an initial bracket of a minimum [a,b,c], i.e. a < b < c, and f(b) < f(a), and f(b) < f(c), we take a guess at a point x either in (a,b) or in (b,c). We evaluate f(x) and update our bracket accordingly.
  - e.g. x is in (a,b): if f(x) < f(b) then the new bracket is [a,x,b], else the new bracket is [x,b,c]
- We repeat this until the bracket is small enough.
  - e.g. suppose that b is fraction a fraction w of the way between a and c, so w = (b-a)/(c-a). Let the next trial point be a fraction z before b, so z = (b-x)/(c-a). Hence the size of the next bracket is either w or or 1-w+z. Minimizing the worst case probability yields z = 2w-1. z is positive only if x is in the larger segment (w > ½). If z is "optimal" then so was w, which suggests that z / w = 1-w, which in turn gives $w^2+w-1=0$, i.e. w ≈ 0.61803.
- Optimal choice of x is (1-w) fraction into the larger interval from the middle.

A Note on Accuracy

- Convergence is linear: count of successive significant figures grows linearly with iterations.
- Do not set tolerance in x to lower than the square root of your numerical precision.
  - Close to a minimum at b, $f(x) \approx f(b) + \frac{1}{2} f''(b)(x - b)^2$, because the first derivative vanishes.
  - The second term above is ε times smaller than the first, i.e. $\varepsilon |f(b)| > \frac{1}{2} f''(b)(x - b)^2$ if

    $|x - b| < \sqrt{\varepsilon}\, |b| \sqrt{(2 |f(b)| / \{ b^2 f''(b) \})}$, where the final square root is a number of order one for most functions.
  - Unless you know a better estimate for the final square root, apply the limit suggested above.

One-dimensional Optimization: Fancier Methods

- Parabolic interpolation: try to fit a parabola through [a,b,c] and jump to its minimum as a guess for x.
  - Caveats: parabolic interpolation may converge to a maximum, or the fit might not be feasible because of collinearity.
- An ideal scheme can avoid unnecessary evaluations, switch between a robust and slow (e.g. Golden Section) technique and parabolic interpolations as f(x) permits, and carefully define a stopping criterion.
- Brent's method is good at all the above at the expense of maintaining six function points instead of three and defining robust rules for acceptance of guesses.

One-dimensional Optimization: Fancier Methods (Cont'd)

- What if we can also use derivatives?
- Keeping the bracketing idea and updating the bracket based on function as opposed to derivative information is more robust.
- You can try to fit a higher order polynomial to function and derivative information.
- You can also try to select the interval to look at based on the derivative at the middle point.
- The latter idea combined with extrapolation to zero of the derivative and robust rules for acceptance of the results as in Brent's method appears to work well.

Multidimensional Optimization: An Overview

- Methods without using derivatives
  - Downhill simplex
  - Line methods in general
  - Direction set methods (line methods)
- Methods with using derivatives
  - Conjugate gradient methods (line methods)
  - Quasi-Newton methods
  - Levenberg-Marquardt method
- Methods using random numbers
  - Simulated annealing
  - Genetic algorithm
  - Ant colony optimization

Downhill Simplex

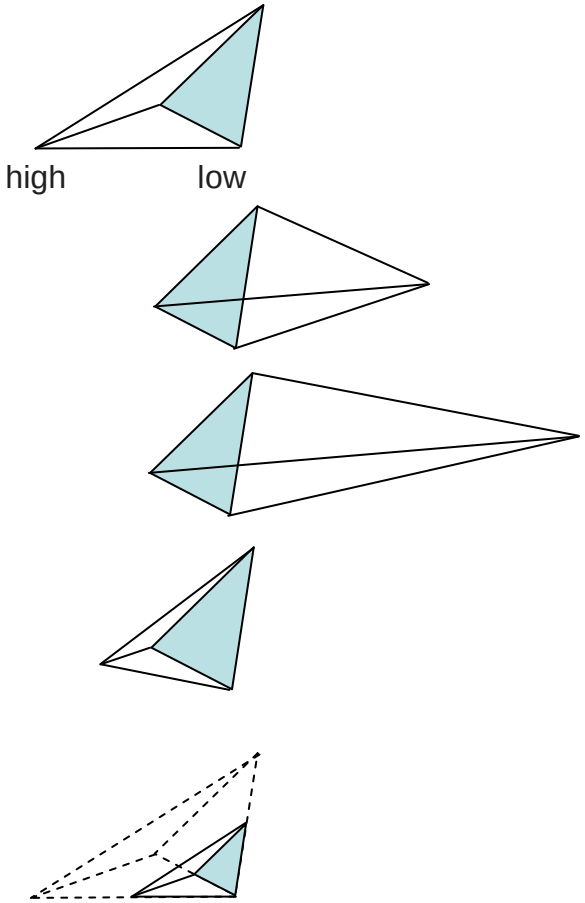
- Definitely not the best in the number of required function evaluations, but can be used for initial trials. Also, sometimes more robust than other methods.
- A simplex is a geometrical figure consisting of N+1 vertices and all their interconnecting line segments or polygonal faces, etc. in N dimensions.
- We are now interested in simplexes with a finite inner N-dimensional volume (>0).
- We need an initial simplex when starting the algorithm.
  - e.g. $\mathbf{P}_0$ is an initial starting point, and $\mathbf{P}_i = \mathbf{P}_0 + \Delta_i \mathbf{e}_i$, where $\mathbf{e}_i$ are the unit vectors and Δ_i are some characteristic length scales.
- The one-dimensional bracketing does not work, so we take a slightly different approach to modify the simplex in the iterations of the algorithm.

Downhill Simplex (Cont'd)

- Given an initial simplex, the following list of moves

  is considered at each iteration:

  - **reflection:** most of the time just move the point with the largest function value through the opposite face of the simplex
  - **reflection and expansion:** the above combined with increasing the search step size, if reflection yields a significantly better point
  - **contraction:** when in a "valley", the simplex is contracted in the transverse direction, i.e., when the reflected point is worse than the one with the second largest function value
  - **multiple contraction:** when passing through the eye of a needle it is best to contract around the point with the lowest function value, i.e., when none of the above works



Downhill Simplex (Cont'd)

- Solution at each iteration is the best point found so far.
- Stopping criteria can be tricky:
  - terminate when vector distance of last move is less than some tolerance limit (not smaller than square root of machine precision)
  - terminate when decrease of function value in the last iteration is less than some tolerance limit (can be approx. machine precision)
  - it is often a good idea to restart the algorithm where it claims to have found a minimum, because an anomalous step might cause the stopping criteria to be triggered

Line Methods in General

- Let us now build on the available one-dimensional routines.
- If we have a starting point **P** and a vector **n** in N dimensions, then (1) we can use our one-dimensional minimization routine to minimize $f(\lambda) = f(\mathbf{P}+\lambda\mathbf{n})$. Thus **n** is the search direction.
- We can then (2) reset our starting point to the minimum found along the search direction and (3) determine a new search direction and continue with step (1) above.
- The various line methods differ by how they choose the search direction.

Direction Set Methods (Line Methods)

- First, not too bad idea:
  - take a set of directions, e.g. the unit vectors $\mathbf{e}_i$,
  - iterate over the set of directions and minimize along the current direction, then from there along the next direction, and so on until the function stops decreasing.
- Problem: what if the function has a narrow valley at an angle to the unit vectors?
- Let us use conjugate directions,
  - which are directions that do not impact the efficacy of minimization along themselves.
  - If we minimize along some direction **u**, then the gradient of the function must be perpendicular to **u** at the minimum.
  - Using Taylor series: $f(\mathbf{x}) = f(\mathbf{p}) + \mathbf{b}\,\mathbf{x} + \frac{1}{2}\,\mathbf{x}\,\mathbf{A}\,\mathbf{x} + \ldots$, where **b** is the gradient at **p** and **A** is the Hessian at **p**.
  - The approximation of the gradient at **x** is $\mathbf{A}\,\mathbf{x} + \mathbf{b}$ (derivative with respect to **x**).
  - The gradient vanishes at **x** that solves $\mathbf{A}\,\mathbf{x} = -\mathbf{b}$, and change in the gradient when moving along **x** is **A** ($\delta\mathbf{x}$).
  - If we move along direction **u** to a minimum, then a new direction **v** does not spoil our minimization as long as $0 = \mathbf{u}\,\mathbf{A}\,\mathbf{v}$, i.e., the change in the gradient along **v** is perpendicular to **u** (note that **u** and **v** are conjugate vectors).

Direction Set Methods (Cont'd)

- If we can find a set of N linearly independent, mutually conjugate directions, then N line minimizations will reach exactly the minimum of a quadratic form.
- If our f(**x**) is not exactly quadratic, we will be close, and repeated cycles will yield quadratic convergence.
- Powell's method: (1) Initialize the set of directions $\mathbf{u}_i$ to the basis vectors $\mathbf{e}_i$. (2) Save your starting position as $\mathbf{P}_0$. (3) For i = 0,…N-1 move $\mathbf{P}_i$ to the minimum along direction $\mathbf{u}_i$ and call this point $\mathbf{P}_{i+1}$. (5) For i = 0,…N-2 set $\mathbf{u}_i := \mathbf{u}_{i+1}$. (6) Set $\mathbf{u}_{N-1} := \mathbf{P}_N - \mathbf{P}_0$. (7) Move $\mathbf{P}_N$ to the minimum along direction $\mathbf{u}_{N-1}$, and call this point $\mathbf{P}_0$. (8) Repeat from step (3).
- N iterations, that is, N(N + 1) line minimizations will exactly minimize a quardatic form.
- Problem: throwing away $\mathbf{u}_0$ in favor of $\mathbf{u}_{N-1}$ tends to produce linearly dependent directions.
- One (and not the best) solution: rerun step (1) after every N or N+1 iterations.

Conjugate Gradient Methods (Line Methods with Derivatives)

- Let us now return to $f(\mathbf{x}) \approx c + \mathbf{b}\,\mathbf{x} + \frac{1}{2}\,\mathbf{x}\,\mathbf{A}\,\mathbf{x}$. Each element of **b** and **A** can affect the location of the minimum, so the information content of this form is $\sim N^2$.
- When minimizing via direction sets, we collect this amount of information via $O(N^2)$ separate line minimizations. If we can easily evaluate the gradient, we get N pieces of new information and hence O(N) carefully chosen line minimizations should suffice.
- Note that the gradient may need O(N) function evaluation time to compute, but there might be repeating computations to take advantage of. Also, each gradient evaluation spares a line minimization, which itself requires possibly many function calls.

Conjugate Gradient Methods (Cont'd)

- Steepest descent performs poorly: takes many steps in a long valley.
- Ideally, we want to proceed along a direction that is chosen to be conjugate to the previous gradient, as well as all previous directions to the extent possible.
- Conjugate gradient methods make use of the technique available for solving sparse linear systems of equations in the context of function minimization.

Conjugate Gradient Methods (Cont'd)

- Let an arbitrary vector $\mathbf{g}_0$ be our first auxiliary vector. Let the first direction be $\mathbf{h}_0 = \mathbf{g}_0$.
- Let $\mathbf{g}_{i+1} = \mathbf{g}_i - \lambda_i \mathbf{A}\,\mathbf{h}_i$, and let $\mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma_i \mathbf{h}_i$, where

  $\lambda_i = \mathbf{g}_i\,\mathbf{g}_i / (\mathbf{h}_i\,\mathbf{A}\,\mathbf{h}_i)$, and $\gamma_i = \mathbf{g}_{i+1}\,\mathbf{g}_{i+1} / (\mathbf{g}_i\,\mathbf{g}_i)$ [improvement: $\gamma_i = (\mathbf{g}_{i+1} - \mathbf{g}_i)\,\mathbf{g}_{i+1} / (\mathbf{g}_i\,\mathbf{g}_i)$].
- The generated sequence will satisfy $\mathbf{g}_i\,\mathbf{g}_j = 0$, $\mathbf{h}_i\,\mathbf{A}\,\mathbf{h}_j = 0$ and $\mathbf{g}_i\,\mathbf{h}_j = 0$ for $j < i$.
- Problem: as opposed to the case of solving the linear system of equations, here we don't know **A**.
- Solution: let $\mathbf{g}_i = -\mathit{grad}\ f(\mathbf{p}_i)$ and we go along the direction $\mathbf{h}_i$ to the local minimum of $f(\mathbf{x})$ at $\mathbf{p}_{i+1}$. Let then $\mathbf{g}_{i+1} = -\mathit{grad}\ f(\mathbf{p}_{i+1})$. This is equivalent to our choice of $\mathbf{g}_{i+1}$ above, and does not require the knowledge of **A**.

Conjugate Gradient Methods (Cont'd)

- Described method is called Fletcher-Reeves. Improvement is called Polak-Ribiere: provides smoother transition between iterations and falls back to using local gradient when additional gain is lost.

Quasi-Newton Methods (Methods with Derivatives)

- Let us now again return to $f(\mathbf{x}) \approx f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i)\, \mathit{grad}\, f(\mathbf{x}_i) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_i)\, \mathbf{A}\, (\mathbf{x} - \mathbf{x}_i)$. Recall that we do not know **A**, and so our aim is to collect its information content somehow.
- Basic idea: try to construct an iterative approximation $\mathbf{H}_i$ to the matrix $\mathbf{A}^{-1}$ to mimic Newton's minimum search method. That is, making the gradient at **x**, $\mathit{grad}\, f(\mathbf{x}_i) + \mathbf{A}\, (\mathbf{x} - \mathbf{x}_i)$, equal to zero yields the equation $\mathbf{x} - \mathbf{x}_i = -\mathbf{A}^{-1}\, \mathit{grad}\, f(\mathbf{x}_i)$, where we use $\mathbf{H}_i$ to replace $\mathbf{A}^{-1}$.
- The name "Quasi"-Newton comes from not using the actual Hessian but the current approximation of its inverse only. It is actually better, because we need to descend, hence we want $\mathit{grad}\, f(\mathbf{x}_i)\, (\mathbf{x} - \mathbf{x}_i) = -(\mathbf{x} - \mathbf{x}_i)\, \mathbf{A}\, (\mathbf{x} - \mathbf{x}_i) < 0$, which means **A** must be positive definite. There is no guarantee that the Hessian will always be positive definite, but the series $\mathbf{H}_i$ can be defined so.

Quasi-Newton Methods (Cont'd)

- Derivation of the iterative approximation $\mathbf{H}_i$ is somewhat involved, but the two main updating formulas are referred to as Davidon-Fletcher-Powell (DFB) and Broyden-Fletcher-Goldfarb-Shanno (BFGS), the latter being recognized as empirically superior.
- Note that it might be worth using Quasi-Newton Methods with finite difference approximations of the gradient to decrease total computation effort compared to using methods without derivatives.

Levenberg-Marquardt Method (Methods with Derivatives)

- Let's say now that our $f(\mathbf{a}) = \chi^2(\mathbf{a}) = \sum[\{y_i - y(x_i|\mathbf{a})\}/\sigma_i]^2$, i.e., we are looking for a least-square fit. We again use the quadratic approximation $\chi^2(\mathbf{a}) \approx c + \mathbf{b}\,\mathbf{a} + \frac{1}{2}\,\mathbf{a}\,\mathbf{A}\,\mathbf{a}$, which suggests using either $\mathbf{a}_{min} = \mathbf{a}_{cur} + \mathbf{A}^{-1}[-\mathit{grad}\ \chi^2(\mathbf{a}_{cur})]$, if the quadratic approximation is good enough, or a steepest descent $\mathbf{a}_{next} = \mathbf{a}_{cur} - \text{constant} * \mathit{grad}\ \chi^2(\mathbf{a}_{cur})$, if it isn't.
- Here we do know the form of the Hessian:

  $\alpha_{kl} = \partial^2\chi^2 / (\partial a_k \partial a_l) = 2\sum 1/\sigma_i^2\ [\ \partial y(x_i|\mathbf{a}) / \partial a_k * \partial y(x_i|\mathbf{a}) / \partial a_l - \{y_i - y(x_i|\mathbf{a})\} * \partial^2 y(xi|\mathbf{a}) / (\partial a_l \partial a_k)\ ]$
- Now, we neglect the second partial derivatives in the above expression, because
  - they are often small enough to be negligible in practice;
  - the term multiplying them should be a random, uncorrelated measurement error in a successful model, which can have either sign, so they will probably cancel;
  - they might lead to destabilizing in the presence of outliers.

Levenberg-Marquardt Method (Cont'd)

- Levenberg-Marquardt solves the equation $\sum \alpha_{kl} \Delta a_l = \beta_k$ (r.h.s. is the gradient) by replacing α_{kl} with z_{kl}, where $z_{jj} = \alpha_{kl} (1 + \lambda)$, and $z_{jk} = \alpha_{jk}$. This will ensure a smooth scaling between the two extremes of using the inverse Hessian and using the steepest descent.
- Outline of algorithm is then: (1) given an initial guess **a**, and an initial setting of λ (=0.001), (2) solve the above system of equations for **Δa**. (3) If $\chi^2(\mathbf{a} + \mathbf{\Delta a}) >= \chi^2(\mathbf{a})$, then increase λ by a significant factor, otherwise decrease λ by a significant factor. (4) Continue with step (2) after setting **a** := **a** + **Δa**.
- The stopping condition can be tricky. Decrease in χ^2 less than 1 is often not significant statistically, but 0.001 will put you on the safe side. Method tends to wander around near the minimum in a flat valley.

Simulated Annealing

- Now think about f(**x**) as a function with multiple local minima.
- Apply the analogy of liquids freezing and crystallizing from thermodynamics.
  - At high temperatures, the molecules move freely.
  - At lower temperatures this mobility is gradually lost.
  - If cooling is slow, the molecules can arrange in an ordered fashion leading to (or near to) the state of the lowest energy.
  - If cooling is fast, the result is a polycrystalline or amorphous state with higher energy.
  - The key is to keep cooling slow to allow for atoms to redistribute.
  - A state of energy E is occupied with probability ~exp(-E/kT) at temperature T. k is Boltzmann's constant.

Simulated Annealing (Cont'd)

- Outline of the Metropolis algorithm
  - Define an initial temperature. (How high?)
  - Generate a random move from state $\mathbf{x}$ to $\mathbf{x}+\Delta\mathbf{x}$. (What moves to pick from? How to choose?)
  - Accept new state with probability $\min(\exp\{-[f(\mathbf{x}+\Delta\mathbf{x})-f(\mathbf{x})]/kT\}, 1)$.
  - Decrease T. (How fast? Repeats at the same temperature?)

Genetic Algorithm

- Think about -f(**x**) as a measure of success of an individual in a population (fitness function).

  Think about **x** as the genome of the individual.

- Define the following rules of evolution
  - Set up an initial population. (What size? Which individuals?)
  - Select the part of the current population for reproduction. (Randomized based on fitness.)
  - Apply crossover and mutation operators to selected subpopulation. (How to define operators? More than two parents?)
  - Evaluate termination criteria (Number of generations? Average fitness?) and continue with selection step, if needed.
- The details are highly problem-specific.

Ant Colony Optimization

- How do ants search for food? How do they form trails connecting the colony with the food source?
- To illustrate the concept, think about **x** as a Hamiltonian cycle in a given graph with given non-negative edge weights. Let f(**x**) be the sum of the weights of edges in the cycle. Minimizing f(**x**) now amounts to solving the traveling salesman problem.

Ant Colony Optimization (Cont'd)

- ACO solution to traveling salesman problem:
  - Let's imagine a number of ants that walk on the graph.
  - We define the following rules for one iteration:
    - Each ant must visit each node exactly once.
    - The smaller the weight and the stronger the pheromone trail the more likely that an ant chooses a particular edge as a next step on its cycle.
  - We iterate over the following steps:
    - Each ant visits a Hamiltonian cycle according to the rules above.
    - Each ant lays pheromone trail on the edges it visited in the current iteration. The shorter the Hamiltonian cycle the stronger the trail.
    - All pheromone trails evaporate (weaken).

Tips&Tricks, Suggestions

- Get to know your f(**x**) as much as possible to identify qualitative behavior, parameter ranges of interest and signs of numerical errors.
- Try to use as much information about f(**x**) as possible, e.g. go for LM if f(**x**) is χ^2.
- Try to match the scales of function parameters (coordinates in **x**).
- Try changing your initial guess.
- When working on a constrained problem, hide the constraints from the optimizer by applying variable transforms.
  - e.g. f(exp(y)) lets the optimizer work in the range $(-\infty, \infty)$ even if the domain of f(x) is R^+.
- Recall the shape of objective function value plotted against iterations taken in a random optimization method.
- There is not any single best recipe, so experiment with different approaches.

References

Most of this presentation is based on

Press-Tseukolsky-Vetterling-Flannery: Numerical Recipes, 3rd ed,

a highly recommended reference.

Q & A

References