Outline

- Cost functions (last class)
- Given a cost function we can calculate
  - The global minimum
  - A local minimum
- Algorithms can be classified according to
  - Derivative information available/not available or expensive
    - Derivatives via finite-differences
    - Linear or nonlinear
    - Local minimum or global minimum
    - Differential or “statistical”
    - Constrained or Unconstrained
- Read Chapter 10-0 of Numerical Recipes.
- Focus will not be on details but educated use of these routines as black-boxes.

Bracketing methods in 1D

- Knowing the function value at 3 points bracket a minimum
- Find a better approximation to the minimum
  - Golden bisection
  - Parabola fitting
  - Methods using derivative information
- 1-D search methods important for multi-dimensional algorithms
- (Read Chapter 10-1 through 10-3 of Numerical Recipes)

Bracketing a minimum in multiple dimensions

- Smallest region bounded by a group of points in
  - 1D is bounded by two points (a line segment)
  - 2D is bounded by three points (a triangle)
  - 3D by four points (a tetrahedron)
  - In ND by N+1 points (a simplex)
- Can find a direction of a decreasing function in
  - 1D by the line from point with higher value to lower
  - 2D by joining point with highest value through point with average value on the opposite side of the triangle
  - And so on for ND
- However cannot guarantee a bracket of a minimum in ND

Downhill Simplex Method (Nelder-Mead)

- Reflection: Project along the direction of decrease with size 1.
- Reflection and expansion: If decrease is large try a step of size 2.
- Contraction: Result of reflection is bad, so try a simple reduction within simplex.
- Multiple contraction: If result of contraction does not give a better result than lowest point.
- Conclude: volume of simplex becomes below tolerance.

Basic calculus

- The direction of maximum increase of a function at a point x is along \( \nabla f(x) \)
- Critical points of a function \( f \) are at \( df/dx = 0 \) or \( \nabla f = 0 \).
  - One way of optimizing is to find \( x \) where \( \nabla f = 0 \)
  - However this can usually be done easily only in one dimension
- Taylor series
  - 1D
  - Multiple dimensions
  - Vector valued function
- Newton’s method for solving \( f(x) = 0 \).
  - Given \( f(x) \neq 0 \) seek a correction, \( h \), to \( x \), so that \( f(x+h) = 0 \)

\[
 f(x+h) = f(x) + h f'(x) + \frac{h^2}{2} f''(x) + O(h^3)
\]
Newton’s Method

- If \( f(x) \) is a scalar valued function of \( n \) variables \( x \)
  \[
  f(x + h) = f(x) + h \frac{d}{dx} f(x) + \frac{h^2}{2} \frac{d^2}{dx^2} f(x) + \cdots
  \]
  - No way to get \( n \) equations from one equation above
  - Use steepest descent methods
- However in optimization problems we are usually solving for the minimum of a scalar valued function of multiple variables \( f(x) \), where \( x \) is an \( n \) dimensional vector
  - We need to solve an equation of the type \( g(x) = \nabla f = 0 \)
  - Same prescription works but now \( \nabla g \) is a matrix called the Jacobian matrix
  - Solve the equation to get corrections and iterate
- However note that we are actually computing Hessian of \( f \)

Gradient Descent

- We have a function \( f \) and an estimate of its gradient \( \nabla f \)
- Decrease \( f \) by a quantity along the direction of \( \nabla f \)
  - Begin initialize \( x, \), tol, \( k = 0 \)
  - do \( k <- k + 1 \)
  - \( x = x - h \cdot \nabla f \)
  - until \( h \cdot \nabla f < \text{tol} \)
  - return \( x \)
- Determining \( h \) is not easy
  - Called “learning rate” in AI
  - Hard to determine \( h \)
    - If \( h \) is too small algorithm will be too slow to converge.
    - If \( h \) is too large the procedure will diverge.
    - Can select it using a line search or using a Newton method.

Selecting step size in Gradient Descent

- Recall \( (x + h) = x + h \cdot \nabla f = 0 \)
- We cannot get \( h \) in general
- However we can minimize along a direction
  - Restrict to the direction of \( \nabla f \). Let \( u \) be a vector in this direction
  - Minimize the one dimensional function of \( t, f(x + t \cdot u) \) by using the one dimensional minimization techniques discussed earlier.
  - Recompute gradient at the new point and repeat the search in the new direction
  - Once \( t \) values become small we have converged
  - Each of the initial searches need not be performed with precision

Powell’s method

- Sometimes it is not possible to estimate the derivative \( \nabla f \)
- To obtain the direction in a steepest descent method
- First guess, minimize along one coordinate axis, then along other and so on. Repeat
- Can be very slow to converge
- Conjugate directions: Directions which are independent of each other so that minimizing along each one does not move away from the minimum in the other directions.
- Powell introduced a method to obtain conjugate directions without computing the derivative.

Function Evaluations

- Often evaluating the function is hard
  - Crash a car to measure a data point
- Analytical expressions for the derivatives are harder, and very much prone to programming error.
  - Analytical derivatives should always be compared with finite difference estimates for accuracy
- Often derivatives are evaluated using finite differences.
  - Recall \( f = f(x + h \cdot \nabla f) \rightarrow 2 \) function evaluations
  - For an \( n \) dimensional function we need at least \( n + 1 \) function evaluations to get the derivative
  - However recall that this is the least accurate
- Promising research area: Use chain rule and semantic parsing of functions to perform automatic differentiation

More complex methods

- Function can be approximated locally near a point \( P \) as
  \[
  f(x) = f(P) + \sum \frac{\partial f}{\partial x_i} (x - P) x_i + \cdots
  \]
  - \( e = x - P \)
  - \( \nabla f \) at \( P \)
  - \( \partial f / \partial x_p \)
  - Gradient of above equation \( \nabla f = A \cdot x - b \)
  - Newton method set gradient equal zero and solve \( A \cdot x = b \)
  - Conjugate directions:
    - Minimize along a direction \( u \). In this case the change in \( \nabla f \) as \( x \) changes by \( \delta x \) is \( A \cdot \delta x \)
    - Minimization in a new direction \( v \) should not modify our previous minimization. Then \( v \) should be chosen so that \( v \cdot A \cdot v = 0 \)
    - Any two directions that satisfy \( v \cdot A \cdot v = 0 \) are called conjugate directions.
Conjugate gradient and quasi-newton

- Use the fact that there is a routine available to calculate \( f \) and the Jacobian \( \nabla f \) to calculate iteratively approximations to the minimum.
  - Conjugate gradients performs minimizations in conjugate directions without constructing \( A \).
  - Quasi Newton methods construct approximations to \( A^{-1} \) iteratively.
- Black boxes, as far as this course is concerned.
- Generally only worth it when we are in the vicinity of a minimum.
- For nonlinear problems they often converge to a local minimum away from the true one.

Levenberg Marquardt

- Return to problem of model fitting by minimizing

\[ \chi^2 = \sum_{i} \frac{(y_i - f(x_i; \mathbf{a}))^2}{\sigma_i^2} \]

- As before set \( \chi^2(\mathbf{a}) \approx \gamma - \mathbf{d} \cdot \mathbf{a} + \frac{1}{2} \mathbf{a} \cdot \mathbf{D} \cdot \mathbf{a} \)

- Observation: steepest descent methods move faster (per function evaluation) far away from the minimum while Newton methods do well near it.
- Idea combine them so that the method adapts according to the location in parameter space.
- Usually for model fitting it is not too difficult to calculate derivatives

\[ \frac{\partial \chi^2}{\partial a_k} = -2 \sum_{i} \frac{y_i - f(x_i; \mathbf{a})}{\sigma_i^2} \frac{\partial f(x_i; \mathbf{a})}{\partial a_k} \quad k = 1, \ldots, M \]

\[ \frac{\partial^2 \chi^2}{\partial a_k \partial a_l} = 2 \sum_{i} \frac{1}{\sigma_i^2} \left( \frac{\partial f(x_i; \mathbf{a})}{\partial a_k} \frac{\partial f(x_i; \mathbf{a})}{\partial a_l} - \frac{y_i - f(x_i; \mathbf{a})}{\sigma_i^2} \frac{\partial^2 f(x_i; \mathbf{a})}{\partial a_k \partial a_l} \right) \]

LM Algorithm

- Compute \( \chi^2(\mathbf{a}) \).
- Pick a modest value for \( \lambda \), say \( \lambda = 0.001 \).
- (1) Solve the linear equations (15.5.14) for \( \mathbf{a} + \delta \mathbf{a} \) and evaluate \( \chi^2(\mathbf{a} + \delta \mathbf{a}) \).
- If \( \chi^2(\mathbf{a} + \delta \mathbf{a}) \geq \chi^2(\mathbf{a}) \), increase \( \lambda \) by a factor of 10 (or any other substantial factor) and go back to (1).
- If \( \chi^2(\mathbf{a} + \delta \mathbf{a}) < \chi^2(\mathbf{a}) \), decrease \( \lambda \) by a factor of 10, update the trial solution \( \mathbf{a} \leftarrow \mathbf{a} + \delta \mathbf{a} \), and go back to (1).
- When the algorithm has converged set \( \lambda = 0 \) and compute the final solution.

Constrained optimization

- We have to optimize \( f(x) \) subject to \( g(x) = 0 \).
  - Makes sense if \( g(x) = 0 \) leaves a few degrees of freedom (N-M).
- Approach 1 (Eliminate constraints)
  - Eliminate variables using constraint equations and solve a reduced problem \( f(x') = 0 \).
  - Not practical, except for simple problems.
- Approach 2 (Penalty function)
  - Construct a new minimization function \( f(x) + Pg(x) \) where \( P >> 1 \).
  - If constraint is violated the minimization function increases rapidly, forcing the optimization routine to solutions where it is not violated.
- Approach 3 (Lagrange Multipliers)
  - Solution has to lie on the surface of \( g(x) = 0 \).
  - Can’t have \( Vf = 0 \) anymore.
  - However we require \( Vf \parallel Vg = 0 \).

Lagrange Multipliers

Optimize \( f(x, y) \) subject to \( g(x, y) = k \):

\( \nabla f(\mathbf{x}, \mathbf{y}) \) is parallel to \( \nabla g(\mathbf{x}, \mathbf{y}) \) and \( g(\mathbf{x}, \mathbf{y}) = k \)

\( \nabla f(\mathbf{x}, \mathbf{y}) = \lambda \nabla g(\mathbf{x}, \mathbf{y}) \) and \( g(\mathbf{x}, \mathbf{y}) = k \)

\( \nabla f(\mathbf{x}, \mathbf{y}) - \lambda \nabla g(\mathbf{x}, \mathbf{y}) = 0 \) and \( g(\mathbf{x}, \mathbf{y}) = k \)
Linear programming

- Black box in this course
- Solve problems with systems of linear equality and inequality constraints

The subject of linear programming, sometimes called linear optimization, concerns itself with linear programming. For a linear program with \( x_1, \ldots, x_n \), maximize the function

\[
 z = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n
\]

subject to the primary constraints

\[
 b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \leq b \quad (b \geq 0) \quad i = 1, \ldots, n_1
\]

and simultaneously subject to \( M = n_1 + n_2 + n_3 \) additional constraints, \( n_1 \) of them of the form

\[
 b_{n_1} x_1 + b_{n_1 + 1} x_2 + \cdots + b_n x_n \leq b \quad (b \geq 0) \quad i = 1, \ldots, n_2
\]

and \( n_3 \) of them of the form

\[
 b_{n_1 + n_2} x_1 + b_{n_1 + n_2 + 1} x_2 + \cdots + b_{n_1 + n_2 + n_3} x_n = b \quad (b \geq 0)
\]