MODULE MA324
TP N° 2 : MÉTHODES DE NEWTON ET QUASI-NEWTON
———-
Optimisation pour la science des données

Aéro. 3
Semestre : 2
A.U. : 2023-2024

Objectifs du TP :
Il s'agit ici de mettre en œuvre et de tester des méthodes de descente qui exploitent un peu plus que la simple information fournie par le gradient de la fonctionnelle $J(x)$ à minimiser tel que discuté dans le **TP n° 1**. Dans cette optique, nous allons

- Utiliser la matrice Hessienne H_J de la fonctionnelle $J(x)$ à minimiser dans la méthode de Newton.
- Ou bien, dans les méthodes Quasi-Newton, utiliser une approximation de la matrice Hessienne H_J, en particulier avec les méthodes BFGS et DFP.

Contrairement à l'approche adoptée dans le **TP n° 1**, nous introduirons un critère d'arrêt de la forme "$|| \nabla J(x^n)|| \leqslant \varepsilon$ ou $n \geqslant iter_{\max}$" pour interrompre l'exécution des algorithmes mentionnés ci-dessus, pour chaque méthode.
Dans ce TP, nous proposons à utiliser les fonctions $J : \mathbb{R}^N \longrightarrow \mathbb{R}$ représentant la sortie d'un réseau de neurone pour prédire l'image binaire $\{0; 1\}$ (0 et 1 représentent respectivement les deux couleurs "noir" et "blanc") :

- $J(x) = \sum_{i=1}^{p} e^{\left(a_{(i)}^T x + b_i\right)}$ où $a_{(i)}^T$ désigne la transposée de la matrice uni-colonne $a_{(i)}$. Pour tester les méthodes en $\mathbb{R}^2$, il suffit de prendre $N = 2$ et $p = 2$, donc $J(x) = e^{(x_1+3x_2-0.1)} + e^{(-x_1-0.1)}$ et $x^0 = (-5; 3)$. Cependant, on fixe $b_i = 0.1$ et on génère aléatoirement $a_{(i)}^T$ via une loi uniforme sur $[0; 1]$ $b_i = -0.1$ pour $n > 2$.
- $J(x) = \log\left(\sum_{i=1}^{N} e^{x_i}\right)$.

Exercice n°1 On s'intéresse à la méthode de Newton définie par l'algorithme ci-dessous pour résoudre le problème de $\min_{x\in\mathbb{R}^N} J(x)$:

1. Choix de $x^0 \in \mathbb{R}^N, \varepsilon > 0$ tolérence fixée
2. Affectation de $x \longleftarrow x^0$
3. Tant que $||\nabla J(x)|| > \varepsilon$ faire :
   $g \longleftarrow \nabla J(x)$
   $h \longleftarrow \nabla^2 J(x)$
   $y \longleftarrow x - hg$
   $x \longleftarrow y$
4. Affichage de x

1. Implémenter une fonction **Newton.py** prenant en argument les paramètres $param$ de la fonctionnelle à minimiser J, le gradient de J, un point de départ x^0, un nombre maximum d'itérations $iter_{\max}$ et un test d'arrêt ε, et retournant le vecteur x^n qui minimise J ainsi que le nombre d'itérations effectuées.
2. Lancer la méthode de Newton pour $N = 2$ et tracer les points calculés.
3. Afin de comparer la vitesse de convergence, on souhaiter afficher et tracer l'image synthétique $x_\star$; la solution du problème de minimisation avec une fonction prédéfinie de Python.
4. Représenter les courbes suivantes pour la fonction objective J : $(||x^n - x_\star||)_{0\leqslant n\leqslant n_{\max}}$ et $(||J(x^n) - J(x_\star)||)_{0\leqslant n\leqslant n_{\max}}$. Commenter le résultat.

5. Illustrer les résultats numériques pour $\varepsilon = 10^{-5}$, et $N \in \{2; 10; 20; 30; 40; 50\}$. Reporter le nombre d'itérations ainsi que le temps de calcul pour les valeurs de N. Comparer avec la méthode du gradient.
6. Tracer sur une même figure en échelle logarithmique les erreurs $J(x^n) - x_\star$ pour la méthode de Newton et pour la méthode du gradient vue dans le **TP n°1**. Est-ce en accord avec les résultats théoriques de la vitesse de convergence (voir le cours) ?

Exercice n°2 Méthodes Quasi-Newton (BFGS et DFP)

Comme indiqué dans l'algorithme de Newton, ce dernier nécessite l'inversion de la matrice Hessienne de la fonction coût J. En pratique, si le système est trop grand, l'inversion de la matrice Hessienne est coûteuse. Ainsi, pour remédier à ce problème, nous pouvons utiliser les méthodes Quasi-Newton, qui consistent à remplacer la matrice hessienne $\nabla^2 J(x^n)$ par une matrice B^n (une approximation de la hessienne, ou l'inverse d'une approximation de la hessienne) afin d'obtenir une généralisation de la méthode de la sécante pour un problème de dimension 1, sous la forme suivante : $x^{n+1} = x^n + B^n \left(\nabla J(x^{n+1}) - \nabla J(x^n)\right)$. Ici, nous proposons d'utiliser les algorithmes ci-dessous :

— Broyden Fletcher Goldfarb Shanno (abrégé en BFGS) :

1. Choix de $x^0 \in \mathbb{R}^N$, $\eta > 0$ un pas fixe, $\varepsilon > 0$ tolérence fixée, $B^0 = I_N$
2. Affectation de $B \longleftarrow B^0$
3. Tant que $||\nabla J(x)|| > \varepsilon$ faire :
   $d \longleftarrow -B\nabla J(x^0)$
   $\eta \longleftarrow \eta > 0$ minimum de $J(x^0 + \eta d)$
   $x \longleftarrow x^0 + \eta d$
   $s \longleftarrow x - x^0$
   $y \longleftarrow \nabla J(x) - \nabla J(x^0)$
   $B \longleftarrow B + \frac{yy^T}{y^T s} - \frac{Bss^T B}{s^T Bs}$
   $x^0 \longleftarrow x$
4. Affichage de x

— Davidon Fletcher Powel (abrégé en DFP) :

1. Choix de $x^0 \in \mathbb{R}^N$, $\eta > 0$ un pas fixe, $\varepsilon > 0$ tolérence fixée, $B^0 = I_N$
2. Affectation de $B \longleftarrow B^0$
3. Tant que $||\nabla J(x)|| > \varepsilon$ faire :
   $d \longleftarrow -B\nabla J(x^0)$
   $\eta \longleftarrow \eta > 0$ minimum de $J(x^0 + \eta d)$
   $x \longleftarrow x^0 + \eta g$
   $s \longleftarrow x - x^0$
   $y \longleftarrow \nabla J(x) - \nabla J(x^0)$
   $B \longleftarrow B + \frac{ss^T}{s^T y} - \frac{Byy^T B}{y^T By}$
   $x^0 \longleftarrow x$
4. Affichage de x

1. Mettre en œuvre les méthodes quasi-Newton BFGS et DFP en implémentant une seule fonction **QuasiNewton.py** en introduisant un booléen $option$ pour choisir entre les méthodes BFGS et DFP.
2. Lancer les deux méthodes pour $N = 2$ et tracer les points sur le même graphique.
3. Vérifier numériquement que la méthode BFGS et DFPs en n itérations très grande convergent avec $B^n \approx H_J$.

4. Représenter les courbes suivantes pour la fonction objective J : $(||x^n - x_\star||)_{0 \leqslant n \leqslant n_{\max}}$ et $(||J(x^n) - J(x_\star)||)_{0 \leqslant n \leqslant n_{\max}}$ où $x_\star$ est l'image synthétique (la solution du problème de minimisation avec une fonction prédéfinie de Python). Commenter le résultat.
5. Illustrer les résultats numériques pour $\varepsilon = 10^{-5}$, et $N \in \{10; 20; 30; 40; 50\}$. Reporter le nombre d'itérations ainsi que le temps de calcul pour les valeurs de N. Commenter le résultat en faisant une petite comparaison avec les méthodes précédemment traitées.
6. Tracer sur une même figure en échelle logarithmique les erreurs $J(x^n) - x_\star$ pour la méthode BFGS et DFP. Commenter le résultat.
7. Conclure sur la comparaison méthodes (Newton, BFGS et DFP) avec les méthodes vues en TP n° 1 dans un tableau contenant le nombre d'itérations, les erreurs $||x^n - x_\star||$, les suites de x^n, la valeur de $J(x^n)$ et la valeur de $\nabla J(x^n)$.