1 Minimization

1.1 Introduction

Minimization (maximization) is the problem of finding the minimum (maximum) of a given—generally non-linear—real valued function $\phi(\mathbf{x})$ of an n-dimensional argument $\mathbf{x} \doteq \{x_1, \ldots, x_n\}$. The function is often called the *objective function* or the *cost function*.

Minimization is a simpler case of a more general poblem—*optimization*—which includes finding the best available values of the objective function within a given domain and/or subject to given constrains.

Minimization is not unrelated to root-finding: at the minimum all partial derivatives of the objective function vanish,

$$\frac{\partial \phi}{\partial x_i} = 0 \Big|_{i=1,\ldots,n} , \tag{1}$$

and one can alternatively solve this system of (non-linear) equations.

1.2 Local minimization

1.2.1 Newton's methods

Newton's method is based on the quadratic approximation of the objective function $\phi(\mathbf{x})$ in the vicinity of the suspected minimum,

$$\phi(\mathbf{x} + \Delta\mathbf{x}) \approx \phi(\mathbf{x}) + \nabla\phi(\mathbf{x})^\mathsf{T} \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^\mathsf{T} \mathrm{H}(\mathbf{x}) \Delta\mathbf{x} \,, \tag{2}$$

where the vector $\nabla\phi(\mathbf{x})$ is the gradient of the objective function at the point $\mathbf{x}$,

$$\nabla\phi(\mathbf{x}) \doteq \left\{ \frac{\partial \phi(\mathbf{x})}{\partial x_i} \right\}_{i=1,\ldots,n} , \tag{3}$$

and $\mathrm{H}(\mathbf{x})$ is the *Hessian matrix*– a square matrix of second-order partial derivatives of the objective function at the point $\mathbf{x}$,

$$\mathrm{H}(\mathbf{x}) \doteq \left\{ \frac{\partial^2 \phi(\mathbf{x})}{\partial x_i \partial x_j} \right\}_{i,j \in 1,\ldots,n} . \tag{4}$$

The minimum of the quadratic form (2), as function of $\Delta\mathbf{x}$, is found at the point where its gradient with respect to $\Delta\mathbf{x}$ vanishes,

$$\nabla\phi(\mathbf{x}) + \mathrm{H}(\mathbf{x}) \Delta\mathbf{x} = 0 \,. \tag{5}$$

This gives an approximate step towards the minimum, called the *Newton's step*,

$$\Delta\mathbf{x} = -\mathrm{H}(\mathbf{x})^{-1} \nabla\phi(\mathbf{x}) \,. \tag{6}$$

The original Newton's method is simply the iteration,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathrm{H}(\mathbf{x}_k)^{-1}\nabla\phi(\mathbf{x}_k)\,, \tag{7}$$

where at each iteration the full Newton's step is taken and the Hessian matrix is recalculated. In practice, instead of calculating H^{-1} one rather solves the linear equation (5).

Usually the Newton's method is modified to take a smaller step $\mathbf{s}$,

$$\mathbf{s} = \lambda\Delta\mathbf{x}, \tag{8}$$

with $0 < \lambda < 1$. The factor λ can be found by a backtracking algoritm similar to that in the Newton's method for root-finding. One starts with $\lambda = 1$ and than backtracks, $\lambda \leftarrow \lambda/2$, until the *Armijo condition*,

$$\phi(\mathbf{x}+\mathbf{s}) < \phi(\mathbf{x}) + \alpha\mathbf{s}^{\mathsf{T}}\nabla\phi(\mathbf{x})\,, \tag{9}$$

is satisfied (or the minimal λ is reached, in which case the step is taken unconditionally). The parameter α can be chosen as small as 10^{-4}.

1.2.2 Quasi-Newton methods

Quasi-Newton methods are variations of the Newton's method which attempt to avoid recalculation of the Hessian matrix at each iteration, trying instead certain updates based on the analysis of the gradient vectors. The update $\delta\mathrm{H}$ is usually chosen to satisfy the condition

$$\nabla\phi(\mathbf{x}+\mathbf{s}) = \nabla\phi(\mathbf{x}) + (\mathrm{H}+\delta\mathrm{H})\mathbf{s}\,, \tag{10}$$

called *secant equation*, which is the Taylor expansion of the gradient.

The secant equation is under-determined in more than one dimension as it consists of only n equations for the n^2 unknown elements of the update $\delta\mathrm{H}$. Various quasi-Newton methods use different choices for the form of the solution of the secant equation.

In practice one typically uses the inverse Hessian matrix (often—but not always—denoted as B) and applies the updates directly to the inverse matrix thus avoiding the need to solve the linear equation (5) at each iteration.

For the inverse Hessian matrix the secant equation (10) reads

$$(\mathrm{B}+\delta\mathrm{B})\mathbf{y} = \mathbf{s}\,, \tag{11}$$

or, in short,

$$\delta\mathrm{B}\mathbf{y} = \mathbf{u}\,, \tag{12}$$

where $\mathrm{B} \doteq \mathrm{H}^{-1}$, $\mathbf{y} \doteq \nabla\phi(\mathbf{x}+\mathbf{s}) - \nabla\phi(\mathbf{x})$, and $\mathbf{u} \doteq \mathbf{s} - \mathrm{B}\mathbf{y}$.

One usually starts with the identity matrix as the zeroth approximation for the inverse Hessian matrix and then applies the updates.

Table 1: Quasi-newton minimisation algorithm with updates.

```
set the inverse Hessian matrix to unity, B = 1
repeat until converged (e.g.  ‖∇φ‖ < tolerance) :
     calculate the Newton's step Δx = −B∇φ
     do linesearch starting with λ = 1 :
         if φ(x + λΔx) < φ(x) accept the step:
             x = x + λΔx
             update B = B + δB
             break linesearch
         λ = λ/2
         if λ is too small accept the step and reset B:
             x = x + λΔx
             B = 1
             break linesearch
         continue linesearch
```

If the minimal λ is reached during the bactracking line-search—which might be a signal of lost precision in the approximate (inverse) Hessian matrix—it is advisable to reset the current inverse Hessian matrix to identity matrix.

Table 1.2.2 lists one possible algorithm of the quasi-newton method with updates.

Broyden's update The Broyden's update is chosen in the form

$$\delta \mathrm{B} = \mathbf{c}\mathbf{s}^{\mathsf{T}} \,. \tag{13}$$

where the vector $\mathbf{c}$ is found from the condition (12),

$$\mathbf{c} = \frac{\mathbf{u}}{\mathbf{s}^{\mathsf{T}}\mathbf{y}} \,. \tag{14}$$

Sometimes the dot-product $\mathbf{s}^{\mathsf{T}}\mathbf{y}$ becomes very small or even zero which results in serious numerical difficulties. One can avoid this by only performing update if the condition $|\mathbf{s}^{\mathsf{T}}\mathbf{y}| > \epsilon$ is satisfied where ϵ is a small number, say 10^{-6}.

Symmetric Broyden's update The Broyden's update (13) is not symmetric (while the Hessian matrix should be) which is an obvious drawback. Therefore a beter approximation might be the symmetric Broyden's update,

$$\delta \mathrm{B} = \mathbf{a}\mathbf{s}^{\mathsf{T}} + \mathbf{s}\mathbf{a}^{\mathsf{T}} \,. \tag{15}$$

The vector $\mathbf{a}$ is again found from the condition (12),

$$\mathbf{a} = \frac{\mathbf{u} - \gamma\mathbf{s}}{\mathbf{s}^{\mathsf{T}}\mathbf{y}} \,, \tag{16}$$

where $\gamma = (\mathbf{u}^T\mathbf{y})/(2\mathbf{s}^\mathsf{T}\mathbf{y})$.

Again one only performs the update if $|\mathbf{s}^\mathsf{T}\mathbf{y}| > \epsilon$.

SR1 update The symmetric-rank-1 update (SR1) in chosen in the form

$$\delta \mathrm{B} = \mathbf{v}\mathbf{v}^\mathsf{T} \,, \tag{17}$$

where the vector $\mathbf{v}$ is again found from the condition (10), which gives

$$\delta \mathrm{B} = \frac{\mathbf{u}\mathbf{u}^T}{\mathbf{u}^T\mathbf{y}} \,. \tag{18}$$

Again, one only performs the update if denominator is not too small, that is, $|\mathbf{u}^\mathsf{T}\mathbf{y}| > \epsilon$.

Other popular updates The wikipedia article "Quasi-Newton method" list several other popular updates.

1.2.3 Downhill simplex method

The *downhill simplex method* [1] (also called "Nelder-Mead" or "amoeba") is a commonnly used minimization algorithm where the minimum of a function in an n-dimensional space is found by transforming a simplex—a polytope with $n+1$ vertexes—according to the function values at the vertexes, moving it downhill until it converges towards the minimum.

The advantages of the downhill simplex method is its stability and the lack of use of derivatives. However, the convergence is realtively slow as compared to Newton's methods.

In order to introduce the algorithm we need the following definitions:

- Simplex: a figure (polytope) represented by $n+1$ points, called vertexes, $\{\mathbf{p}_1, \ldots, \mathbf{p}_{n+1}\}$ (where each point $\mathbf{p}_k$ is an n-dimensional vector).
- Highest point: the vertex, $\mathbf{p}_{\mathrm{hi}}$, with the highest value of the function: $\phi(\mathbf{p}_{\mathrm{hi}}) = \max_k \phi(\mathbf{p}_k)$.
- Lowest point: the vertex, $\mathbf{p}_{\mathrm{lo}}$, with the lowest value of the function: $\phi(\mathbf{p}_{\mathrm{lo}}) = \min_k \phi(\mathbf{p}_k)$.
- Centroid: the center of gravity of all points, except for the highest: $\mathbf{p}_{\mathrm{ce}} = \frac{1}{n}\sum_{(k\neq \mathrm{hi})} \mathbf{p}_k$

The simplex is moved downhill by a combination of the following elementary operations:

1. Reflection: the highest point is reflected against the centroid, $\mathbf{p}_{\mathrm{hi}} \to \mathbf{p}_{\mathrm{re}} = \mathbf{p}_{\mathrm{ce}} + (\mathbf{p}_{\mathrm{ce}} - \mathbf{p}_{\mathrm{hi}})$.

Table 2: Downhill simplex (Nelder-Mead) algorithm

```
REPEAT :
  find highest, lowest, and centroid points of the simplex
  try reflection
  IF φ(reflected) < φ(lowest) :
    try expansion
    IF φ(expanded) < φ(reflected) :
      accept expansion
    ELSE :
      accept reflection
  ELSE :
    IF φ(reflected) < φ(highest) :
      accept reflection
    ELSE :
      try contraction
      IF φ(contracted) < φ(highest) :
        accept contraction
      ELSE :
        do reduction
UNTIL converged (e.g. size(simplex)<tolerance)
```

2. Expansion: the highest point reflects and then doubles its distance from the centroid, $\mathbf{p}_{\rm hi} \to \mathbf{p}_{\rm ex} = \mathbf{p}_{\rm ce} + 2(\mathbf{p}_{\rm ce} - \mathbf{p}_{\rm hi})$.

3. Contraction: the highest point halves its distance from the centroid, $\mathbf{p}_{\rm hi} \to \mathbf{p}_{\rm co} = \mathbf{p}_{\rm ce} + \frac{1}{2}(\mathbf{p}_{\rm hi} - \mathbf{p}_{\rm ce})$.

4. Reduction: all points, except for the lowest, move towards the lowest points halving the distance. $\mathbf{p}_{k\neq{\rm lo}} \to \frac{1}{2}(\mathbf{p}_k + \mathbf{p}_{\rm lo})$.

Table 2 shows one possible algorithm for the downhill simplex algorithm.

References

[1] J.A.Nelder and R.Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.