Physics Coding Club

Optimisation of a Function

Phil Hasnip
phil.hasnip@york.ac.uk

What do we mean by 'optimisation'?

Given a function, $f(x)$:

- What is the 'best' possible value of $f(x)$?
- What inputs x_{opt} give this 'best' output $f(x_{\text{opt}})$?

What do we mean by 'optimisation'?

Given a function, $f(x)$:

- What is the 'best' possible value of $f(x)$?
- What inputs x_{opt} give this 'best' output $f(x_{\mathrm{opt}})$?

What do we mean by 'best'?

'Best?'

- Population of a species
  Find the equilibrium population (stationary point)

‘Best?’

- Population of a species
  Find the equilibrium population (stationary point)
- Financial return from shares
  Show me the money! Find the shares that give maximum profit

‘Best?’

- Population of a species
  Find the equilibrium population (stationary point)
- Financial return from shares
  Show me the money! Find the shares that give maximum profit
- Energy of electrons in a crystal
  Find the lowest energy state

It’s up to us!

‘Best?’

- Population of a species
  Find the equilibrium population (stationary point)
- Financial return from shares
  Show me the money! Find the shares that give maximum profit
- Energy of electrons in a crystal
  Find the lowest energy state

It’s up to us! Usually want one of:

- Minimum
- Maximum
- Specific value, $f(x) = a$
- Stationary point, $f(x) = x$

Optimisation

The function we wish to optimise is called the objective function.

Most optimisation problems can be converted into a minimisation problem, e.g.

Optimisation

The function we wish to optimise is called the objective function.

Most optimisation problems can be converted into a minimisation problem, e.g.

- Minimum: minimise $f(x)$

Optimisation

The function we wish to optimise is called the objective function.

Most optimisation problems can be converted into a minimisation problem, e.g.

- Minimum: minimise $f(x)$
- Maximum: minimise $-f(x)$

Optimisation

The function we wish to optimise is called the objective function.

Most optimisation problems can be converted into a minimisation problem, e.g.

- Minimum: minimise $f(x)$
- Maximum: minimise $-f(x)$
- Specific value, a: minimise $(f(x) - a)^2$

Optimisation

The function we wish to optimise is called the objective function.

Most optimisation problems can be converted into a minimisation problem, e.g.

- Minimum: minimise $f(x)$
- Maximum: minimise $-f(x)$
- Specific value, a: minimise $(f(x) - a)^2$
- Stationary point: minimise $(f(x) - x)^2$

We'll focus on minimisation problems for the rest of the talk.

Is this slide derivative?

There are two distinct classes of problem:

- Derivatives known
- Derivatives unknown

We're going to focus on the first class, where we know the derivatives of f.

Models with one variable

If we have a simple differentiable model of only one variable, then we can use ordinary calculus.

- $f = f\left(x\right)$

Models with one variable

If we have a simple differentiable model of only one variable, then we can use ordinary calculus.

- $f = f(x)$
- Differentiate to get $\frac{df}{dx}$
- Find the stationary points $\frac{df}{dx} = 0$
- Classify stationary points (min, max, inflexion)

What if we can't solve $\frac{df}{dx} = 0$?

Newton's Method

Solve iteratively. At iteration i, Taylor-expand function about the current point $x^{(i)}$:

$$
\begin{aligned}
f(x - x^{(i)}) &\approx f(x^{(i)}) + (x - x^{(i)}) \left.\frac{df}{dx}\right|_{x^{(i)}} + \frac{1}{2}(x - x^{(i)})^2 \left.\frac{d^2 f}{dx^2}\right|_{x^{(i)}} \\
&= f(x^{(i)}) + (x - x^{(i)}) f'(x^{(i)}) + \frac{1}{2}(x - x^{(i)})^2 f''(x^{(i)})
\end{aligned}
$$

Start iteration $i = 0$ with a guess x_0, and update iteratively as:

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

Models with many variables

With N variables, this generalises to:

- $f = f\left(x_1, x_2, x_3, \ldots, x_N\right)$

Models with many variables

With N variables, this generalises to:

- $f = f\left(x_1, x_2, x_3, \ldots, x_N\right)$
- Multivariate calculus: derivative is now a vector $\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_N}\right)$

Models with many variables

With N variables, this generalises to:

- $f = f(x_1, x_2, x_3, \ldots, x_N)$
- Multivariate calculus: derivative is now a vector $\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_N}\right)$
- Find the stationary points $\nabla f = (0, 0, \ldots, 0)$

Models with many variables

With N variables, this generalises to:

- $f = f\left(x_1, x_2, x_3, \ldots, x_N\right)$
- Multivariate calculus: derivative is now a vector $\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_N}\right)$
- Find the stationary points $\nabla f = (0, 0, \ldots, 0)$
- Classify stationary points (min, max, inflexion, saddle point)

Models with many variables

With N variables, this generalises to:

- $f = f\left(x_1, x_2, x_3, \ldots, x_N\right)$
- Multivariate calculus: derivative is now a vector $\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_N}\right)$
- Find the stationary points $\nabla f = (0, 0, \ldots, 0)$
- Classify stationary points (min, max, inflexion, saddle point)

But solving $\nabla f = (0, 0, \ldots, 0)$ can be extremely difficult!

Newton's Method

At iteration i, Taylor-expand function about the current point $x^{(i)}$:

$$f(\mathbf{x}-\mathbf{x}^{(i)}) \approx f(\mathbf{x}^{(i)}) + (\mathbf{x}-\mathbf{x}^{(i)})^{\mathrm{T}}\mathbf{G} + \frac{1}{2}(\mathbf{x}-\mathbf{x}^{(i)})^{\mathrm{T}}\mathrm{B}(\mathbf{x}-\mathbf{x}^{(i)})$$

where $\mathbf{G}$ is the vector of first derivatives and B is the matrix of second derivatives, called the Hessian.

Start iteration $i=0$ with a guess $x^{(0)}$, and update iteratively as:

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - (\mathrm{B}^{(i)})^{-1}\mathbf{G}^{(i)}$$

Steepest Descents

It's common for the Hessian (second derivative matrix) B to be unknown or too expensive to compute and/or invert.

If we approximate the Hessian by a scaled identity matrix, $\frac{1}{\alpha}\mathrm{I}$, we get

$$\begin{aligned} x^{(i+1)} &= x^{(i)} - \mathrm{B}^{-1}\mathbf{G} \\ &\approx x^{(i)} - \alpha\mathbf{G} \end{aligned}$$

What value of α is appropriate? It should be the mean eigenvalue of B^{-1} but we probably don't know that – α is a parameter of the method.

We could *search* for the optimal α...

Steepest Descents

We want to find the minimum of the function $f(x)$. First we pick a starting point x_0 (guess!), then:

Steepest Descents

We want to find the minimum of the function $f(x)$. First we pick a starting point x_0 (guess!), then:

1. Set $x^{(i)} = x_0$
2. Differentiate to get vector $\nabla f(x^{(i)})$

Steepest Descents

We want to find the minimum of the function $f(x)$. First we pick a starting point x_0 (guess!), then:

1. Set $x^{(i)} = x_0$
2. Differentiate to get vector $\nabla f(x^{(i)})$
3. Search direction $\mathbf{d}_i = -\nabla f(x^{(i)})$
4. Move from $x^{(i)}$ along $\mathbf{d}$ to find minimum – line minimisation
   - $\mathbf{x}_{i+1} = \mathbf{x}^{(i)} + \alpha \mathbf{d}_i$
   - Find α_{opt} that minimises $f(\mathbf{x}_{i+1})$

Steepest Descents

We want to find the minimum of the function $f(x)$. First we pick a starting point x_0 (guess!), then:

1. Set $x^{(i)} = x_0$
2. Differentiate to get vector $\nabla f(x^{(i)})$
3. Search direction $\mathbf{d}_i = -\nabla f(x^{(i)})$
4. Move from $x^{(i)}$ along $\mathbf{d}$ to find minimum – line minimisation
    - $\mathbf{x}_{i+1} = \mathbf{x}^{(i)} + \alpha \mathbf{d}_i$
    - Find α_{opt} that minimises $f(\mathbf{x}_{i+1})$
5. Increment i and repeat from step 2

Steepest Descent Example

Preconditioning

The steepest descent method works, but convergence can be extremely slow. We have a single parameter α to approximate the Hessian. It works best when all the eigenvalues of the Hessian are the same, i.e. the function has the same curvature in all directions (spherical contours).

If we can find a better approximation to the Hessian, particularly its eigenvalues, then we can use that instead – called preconditioning.

Preconditioning is equivalent to a coordinate transformation; it is a transformation which takes the Hessian and make its contours 'more spherical'.

If we have an approximate Hessian A then the method is:

$$\begin{aligned} x^{(i+1)} &= x^{(i)} - \mathrm{B}^{-1}\mathbf{G} \\ &\approx x^{(i)} - \alpha\mathrm{A}^{-1}\mathbf{G} \end{aligned}$$

Preconditioned Steepest Descent Example

Quasi-Newton Methods

After one iteration we have a new input $\{x^{(i)}\}$ and a new gradient $\{G^{(i)}\}$. We know

$$\mathbf{G}(\mathbf{x}^{(i+1)}) \approx \mathbf{G}(\mathbf{x}^{(i)}) + \mathrm{B}^{-1}\left(\mathbf{x}^{(i+1)} - \mathbf{x}^{(i+1)}\right)$$

so we can use this to approximate $\mathrm{B} \longrightarrow$ Quasi-Newton method.

For N variables, B is an $N \times N$ matrix so we don't have enough information to determine it fully. There are many different proposals for how to construct an approximate B, the most common are:

- BFGS (and L-BFGS)
- Broyden (class of methods)
- Conjugate gradients

Each of these can be combined with preconditioning.

Preconditioned Conjugate Gradients Example

Constraints

Sometimes our solution might have to obey some constraints, for example:

- Population of a species
  - Find the equilibrium population (stationary point)
  - Population must not be negative

Constraints

Sometimes our solution might have to obey some constraints, for example:

- Population of a species
  - Find the equilibrium population (stationary point)
  - Population must not be negative
- Financial return from shares
  - Find the shares that give maximum profit
  - Amount to invest must not be more than our bank balance

Constraints

Sometimes our solution might have to obey some constraints, for example:

- Population of a species
  - Find the equilibrium population (stationary point)
  - Population must not be negative
- Financial return from shares
  - Find the shares that give maximum profit
  - Amount to invest must not be more than our bank balance
- Energy of electrons in a crystal
  - Find the lowest energy state
  - Number of electrons is constant
  - Must obey Pauli exclusion principle

Usually best to use method of Lagrange multipliers to make a modified objective function.

Typical optimisation scenario

Optimise the set of objective functions

$$f_i\left(x_1, x_2, \ldots, x_N\right)$$

subject to the set of constraints

$$g_j\left(x_1, x_2, \ldots, x_N\right) = 0$$

to find the optimal inputs

$$\mathbf{x} = \left(x_1, x_2, \ldots, x_N\right)$$

Libraries?

- Quasi-Newton methods
  Several decent libraries, e.g. scipy
- Preconditioning
  Usually need to write this yourself

Beyond conventional Quasi-Newton

There are challenges for standard quasi-Newton methods, e.g.

- How to handle noise
  Stabilised quasi-Newton methods (SQNM)
- How to handle higher-order terms
- How to control algorithmic greed
  Greed is the name given to how ‘ambitious’ an algorithm is. A greedy algorithm will make large changes to the inputs.

Increasingly, modern optimisation methods use trust regions.

Roughly speaking, a Trust Region is the neighbourhood of $\mathbf{x}$ where the quadratic expansion is expected to be good.

Special cases

There are two particular special cases worth mentioning:

- f and g are linear in $\mathbf{x} \longrightarrow$ linear programming
- x_i are integers $\longrightarrow$ integer programming

Each of these is a field in its own right!

Multiple minima

If there is more than one minimum, how do we know the one we've found is the lowest?

Multiple minima

If there is more than one minimum, how do we know the one we've found is the lowest?

We don't! Our methods will find the local optimum, i.e. the one closest to our starting guess.

Multiple minima

If there is more than one minimum, how do we know the one we've found is the lowest?

We don't! Our methods will find the local optimum, i.e. the one closest to our starting guess. We'd really like to find the *actual* optimum value, called the global optimum. Many methods, e.g.

- Monte-carlo
- Basin-hopping
- Genetic algorithms

We'll cover some of these at a later date...

Summary

- Form an objective function $f\left(\mathbf{x}\right)$
- Form constraint functions $g_i\left(\mathbf{x}\right) = 0$
- Transform to minimisation problem, e.g.
    - Maximum: $f(x) = -E(\mathbf{x})$
    - Target value, E_0: $f(x) = (E(\mathbf{x}) - E_0)^2$
- Two distinct cases:
    - Derivatives known
    - Derivatives unknown