Numeric Optimization

Background

Methods

Reading

- Wilkinson & Allen
  - Ch. 13: Searching and Optimization
- A Survey of Global Optimization Methods
  - a web site at Sandia National Labs:
  - from 1997, but it's a good overview

Terminology

The goal of an optimization problem is to find arguments $x$ of a function $f$ that provide the highest (or lowest) value:

- minimization: $x$ s.t. $f(x) < f(x')$ for all $x \neq x'$
- maximization: $x$ s.t. $f(x) > f(x')$ for all $x \neq x'$

- The function $f$ is the objective function
- For functions of more than one variable $x$ is a vector (a set of arguments)

Terminology (cont’d)

- In global optimization we’re looking for a minimum or maximum over all possible input arguments
- Plot is for scalar $x$ (one-dimensional $x$)
### Combinatorial Optimization
- In some problem areas the function domain is discrete.
- The goal is to find a combination of arguments that minimizes or maximizes the function.
- Example: VLSI circuit layout
  - A chip is built up over several layers.
  - Complex rules for elements in different layers crossing each other:
    - Metal: OK to cross.
    - Poly over diffusion: transistor.
  - Goal: minimum length paths, maximum number of circuits per chip.
  - Constraints: wire thickness, path separation, intersection areas.

### Unconstrained Optimization
- In other domains variables can take on any real values.
- Example: maximum likelihood estimation
  - The "distance" between two sequences is a function of several evolutionary parameters.
    - \( p(A \rightarrow T) \)
    - \( p(A \rightarrow C) \)
  - Optimization problem: what parameter settings correspond to the most likely sequence of evolutionary events?

### Examples (cont’d)
- DNA structure
  - Potential energy is the sum of seven different terms:
    - At right: bond length and bond angle.
    - Goal: find coordinates of atoms that give lowest energy.
  - There are some limitations (e.g., min or max bond lengths) but they usually don’t play a role in the optimization algorithm.

### Examples (cont’d)
- Neural network models
  - A neuron (a node in this graph) has several input signals and one output.
  - The strength of the output \( y \) is a function of the inputs \( s \) and other parameters.
  - Networks are often “trained” to produce outputs in response to given inputs or sequences of inputs.
  - Training is an optimization problem: what parameter settings give the smallest difference between observed and expected outputs?

\[
S_j = \sum_{i=0}^{n_j} w_{i,j} x_j \quad y_j = f(s_j)
\]
**Hill Climbing**
- The simplest optimization algorithm is *hill climbing*
- For minimization in one dimension:
  
  ```
  \begin{align*}
  x &= \text{random} \\
  m &= f(x) \\
  \text{if } f(x + dx) > f(x) & \Rightarrow dx = -dx \\
  \text{while } f(x + dx) < m & \Rightarrow m = f(x + dx) \\
  x &= x + dx
  \end{align*}
  ```

**Hill Climbing (cont’d)**
- There are some obvious problems
  - potential for getting trapped in a local minimum
  - magnitude of $dx$
    - too big: won’t find min
    - too small: inefficient
- Difficult to generalize to multiple dimensions
  - e.g. “saddle points”

**Hill Climbing (cont’d)**
- Hill-climbing can be used as the basis for a *Monte Carlo* optimization algorithm
  - choose several random starting points
  - one or more searches should end up at the global minimum
- Monte Carlo methods were originally defined for numeric integration
- The Monte Carlo optimizer is easily parallelized
  - example of an “embarrassingly parallel” application
  - no communication between processes

**Newton’s Method**
- A better way to choose both the direction and size of the step is to use Newton’s method
- Recall from our discussion of difference methods:
  
  \[ f(x + \Delta x) \approx f(x) + f'(x)\Delta x + f''(x)(\Delta x)^2/2 \]
- Choose step size using the slope of the surface at the current point:
  
  \[ \Delta x = -f'(x)/f''(x) \]

  ```
  \begin{align*}
  xi &= \text{random} \\
  \text{repeat} & \\
  xj &= xi \\
  xi &= xj - f'(xj)/f''(xj) \\
  \text{until } \text{abs}(xi-xj) < \text{epsilon}
  \end{align*}
  ```
Multidimensional Searches

- A method similar to hill climbing can be applied to functions of more than one variable
- The gradient of a function $f$ is
  \[ \nabla f(x) = [f_{x_1}(x), f_{x_2}(x), \ldots, f_{x_n}(x)] \]
  where $f_{x_i}(x)$ is the partial derivative of $f$ with respect to $x_i$

Steepest Descent

- The steepest descent algorithm is a generalization of hill climbing to functions of more than one variable
- The gradient of a function $f$ is
  \[ \nabla f(x) = [f_{x_1}(x), f_{x_2}(x), \ldots, f_{x_n}(x)] \]
  where $f_{x_i}(x)$ is the partial derivative of $f$ with respect to $x_i$
- When the gradient function is evaluated at $x$, it points in the direction of the steepest descent
- Now we know the direction of the step -- but how far do we step?

- Find $a$ that minimizes $f(x + a\nabla f(x))$
  - the gradient defines a direction
  - $a$ is a scalar that specifies a distance in that direction

See Heath (2002) for more on this and other methods
- Newton’s method for multiple dimensions
- Conjugate gradient
Probabilistic Methods

- Newton’s method, steepest descent, and related algorithms all have two drawbacks:
  - the objective function must have a derivative
  - they can be caught in local minima
- The remaining slides introduce two probabilistic algorithms for finding the global minimum
  - use several different starting points
  - explore multiple paths from these points
  - no guarantee of finding the true minimum, but should find a better solution than a single search

Simulated Annealing

- In manufacturing, annealing is a process used to strengthen or remove defects from metals and ceramics
  - heat the metal to about half its melting point
  - slowly allow it to cool
  - slow cooling allows crystals to re-form
- Simulated annealing is a numeric optimization algorithm
  - initially, at high temperatures, search a wide area around the current best guess
  - as the algorithm progresses, lower the temperature: use smaller and smaller step sizes

Simulated Annealing (cont’d)

- Use the temperature $T$ as the mean of a random number generator
  - $x_i = \text{random}$
  - $m = f(x_i)$
  - $T = T_{\text{max}}$
  - while $T > T_{\text{min}}$
    - repeat N times
      - $x_i = \text{random}(T)$
      - if $f(x_i) < m$
        - $x_m = x_i$
      - reduce $T$

Notes on Simulated Annealing

- The temperature reduction schedule is an important parameter
  - a common schedule is logarithmic
  - initially $T$ decreases quickly, then more slowly
- Variations:
  - instead of remembering the best $x$ at each $T$, do a “random walk”
  - take a step with high probability if it’s downhill
  - take uphill steps with $p(T)$
Performance of Simulated Annealing

- In practice annealing does a good job of avoiding local minima
- "Black art" in choosing annealing parameters
  - many domain-dependent heuristics, rules of thumb
- Annealing is often augmented with gradient descent at low temperatures
  - use high temps to avoid local minima
  - steepest descent will be better for the last few steps
- Annealing is easily parallelized
  - multiple starting points (embarassingly parallel)
  - partition N samples at current temp to N/P workers

Genetic Algorithms

- A genetic algorithm (GA) approach to optimization uses population modeling as a metaphor
  - a description of a solution is an individual
    - numeric optimization: x
    - neural net: complete set of weights and other params
    - maximum likelihood: transition matrix, other params
    - maintain a collection of solutions, called a population or generation
- Goal: let the population evolve, generating more fit individuals (better solutions) over time

New Generation

- Initialize the first population with random solutions
- Repeat until a good solution is found:
  - initialize the new generation with the N most fit individuals from the current generation
  - create new individuals to fill out the population
- Key step: new individuals are derived from current generation
- Point mutations make small changes

Crossovers

- Crossovers combine attributes from two individuals
  - the "descendants" have to be complete problem descriptions
  - make sure domain constraints are satisfied (e.g. in ML the sum of transition probabilities is 1.0 for each nucleotide)

Comments on GA

- Crossovers allow the algorithm to combine “subproblem” solutions
  - maybe one individual has an effective combination of parameters
  - this combination may be transferred intact to a new generation
- Honor diversity: GA will be like hill climbing if all individuals zero in on the same target
  - choose a few less fit individuals at each generation
  - maintain variation in the population

Project 5

- If you choose to do the optimization problem for project 5:
  - the distribution will include a simple neural network class
  - invoke methods to set or query network parameters
  - invoke another method to “run” the network on a set of inputs, get back an error value
- Your job: write the hill climbing and simulated annealing code that will find the best parameter combination