

Aide à la décision - R6.06

## Optimisation

Déterminer le minimum d'une fonction

Descente de gradient *versus* recuit simulé

Michel Salomon

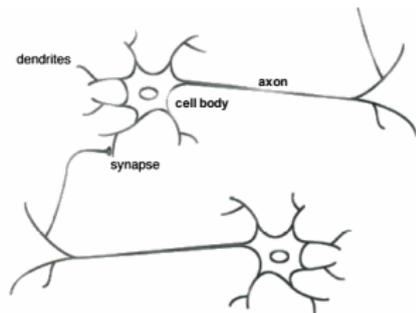
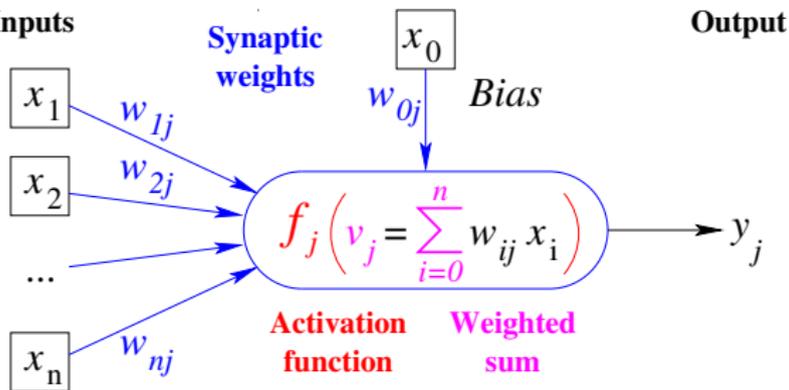
IUT Nord Franche-Comté  
Département d'informatique

## Classification des algo. de *Machine Learning* / Apprentissage auto.

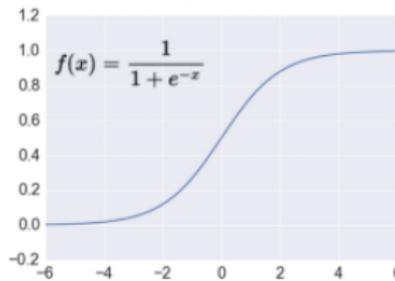
- Apprentissage supervisé
  - Ensemble d'entraînement (ou d'apprentissage) avec les réponses correctes (cibles ou étiquettes - paires  $(X, Y^t)$ )
  - L'entraînement vise à généraliser pour fournir des réponses correctes pour de nouvelles entrées  $(X')$  en **minimisant** les erreurs de prédiction
- Apprentissage non supervisé
  - Les étiquettes/réponses correctes ne sont pas disponibles
- Apprentissage semi-supervisé
  - Des données d'entrée partiellement étiquetées sont disponibles
- Apprentissage par renforcement
  - L'algorithme reçoit une mesure de la qualité d'une action
  - Il explore et essaie différentes possibilités pour trouver comment effectuer correctement une tâche donnée (apprentissage par essais) et **maximiser** la qualité à long terme

# Réseau de neurones - composant de base : le neurone

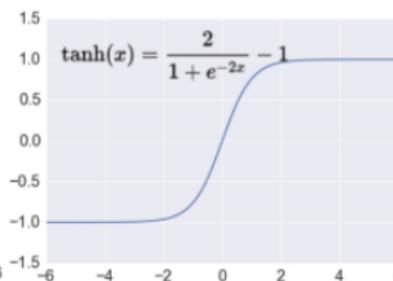
Inputs



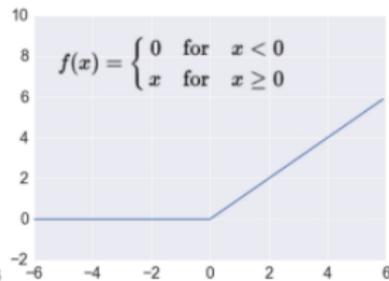
Sigmoid



TanH

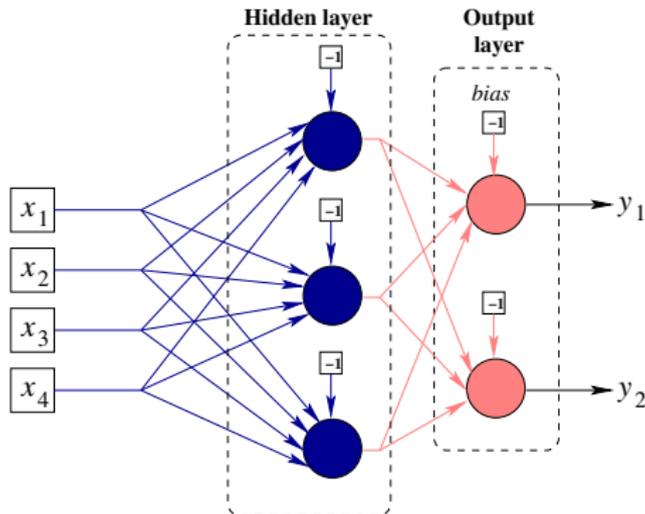


ReLU



# Réseau de neurones - perceptron multicouches

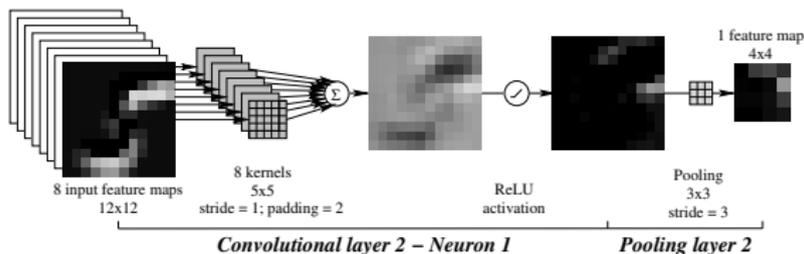
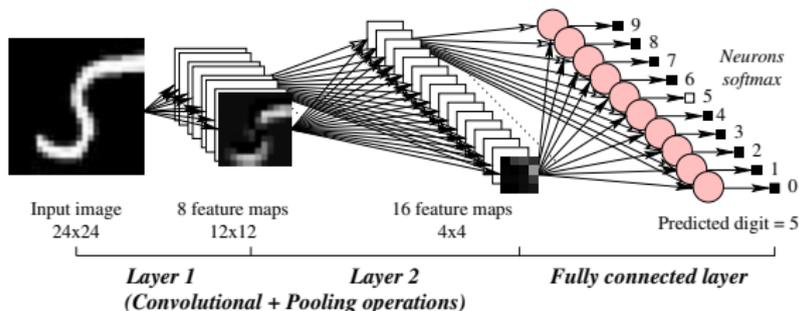
- Des neurones organisés en couches
- Classification (exemples avec [Playground TensorFlow](#)) ; régression



Apprentissage supervisé des poids à partir de données du jeu d'entraînement  $(X, Y^t) = ((x_1, x_2, x_3, x_4), (y_1^t, y_2^t))$  de manière à minimiser  $\| Y - Y^t \|^2$

# Réseau de neurones - apprentissage profond (*Deep Learning*)

- Un “grand” nombre de couches, d’autres types de neurones, ...
- Une des archi. phares : réseau de neurones convolutifs



MNIST problem - Poids synaptiques = convolution kernels

## Qu'attend-t-on d'un réseau de neurones ?

- Qu'il apprenne de "stimulations" provenant de son environ.

## Que fait la phase d'apprentissage ?

- Modifie les paramètres ajustables (poids synaptiques et biais)
- Utilise des règles pour les mettre à jour itérativement

## Que signifie "stimuler" ou entraîner un réseau de neurons ?

- Utiliser données pour le guider la mise à jour des paramètres

## Qu'est-ce que l'apprentissage supervisé ?

L'optimisation des poids synaptiques et biais pour minimiser l'erreur de sortie / la prédiction ( $MSE = \| Y - Y^t \|^2$ )

Entraîner un réseau de neurones revient à résoudre problème d'optimisation généralement non linéaire

## Un réseau bien entraîné est capable de généraliser

- N'importe quelle méthode d'optimisation pourrait être utilisée
- En pratique **une méthode de descente de gradient**
  - Une méthode d'optimisation locale
    - converge vers un minimum local
  - Comment est calculer le gradient (des poids synaptiques et biais)
    - Grâce à l'algorithme de rétropropagation du gradient
  - Variantes de la descente de gradient
    - Descente batch ou lot (*batch gradient descent*)
    - Descente stochastique (*stochastic gradient descent - SGD*)
    - Descente par mini-batch (*mini-batch gradient descent*)
  - Méthodes d'optimisation avancées à base du gradient
    - *Momentum, Adagrad, etc.*

## Description générale d'un problème d'optimisation

- Étant donné un ensemble  $\Omega$  de
  - configurations ;
  - ou solutions admissibles
- du problème à résoudre et une fonction objectif  $f$  ;
- trouver le minimum  $x'$  de  $f$  par rapport à l'ensemble  $\Omega$
- Soit  $x'$  qui vérifie

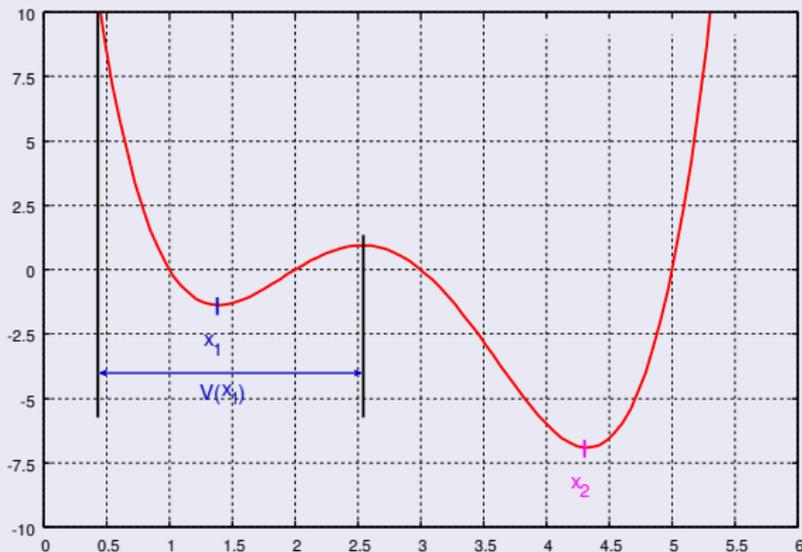
$$x' \in \Omega \text{ et } f(x') = \min_{x \in \Omega} f(x)$$

## Méthodes étudiées

- Méthode d'optimisation locale, déterministe
  - Descente de gradient
- Méthode d'optimisation globale, stochastique
  - Recuit simulé

# Fonction objectif considérée

$$f(x) = (x - 1) \cdot (x - 2) \cdot (x - 3) \cdot (x - 5) \text{ sur } [0; 6]$$



- $x_1$  est un minimum local sur le voisinage  $V(x_1)$  ;
- $x_2$  est le minimum global

# Descente de gradient - fonction objectif de dimension 1

- Permet de trouver un minimum d'une fonction dérivable
- Si la fonction est dérivable, alors un minimum  $x'$  vérifie :

$$f'(x') = \frac{\partial f}{\partial x}(x') = 0$$

dans le cas de la fonction considérée sa dérivée  $f'(x)$  est :

$$f'(x) = \frac{\partial f}{\partial x}(x) = 4 \cdot x^3 - 33 \cdot x^2 + 82 \cdot x - 61$$

- Résoudre l'équation  $f'(x) = 0$  revient donc à trouver les racines d'un polynôme de degré 3 → difficile
- La descente de gradient trouve un minimum  $x'$  itérativement :
  - À partir d'une valeur initiale  $x^0$  (plus ou moins bien choisie)
  - on construit une suite de valeurs  $x^k$  qui converge vers un minimum de la fonction objectif

- La méthode repose sur l'observation qu'en un point  $a$  la fonction  $f$  décroît le plus rapidement dans la direction opposée à celle de  $f'$  en  $a$ . En effet :

$$f'(a) = \frac{\partial f}{\partial x}(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$$

d'où :

- si  $f'(a) > 0 \Rightarrow f(a+h) > f(a)$   
→  $f$  décroît dans la direction des  $x$  négatifs
- si  $f'(a) < 0 \Rightarrow f(a+h) < f(a)$   
→  $f$  décroît dans la direction des  $x$  positifs
- Ainsi, en définissant  $x^{k+1}$  comme suit :

$$x^{k+1} = x^k - \gamma \cdot f'(x^k) = x^k - \gamma \cdot \frac{\partial f}{\partial x}(x^k) \quad (1)$$

pour  $\gamma > 0$  suffisamment petit on a  $f(x^{k+1}) \leq f(x^k)$

# Descente de gradient

## Remarques sur $\gamma$

- ce nombre s'appelle le pas ;
- sa valeur influe sur la vitesse de convergence ;
- elle influe également sur le minimum trouvé qui sera plus ou moins proche du minimum exact ;
- si le pas est trop grand, la méthode peut avoir un comportement chaotique voire diverger ;
- le pas peut éventuellement varier à chaque itération

## Détection de la convergence

- Arrêter les calculs lorsque les valeurs successives de  $x$  sont suffisamment voisines

$$\left| x^{k+1} - x^k \right| \leq \epsilon$$

- Arrêter les calculs lorsqu'on a effectué un max. d'itérations

- La descente de gradient trouve un minimum  $x'$  itérativement :
  - À partir d'un vecteur initial  $x^0$  (plus ou moins bien choisi)
  - on construit une suite de vecteurs  $x^k$  qui converge vers un minimum de la fonction objectif
- La formule (1) définissant  $x^{k+1}$  devient :

$$x^{k+1} = x^k - \gamma \cdot \nabla f(x^k)$$

où  $\nabla f$  désigne le gradient de  $f$  (d'où le nom de la méthode)

- Comme

$$\nabla f(x) = \left( \frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_n}(x) \right)^T$$

on a la mise à jour suivante pour la  $i$ -ème composante de  $x^k$

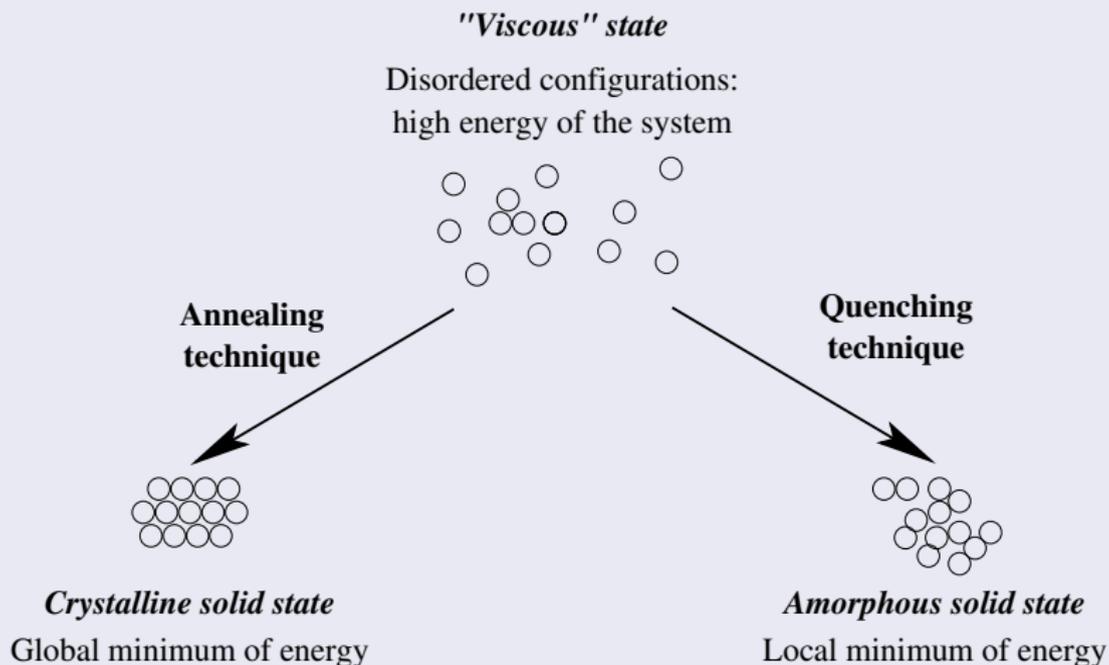
$$x_i^{k+1} = x_i^k - \gamma \cdot \frac{\partial f}{\partial x_i}(x^k)$$

## Origine du recuit simulé (*Simulated Annealing*)

- Obtenu par analogie avec :
  - le phénomène thermodynamique de recuit des métaux ;
  - le processus de refroidissement pour cristalliser un liquide
- Principe
  - Initialement, le métal est porté à haute température
  - Ensuite on le refroidit progressivement :
    - à haute température les atomes sont agités  
→ configurations atomiques équiprobables ;
    - à basse température les atomes s'organisent  
→ structure atomique parfaite, proche de l'état d'énergie minimale
  - Contrainte
    - refroidissement lent → ne pas rester bloqué par un min. local

# Recuit simulé - origine et principe

## Origine du recuit simulé (*Simulated Annealing*)



## Distribution de Gibbs / Boltzmann (mécanique statistique)

$$P(x) = \frac{1}{Z_T} \cdot \exp\left(-\frac{E(x)}{K_B \cdot T}\right)$$

où :

- $x \in \Omega$  (une configuration du système physique) ;
- $E$  est une fonction d'énergie définie sur  $\Omega$  ;
- $T$  est la température

## Algorithme de Metropolis (1953) - Simulation à $T$ constante

- Suite de configurations obtenues par réarrangement élémentaire

$$P(y|x) = \min\left(1, \exp\left(-\frac{E(y) - E(x)}{T}\right)\right)$$

## Impact de la valeur de la température sur $P(y|x)$

- Quand  $T$  est grand alors  $P(y|x)$  est proche de 1
  - Même si  $E(y)$  est bien plus grand que  $E(x)$
  - La fraction est proche de 0

**À haute température  
n'importe quelle configuration est acceptée**

- Quand  $T$  est petit
  - Si  $E(y) \leq E(x)$  alors  $P(y|x) = 1$
  - Si  $E(y) > E(x)$  alors  $P(y|x)$  tend vers 0

**À basse température  
seule une config. faisant baisser l'énergie est acceptée**

# Recuit simulé - analogie optimisation / syst. physique

Problème d'optimisation	Système physique
Fonction de coût/objectif $f(x)$	Énergie libre $E(x)$
Variables du problème	«Coordonnées» des atomes
Trouver une bonne configuration	Trouver un état de basse énergie

⇒ **algorithme du recuit simulé (Kirkpatrick *et al.* - 1983)**

## Algorithme basé sur deux procédures

- Procédure d'échantillonnage de la distribution
  - Phase d'exploration → Notion de voisinage
  - Phase d'acceptation → Utilisation de l'algo. de Metropolis
- Procédure de refroidissement
  - Schéma de décroissance de la température

- Capacité à éviter les minima locaux
  - Acceptation probabiliste de config. d'énergie plus élevée
  - Probabilité d'autant plus élevée que  $T$  est grande
- Choix des paramètres (convergence en un temps fini)
  - $T^0$  doit être choisie de sorte que pratiquement toutes les configurations proposées soient acceptées
  - Les paliers de température doivent être suffisamment long  
→ atteindre la distribution stationnaire de Gibbs/Boltz.
  - Baisse de température entre deux paliers pas trop rapide
- Choix des paramètres (en pratique)
  - $T^0$  et longueur d'un palier choisis à l'issue d'expérimentations
  - Schéma de température exponentiel
$$T^k = T^0 \times \alpha^k, k \in \mathbb{N} \text{ et } 0 < \alpha < 1$$
  - Critères d'arrêt possibles
    - Le pourcentage de config. acceptées passe sous un seuil fixé ;
    - la variance de l'énergie est faible ;
    - choix d'une température minimale

## Description de la version de Metropolis

```
1:  $k = 0$ 
2:  $T^k = T^0$  // Température initiale
3:  $x^k = x^0$  // Configuration initiale
4: repeat
5:   repeat
6:     Tirer aléatoirement  $y \in V(x^k)$  // Une configuration voisine de  $x^k$ 
7:     if  $\Delta E = (E(y) - E(x^k)) < 0$  ou  $\exp(-\frac{\Delta E}{T^k}) > \mu$ ,  $\mu \in [0; 1]$  tiré selon une
       loi uniforme then
8:        $x^{k+1} = y$ 
9:     else
10:       $x^{k+1} = x^k$ 
11:    end if
12:  until fin de palier
13:   $T^{k+1} = g(T^k)$  //  $g$  est strictement décroissante
14:   $k = k + 1$ 
15: until critère d'arrêt vérifié
```

# Équation de la diffusion - définition

- Définie par Aluffi *et al.* - 1985 / Geman *et al.* - 1986
- Équation différentielle stochastique

- Configuration du type  $x^t = (x_1^t, \dots, x_n^t)^T$

$$dx^t = -\nabla E(x^t) \cdot dt + \sqrt{2 \cdot T^t} \cdot dw^t$$

- La fonction doit vérifiée :  $\lim_{\|x\| \rightarrow \infty} E(x) = +\infty$
- $T^t$  est la température à l'instant  $t \in \mathbb{R}^+$
- $w^t$  est un mouvement brownien dans  $\mathbb{R}^n$  (marche au hasard)
- Échantillonne également la distribution de Gibbs / Boltzmann

$$P(x^t) = \frac{1}{Z_T} \cdot \exp\left(-\frac{E(x^t)}{K_B \cdot T^t}\right)$$

⇒ variante du recuit simulé

# Équation de la diffusion - principe

- Décomposition de l'équation en deux termes

$$\delta_1^t = -\nabla E(x^t) \cdot dt$$

$$\delta_2^t = \sqrt{2 \cdot T^t} \cdot dw^t$$

- Terme 1 → descente de gradient
- Terme 2 → perturbation aléatoire
  - À haute temp. → possibilité de sortir d'un minimum local
  - À basse temp. → perturbation négligeable → descente de gradient
- Algorithme basé sur deux étapes
  - Résolution itérative de l'équation
    - Discrétisation de l'équation par la méthode d'Euler-Cauchy
  - Étape de refroidissement

# Équation de la diffusion - algorithmme

## Description pseudo-code

- 1:  $t = 0$
- 2:  $T = T^0$  // Température initiale
- 3:  $\Delta P^t = P^0$  // Pas de temps initial
- 4:  $x^t = x^0$  // Configuration initiale
- 5: **repeat**
- 6:  $\Delta w_i^t = N(0, \Delta P^t)$
- 7:  $x_i^{t+\Delta P^t} = x_i^t - \frac{\partial E}{\partial x_i}(x^t) \cdot \Delta P^t + \sqrt{2 \cdot T^t} \cdot \Delta w_i^t, i \in 1, \dots, n$
- 8:  $T^{t+\Delta P^t} = g(T^t)$  //  $g$  et  $h$  sont strictement décroissantes
- 9:  $\Delta P^{t+\Delta P^t} = h(\Delta P^t)$
- 10:  $t = t + \Delta P^t$
- 11: **until** critère d'arrêt vérifié

# Équation de la diffusion - convergence et paramètres

- Conditions de convergence
  - Condition sur la fonction d'énergie  $E$
  - Condition sur la température  $T^t$

$$\lim_{t \rightarrow \infty} T^t \geq \frac{c}{\ln(2+1)}, c \text{ suffisamment grand}$$

- Condition sur le pas de temps  $dt$ 
  - $dt$  doit être suffisamment «petit» et tendre vers zéro
- Choix des paramètres
  - Paramètres commun avec le recuit simulé classique
    - Même approche que pour le recuit simulé
  - Paramètre spécifique à cette méthode : le pas de temps
    - Schéma de décroissance exponentiel (idem que pour la température)

$$\begin{aligned} T^{t+\Delta P^t} &= \alpha_T \times T^t, 0 < \alpha_T < 1 \\ \Delta P^{t+\Delta P^t} &= \alpha_P \times \Delta P^t, 0 < \alpha_P < 1 \end{aligned}$$

## Écrire du code implémentant les étapes suivantes

Utiliser Python, pyplot et animation de matplotlib

- 1 Écrire une fonction calculant  $f(x)$  et une autre  $f'(x)$
- 2 Afficher  $f(x)$  avec  $x \in [0; 6]$  avec pyplot
- 3 Écrire une fonction `descenteGradient` qui prend en entrée un point de départ  $x^0$ , une valeur pour  $\gamma$ , et un nombre max. d'itérations pour arrêter la descente (nombre de  $x$  calculés)
- 4 Afficher sur le graphe de  $f(x)$  les points successifs trouvés
- 5 Écrire une fonction `recuitSimule` qui prend en entrée un point de départ  $x^0$ , une température initiale  $T^0$ , une longueur de palier de temp., une valeur pour  $\alpha$  pour faire décroître  $T$ , et un nombre max. de températures comme critère d'arrêt
- 6 Afficher sur le graphe de  $f(x)$  les points successifs trouvés
- 7 Faire une animation (vidéo) montrant l'exploration du recuit