

Medical Image Analysis

CS 593 / 791

Computer Science and Electrical Engineering Dept.
West Virginia University

16th March 2007

Outline

- 1 Minimizing the energy function
- 2 Methods based on Taylor series expansion
- 3 Search methods

Outline

- 1 Minimizing the energy function
- 2 Methods based on Taylor series expansion
 - Gradient Descent
 - Newton's Method
 - Quasi-Newton Methods
- 3 Search methods

Minimizing the energy function

- We wish to find (at least) a local minimum of some function $f(x)$.
- The function may or may not be differentiable.
- The function may be expensive to evaluate.

There is need for a variety of minimization techniques:

- Techniques which make use of analytically computed derivatives.
- Techniques which use approximations to the derivatives.
- Techniques which require a small number of function evaluations.

Outline

- 1 Minimizing the energy function
- 2 **Methods based on Taylor series expansion**
 - Gradient Descent
 - Newton's Method
 - Quasi-Newton Methods
- 3 Search methods

Gradient Descent

- Technique is based on a first-order Taylor series expansion.
- Local linear approximation to f .

$$\begin{aligned}f(x) &= f(x_0) + f'(x_0)(x - x_0) \\f(x) - f(x_0) &= f'(x_0)(x - x_0)\end{aligned}$$

We want to decrease the function f , so $f(x) - f(x_0) < 0$. This implies that

$$f'(x_0)(x - x_0) < 0$$

or, equivalently

$$x - x_0 = -\alpha f'(x_0) \text{ where } \alpha > 0$$

Gradient Descent

In higher dimensions the descent direction is parallel to the gradient.

$$x^{t+1} = x^t - \alpha^t \nabla f(x^t)$$

- Slow convergence
- Stability depends on α

Newton's Method

- Technique is based on a second-order expansion.
- Local quadratic approximation to f .

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2$$

We want to find the minimum of our local quadratic approximation ($f'(x) = 0$). Differentiating the Taylor series expansion gives

$$f'(x) = f'(x_0) + f''(x_0)(x - x_0)$$

Setting $f'(x) = 0$ and solving for x gives

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

Newton's Method

In higher dimensions, the descent direction depends on the gradient and the Hessian, $H(f)$.

$H(f)$ is the matrix of second partial derivatives:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Newton's Method

In higher dimensions, Newton's method becomes

$$x^{t+1} = x^t - H(f(x^t))^{-1} \nabla f(x^t)$$

This converges in one iteration if $f(x)$ is quadratic. Usually we will need to modify the step size to account for $f(x)$ not being quadratic.

$$x^{t+1} = x^t - \alpha H(f(x^t))^{-1} \nabla f(x^t)$$

where α is chosen so that $f(x^{t+1}) < f(x^t)$.

Note that if $H(f)$ is approximated as I , Newton's method becomes equivalent to gradient descent.

Newton's Method

- Faster convergence than gradient descent.
- Requires computation and inversion of Hessian matrix.

Ideally, the gradient and Hessian can be computed analytically, by differentiating $f(\mathbf{x})$.

May also be approximated using finite differences, but this can require $O(n^2)$ evaluations of $f(x)$.

Quasi-Newton Methods

Approximate the Hessian using the gradient information from previous iterations.

Using

$$\nabla f(x^{t+1}) - \nabla f(x^t) \approx H(f(x^{t+1}))(x^{t+1} - x^t)$$

there are a variety of ways of approximating the new Hessian approximation. These are often in the form of an additive update

$$H(f(x^{t+1})) = H(f(x^t)) + C(H(f(x^t)), \nabla f(x^t), x^t)$$

Some techniques even directly approximate $H(f(x^{t+1}))^{-1}$.

These techniques are used in the Matlab function "fminunc".

Levenberg-Marquardt

Levenberg-Marquardt approximates the Hessian as

$$H = J^T J + \mu I$$

- J is the Jacobian matrix (matrix of first derivatives)
- Where μ is a damping parameter
 - ▶ When $J^T J$ is a good approximation make μ small - Newton's method
 - ▶ When $J^T J$ is a bad approximation make μ large - Gradient descent

Outline

- 1 Minimizing the energy function
- 2 Methods based on Taylor series expansion
 - Gradient Descent
 - Newton's Method
 - Quasi-Newton Methods
- 3 Search methods

Search methods - General idea

- Very general techniques.
- Simple to implement.
- Few parameters.
- Useful in hybrid approaches as initialization of a quasi-Newton scheme.
- Can handle discontinuous functions.

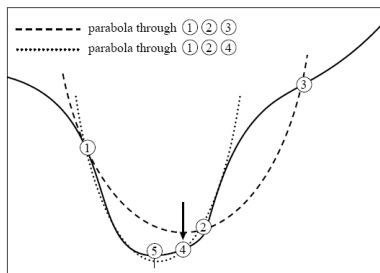
Direct search methods

Hooke and Jeeves (1961):

Direct Search: "sequential examination of trial solutions involving comparison of each trial solution with the "best" obtained up to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be."

Brent's Line Search

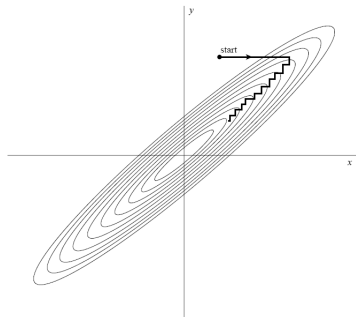
Derivative-free technique for optimizing a 1-dimensional function.



- Evaluate $f(x)$ at 3 points (1), (2), (3), and interpolate with a quadratic.
- Find the minimum of the quadratic (4).
- If (4) is between (1),(2), the next 3 points are (1), (4), (2)
- If (4) is between (2),(3), the next 3 points are (2), (4), (3)

n-dimensional Optimization

Find a sequence of directions to optimize along in 1-D.
One approach: simply use the n coordinate directions.



- Inefficient when valleys of $f(x)$ are not aligned with coordinate axes.

n-dimensional optimization

Powell's method: Try to find an optimal sequence of directions to optimize along in 1-D.

- n linearly independent search directions plus the last step direction.
- Perform the quadratic line search in each search direction.
- Update the direction set.
- Repeat until converged.

n-dimensional optimization

Powell's method:

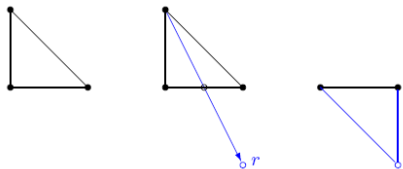
- ① Start position = x_0 , $u_k = e_k$, $k = 1$ to N , $i=0$
- ② $p_0 = x_i$
- ③ For $k = 1$ to N
 $\{p_k = \text{min in direction } u_k \text{ through } p_{k-1}\}$
- ④ For $j = 1$ to $N-1$
 $\{u_j = u_{j+1}\}$
- ⑤ $u_N = p_N - p_0$
- ⑥ $i = i + 1$
- ⑦ $x_i = \text{min in direction } u_N \text{ through point } p_0$.
- ⑧ If not converged go to 2.

Simplex Search

A simplex is a set of $n + 1$ points in R^n .

- 1-D: simplex is a line segment
- 2-D: simplex is a triangle
- 3-D: simplex is a tetrahedron

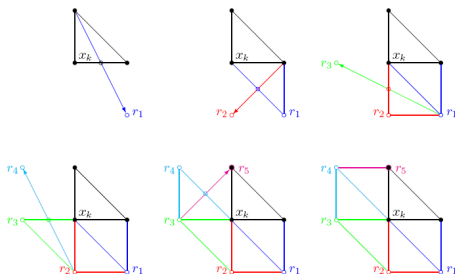
Simplex search methods use the values of the function, $f(x)$ at the vertices of a simplex to drive the search.



Basic operation: reflect the "worst" vertex through the center of the opposite face.

Simplex Search

If the reflected vertex is still the "worst" vertex, choose the "next worst" vertex.

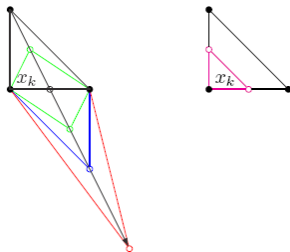


- If we have found the optimum, the simplex may rotate about the "best" vertex.
- When this is detected, assume the optimum value is near the "best vertex".

Simplex Search

Some variants:

- Spendly et. al : Simplex reflection and shrink operation
- Nelder-Mead Simplex Search : Expansion and contraction operations.



Matlab: `fminsearch` uses the Nelder-Mead simplex search.