Introduction to (Mathematical) Optimization

Nick Henderson, AJ Friend
(w/ material from Stephen Boyd and Steven Diamond)
Stanford University

June 28, 2015

Optimization

Optimization finding a best (or good enough) choice among the set of options for a certain objective subject to a set of constraints

Mathematical optimization

Mathematical optimization problem has form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \end{array}$$

- $x \in \mathbf{R}^n$ is **decision variable** (to be found)
- f_0 is objective function; f_i are constraint functions
- problem data are hidden inside $f_0, \ldots, f_m$
- variations: add equality constraints, maximize a utility function, satisfaction (feasibility), optimal trade off, and more

The good news

Everything is an optimization problem

- *choose parameters* in model to fit data (minimize misfit or error on observed data)
- *optimize actions* (minimize cost or maximize profit)
- *allocate resources* over time (minimize cost, power; maximize utility)
- *engineering design* (trade off weight, power, speed, performance, lifetime)

The bad news

In full generality, optimization problems can be quite difficult

- generally NP-hard
- heuristics required, hand-tuning, luck, babysitting

The bad news

In full generality, optimization problems can be quite difficult

- generally NP-hard
- heuristics required, hand-tuning, luck, babysitting

But...

- we can do a lot by restricting to convex models (AJ's talk)
- we have good computational tools
  - modeling languages (CVX, CVXPY, JuMP, AMPL, GAMS) to write problems down
  - solvers (IPOPT, SNOPT, Gurobi, CPLEX, Sedumi, SDPT3, ...) to obtain solutions

Example: The Raptor Problem

See other slides

Optimization in one variable

$$\text{minimize} \quad f(x) \in C^2 : \mathbf{R} \to \mathbf{R}$$

Optimization in one variable

$$\text{minimize} \quad f(x) \in C^2 : \mathbf{R} \to \mathbf{R}$$

- x is a real variable

Optimization in one variable

$$\text{minimize} \quad f(x) \in C^2 : \mathbf{R} \to \mathbf{R}$$

- x is a real variable
- $f(x)$ is the objective function, which returns a single real number

Optimization in one variable

$$\text{minimize} \quad f(x) \in C^2 : \mathbf{R} \to \mathbf{R}$$

- x is a real variable
- $f(x)$ is the objective function, which returns a single real number
- Local optimization: look for a point x^* such that $f(x^*) \leq f(x)$ for all points x near x^*

Optimization in one variable

$$\text{minimize} \quad f(x) \in C^2 : \mathbf{R} \to \mathbf{R}$$

- x is a real variable
- $f(x)$ is the objective function, which returns a single real number
- Local optimization: look for a point x^* such that $f(x^*) \leq f(x)$ for all points x near x^*
- Global optimization: look for a point x^* such that $f(x^*) \leq f(x)$ for all points x in domain of interest

Optimization in one variable

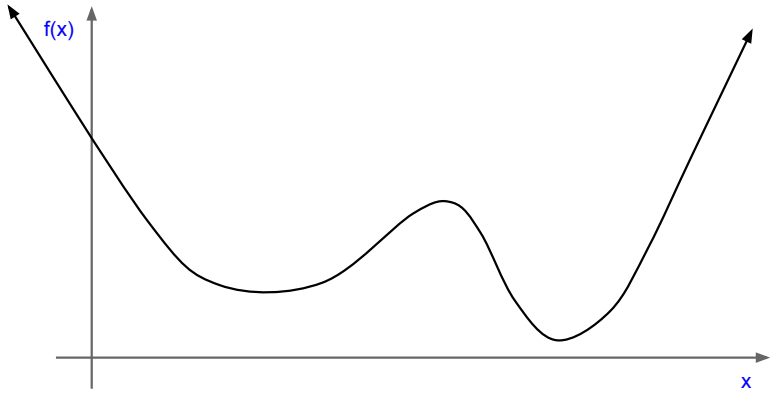
$$\text{minimize} \quad f(x) \in C^2 : \mathbf{R} \to \mathbf{R}$$

- x is a real variable
- $f(x)$ is the objective function, which returns a single real number
- Local optimization: look for a point x^* such that $f(x^*) \leq f(x)$ for all points x near x^*
- Global optimization: look for a point x^* such that $f(x^*) \leq f(x)$ for all points x in domain of interest
- When $f(x)$ is twice continuously differentiable, then local optimization involves finding a point x^* such that $f'(x^*) = 0$ and $f''(x^*) > 0$

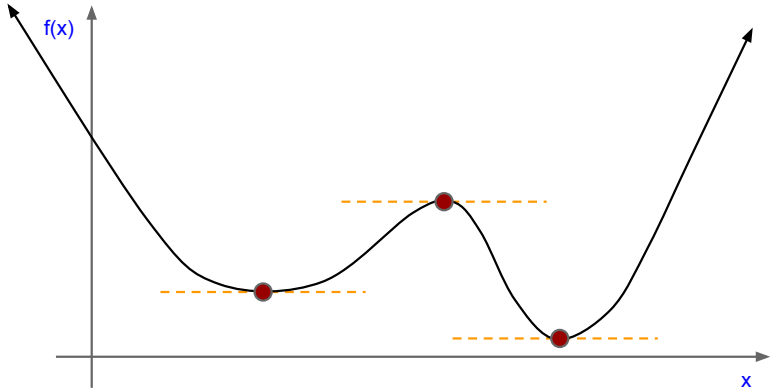
Optimization in one variable: axis

Optimization in one variable: definitions

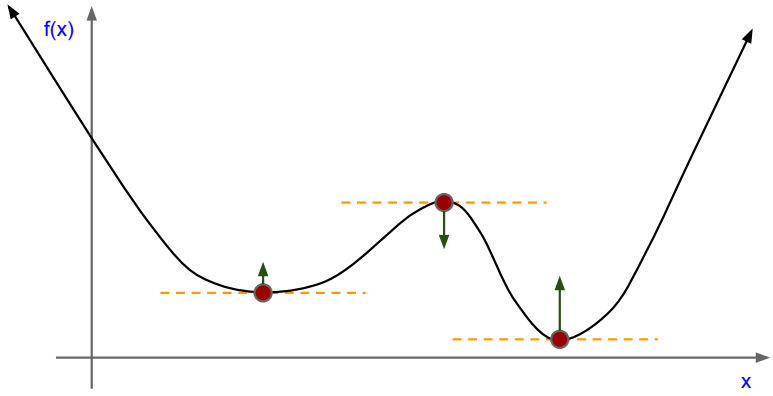
Optimization in one variable: example objective function



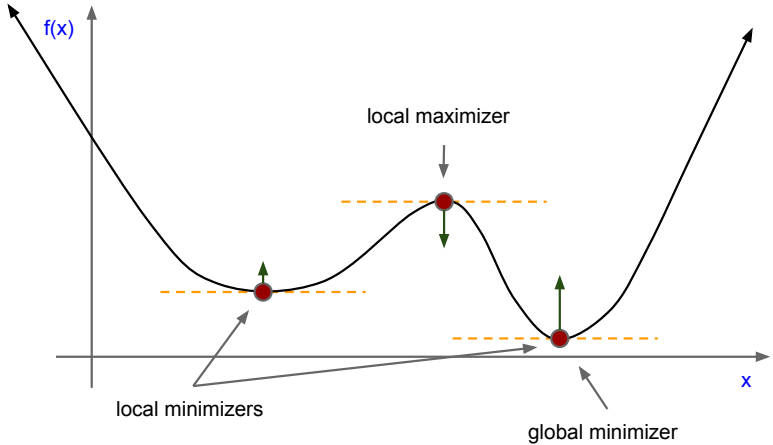
Optimization in one variable: critical points, $f'(x) = 0$



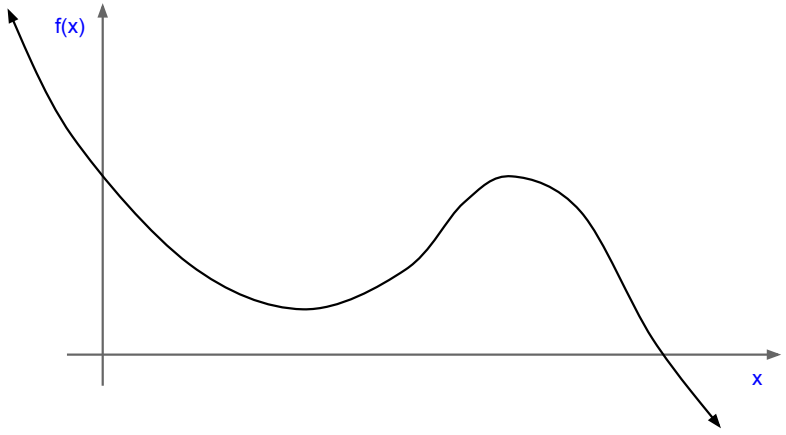
Optimization in one variable: local optima



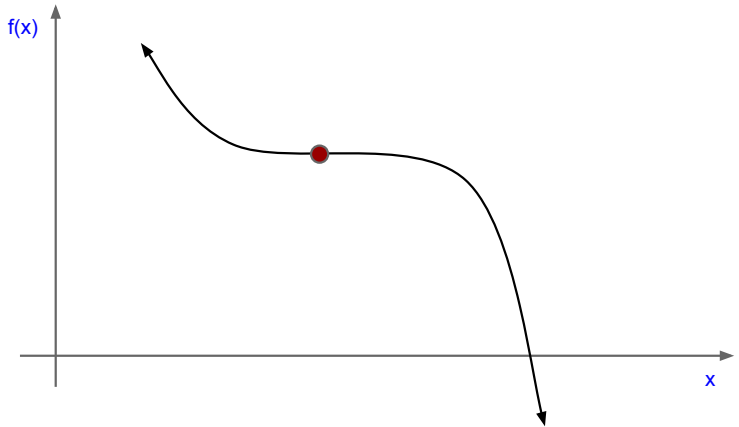
Optimization in one variable: local optima, $f''(x) = ?$



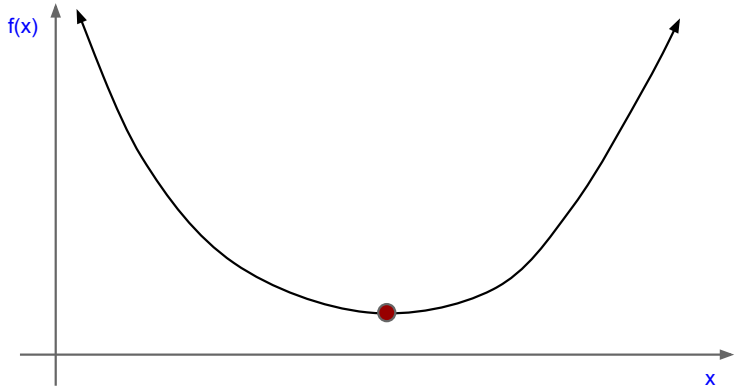
Optimization in one variable: unbounded below



Optimization in one variable: saddle point, $f'(x) = 0$ and $f''(x) = 0$



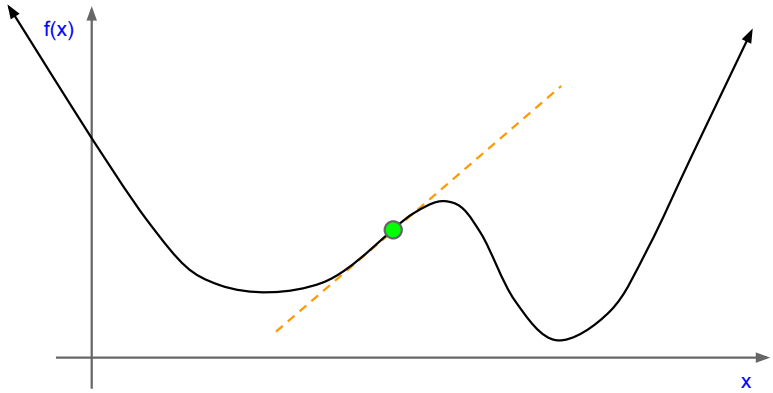
Optimization in one variable: convex objective



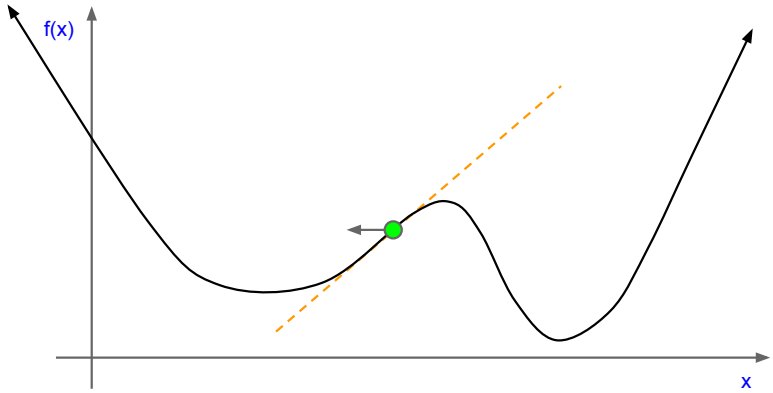
Key definitions

- *domain*: space for input variable x
- *range*: space for output of objective function $f(x)$
- *critical point*: $f'(x) = 0$
- *local minimizer*: $f'(x) = 0$ and $f''(x) > 0$
- *local maximizer*: $f'(x) = 0$ and $f''(x) < 0$
- *saddle point*: $f'(x) = 0$ and $f''(x) = 0$
- *global minimizer*: x^* such that $f(x^*) \leq f(x)$ for all x in domain

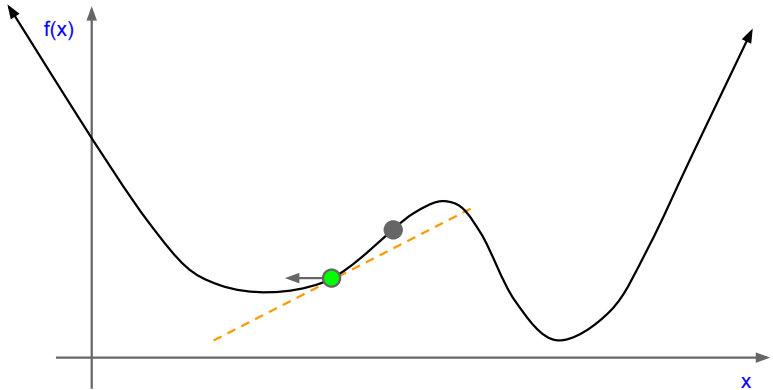
Optimization in one variable: algorithm basics



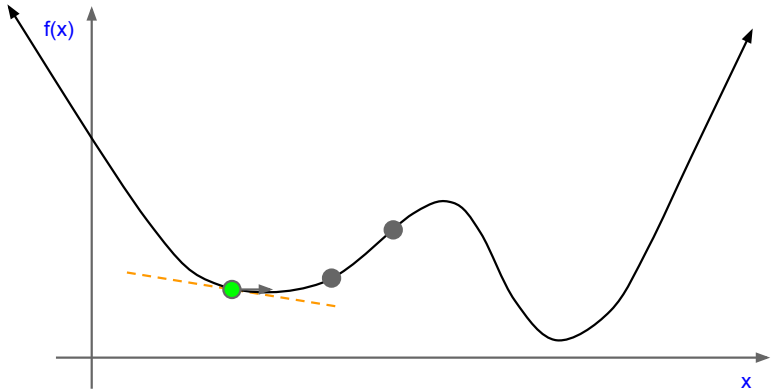
Optimization in one variable: algorithm basics



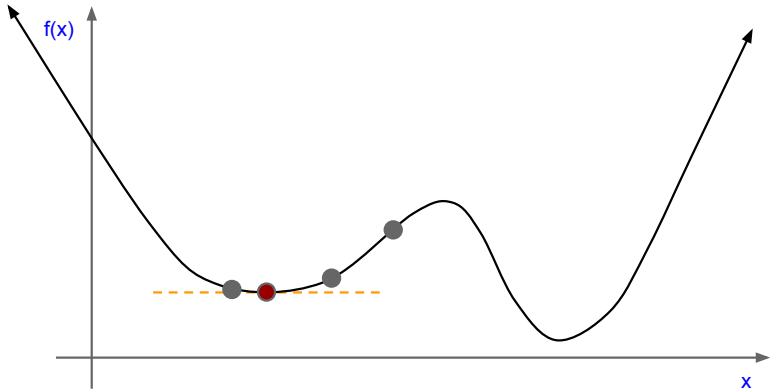
Optimization in one variable: algorithm basics



Optimization in one variable: algorithm basics



Optimization in one variable: algorithm basics



Optimization in one variable: algorithm basics

Optimization in one variable: algorithm basics

- Start with an initial guess x_0

Optimization in one variable: algorithm basics

- Start with an initial guess x_0
- Goal: generate sequence that converges to solution

$$x_0, x_1, x_2, x_3, \cdots \to x^*$$

Optimization in one variable: algorithm basics

- Start with an initial guess x_0
- Goal: generate sequence that converges to solution

$$x_0, x_1, x_2, x_3, \cdots \to x^*$$

- Notation for sequence and convergence: $\{x_k\} \to x^*$

Optimization in one variable: algorithm basics

- Start with an initial guess x_0
- Goal: generate sequence that converges to solution

$$x_0, x_1, x_2, x_3, \cdots \to x^*$$

- Notation for sequence and convergence: $\{x_k\} \to x^*$
- Key algorithm property: ***descent condition***

$$f(x_{k+1}) < f(x_k)$$

Optimization in one variable: algorithm basics

- Start with an initial guess x_0
- Goal: generate sequence that converges to solution

$$x_0, x_1, x_2, x_3, \cdots \to x^*$$

- Notation for sequence and convergence: $\{x_k\} \to x^*$
- Key algorithm property: ***descent condition***

$$f(x_{k+1}) < f(x_k)$$

- Technical algorithm property: ***convergence to solution***

$$|x_{k+1} - x_k| \to 0 \text{ if and only if } f'(x_k) \to 0 \text{ and } \lim_{k\to\infty} f''(x_k) \geq 0$$

Optimization in many variables

$$\text{minimize} \quad f(x) \in C^2 : \mathbf{R}^n \to \mathbf{R}$$

- x is an n-dimensional vector of real variables
- $f(x)$ is the objective function (twice continuously differentiable)
  - First derivative or gradient of f is written $\nabla f(x)$
  - Second derivative or Hessian of f is written $\nabla^2 f(x)$
- We are looking for a point x^* such that $\nabla f(x) = 0$ and $\nabla^2 f(x) \succeq 0$. Note that this is a *local* optimizer
  - $\nabla^2 f(x) \succeq 0$ means that all the eigenvalues of $\nabla^2 f(x)$ are non-negative

The gradient $\nabla f(x)$ in 2 variables

Vector of variables:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Gradient of f:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

The Hessian $\nabla^2 f(x)$ in 2 variables

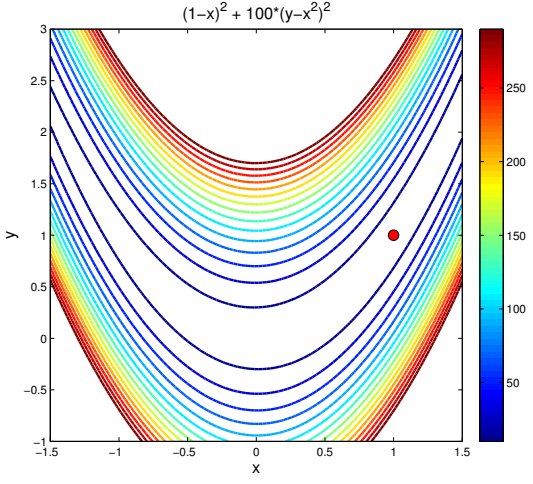
$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

Let's look at an example

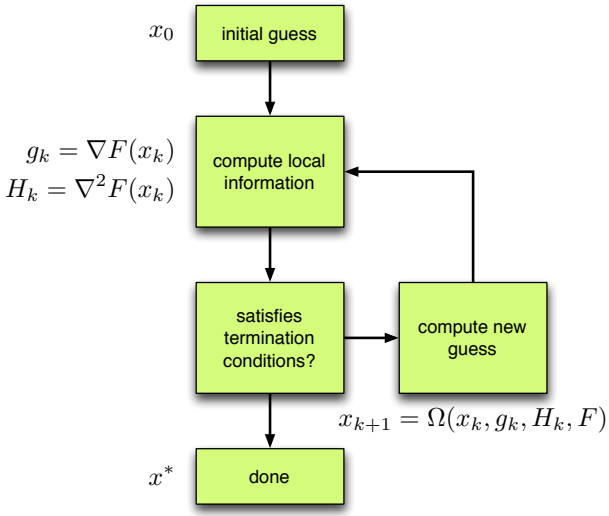
The Rosenbrock function:

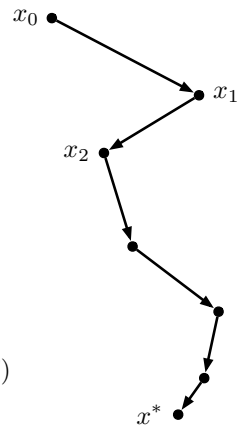
$$f(x, y) = (1 - x)^2 + 100\left(y - x^2\right)^2$$

Rosenbrock contours



Basic optimization algorithm





Line search algorithms

1. compute a search direction p_k
   - for minimization, p_k must be a descent direction, that is $p_k^T g_k < 0$
2. select a step length α_k along p_k such that $f(x_k + \alpha_k p_k) < f(x_k)$
   - (we need more technical requirements here)
3. update the guess $x_{k+1} \leftarrow x_k + \alpha_k p_k$

Example line search algorithms

Algorithm:

$$x_{k+1} \leftarrow x_k + \alpha_k p_k$$

Gradient descent:

$$p_k = -g_k = -\nabla f(x_k)$$

Modified Newton's method:

$$p_k = -(H_k + \lambda_k I)^{-1} g_k = -(\nabla^2 f(x_k) + \lambda_k I)^{-1} \nabla f(x_k)$$

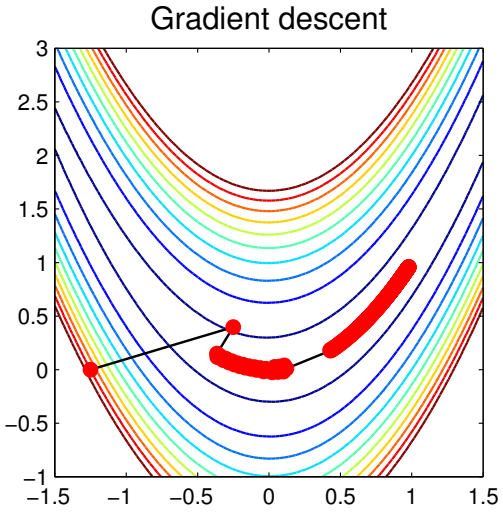
Step length selection: backtracking

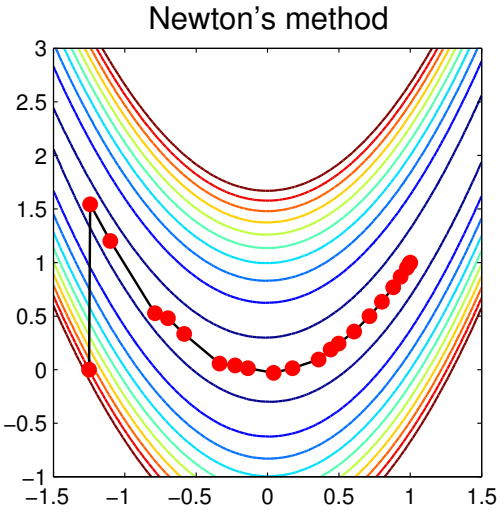
Goal: given p_k find α such that $f(x_k + \alpha p_k) < f(x_k)$.

Procedure: start with initial guess $\alpha > 0$ (use $\alpha = 1$ for Newton's method)

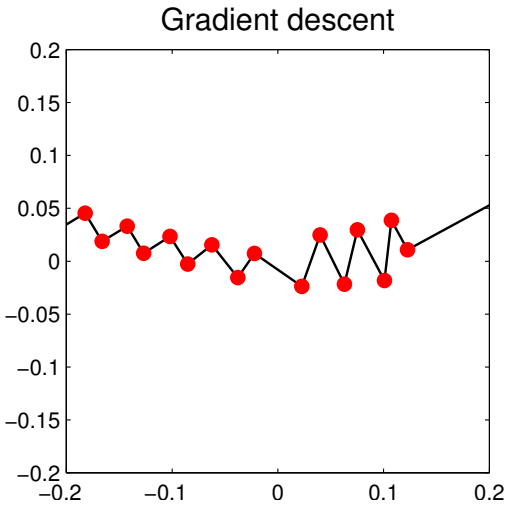
1. if $f(x_k + \alpha p_k) < f(x_k)$, then return α, otherwise continue
2. decrease α by some factor $0 < \delta < 1$: $\alpha \leftarrow \delta\alpha$
3. repeat

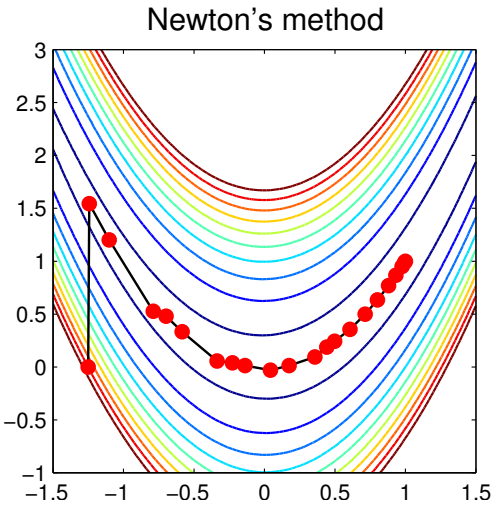
Optimization on Rosenbrock function



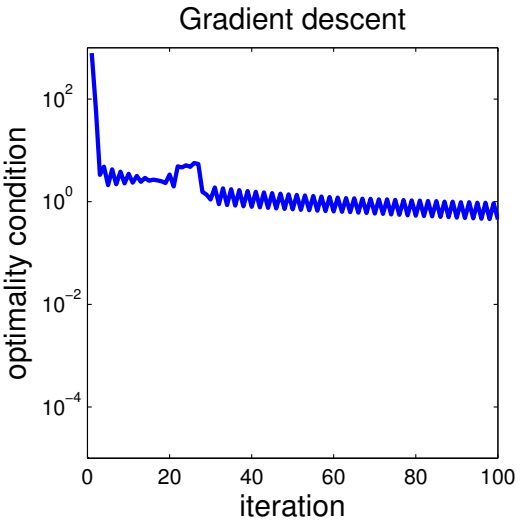


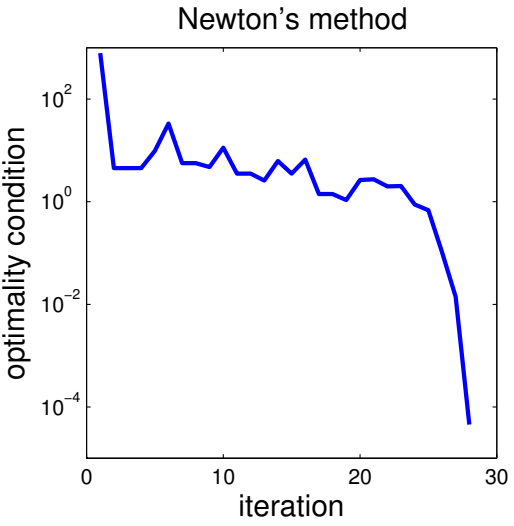
Optimization on Rosenbrock function





Optimization on Rosenbrock function





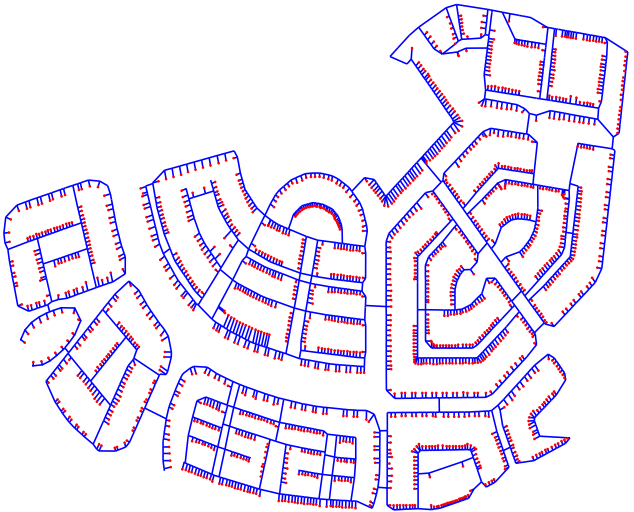
Considerations in selecting optimization algorithms

- Computational cost/scale of objective function
- Computational cost of linear algebra associated with optimization algorithm
- Accuracy requirement in your application

Laying cable: another example

- Housing developer wants to minimize the length of cable needed to connect houses to the internet
- Each house must be connected to a distribution center (DC) through the trench network
- Question: how to choose location of distribution centers and the assignment of houses to minimize cable length?

Laying cable: example development



Laying cable: data

- 1676 houses
- 1904 “junctions”
- 3580 edges
- 28 km of trench

Laying cable: problem

Problem statement

- Given a network of houses in a fully connected trench network, find the location of distribution centers (DCs) and assignment of houses to minimize cable length.

Properties:

- The combinatorial nature of this problem likely puts it in the class of NP-hard problems
- there is no known algorithm to compute an optimal solution in a period of time that does not grow exponentially with the size of the problem

Solution: alternating minimization

- The first part is the assignment of DCs to junction nodes
- The second is the assignment of houses to DCs
- Not guaranteed to find an optimal solution. However, experiments show it is a practical solution

Laying cable: notation and variables

h	number of houses
i	index over houses
c	number of DCs
j	index over DCs
n	number of junctions
k	index over junctions
$A \in \mathbf{R}^{h+n \times h+n}$	adjacency matrix
$D \in \mathbf{R}^{h \times n}$	distance matrix between houses and junctions
$X \in \{0,1\}^{h \times c}$	assignment matrix of houses to DCs
$Y \in \{0,1\}^{c \times n}$	assignment matrix of DCs to junctions
δ	DC capacity

Laying cable: distribution center placement

Given housing assignment matrix X and distance matrix D the optimal DC assignment matrix Y is easily computed. First, compute the matrix of cable lengths for any DC placement:

$$Z = X^T D \in \mathbf{R}^{c \times n}.$$

DC assignment matrix Y is constructed to assign DC j to junction $\text{argmin}_k \; Z_{jk}$ for all j.

This is a simple computation and does not require optimization software.

Laying cable: assignment of houses to DCs

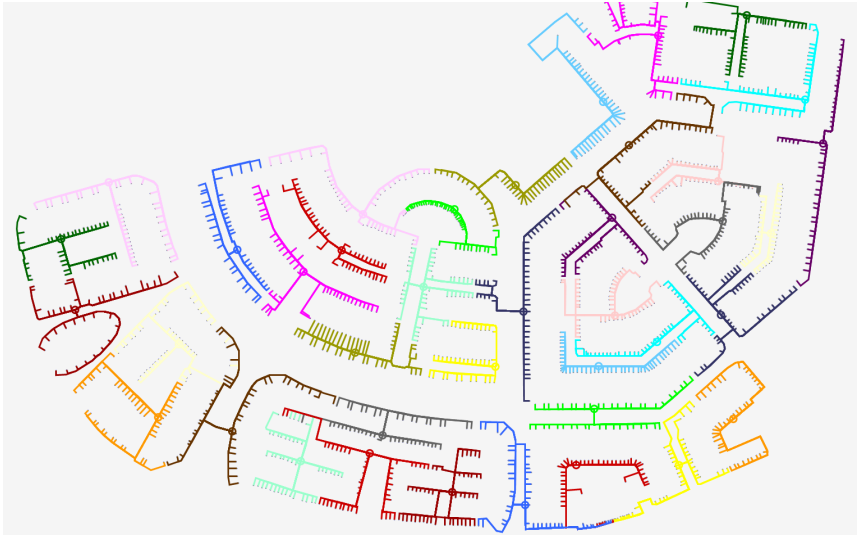
Given DC assignment matrix Y and distance matrix D the optimal housing assignment matrix X is computed using a LP/MIP solver. First, compute the matrix of distances from each house to the fixed DC locations:

$$C = DY^T \in \mathbf{R}^{h \times c}.$$

Second, use LP/MIP software to solve the optimization problem:

$$\begin{array}{ll} \text{minimize} & \text{vec}(C)^T \text{vec}(X) \\ \text{subject to} & X \in \{0,1\}^{h \times c} \\ & Xe_c = e_h \\ & X^T e_h \leq \delta e_c \end{array}$$

Laying cable: solution



Laying cable: results

configuration	length	cost (of cable)
initial	194 km	$110,580
optimized	127 km	$72,390
difference	-67 km	-$38,190

Laying cable: wrapping up

- it is often useful (necessary) to decompose a hard problem into a sequence of easier ones
- good (and useful) solutions are not necessarily globally optimal
- also good to map your problem to existing computational tools

Summary

- Mathematical optimization is an important and useful tool in science, engineering, and industry
- The optimization community has produced a large set of good tools to solve problems
  - there are a mix of open-source and commercial packages
- Art: mapping your problem into a mathematical model that can be attacked using an existing tool