Comparaison de différentes méthodes d'optimisation sans contrainte

Introduction

Dans ce TP d'introduction à l'optimisation pour l'image nous allons introduire des outils basiques d'optimisation et explorer quelques problèmes reliés à l'image. Le but, est qu'à la fin du TP vous sachiez ce qu'est un gradient, comment chercher le minimum d'une fonction par descente de gradient à pas constant, avoir un aperçu des problèmes d'image résolubles par optimisation.

Une fonction F définie de $\mathbb{R}^N$ dans $\mathbb{R}$ sera souvent appelée "fonctionnelle" ou énergie. Dans ce TP on supposera F différentiable et sa différentielle en un point $x \in \mathbb{R}^N$ est notée $DF(x)$. C'est une fonction linéaire de $\mathbb{R}^N$ dans $\mathbb{R}$. Comme $\mathbb{R}^N$ est muni d'un produit scalaire canonique ($< x|y >= \sum_i x_i y_i$) cette fonction peut être représentée par un vecteur de $\mathbb{R}^N$ que l'on nomme gradient de F et qui se note $\nabla F(x)$.

$$F(x+\delta) = F(x) + < \nabla F(x)|\delta > + o(\|\delta\|)$$

Si on cherche à minimiser la fonction F, on voit à l'équation ci-dessus que (dans le cas d'une fonctionnelle différentiable) il faut au moins que le gradient de F soit nul en x pour que ce point soit un minimiseur. Calculer les gradients des fonctions suivantes :

1. $F_1(x) = x^2, \quad F(x) = ax^2 + bx, \ (\mathbb{R} \to \mathbb{R})$
2. $F_2(x) = (x_1 - x_2)^2 \ (\mathbb{R}^2 \to \mathbb{R})$
3. $F_3(x) = \sum_{1 \le i < N} (x_{i+1} - x_i)^2 (\mathbb{R}^N \to \mathbb{R})$
4. $F_4(x) = \sum_{j, 1 \le i < N} (x_{i+1,j} - x_{i,j})^2 + \sum_{1 \le j < N, i} (x_{i,j+1} - x_{i,j})^2 \ (\mathbb{R}^{N \times N} \to \mathbb{R})$. Ici, la variable x est indexée par deux coordonnées i et j, c'est une image.
5. $F_5(x) = \sum_{i,j} (x_{i,j} - a_{i,j})^2 + F_4(x)$. Les $a_{i,j}$ sont des constantes. C'est une image.

0.1 Minimisation par descente de gradient

L'algorithme de descente de gradient à pas constant est le suivant [1] :

— Choisir un point de départ arbitraire x_0.
— Choisir une constant ρ qu'on appellera le pas.
— Effectuer l'itération suivante $x_{n+1} = x_n - \rho \nabla F(x_n)$ jusqu'à convergence.

Appliquer cet algorithme avec la fonction $F(x) = (x-1)^2$ en choisissant comme point de départ $x_0 = 0$, $\rho = 0.1$ et un nombre d'itérations fixé à 100.

Tracer la fonction $\log(|x_n - 1|)$. Qu'en déduisez-vous sur la vitesse de convergence ? Trouver, par l'expérience et la preuve, toutes les valeurs de ρ qui conduisent à une convergence de l'algorithme.

1 Le gradient d'une image

Une image peut être vue comme une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$. Elle admet donc un gradient qui est un vecteur à deux dimension en chaque point de l'image. À l'aide de la fonction **grad_im** afficher le gradient d'une image (en particulier sa norme). Quels commentaires pouvez-vous faire ? Afficher le gradient d'une image de bruit blanc (**randn(256,256)**, par exemple).

Une caractéristique des images naturelles est la faiblesse de leur gradient et la concentration de ses valeurs non-nulles sur de petites zones de l'image. C'est cette caractéristique que l'on va utiliser dans la suite pour résoudre des problèmes simples de traitement des images par des techniques d'optimisation. La manière de résoudre ces problèmes dans ce TP d'introduction est très simpliste. L'objet de la suite du cours est l'introductions de fonctionnelles mieux adaptées aux images et des algorithmes (plus complexes) pour les minimiser.

1. Pour vous aider, téléchargez les scripts matlab qui se trouvent à l'adresse : `http://perso.telecom-paristech.fr/~ladjal/TPMVA/TP0`

2 Un problème de traitement des images : Le zoom

On se pose le problème suivant : On connaît une image $a(1 \dots M, 1 \dots N)$ et l'on veut afficher une version zoomée. L'image recherchée x a la taille $(2 \times M - 1, 2 \times N - 1)$ et vérifie

$$x_{2i-1,2j-1} = a_{i,j} \tag{1}$$

On voit qu'un 3/4 des pixels de l'image x ne sont pas connus. Nous allons les fixer en disant que notre image zomée doit avoir un gradient faible. Cela se formalise par

Minimiser $F_4(x)$ en fixant la valeur des pixels connus

Pour adapter l'algorithme de la descente de gradient au problème, on remarque que si x est la concaténation de deux vecteurs x_1 et x_2

$$\nabla F(x_1, x_2) = \begin{bmatrix} \nabla F(x)_1 \\ \nabla F(x)_2 \end{bmatrix}$$

Ainsi, on peut se contenter de descendre le gradient déjà calculé dans la première section pour F_4 seulement sur les composantes de l'image qui sont inconnues.

Pour quelles valeurs du pas l'algorithme proposé converge-t-il ? Trouver une explication théorique.

3 Un autre problème : retrouver une image à partir de ses gradients dégradés

Une image a subi les traitements suivants

1. On calcule le champ de gradient.
2. On annule tous les gradients aux pixels dont la position n'est pas multiple d'un certain entier m.
3. On annule aussi les gradients plus petits qu'un seuil s.
4. Les gradients restants sont quantifiés sur 8 directions (on ne garde que le signe de chaque composante et on perd la norme précise du gradient).

Il ne nous reste qu'une version très parcellaire du champ de gradient. On reconstruit une image (m fois plus petite) à partir de cette donnée en cherchant une image dont le champ de gradient est le plus proche possible du champ de gradient qui nous est donné en entrée.

Ces étapes sont décrites dans le fichier **scratch.m**. Comment est résolu le problème ? Expliquer la solution.