E7441: Scientific computing in biology and biomedicine

Short introduction to stochastic optimization

Vlad Popovici, Ph.D.

Fac. of Science - RECETOX

Outline

Introduction

Let $A \subset \mathbb{R}^n$ and $f : A \to \mathbb{R}$ a continuous function,

$$A^* = \arg\min f = \{a \in A | f(a) \leq f(x),\ \forall x \in A\}$$

If A is compact then $A^* \neq \emptyset$.
Goal: find a good approximation of A^*.
Metaheuristics: generate a sequence $X_t : \Omega \to A$ of *random* n–dimensional vectors from a probability space.

- *stochastic convergence*

$$\forall \epsilon > 0, \Pr(\text{dist}(X_t, A^*) < \epsilon) \to 1,\ \text{as } t \to \infty$$

- *almost sure convergence*

$$\Pr(X_t \to A^*,\ \text{as } t \to \infty) = 1$$

Random search algorithms

Let $A = [0, 1]^n \subset \mathbb{R}^n$ and $U(A)$ be the uniform distribution on A.

Algorithm 1: Pure Random Search

$t \leftarrow 1$;
generate $x_0 \sim U(A)$;
while *true* **do**
    generate $x_t \sim U(A)$;
    if $f(x_t) < f(x_{t-1})$ **then**
        $t \leftarrow t + 1$;
    end
end

From Borel-Cantelli Lemma:
$\Pr(X_t \rightarrow A^*, \text{ as } t \rightarrow \infty) = 1$

Exercise 1 - implement the PRS - see the Jupyter notebook.

Accelerated Random Search

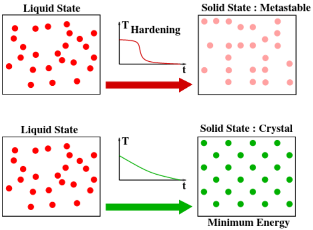
- in PRS the information from previous steps/attempts is not used
- ASR is confining the search around the current best choice
- when a better choice is found, the search space is reinitialized to full space
- let $c > 0$ be a shrinking factor, $\rho > 0$ the desired precision and let $B(x, r)$ denote a ball of radius r centered at x

Algorithm 2: Accelerated Random Search

```
t → 1, r_1 ← 1;
generate x_1 ~ U(A);
while true do
    generate y_t ~ U(B(x_t, r_t) ∩ A);
    if f(y_t) < f(x_t) then
        x_{t+1} ← y_t;
        r_{t+1} = 1;
    else
        if r_t ≥ ρ then
            x_{t+1} ← x_t;
            r_{t+1} ← r_t/c;
        else
            r_{t+1} ← 1;
    t ← t + 1;
```

Simulated Annealing

- inspired from physics: to reach an optimal (minimum) energy, the process of cooling (called annealing for metals) must not be too fast
- incorporates a stochastic process of escaping local minima
- the function *f* is now called “energy” or “cost function” and the parameter governing the escape process - “temperature”



Algorithm 3: Simmulated Annealing generic algorithm

```
initialize randomly x ∈ ℝ^n;
x* ← x;                                   // current best choice
for k = 0, 1, ..., K_max do
    x' ← nearby(x);
    if f(x') < f(x) then
        x ← x';
    else
        x ← x' with probability Pr(f(x'), f(x), T);
    if f(x) < f(x*) then
        x* ← x;
return x*
```

- the acceptance of worse values for x allows exploring a space away from the current local minima
- a common probability function used is

$$\exp\left(-\frac{f(x') - f(x)}{T}\right)$$

- the temperature T may be constant or decreasing with time: start with $T = 1$ and then update $T = \frac{K_{max}-k}{K_{max}}$

Algorithm 4: Simulated annealing for continuous functions

```
initialize randomly x ∈ ℝ^n;
x* ← x;                                    // current best choice
for k = 0, 1, ..., K_max do
    x' ← N(0, 1);
    T ← (K_max − k)/K_max;
    if f(x') < f(x) or exp(−(f(x') − f(x))/T) > rand() then
        x ← x';
        if f(x) < f(x*) then
            x* ← x;
return x*
```

Exercise 2 - implement the SA - see the Jupyter notebook.

Practical stochastic optimization:

In PYTHON,

- PYMOO - https://pymoo.org/
- STOCHOPY - https://keurfonluu.github.io/stochopy/
- OPTUNA - https://optuna.org/ <- very interesting also for machine learning
- etc etc

InR,

- GENSA - https://cran.r-project.org/web/packages/GenSA/
- MCO - https://cran.r-project.org/web/packages/mco/ind

Questions?