CogSci 109: Lecture 19

Wednesday Nov. 21, 2007
More optimization, function minimization: Nelder-Mead explanation/examples, introduction to gradient descent

Outline for today

- Announcements
- Nelder-Mead Simplex explanation, more examples and matlab
- Introduction to gradient descent

Announcements

- Compliment vs. complement
  - **Compliment -** An expression of praise, admiration, or congratulation.
  - **Complement -**
    - An angle related to another so that the sum of their measures is 90°
    - Either of two parts that complete the whole or mutually complete each other
    - More formally:

In general, the word "complement" refers to that subset F' of some set S which excludes a given subset F. Taking F and its complement F' together then gives the whole of the original set. The notations F' and $\overline{F}$ are commonly used to denote the complement of a set F.

- (**http://mathworld.wolfram.com/Complement.html**)

Announcements

- Reminder - to get 5 bonus points, homework must be turned in today before midnight via email or in person
  - **If you turn in via email you must turn in a printed version on Monday**
- 100 words for 3.4 ->200, still 500 max per assignment

Last time we discussed the Nelder-Mead Simplex method

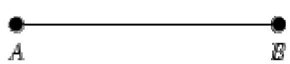
- It's the built-in nonlinear function minimization routine in Matlab
- **fminsearch()**
- One of the most widely used methods of unconstrained nonlinear optimization
- Published in 1965
  - J. A. Nelder and R. Mead, A simplex method for function minimization, Computer Journal 7 (1965), 308–313.
  - See linked page on website for (short) collection of NM papers

What does NM do?

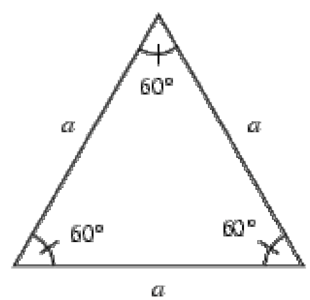
- Uses a simplex (a polytope in N+1 vertices in N dimensions)
  - **A line segment on a line**
  - **A triangle on a plane**
  - **A tetrahedron in 3d space, etc**
- Finds an approximate locally optimal solution to a problem with N variables (if the objective function varies smoothly)

What does a simplex look like?

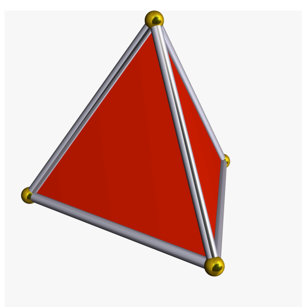
- Think of it as an N-Dimensional triangle
  - For specifics, start by reading *mathworld* and *wikipedia* definitions of ***simplex*** and related important details like ***convexity*** and ***convex hulls***:
    - ***http://en.wikipedia.org/wiki/Simplex***
    - ***http://mathworld.wolfram.com/Simplex.html***



1D ->line



2D ->triangle



3D ->tetrahedron

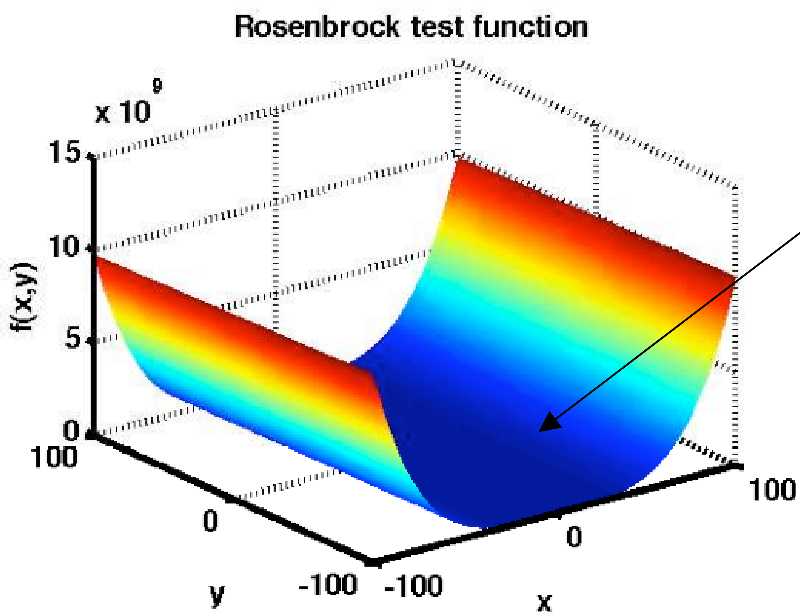
How does NM use the simplex?

- Let's see - first consider the following challenging objective function we want to minimize over the variables x and y ***(this is a typical test problem for optimization algorithms)***

$$\boxed{f(x,y) = (1-x)^2 + 100(y-x^2)^2}$$

Why is this challenging?

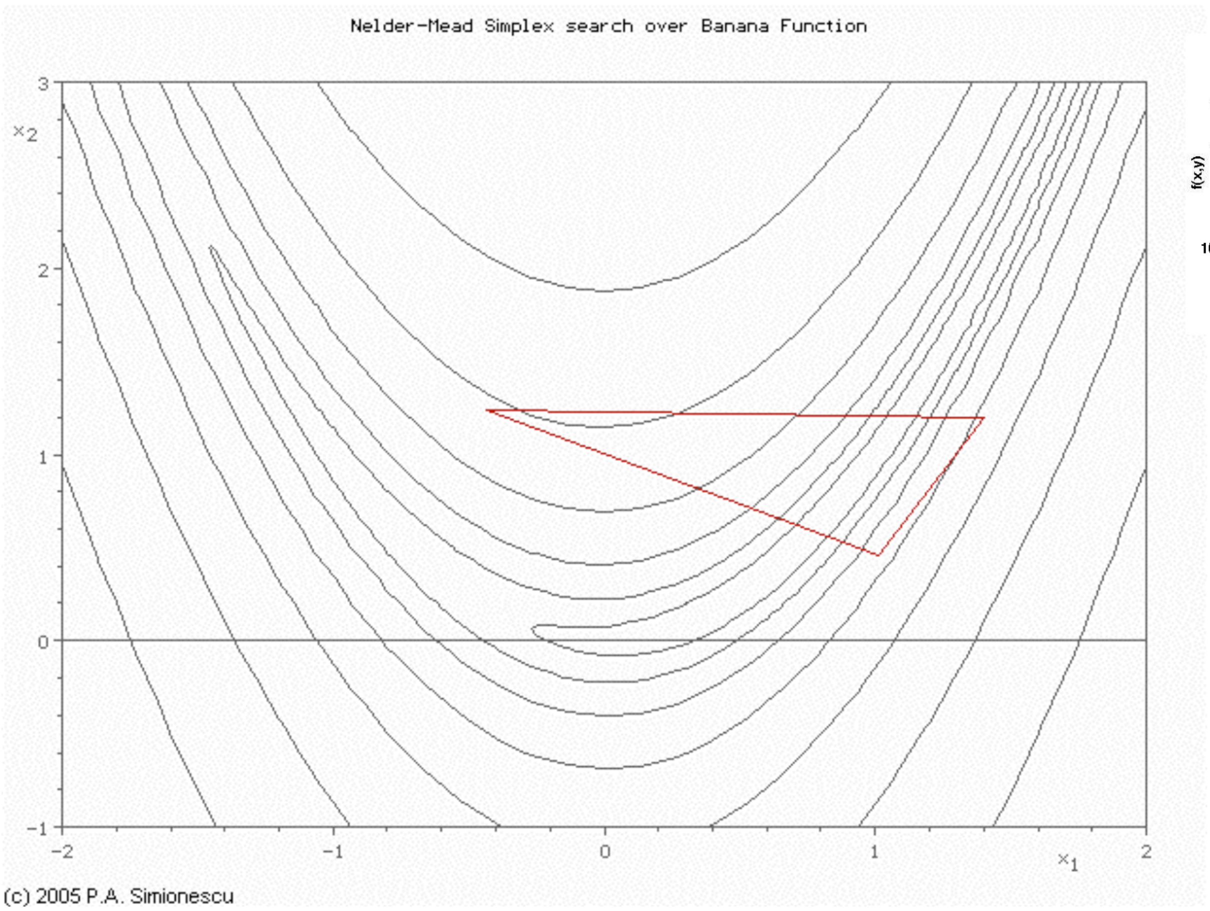
***A.k.a. - Rosenbrock's valley** or **Rosenbrock's banana function**.*

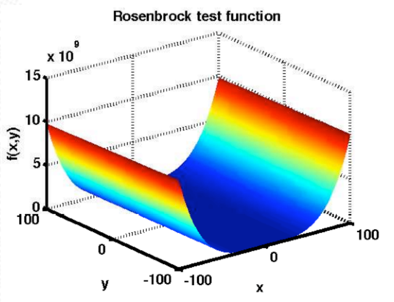


Note the long narrow valley. That makes it tough to find the global minimum with an optimization algorithm

Let's take a look at the NM simplex algorithm in action

- The NM algorithm trying to minimize the Rosenbrock function:





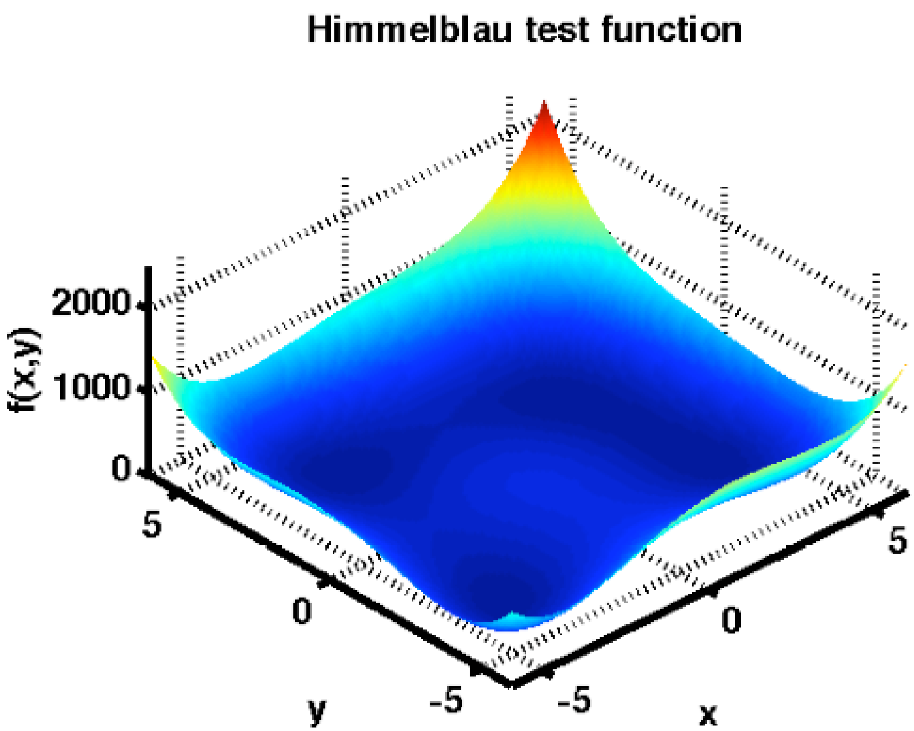
NM computes the simplex, and compares points

- If one is worse (higher) on the cost (objective) function, the simplex reflects that point about the centroid (generalized center) of the simplex and thus makes a new simplex which is hopefully better
- If the points are close in their value the simplex shrinks
- If the points are far away in their value (steep slope) the simplex expands
- See the readings for details
  - **Intro - wikipedia link**
  - **Original 1965 paper**
  - **Convergence properties paper**

Let's look at another function

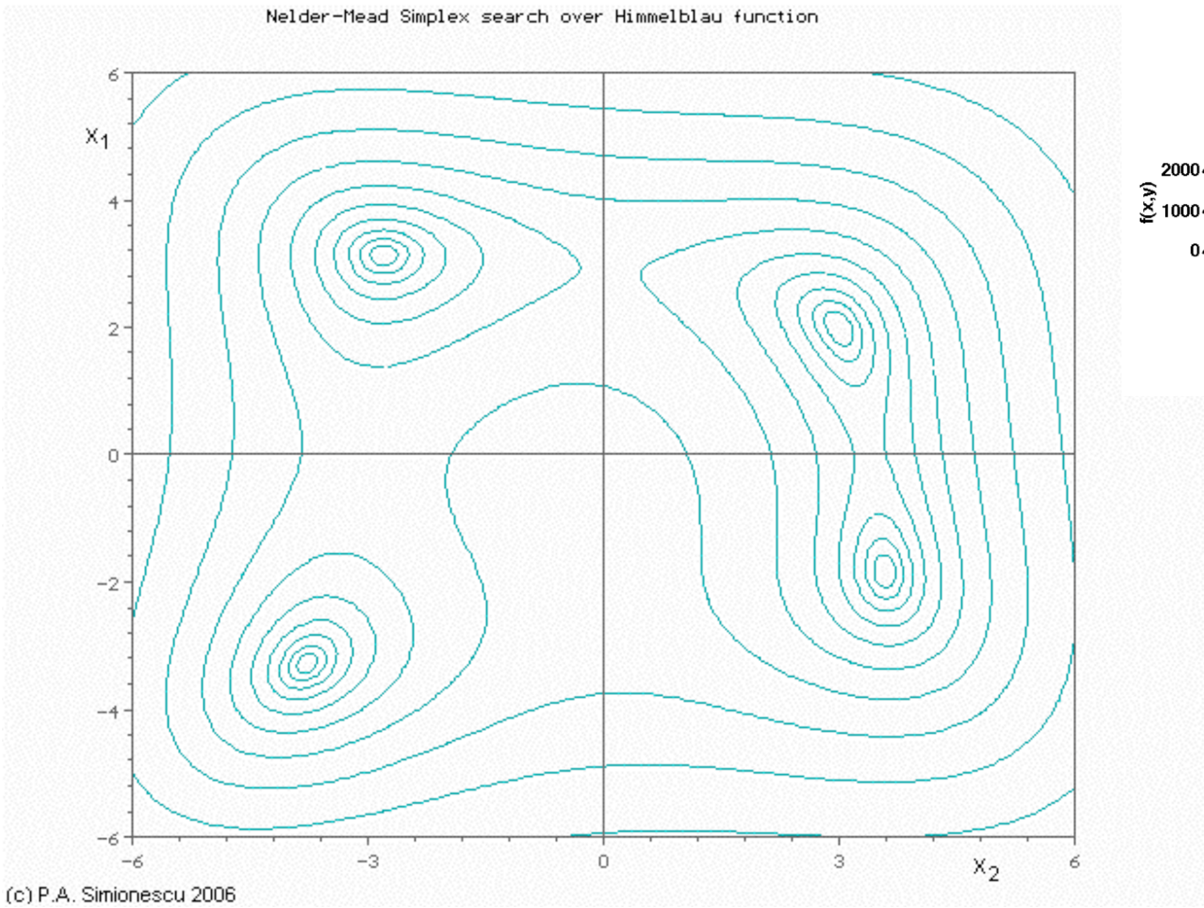
$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

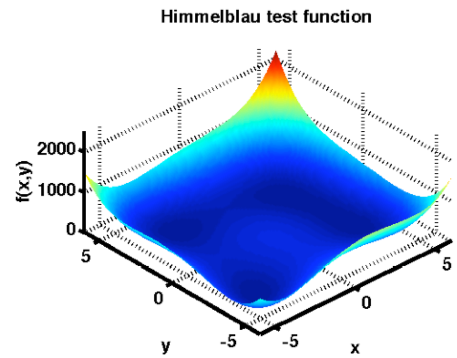
- Himmelblau's function
- Global Minimum
  - f (3,2) = 0
- Local Minima:
  - f (-3.78, -3.28) = 0.0054
  - f (-2.81, 3.13) = 0.0085
  - f (3.85,-1.85) = 0.0011
- In this case, multiple minima exist



How does NM approach this?

- NM finding local minimum of Himmelblau function





A few notes that are important

- Convex functions have one global minimum and no additional local minima
  - **They can still be hard to minimize though - like for the Rosenbrock function**
  - **There exist many techniques which rapidly converge to the solution of convex functions**

A few more notes

- Non-convex functions may have multiple local minima which are not anywhere near the global minimum
  - **For example, the Himmelblau function**
  - **What can we do?**
    - Many strategies - it's hard to know what is the absolute global minimum when you can't explicitly compute it
      - **Can restart with multiple different initial conditions and see if you get the same minima**
      - **Global optimization is a whole branch of mathematics where one attempts to find deterministic algorithms guaranteed to converge to globally optimal solutions in finite time**
- Take home message - use any algorithm with caution and awareness

Why not just compute all the minima of a function over all the space of interest?

- You might not know the function!
  - Think if I told you to find the lowest part of campus blindfolded and with your ears and sense of smell somehow ‘disabled’
  - You’d have to feel your way there, you couldn’t predict the final lowest point, if you had no prior knowledge

What if you know the function?

- It might be that you know the function but it's unreasonable to calculate all the minima
  - **Too computationally expensive!**
    - you'd have to compute the function at n points, and if it's an m-dimensional function (ie we have m parameters to find), m being big and n being big, you would have to compute n^m points
    - e.g. - 10D, 100pts would be 100^10=1e20 computations of the function
    - Comparison - our computers presently are on the order of 10^9 computations per second (GHz), so assuming in one cycle we can compute the function, which isn't true, but for the sake of argument, consider that even this would take 10^11 seconds
      - **This is 3.1710e+03 years!!! Oops:)**
- **There has to be a better way!!! And we can use search to do it in a few computations**

Detailed description of matlab application of *fminsearch()*

<<to matlab!!!>>

One common theme in optimization is trying to find a minimum

- Sometimes we don't need to deal with nonlinearity, and as such can use search methods which are specifically designed/optimized for such problems
- Skiing - you want to get to the bottom of the hill as fast as possible to get the hot chocolate
  - **Obvious approach is to choose the direction of steepest descent down the mountain**
- Leads us to
  - ***Gradient descent* (a.k.a. the *method of steepest descent)***
  - **Do exactly what we just said**

How does gradient descent work (an introduction)?

- Start with the cost function
  - **Make it (hopefully) quadratic so it has the nice bowl shape, and a definite global minimum (though complicated functions may have local minima)**
  - **We want to find a way to make**
    - Mp-k = *something* as *small as possible*
    - *So we'll start at some guess for p, then change p at each step to be going 'down the hill' of the cost function*

The algorithm

- Algorithm:
  Choose a starting point p(0)
  - **Repeat this until we're satisfied that we're close**
    - Compute the distance to change the vector p
    - Compute the direction to change the vector p
    - Update p
  - **Goto repeat**
- It turns out that the steepest direction and step distance is found by looking at the 'gradient' of the cost function

What does the resulting behavior look like?