Overview of slides 05

1. MCMC for Classical Stat. Phys.
   - Metropolis algorithm
   - Alternatives and improvements to Metropolis
   - The heat bath algorithm
   - Other Stat.Phys. systems

2. Minimisation
   - One-dimensional minimisation
   - Downhill Simplex (Nelder-Mead) algorithm
   - Direction set methods in many dimensions
   - Global minimisation, simulated annealing
   - Summary

Metropolis algorithm

Recap: Markov chain Monte Carlo

Generate ergodic Markov chain of configurations with Botzmann distribution $P(X) \propto \exp(-\beta H(X))$, using detailed balance:

$$P(X)T_{X\to Y} = P(Y)T_{Y\to X}$$

Decompose transition probability: $T_{X\to Y} = \omega_{XY} A_{XY}$

- $\omega_{XY} = \omega_{YX} =$ proposal probability, $\sum_Y \omega_{XY} = 1$.
- $A_{XY} \neq A_{YX} =$ acceptance probability

Metropolis algorithm

Ensure symmetric proposal probability: $\omega_{XY} = \omega_{YX}$
Use acceptance probability

$$A_{XY} = \min\left(1, \frac{P(Y)}{P(X)}\right) = \begin{cases} 1 & P(X) \leq P(Y) \\ \frac{P(Y)}{P(X)} & P(X) > P(Y) \end{cases}$$

Alternatives and improvements

Metropolis acceptance probability, $A_{XY} = \min(1,\, P(Y)/P(X))$, ensures detailed balance. But detailed balance can be satisfied by other acceptance probabilities for local updates:

- Glauber algorithm: $\boxed{A_{XY} = \dfrac{1}{1 + e^{\beta \Delta H}}}$ { Exercise! show that detailed balance is satisfied
- Heat-bath algorithm (next)
- Optimization of heat-bath: Overrelaxation

and also for cluster updates:

- Swendsen-Wang algorithm (1987)
- Wolff algorithm (1989)

The heat bath algorithm

Like Metropolis, another local update algorithm

Local value x is updated according to Boltzmann probabilities:

- independent of current value of x, but
- depending on current state of rest of system.

Algorithm

We update only a few degrees of freedom (eg single site) at a time.
Call these x; the surrounding 'environment' (rest of system) is $X \setminus x$
Choose new x randomly with distribution

$$P(x) \propto e^{-\beta H(x|X \setminus x)}$$

The new x does not depend on the old x, only on old $X \setminus x$.

- For Ising, heat bath $\equiv$ Glauber. Show!
- Less efficient than Metropolis for Ising.
  Wins when system has more local degrees of freedom, e.g. Potts.

Proof of detailed balance

We write $X = \{x, X \setminus x\}$; $Y = \{y, X \setminus x\} \equiv \{y, Y \setminus y\}$.
The transition probability is then given by

$$T(X \to Y) = \frac{1}{Z} e^{-\beta H(y|X \setminus x)} = P(y|X \setminus x)\,.$$

Using Bayes' theorem $P(A, B) = P(A|B)P(B)$ we hence have

$$\begin{aligned} P(X)T(X \to Y) &= P(x, X \setminus x)P(y|X \setminus x) \\ &= P(x|X \setminus x)P(X \setminus x)P(y|X \setminus x)\,, \\ P(Y)T(Y \to X) &= P(y, X \setminus x)P(x|X \setminus x) \\ &= P(y|X \setminus x)P(X \setminus x)P(x|X \setminus x)\,, \end{aligned}$$

and we see the two expressions are the same.

Error estimates

Monte Carlo calculations lead to stochastic estimates

Always provide an estimation of the error, if at all possible.

Error of estimate of a mean, using N samples: $\dfrac{\text{st.dev.}}{\sqrt{N}}$

Beyond Ising

Other classical models of magnetism, often simulated by Monte Carlo:

- Potts model: each spin has q degrees of freedom ('states').

$$\beta H_{\text{Potts}} = -J \sum_{\langle ij \rangle} \delta(s_i, s_j) - B \sum_i s_i \qquad \begin{cases} s_i \text{ takes values} \\ \in \{1, 2, 3, \ldots q\} \end{cases}$$

- clock model or vector Potts model: s_i's interpreted as spin vector pointing at different angles:

$$\theta_n = \frac{2\pi n}{q}, \qquad H_{\text{clock}} = J_c \sum_{(i,j)} \cos\left(\theta_{s_i} - \theta_{s_j}\right)$$

- XY model: The spins are 2D vectors of fixed size, hence defined by one angle:

$$H_{XY} = -\sum_{\langle ij \rangle} \vec{S}_i \cdot \vec{S}_j = -\sum_{\langle ij \rangle} \cos(\theta_i - \theta_j)$$

- Heisenberg model: The spins are 3D vectors of fixed size.

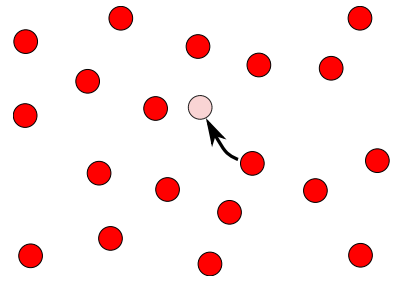
Beyond magnetism

MCMC widely used for thermodynamics, also for non-magnetic systems

- Lennard-Jones fluid: particles, pairwise interacting:

$$\beta H_{\mathrm{LJ}} \propto \left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}$$

For randomly selected particle, move to new position is proposed.
Accepted/rejected via Metropolis rule.



Possible liquid-gas phase transition

- Polymers, e.g., proteins. Monomers, mutually attracting or repelling.

Competing technique: molecular dynamics:

- Solve Newton's laws directly for each particle
- Add artificial 'heat bath' to achieve thermal equilibrium

Quantum statistical physics

Quantum many-body problem (a.k.a. quantum condensed matter physics) is computationally challenging:

Hilbert space dimension grows exponentially

Among available computational methods, there are many different types of quantum Monte Carlo $\rightarrow$

- Variational Monte Carlo
- Diffusion Monte Carlo
- Path integral Monte Carlo
- Green's function Monte Carlo
- Auxiliary-field Monte Carlo
- Determinant Monte Carlo
- World-line Monte Carlo
- .....

Done with Monte Carlo methods

Next chapter: MINIMIZATION / OPTIMIZATION

Minimisation

- Physical problems might involve minimizing or maximizing some quantity (energy, free energy, action etc)
  - Example: variational methods for approximating complex quantum systems. E.g., minimize the multivariable function

$$E_v(\alpha_1, \alpha_2, \ldots, \alpha_N) = \int d\vec{x} \; \frac{\left|\psi(\{\alpha_i\}; \vec{x})\right|^2}{\langle\psi|\psi\rangle} \; \frac{\hat{H}\,\psi(\{\alpha_i\}; \vec{x})}{\psi(\{\alpha_i\}; \vec{x})}$$

- Data fitting = minimising χ^2, "loss function" of model parameters
  - Modern machine learning $\approx$ multi-dimensional minimization
- Other problems may be rewritten as minimisation
  - matrix equation $Ax = b \quad \Leftrightarrow \quad$ minimise $f(x) = \frac{1}{2}x^{\mathrm{T}}Ax - x^{\mathrm{T}}b$

Minimisation

- Many, many, many algorithms
- Textbooks titled ‘Optimization’, ‘Nonlinear programming’, ‘Operations Research’
- These slides: partly based on Numerical Recipes chap 10 (peruse for details)

Constrained vs Unconstrained

General N-dimensional problem

f, g_i, h_i are scalar functions of N variables $\vec{x} = (x_1, x_2, \ldots, x_N)$

Want to $\boxed{\text{minimize } f(\vec{x})}$ $\begin{cases} \text{subject to inequalities } \boxed{g_i(\vec{x}) > 0} \\ \text{and equalities } \boxed{h_i(\vec{x}) = 0} \end{cases}$

- Maximization is the same problem: minimize $-f(\vec{x})$
- Linear programming, quadratic programming, nonlinear programming: when f, g_i, h_i are linear/quadratic/ nonlinear functions
- Example of constraint: for variational wavefunction e^{-ax^2}, may want $a > 0$
- We focus on unconstrained minimization. Constraints unimportant or expected to be automatically satisfied.

One dimension vs many dimensions

Univariate minimization

- Find value of x where $f(x)$ is minimum.
- Use values of $f(x)$ at various x.
- Perhaps also use first and second derivatives, $f'(x)$ and $f''(x)$.

Multivariate minimization

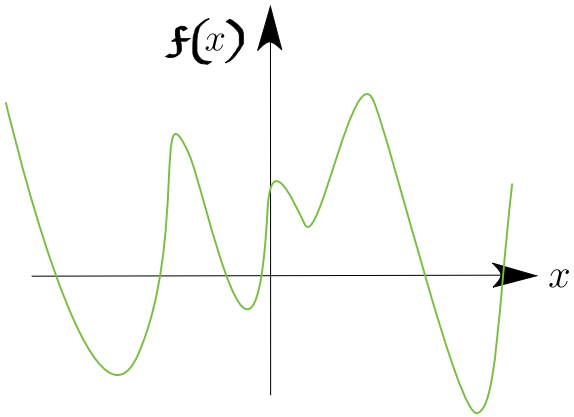
- Find point in multi-dimensional $\vec{x}$-space where $f(\vec{x})$ is minimum.
- Use values of $f(\vec{x})$, possibly also the gradients $\nabla f(\vec{x})$ (1st derivatives) and Hessian matrix $\dfrac{\partial^2 f}{\partial x_i \partial x_j}$ (2nd derivatives).
- Nowadays, millions of dimensions might be common.
  "Curse of dimensionality"

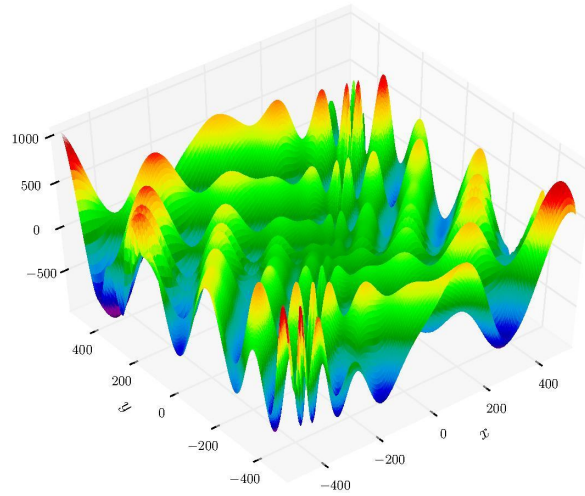
Minimisation

Typical issues

- Want to evaluate f as few times as possible
- Derivatives of f may not be possible/cheap to evaluate

Local versus global





Local minima vs global minimum

- Local minimum: in principle always possible
  - Go downhill till you start going uphill again
- Global minimum: extremely hard
  - May have ragged function with very many local minima
  - How sure can we be that we have found the minimum?

We will mainly discuss local minimization.

Some minimisation algorithms

1 dimension: analogy to root finding

- Equivalent of bracketing a root: Bracketing a minimum requires 3 points $a < b < c$: $f(b) < f(a)$ & $f(b) < f(c)$
- Equivalent of bisection: Golden section [NR 10.1]
- Equivalent of false position: Parabolic interpolation / Brent [NR 10.2]
- Newton iteration for $f'(x)$ instead of $f(x)$

Many dimensions

- Simplex: requires no knowledge of derivatives [NR 10.4]
- Direction set methods: minimise successively in different directions
  - Succession of 1D problems: line search or line minimization)
  - Important issue: Choice of directions
- Gradient descent
- quasi-Newton methods

1-dimensional minimisation

Minimizing single-variable functions

Grid search or uniform search

If function evaluation is cheap, just evaluate the function on a fine grid (e.g., plot the function).

Look for the minimum value.

Done.

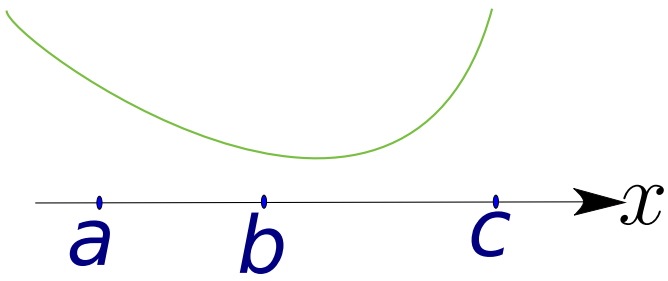
Golden section

Bisection: After bracketing a root, subdivide the interval in halves until we have zoomed in on the root.

Bracketing a minimum involves finding three points.
What is the optimal way of subdividing an interval in three?

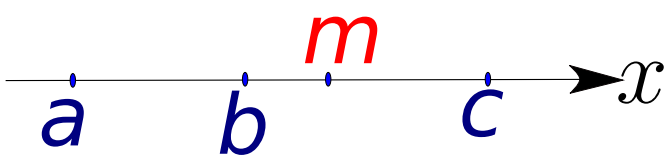
Assume we have three points $a < b < c$ such that $a - b < b - c$ and

$$\frac{b-a}{c-a} = w\,, \qquad \frac{c-b}{c-a} = 1 - w\,.$$



We choose a new point m a fraction z into the (larger) interval $\langle b, c\rangle$,

$$\frac{m-b}{c-b} = z\,.$$

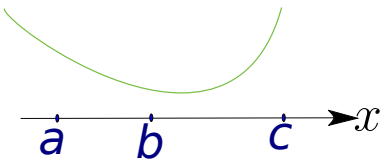


The minimum is now bracketed either by (a, b, m) or by (b, m, c).

Golden section (1-dimensional minimisation)

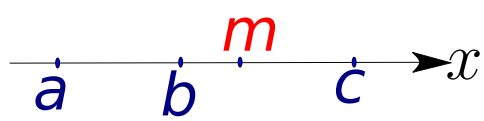
What is the optimal way of subdividing an interval in three?

$$\frac{b-a}{c-a} = w\,, \qquad \frac{c-b}{c-a} = 1-w\,.$$



We choose a new point m a fraction z into the (larger) interval $\langle b, c\rangle$,

$$\frac{m-b}{c-b} = z\,.$$



The minimum is now bracketed either by (a, b, m) or by (b, m, c).

Golden section

Choose $x - a = c - b$ (possible new intervals equal)
and $w = z$ (subdivision ratio unchanged)

$$\Longrightarrow \quad w^2 - 3w + 1 = 0 \quad \Longrightarrow \quad \begin{cases} w = \frac{3-\sqrt{5}}{2} \approx 0.38197 \\ 1-w = \frac{\sqrt{5}-1}{2} \approx 0.61803 \end{cases}$$

Parabolic interpolation + Brent's method

Like bisection, golden section ignores the values of the function, ie how close you are to the minimum.
The simplest way of estimating the position of the minimum is
fit the three points to a parabola, $f(x) = Ax^2 + Bx + C$

$$Aa^2 + Ba + C = y_1 \quad \& \quad Ab^2 + Bb + C = y_2 \quad \& \quad Ac^2 + Bc + C = y_3$$

$$\Longrightarrow \quad x_{\text{min}} = -\frac{B}{2A} = b + \frac{1}{2}\frac{(b-a)^2(y_2-y_3) + (b-c)^2(y_2-y_1)}{(b-a)(y_2-y_3) + (b-c)(y_2-y_1)}$$

This can go wrong:

- We may find a maximum, not a minimum
- It can jump around without converging

Brent's method: switches between golden section and parabolic interpolation as appropriate. [NR 10.2]

Newton iteration

Reminder: Newton's method for root-finding (Newton-Ralphson)

Iterate $\boxed{x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}}$ until converged

Use root of linear approx. $f(x) \approx f\left(x^{(k)}\right) + (x - x^{(k)}) f'(x^{(k)})$ as next iterate

Newton's method for finding extrema

Look for root of $f'(x)$, which hopefully might be a minimum of $f(x)$.

Iterate $\boxed{x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}}$ until converged

Interpretation: (1) use root of linear approximation of slope,
$f'(x) \approx f\left(x^{(k)}\right) + (x - x^{(k)}) f''\left(x^{(k)}\right)$, as next iterate
(2) use minimum of quadratic approximation,
$f(x) \approx f\left(x^{(k)}\right) + (x - x^{(k)}) f'\left(x^{(k)}\right) + \frac{1}{2}(x - x^{(k)})^2 f''\left(x^{(k)}\right)$, as next iterate

Newton iteration

Newton's method for finding extrema

$$\text{Iterate } \boxed{x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}} \text{ until converged}$$

Newton's method for multi-dimensions

$$\text{Iterate } \boxed{\vec{x}_{k+1} = \vec{x}^{(k)} - \left[\mathbb{H}\left(\vec{x}^{(k)}\right)\right]^{-1} \nabla f(\vec{x}^{(k)})} \text{ until converged}$$

Usable when gradient and Hessian are not too expensive, and Hessian can be inverted cheaply.

Use minimum of quadratic approximation,

$$f(\vec{x}) \approx f\left(\vec{x}^{(k)}\right) + (\vec{x} - \vec{x}^{(k)})^{\mathrm{T}} \nabla f\left(\vec{x}^{(k)}\right) + \frac{1}{2}(\vec{x} - \vec{x}^{(k)})^{\mathrm{T}} \mathbb{H}\left(\vec{x}^{(k)}\right)(\vec{x} - \vec{x}^{(k)}),$$

as next iterate

Multi-dimensional minimisation

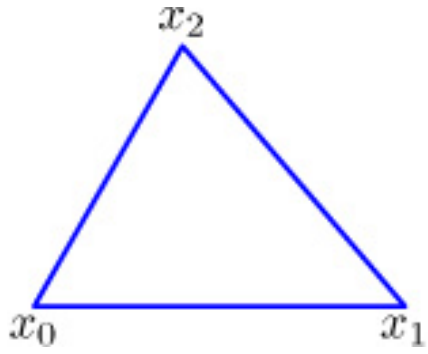
Minimizing a function of many variables

Many dimensions

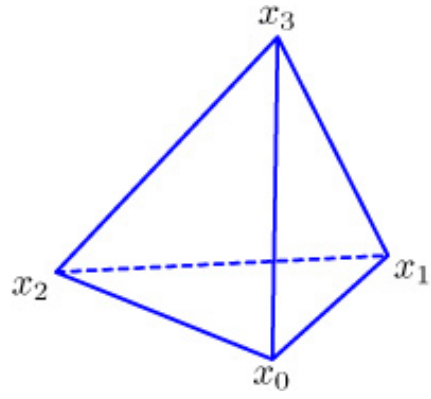
- Simplex: requires no knowledge of derivatives [NR 10.4]
- Direction set methods: minimise successively in different directions
  - Succession of 1D problems: line search or line minimization)
  - Important issue: Choice of directions
- Gradient descent
- quasi-Newton methods

Downhill Simplex (Nelder-Mead) algorithm, a.k.a. amoeba algorithm

For n-variable problem,
choose $n+1$ points and calculate $f(\mathbf{x})$ at all these points.



Simplex in 2D



Simplex in 3D

'Move' the point with largest value of $f(\mathbf{x})$. Repeat. Repeat.

More details: NR 10.4, wikipedia "Nelder-Mead method"

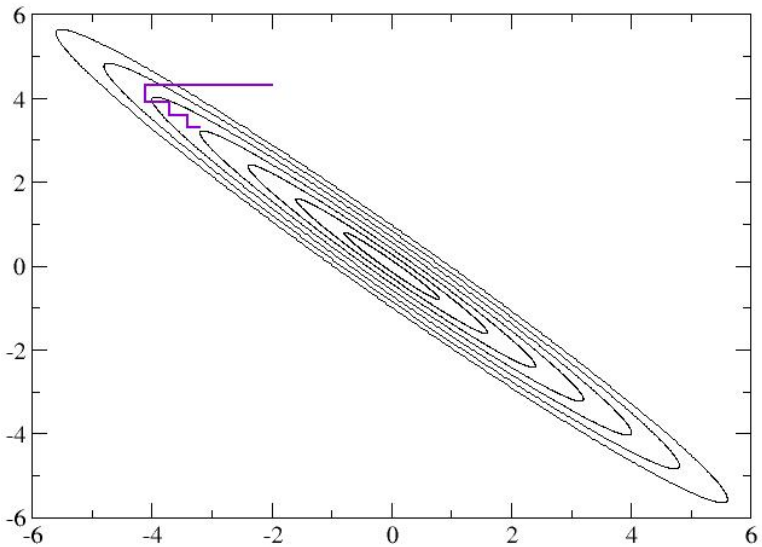
Line minimisation in many dimensions

The most obvious way of minimising $f(\mathbf{x}) = f(x_1, x_2, \ldots)$ is

1. Choose a starting guess $\mathbf{x}^{(0)}$
2. Find the minimum in the x_1-direction, ie, find value of λ that minimizes $f(\mathbf{x}^{(0)} + \lambda \mathbf{e}_1)$. Take $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \mathbf{e}_1$
3. Minimise in x_2-direction: find λ minimizing $f(\mathbf{x}^{(1)} + \lambda \mathbf{e}_2)$; Set $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \lambda \mathbf{e}_2$. Repeat for all directions
4. Repeat steps 2-3 until we are at a minimum for all directions

i.e., minimize successively in directions $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$

This should work eventually, but ineffiecient.
E.g., bad for long, narrow valleys in diagonal directions



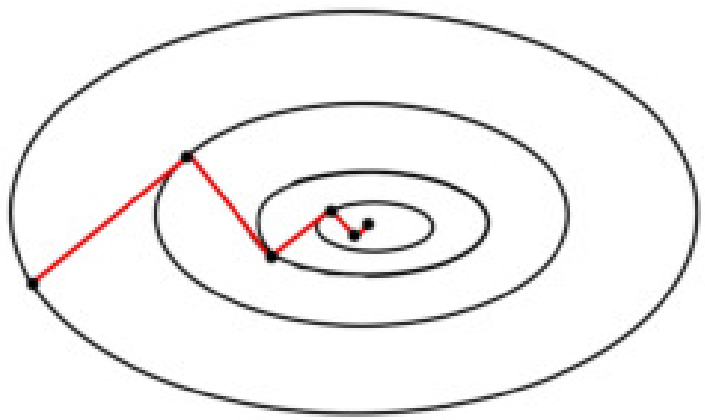
Once in valley, would prefer to head straight to minimum

Steepest descent and Gradient descent

What if we can calculate the gradient, $\nabla f(\mathbf{x})$?
$\nabla f(\mathbf{x})$ is a n-dimensional vector.
$-\nabla f(\mathbf{x})$ is the direction of steepest descent.

Instead of minimizing successively in directions $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$,
Minimize successively in the direction of $-\nabla f(\mathbf{x})$.

1. Choose a starting guess $\mathbf{x}^{(0)}$
2. Find minimum in direction of $-\nabla f(\mathbf{x}^{(0)})$, ie
   find $\lambda = \lambda_m$ minimizing $f(\mathbf{x}^{(0)} - \lambda \nabla f(\mathbf{x}^{(0)}))$.
   Take $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \lambda_m \nabla f(\mathbf{x}^{(0)})$.
3. From $\mathbf{x}^{(1)}$, minimise in direction of $-\nabla f(\mathbf{x}^{(1)})$:
4. Repeat until minimum is reached.



Successive directions perpedicular!
Lines stop & turn when tangential to a contour.

Steepest descent and Gradient descent

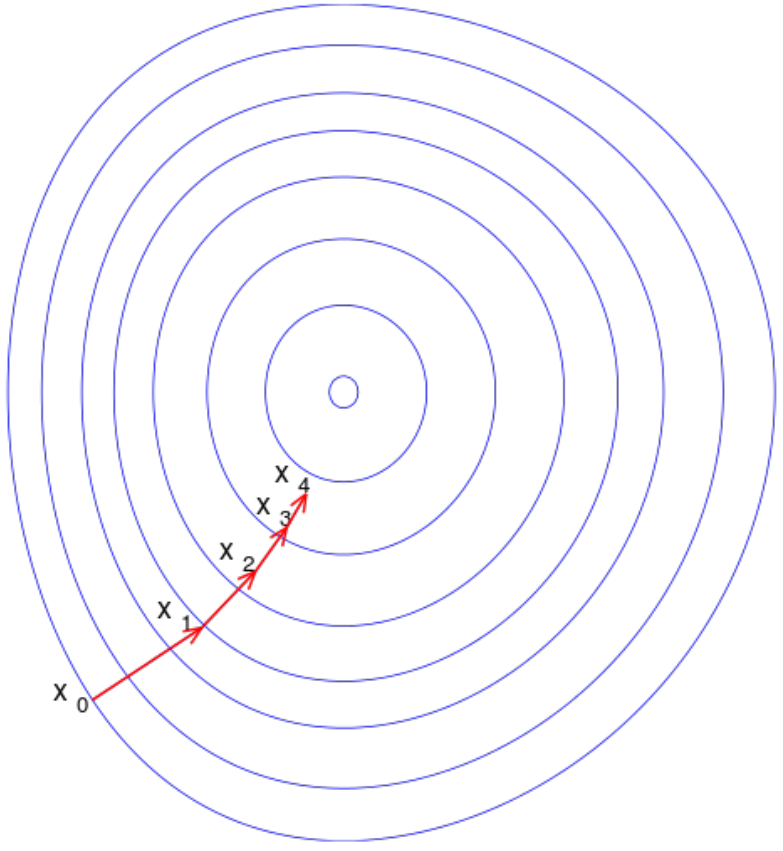
Could be easier to just move a bit downward in the direction of $-\nabla f(\mathbf{x})$, successively.

Saves the effort of minimising along each direction —
At the expense of calculating gradient more often, possibly.

Use if gradient calculation is cheap.

Usually called gradient descent.

Now popular in machine learning, esp. neural networks.



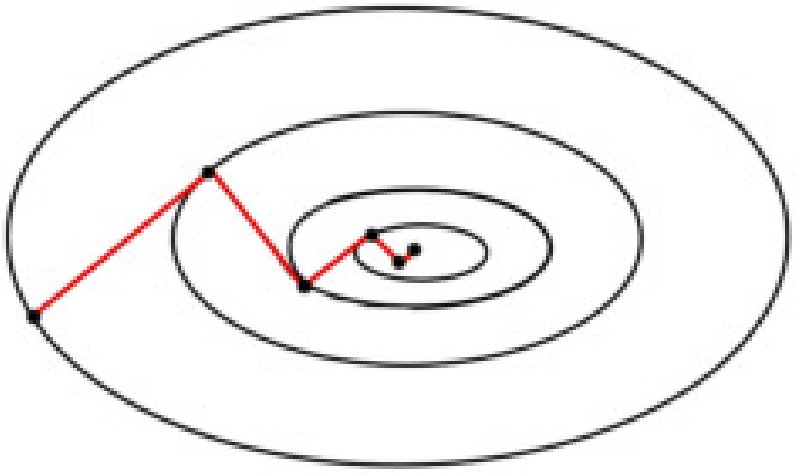
$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \alpha \nabla f(\mathbf{x}^{(i)})$$

$\alpha \longrightarrow$ learning rate

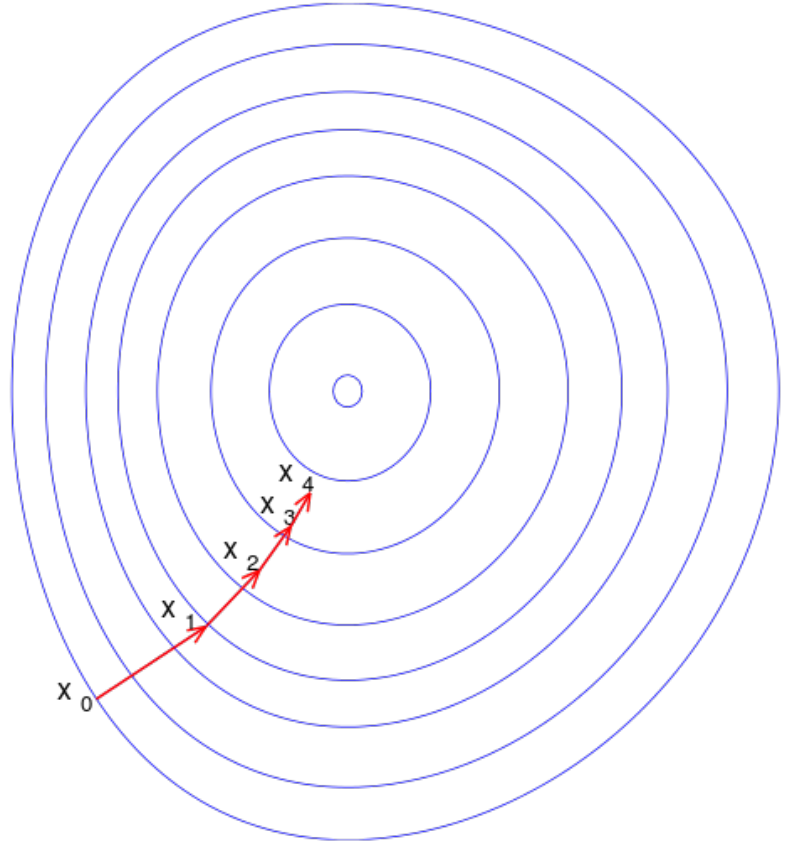
Steepest descent versus Gradient descent

Hard to tell difference from the names. Sometimes used interchangeably.

Most common usage (I think): Steepest descent refers to successive minimizations; gradient descent refers to moving 'a little bit each time'.



Steepest descent



Gradient descent

Conjugate directions + Conjugate Gradient Method

Instead of minimising along perpendicular directions, e.g., $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$, what if we could choose directions that don't spoil previous minimisations?

$\longrightarrow$ Choose conjugate directions, $(\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ relative to matrix A:

$$\boxed{\mathbf{p}_i^{\mathrm{T}} A \mathbf{p}_j = 0 \quad \text{for} \quad i \neq j}$$

- Conjugacy = generalization of orthogonality
- treating A like a metric
- (For $A = \mathbb{I}$, conjugate $\rightarrow$ orthogonal)

$\longrightarrow$ Conjugate gradient method for minimising $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}} A \mathbf{x} - \mathbf{x}^{\mathrm{T}} \mathbf{b}$
i.e., method for solving matrix equation $A\mathbf{x} = \mathbf{b}$.
Reaches minimum in n steps.

Directions: $\mathbf{p}_0 = -\nabla f\left(\mathbf{x}^{(0)}\right)$; after that $\mathbf{p}_{k+1} = -\nabla f\left(\mathbf{x}^{(k+1)}\right) + \beta_k \mathbf{p}_k$.

Imposing conjugacy determines β_k:

$$\beta_k = \frac{\nabla f\left(\mathbf{x}^{(k+1)}\right)^{\mathrm{T}} \cdot A \cdot \mathbf{p}_k}{\mathbf{p}_k^{\mathrm{T}} \cdot A \cdot \mathbf{p}_k}$$

Conjugate Gradient Method for quadratic functions

Conjugate gradient method for quadratic functions

Minimizing $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}} A\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{b}$, or solving $A\mathbf{x} = \mathbf{b}$:

Minimize successively along conjugate directions, $(\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}_k$$

$$\mathbf{p}_0 = \mathbf{r}_0$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$

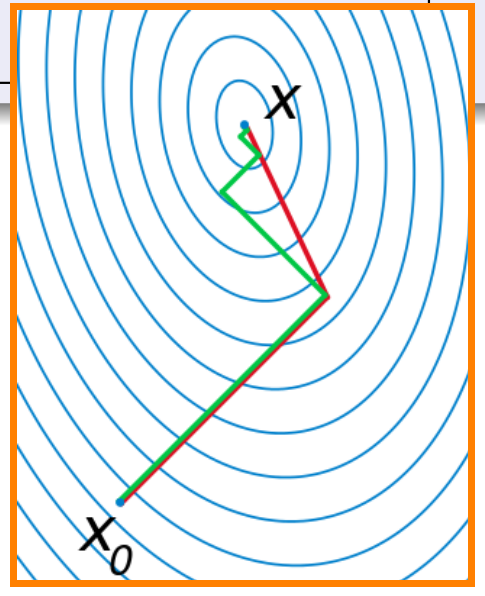
Defined $\mathbf{r}_k = -\nabla f\left(\mathbf{x}^{(k)}\right) = \mathbf{b} - A\mathbf{x}^{(k)}$

$$\beta_k = -\frac{\mathbf{r}_{k+1}^{\mathrm{T}} \cdot A \cdot \mathbf{p}_k}{\mathbf{p}_k^{\mathrm{T}} \cdot A \cdot \mathbf{p}_k}$$

$$\alpha_k = \frac{\mathbf{r}_k^{\mathrm{T}} \cdot \mathbf{r}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}} \cdot A \cdot \mathbf{p}_k}$$

- Reach minimum in n steps

Fig: 2D example →
vs. Steepest Desc



- Don't have to modify A
  All we need to do with A is multiply vectors
  → suitable for using sparse storage

Nonlinear (General) Conjugate Gradient Method

Minimizing arbitrary (not quadratic) function?

Near a minimum, any function is a quadratic form (Taylor expand):

$$f(\mathbf{x}) \approx f(P) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j = c - \mathbf{x}^{\mathrm{T}} \mathbf{b} + \frac{1}{2} \mathbf{x}^{\mathrm{T}} H \mathbf{x}$$

Generally: functions not exactly quadratic; Hessians not exactly known.

Nevertheless, (approximate) conjugate gradient works well.
Hessian is approximated

Various versions of algorithm for general (non-quadratic) case
scipy.optimize.fmin_cg uses version of Polak & Ribiere

Quasi-Newton iteration

Newton's method for multi-dimensions

Iterate $\boxed{\vec{x}_{k+1} = \vec{x}^{(k)} - \left[\mathbb{H}\left(\vec{x}^{(k)}\right)\right]^{-1} \nabla f(\vec{x}^{(k)})}$ until converged

Usable when gradient+Hessian are not too expensive, and inverting Hessian is acceptable

Quasi-Newton

- Estimate Hessian, usually from successive gradients
- Typically $\vec{x}_{k+1} = \vec{x}^{(k)} - A^{(k)} \nabla f(\vec{x}^{(k)})$, with $A^{(k)} \xrightarrow{\text{large } k} \left[\mathbb{H}\left(\vec{x}^{(k)}\right)\right]^{-1}$
- Several variants; some very widely used
  e.g., BFGS, included in scipy.optimize
- Also known as variable metric methods [NR 10.7]

Levenberg-Marquardt: combining Newton iteration & gradient descent

Levenberg-Marquardt

Iterate $\boxed{\vec{x}_{k+1} = \vec{x}^{(k)} - \left[\mathbb{H}\left(\vec{x}^{(k)}\right) + \lambda_k \mathbb{I}\right]^{-1} \nabla f\left(\vec{x}^{(k)}\right)}$

until converged. Adjust λ as appropriate

- Small λ: Newton iteration. Use when close to minimum.
- Large λ: Gradient descent. Use when far from minimum.
- If iteration leads to improvement, probably near quadratic region. Decrease λ.
- If iteration worsens value of f, probably not in quadratic region, best to use more robust method. Increase λ.

Global minimisation

Many physical systems (esp complex or disordered ones) have a large number of 'local' minima (metastable states):

- configurations of large molecules (proteins)
- spin glasses
- neural networks
- integrated circuit design (minimise interference)
- travelling salesman:
  find the shortest round trip for a salesman visiting N towns

Simulated annealing

Analogy

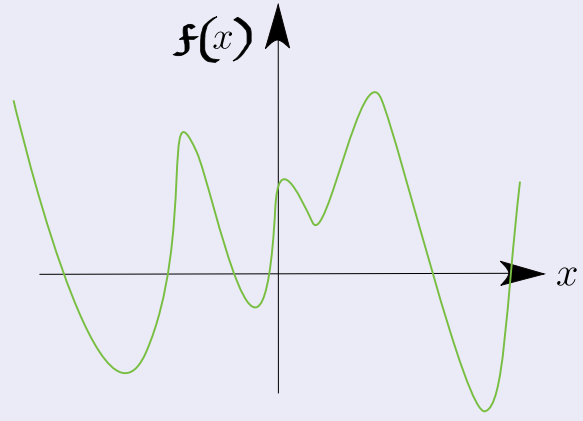
Cooling or crystallisation of liquids (annealing)
They will settle into their ground states if cooled slowly enough

Simulated annealing

Explore X-space with thermal probabilities $P(X) \propto \exp(-f(X)/T)$, using Metropolis. Start with high T, lower it gradually and slowly

We need:

1. A plan for reducing 'temperature' T:
   - how many moves to try at each T
   - how much to lower T each time
2. a stopping criterion



Summary

Last part on Monte Carlo, Stat.Phys.

- Metropolis, heat bath, cluster methods
- Monte Carlo used far beyond educational (Ising) model

Minimisation

- Local minimisation in many dimensions:
  - Direction set methods in many dimensions:
    minimise successively in different directions
  - Gradient descent:
    avoid line search at each step, just move a bit in steepest direction
  - Conjugate gradient:
    choose directions which do not spoil previous minimisations
- Global minimisation: hard (or impossible!)
  - Simulated annealing:
    "cool" the system down slowly till it settles in its ground state