Optimisation for Computer Vision

• Many tasks in CV require minimising or maximising a function.
  – When fitting a model to an image, one minimises an error function
    • Fitting lines to data
    • Fitting a pdf to data
    • Finding image features etc
• Suggested Reading:

Maxima and Minima

• Any maximum point of \( f(x) \) is a minimum of \(-f(x)\)
  – Need only consider minimisation
• Global minima
  – Lowest value of \( f(x) \) across all valid \( x \)
• Local minima
  – \( f(x) \) lower than all neighbours,
    – \( f(x) < f(x+d) \) for all small \( |d| \)

Methods

• Local Methods
  – find local minimum near to a starting point
• Global methods
  – find the global minima over a region (very hard)
  – Typically we polish the resulting global minima by applying a local optimiser

Choice of method

• Choice of method will depend on
  – Whether we need local/global solution
  – Whether we can calculate derivatives efficiently
  – Number of dimensions
  – Amount of memory available

Optimisation in 1D: Brackets

• A minimum is \textit{bracketed} by three points, \( a < b < c \) such that \( f(b) < f(a) \) and \( f(b) < f(c) \)
• Minimum must be in range \([a, c]\)
Golden section search

- Find initial bracket \([a,b,c]\)
- Choose new point \(x\) in \([a,c]\)
  - If \(x < b\)
    - if \(f(x) < f(b)\) then new interval \([a,x,b]\) else \([x,b,c]\)
  - If \(x > b\)
    - if \(f(x) < f(b)\) then new interval \([b,x,c]\) else \([a,b,x]\)
  - New interval always smaller
- Repeat until interval sufficiently small

Examples

<table>
<thead>
<tr>
<th>(f(x))</th>
<th>(x)</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f(x))</td>
<td>(x)</td>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
</tr>
<tr>
<td>(f(x))</td>
<td>(x)</td>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
</tr>
</tbody>
</table>

Choice of \(x\)

- Choose \(x\) a fraction \(g\) along largest interval, measured from central point
  - \(g = 0.5(3 - \sqrt{5})\) (Golden mean)
- If initial bracket is \([a,b,c]\) with \((c-a)/(b-a) = g\).
  - after \(n\) iterations bracket width will be \((1-g)^n(c-a)\)

Obtaining initial bracket

- Plot function and estimate position of min.
- General approach:
  - Guess at two points
  - Step in downward direction until function rises again
  - Bracket is the two points found before the function rose, and the one after.

Golden Section search

- Guaranteed to find the local minimum given an initial bracket
- Reliable but not necessarily fast
Parabolic Interpolation
- Near a minimum, most smooth functions will be approximately parabolic
- Given 3 points, fit parabola to estimate minima

Brent’s method
- 1D optimisation method useful when derivatives not available
- Uses a cunning combination of golden section search and parabolic interpolation
- Fast and reliable

Using Derivatives in 1D search
- Derivative information can be used to do interpolation with cubics or higher order polynomials
- More conservative approach recommended:
  - Given bracket \([a,b,c]\),
    - If \(f'(b)<0\) choose \(x\) in \([b,c]\)
    - If \(f'(b)>0\) choose \(x\) in \([a,b]\)
  - Use linear interpolation to select \(x\)

Using 1D derivatives

n-D Optimisation
- Multi-Dimensional optimisation is hard
  - 1-D: Find lowest point on a curve
  - 2-D: Find lowest point on a surface
    - (eg mountain range)
  - 3-D: Find point of lowest density in a block
  - >3D: err...

Example: Circle fitting
- Suppose we have \(n\) points in 2D and wish to find the best circle through the points
- Circle has 3 parameters defining its centre, \((x_c, y_c)\), and radius, \(r\)
- Define a quality of fit function, \(f(x_c, y_c, r)\)
- Best fitting circle is one whose parameters minimise \(f(x_c, y_c, r)\)
Pattern Search methods

- Simple and reliable (but inefficient)
- Need only ever compare function at two points
  - no need for derivatives, or even a smooth function

Pattern Search

- Assume searching \( n \)-D space
- Unit vectors \( \mathbf{u}_i \), \( i = 1 \ldots n \)
- Start at \( x \), with step size \( d \)
- Repeat
  - for \( i = 1 \ldots n \)
    - Search along direction \( \mathbf{u}_i \) with step \( d \) until bracket minima:
      \( f(\mathbf{x} - d \mathbf{u}_i) > f(\mathbf{x}) < f(\mathbf{x} + d \mathbf{u}_i) \)
  - Until no further improvement
  - Reduce step size by half

Downhill Simplex

- Reliable and moderately quick pattern search algorithm
- Can jump over local minima
- Method of choice if speed not an issue and function known to have many local minima
- Stores function evaluation at \( n+1 \) points (hence simplex)

Downhill Simplex

- Stores function evaluation at \( n+1 \) points
- Modifies position of worst point using a variety of predefined moves, eg
  - line search along line perpendicular to plane of other points
  - reflect in plane of other points
  - shrink all points toward CoG

Downhill Simplex in 2D

\( \square \)

Initial Points

Reflection
Reflection + Expansion
Contraction
Multiple Contraction

Direction Set Methods

- To minimise \( f(x) \) starting at \( x_0 \)
- Repeat
  - Select a direction \( \mathbf{u} \)
  - Find \( a \) which minimises \( f(x_0 + a\mathbf{u}) \)
  - 1-D problem, so use a 1D approach (eg Brent)
  - Set \( x \) to minimum point
- Until convergence

\( \square \)
Direction Set Methods

• Trick is to select directions in a suitable way
• Unit directions can work sometimes

Powell’s Method

• Requires no derivatives
• Direction set method
• Chooses conjugate directions
  – Searching along one does not affect optimum position along previous search line
  – Gives fast convergence
• Method of choice for smooth functions where minima approximately known

Including Derivatives

• If we can compute the derivatives of the function efficiently, we can use them to speed up search for minima
• Let $Df(x)$ be vector of partial derivatives
• Best use of derivatives is in Conjugate Gradient methods
  – (see Numerical Recipes for a good overview)

Steepest Descent

• Warning: Not very good!
• Start at $x_0$
• Repeat
  – Select a direction $u = Df(x)$
  – Find $a$ which minimises $f(x_i + au)$
  – Set $x_{i+1}$ to minimum point
• Until convergence

Other Methods

• Dynamic Programming
• Simulated Annealing
• Genetic Algorithms