計算機実験II (L4) — 最適化

藤堂眞治
wistaria@phys.s.u-tokyo.ac.jp

2019/12/20

最適化問題

- 目的関数 $f(x)$ の最小値 (あるいは最大値) とその場所を求めたい
- どういう問題を解くのに使えるか？
  - 変分原理が成り立つ問題: 最小作用の原理、最小エネルギーの原理
  - コスト関数が定義できる問題: 最小二乗法 (線形回帰、非線形回帰)、(連立) 方程式の求階、機械学習
- ほとんど全ての問題はコスト関数をうまく定義することで、最適化問題に書き換えることができる
  - (一般に) 最適化問題として解くのは最終手段
  - もっと良い方法があるときはそちらを使う

最適化問題

- 最適化問題の種類
  - 連続最適化問題 ⇐ 難しい
  - 離散最適化 (組み合わせ最適化) 問題 ⇐ もっと難しい
- 真の (大局的な) 最小値 (最大値) を求めるのは難しい
- 一般的には極値を求めることしかできない
- 多次元では極小を囲い込むことができない
- 導関数を使う方法: ニュートン法、準ニュートン法、最急降下法、勾配降下法、共役勾配法 ...
- 使わない方法: 囲い込み法、Nelder-Mead の滑降シンプレックス法、シミュレーテッド・アニーリング ...
- コスト関数・導関数の評価回数と収束までの反復回数のトレードオフ

ニュートン法

- 反復法により方程式 $f(x) = 0$ の解を求める
- 真の解を x_0、現在の解の候補を $x_n = x_0 + \epsilon$ とすると

$$0 = f(x_0) = f(x_0 + \epsilon - \epsilon) = f(x_n) - f'(x_n)\epsilon + O(\epsilon^2)$$

- 次の解の候補 (反復法、逐次近似法)

$$\epsilon \approx \frac{f(x_n)}{f'(x_n)} \qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- 複素変数の複素関数や多変数の場合にも自然に拡張可

多次元の場合

- $f(x) = 0$: d 次元 (非線形) 連立方程式
- x は d 次元のベクトル: $x = (x_1, x_2, \cdots, x_d)$
- $f(x)$ も d 次元のベクトル: $f(x) = (f_1(x), f_2(x), \cdots, f_d(x))$
- 真の解のまわりでの展開 $(x_n = x_0 + \epsilon)$

$$0 = f(x_0) = f(x_0 + \epsilon - \epsilon) = f(x_n) - \frac{\partial f(x_n)}{\partial x} \cdot \epsilon + O(|\epsilon|^2)$$

- ヤコビ行列 $(d \times d)$: $\left(\frac{\partial f(x_n)}{\partial x}\right)_{ij} = \frac{\partial f_i(x_n)}{\partial x_j}$
- 次の解の候補: $x_{n+1} = x_n - \left(\frac{\partial f(x_n)}{\partial x}\right)^{-1} f(x_n)$

ニュートン法による最適化

- x は d 次元のベクトル: $x = (x_1, x_2, \cdots, x_d)$, $g(x)$ はスカラー
- 勾配ベクトル: $(\nabla g(x))_i = \dfrac{\partial g(x)}{\partial x_i}$
- 極小値 (最小値) となる条件: $\nabla g(x) = 0$
- ニュートン法で $f(x)$ を $\nabla g(x)$ で置き換えればよい
- 次の解の候補: $x_{n+1} = x_n - H^{-1}(x_n)\nabla g(x_n)$
- ヘッセ行列 (Hessian): $H_{ij}(x_n) = \dfrac{\partial^2 g}{\partial x_i \partial x_j}(x_n)$

準ニュートン法

- ニュートン法では、ヘッセ行列の計算・保存が必要
- 準ニュートン法: それまでの反復で計算した勾配ベクトルから、ヘッセ行列を近似 (B_n)
- BFGS 法 (Broyden-Fletcher-Goldfarb-Shanno)

$$B_{n+1} = B_n + \frac{y_n y_n^T}{y_n^T s_n} - \frac{B_n s_n (B_n s_n)^T}{s_n^T B_n s_n}$$

- $s_n = x_{n+1} - x_n$、$y_n = \nabla g(x_{n+1}) - \nabla g(x_n)$
- 直接 B_n の逆行列 C_n を更新することも可能

$$C_{n+1} = B_{n+1}^{-1} = C_n + \left(1 + \frac{y_n^T C_n y_n}{y_n^T s_n}\right) \frac{s_n s_n^T}{y_n^T s_n} - \frac{C_n y_n s_n^T + s_n y_n^T C_n^T}{y_n^T s_n}$$

- 他にも、SR1 法、BHHH 法、記憶制限 BFGS 法

囲い込み法 (一次元の最適化)

- $f(a) > f(b) < f(c)$ を満たす 3 点の組 $a < b < c$ の領域を狭めていく
- $[a, b]$、$[b, c]$ の広い方 (例えば後者) を b から見て、黄金比 $[1 : (1 + \sqrt{5})/2 \approx 0.382 : 0.618]$ に内分する点を x とする
  - $f(b) > f(x)$ の場合: $[b, c]$ を新しい領域にとる
  - $f(b) < f(x)$ の場合: $[a, x]$ を新しい領域にとる
- もともとの b が $[a, c]$ を $0.382 : 0.618$ に内分する点だった場合、新しい領域の幅は、どちらの場合も 0.618
- 最初の比率が黄金比からずれていたとしても、黄金比に収束
- 黄金分割法 (golden section) とも呼ばれる

最初の囲い込み

- 1 点を選び、適当な Δx を取る
- 左右に Δx 動かしてみて、関数値が小さくなる方へ動く
- どちらに進んでも関数値が大きくなる場合には、囲い込み完了
- 小さくなった場合、その方向へ再び増えるまで Δx を倍々に増やしながら進む
- 最後の 3 点で極小値を囲い込むことができる
- 囲い込み法のプログラムの例: golden_section.c

極小値をとる x の精度

- 実数の有効桁数を 16 桁 ($\epsilon \approx 10^{-16}$) とする (倍精度)
- 真の極小 (x_0) のまわりでテイラー展開

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

- $f''(x_0)/f(x_0)$ が $O(1)$ だとすると

$$|x - x_0| \sim \sqrt{\epsilon} \sim 10^{-8}$$

  以下になると、第二項の第一項に対する比が ϵ よりも小さくなる

- それ以上領域を狭めても、関数値は変化しない

最急降下法 (steepest descent)

- 関数の微分の情報を使う
- 現在の点 **x** における勾配を計算

$$-\nabla f|_i = -\frac{\partial f}{\partial x_i}$$

- 坂を下る方向にそって、一次元最適化
- 動いた先の勾配の方向でさらに最適化を繰り返す
- 関数値は単調減少 $\Rightarrow$ 極小値に収束

勾配降下法 (gradient descent)

- 勾配方向に一次元最適化を行うかわりに、あらかじめ決めた一定量 (ϵ) だけ坂を下る

$$x_{n+1} = x_n - \epsilon \nabla f$$

- あらかじめ最適な ϵ を知るのは困難
- 機械学習の分野では、(なぜか) $\epsilon = 0.1$ が良いとされている
- この方法を「最急降下法」、一次元最適化を行う勾配法を「最適降下法 (optimum descent)」と呼ぶ場合も

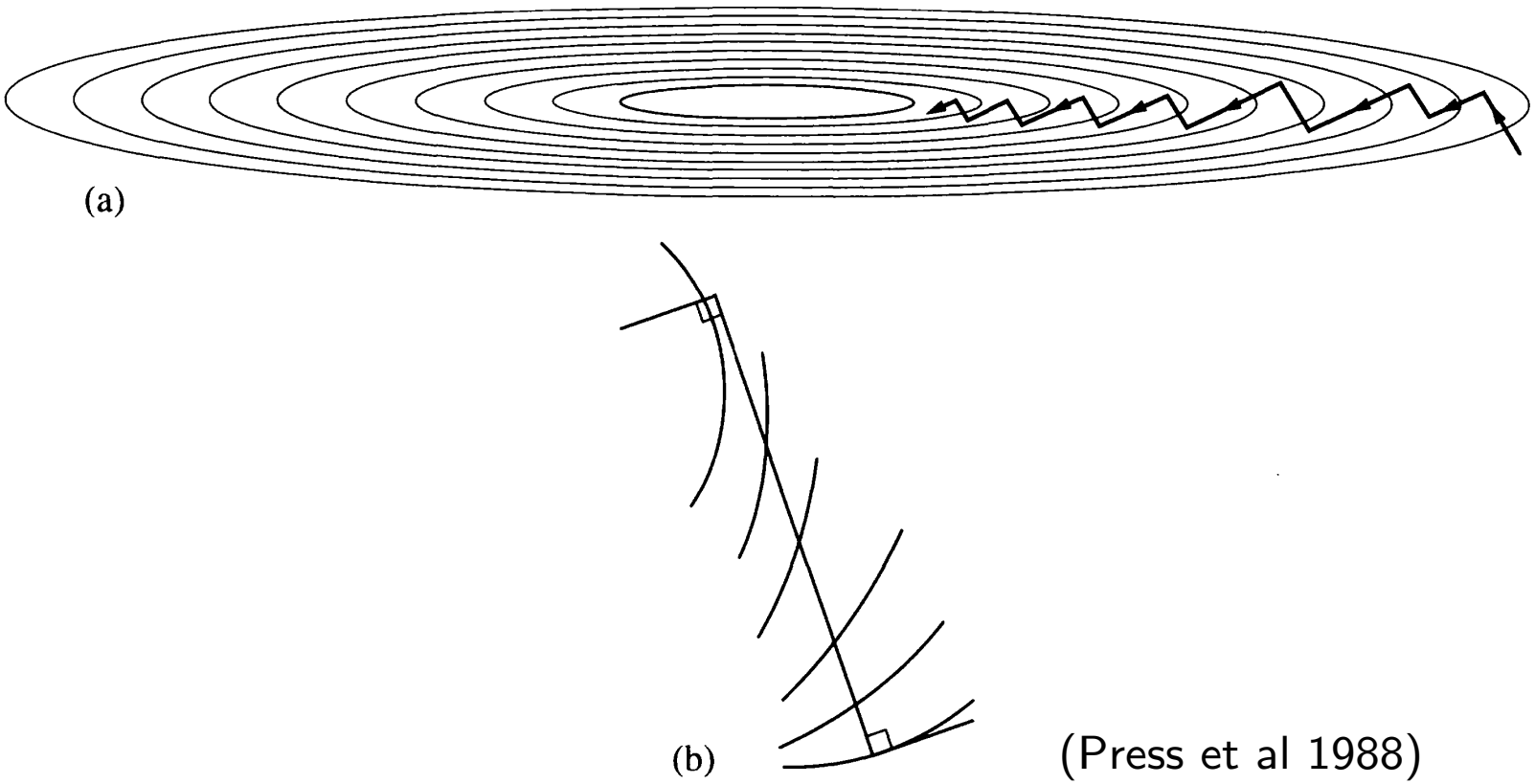
制約条件付きの場合

- 目的関数: $f(x)$
- 制約条件:
  - $g_i(x) = 0 \quad (i = 1, \cdots, m)$ (等式制約条件)
  - $h_j(x) \geq 0 \quad (j = 1, \cdots, n)$ (不等式制約条件)
- 等式制約条件の付いている場合: Lagrange の未定乗数法

$$L(x, \lambda_1, \cdots, \lambda_m) = f(x) + \sum_i \lambda_i g_i(x)$$

  を考え、$x, \lambda_1, \cdots, \lambda_m$ に関する微分が零となる点を探す
- 不等式制約条件の付いている場合: 線形計画法, ペナルティ関数法

細長い谷の場合



(Press et al 1988)

共役勾配法 (conjugate gradient)

- 目的関数がある点のまわりで

$$f(\mathbf{x}) \approx c - \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T A\mathbf{x}$$

  と近似できるとする

- $\mathbf{x}$ における勾配は、連立方程式 $A\mathbf{x} = \mathbf{b}$ の「残差」の形で書ける

$$-\nabla f = \mathbf{b} - A\mathbf{x}$$

- 新しい勾配方向ではなく、それまでとは「共役な方向」に進みたい

「共役な方向」とは

- あるベクトル $\mathbf{p}$ にそった一次元の最適化が完了したとする
  - その点における $\mathbf{p}$ 方向の勾配は零。すなわち $\mathbf{p}^T(\nabla f) = 0$
  - $\mathbf{p}$ 方向の勾配の値を変化させないようにしたい
- 次に、$\mathbf{q}$ にそって、$\mathbf{x} + \epsilon\mathbf{q}$ と移動するとする。その時の勾配の変化は

$$\delta(\nabla f) = A \times (\epsilon\mathbf{q}) \sim A\mathbf{q}$$

  これが $\mathbf{p}$ に垂直であるためには

$$\mathbf{p}^T A\mathbf{q} = 0$$

- この関係が成り立つ時、$\mathbf{p}$ と $\mathbf{q}$ は「互いに共役」という

共役勾配法 (conjugate gradient)

- 初期条件: 位置 $\mathbf{x} = \mathbf{x}_0$

  勾配: $\mathbf{p} = \mathbf{p}_0 = -\nabla f(\mathbf{x}_0) = \mathbf{b} - A\mathbf{x}_0$

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

- 最適化 (第 n ステップ)

$$\alpha_n = \frac{\mathbf{r}_n^T \mathbf{p}_n}{\mathbf{p}_n^T A \mathbf{p}_n}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{p}_n$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n - \alpha_n A \mathbf{p}_n = \mathbf{b} - A\mathbf{x}_{n+1}$$

$$\beta_n = -\frac{\mathbf{p}_n^T A \mathbf{r}_{n+1}}{\mathbf{p}_n^T A \mathbf{p}_n}$$

$$\mathbf{p}_{n+1} = \mathbf{r}_{n+1} + \beta_n \mathbf{p}_n$$

共役勾配法 - 直線上の最適化

- 直線 $\mathbf{x} = \mathbf{x}_n + \alpha \mathbf{p}_n$ 上での最適化
- 点 $\mathbf{x}$ で $f(\mathbf{x})$ が最小値をとるためには、$\mathbf{x}$ における勾配と直線の方向ベクトル $\mathbf{p_n}$ が垂直にならなければならない

$$-[\nabla f(\mathbf{x})]^T \mathbf{p}_n = (\mathbf{b} - A\mathbf{x})^T \mathbf{p}_n = \mathbf{r}^T \mathbf{p}_n = (\mathbf{b} - A(\mathbf{x}_n + \alpha \mathbf{p}_n))^T \mathbf{p}_n$$
$$= (\mathbf{r}_n - \alpha A \mathbf{p}_n)^T \mathbf{p}_n = 0$$

$$(\mathbf{r}_n \equiv \mathbf{b} - A\mathbf{x}_n)$$

- 最適解: $\alpha = \dfrac{\mathbf{r}_n^T \mathbf{p}_n}{\mathbf{p}_n^T A \mathbf{p}_n}$

共役勾配法 - 共役な方向の決め方

- 新たな最適化方向: $\mathbf{p}_{n+1} = \mathbf{r}_{n+1} + \beta_n \mathbf{p}_n$
- $\beta_n = 0$ とすると最急降下法と等価
- 共役勾配法では、$\mathbf{p}_{n+1}$ と $\mathbf{p}_n$ が共役となるように $\beta_n \mathbf{p}_n$ だけ補正を加える

$$\mathbf{p}_n^T A \mathbf{p}_{n+1} = \mathbf{p}_n^T A(\mathbf{r}_{n+1} + \beta_n \mathbf{p}_n) = 0$$

- β_n について解くと

$$\beta_n = -\frac{\mathbf{p}_n^T A \mathbf{r}_{n+1}}{\mathbf{p}_n^T A \mathbf{p}_n}$$

共役勾配法 - 共役性と直交性

- 共役性: $\{\mathbf{p}_i\}$ は自動的に互いに共役となる

$$\mathbf{p}_i^T A \mathbf{p}_j = 0 \qquad (i < j \leq n)$$

- $\mathbf{x}_n$ を通り $\mathbf{p}_0 \cdots \mathbf{p}_n$ に並行な「平面」上で $f(\mathbf{x}_{n+1})$ は最小

$$\mathbf{p}_i^T \mathbf{r}_{n+1} = 0 \qquad (i \leq n)$$

- 直交性: $\{\mathbf{r}_i\}$ は自動的に互いに直交する

$$\mathbf{r}_i^T \mathbf{r}_j = 0 \qquad (i < j \leq n+1)$$

- N 回反復すると残差は零 (実際は数値誤差により直交性がくずれる)

最適化問題として連立一次方程式の解を求める

- 行列 A を正定値対称行列とする
- 連立方程式 $A\mathbf{x} = \mathbf{b}$ の解を $\hat{\mathbf{x}}$ とすると、目的関数

$$f(\mathbf{x}) = \frac{1}{2}(\hat{\mathbf{x}} - \mathbf{x})^T A(\hat{\mathbf{x}} - \mathbf{x})$$

  は $\mathbf{x} = \hat{\mathbf{x}}$ の時、最小値 0 をとる
- $\mathbf{x}$ における目的関数の勾配は、連立方程式の「残差」の形で書ける

$$-\nabla f = A(\hat{\mathbf{x}} - \mathbf{x}) = \mathbf{b} - A\mathbf{x} \equiv \mathbf{r}$$

- $f(\mathbf{x})$ の値を計算するには真の解 $\hat{\mathbf{x}}$ が必要だが、$f(\mathbf{x})$ の値そのものではなく勾配のみがあれば良い
- 行列ベクトル積だけで計算できるので、A が疎行列の時、特に有効 ⇒ 共役勾配法を利用

共役勾配法による一般の関数の最適化

- $f(\mathbf{x})$ は厳密な二次形式ではない
- 係数行列 A も分からない
- α_n は一次元最適化を使って反復法で求める
- β_n は、A を知らなくても計算可能

$$\beta_n = \frac{\mathbf{r}_{n+1}^T \mathbf{r}_{n+1}}{\mathbf{r}_n^T \mathbf{r}_n}$$

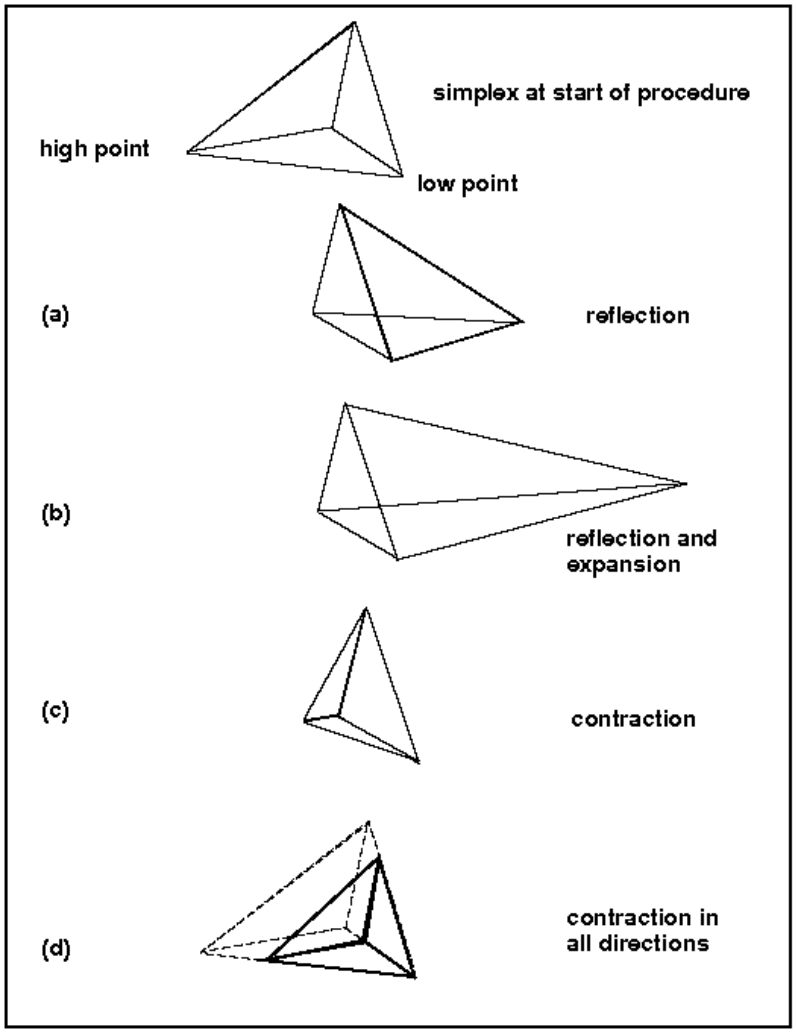
Nelder-Mead の滑降シンプレックス法

- 関数値のみ。導関数の情報を必要としない
- プログラミングが簡単
- 収束は遅いが、安定に極小値が求まる
- $N+1$ 個の頂点からなる N 次元の単体 (シンプレックス) を変形しながら、極小値を探す
  - 2 次元: 三角形
  - 3 次元: 四面体
- 別名「アメーバ法」

Nelder-Mead の滑降シンプレックス法

- $N+1$ 個の点 $x_0, x_1, \cdots, x_N$ は $f(x_0) \leq f(x_1) \leq \cdots \leq f(x_N)$ と並べられているとする
- 最大値を取る点 x_N を除く N 点の重心を x_g とする
- Nelder-Mead 法では以下のステップを繰り返す
  - x_N の x_g に関する対称な点と x_N の関数値を比較し、小さい方に移動 (反射)
  - x_0 の関数値よりも小さくなるようであればさらに先に進む (拡大)
  - x_N の関数値が x_{N-1} のものよりもまだ大きい場合には x_N を x_g に近づける (縮小)
  - それでも x_N の関数値が小さくならない場合、x_0 以外の点を x_0 に一様に近づける (収縮)
- プログラム例: nelder_mead_2d.c

Nelder-Mead の滑降シンプレックス法

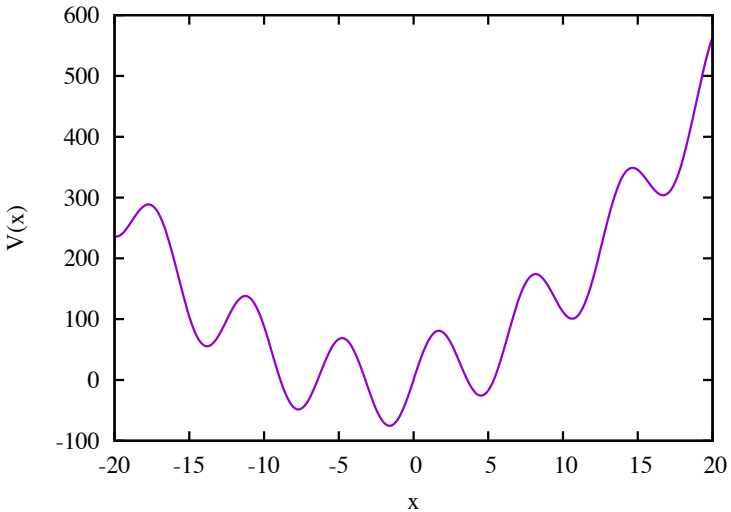


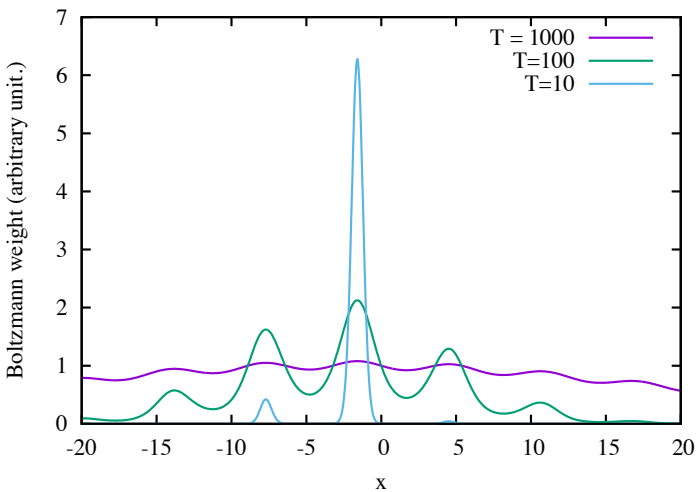
http://www.kniaz.net/software/RosNM.aspx

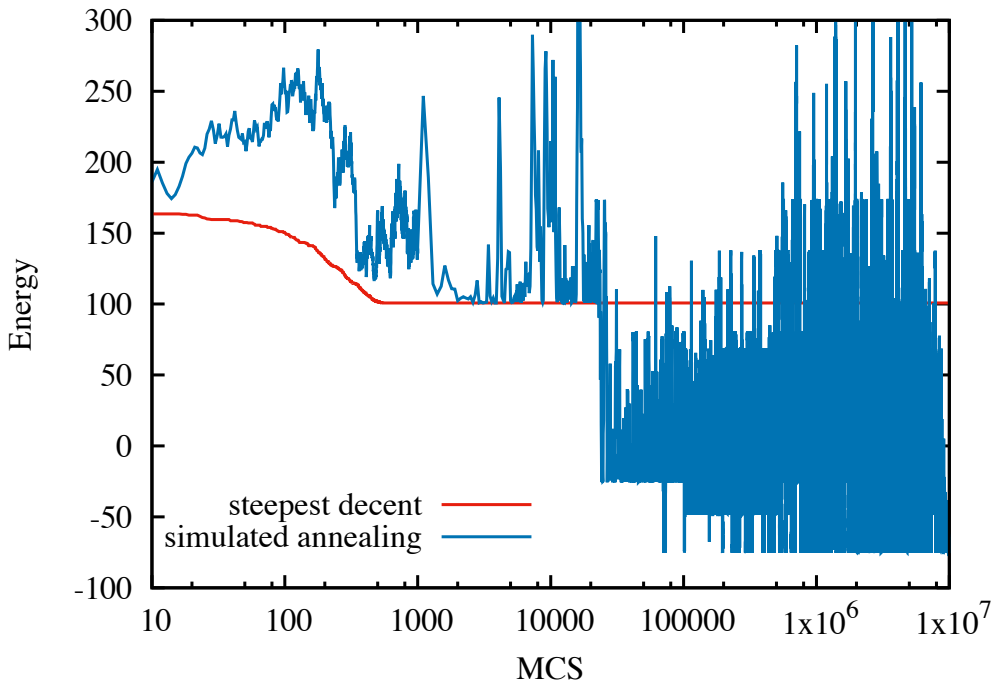
確率過程を用いた最適化

- 最急降下法 (steepest decent) mc_steepest_descent_1d.c
  - 初期状態をランダムに定める
  - 配位を少しだけ変化させる
  - エネルギー (コスト関数) が小さくなるなら採択、大きくなるなら棄却
  - 状態が変化しなくなるまでくり返す
  - 問題点：エネルギー極小状態にすぐに捕まってしまう
- 徐冷法 (simulated annealing) simulated_annealing_1d.c
  - いきなり温度を零にするのではなく少しずつ下げていく
  - どれくらいゆっくり下げれば良いか? $T(t) \geq cN/\log(t+2)$
  - 実際には適当なスケジューリングで温度を下げ、何回か繰り返して最も良い結果を採択

最急降下法とシミュレーテッド・アニーリング







$$T(t) = 100 - \frac{99}{10^7}\,t$$

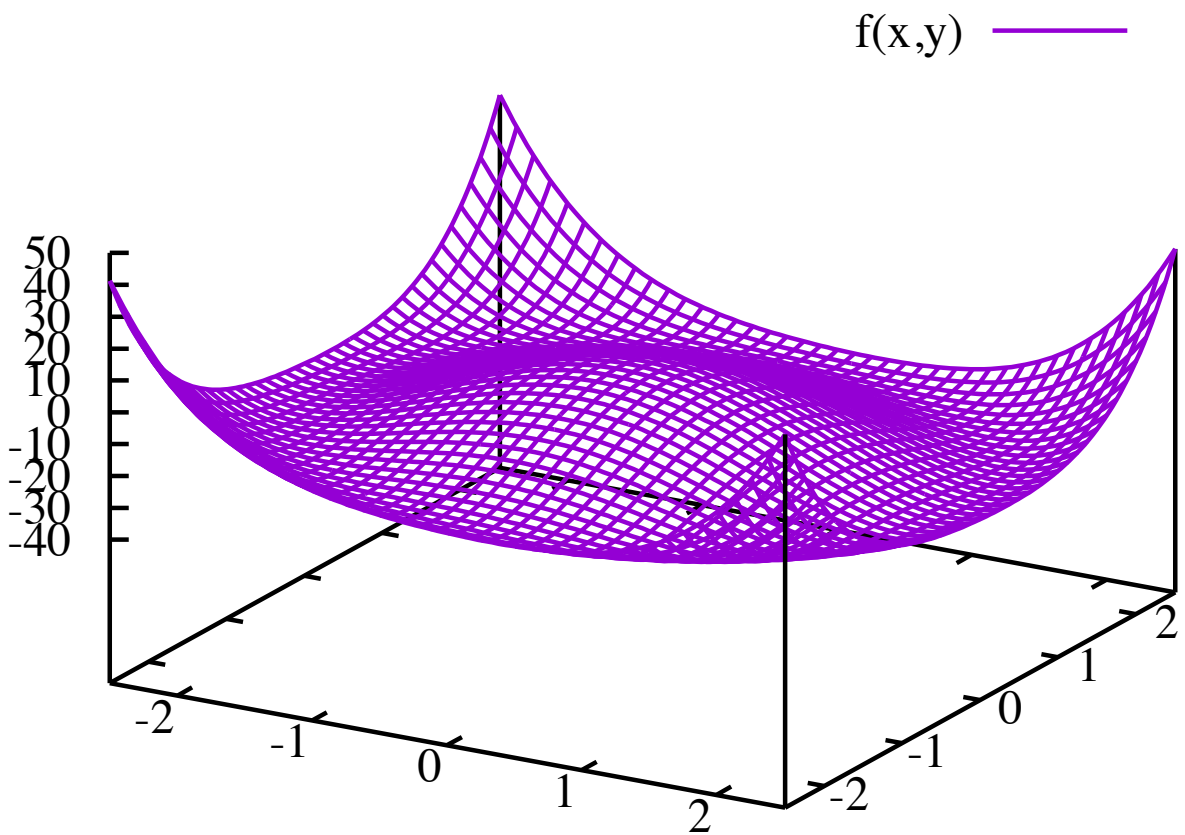
離散最適化問題への応用

- 微分を必要としないので、離散最適化問題にも適用可
  - 例: 巡回セールスマン問題、数独、ナップザック問題
- いかに状態とエネルギーを定義するかが重要
  - 例: $n \times n$ 魔法陣 (行・列・ななめの和 $M = n(n^2+1)/2$)
  - 「状態」C: $1 \sim n^2$ の自然数をある順序でます目に並べたもの
  - 「エネルギー」

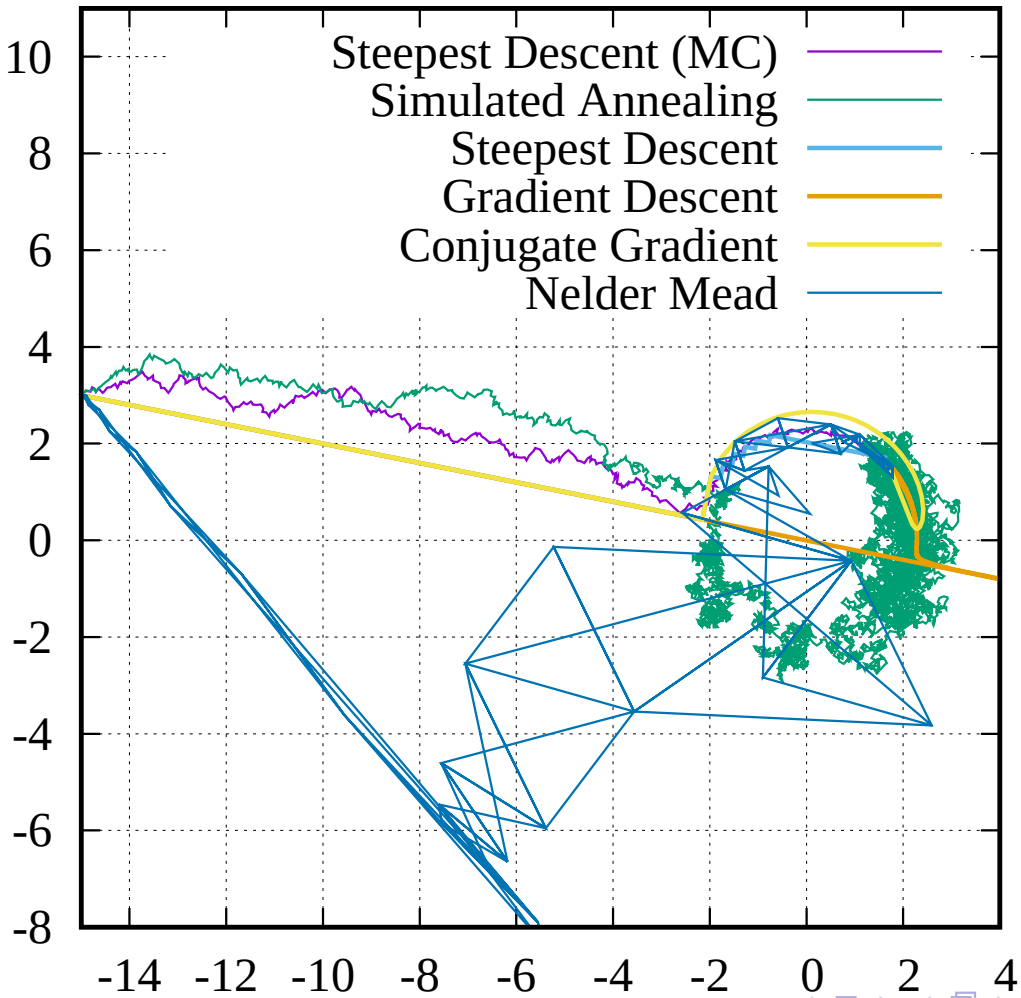
$$E(C) = \sum_{\text{row}} (S_r - M)^2 + \sum_{\text{col}} (S_c - M)^2 + \sum_{\text{diag}} (S_d - M)^2$$

  - 「正しい」魔方陣: $E(C) = 0$
- 解の数 (絶対零度のエントロピー) を求めるのにも利用できる

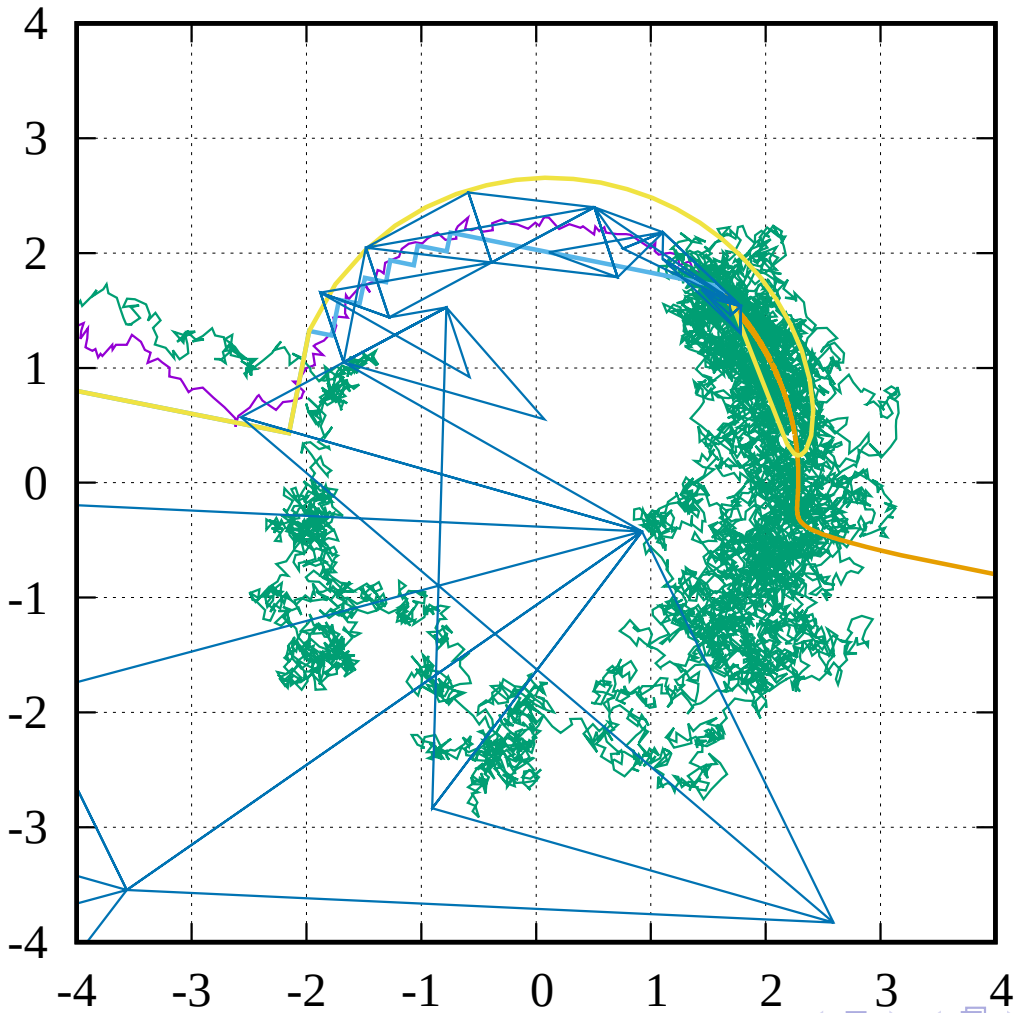
例題 (二次元の最適化)



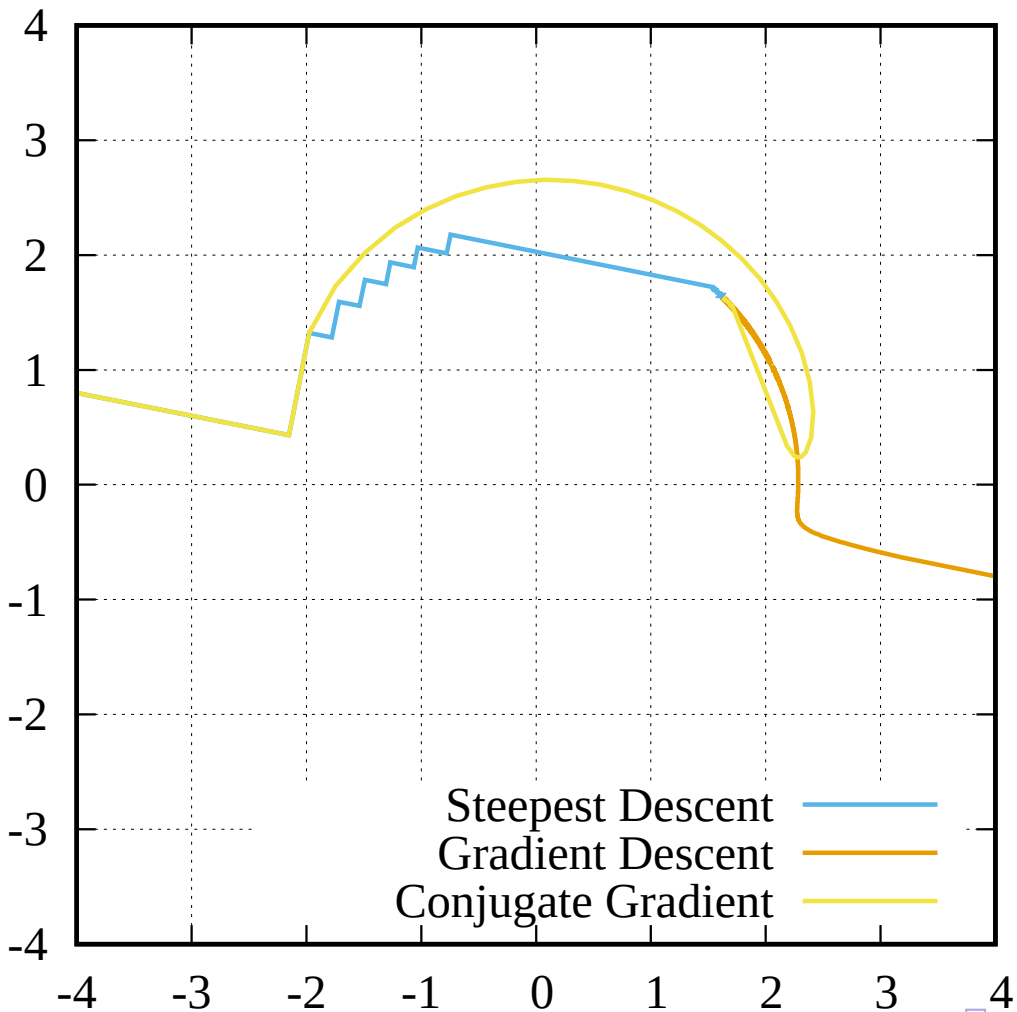
様々な最適化手法の比較 (1/4)



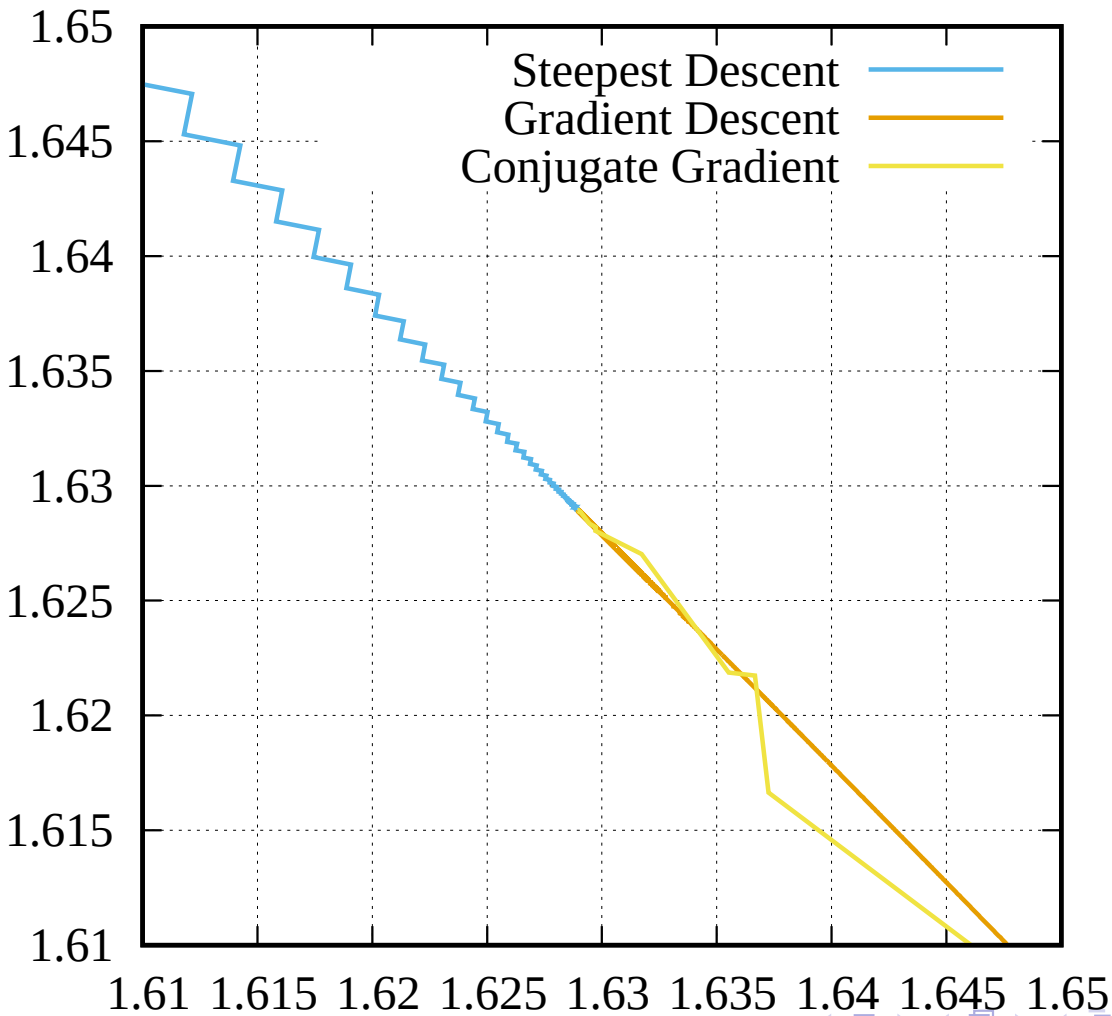
様々な最適化手法の比較 (2/4)



様々な最適化手法の比較 (3/4)



様々な最適化手法の比較 (4/4)



真の解への近づき方