TP – Optimisation convexe avec des descentes de gradient stochastique

François-Xavier Dupé & Hachem Kadri

Remarque préliminaire

Ce TP repose en partie sur le précédent TP autour de la régression linéaire. Nous recommandons donc de se munir de ses notes de cours sur l'optimisation convexe et de la correction du TP (disponible sur Ametice).

Cet exercice est découpé en deux parties chacune autour de l'implémentation d'un algorithme d'optimisation pour la résolution d'un système d'équation de très grande taille. Une réponse justifiée est attendue pour chacune des questions.

Rendu du travail

Ce TP sera à rendre pour le dimanche 27 octobre dernier délai sur le dépôt Ametice prévu à cet effet. Le travail rendu comportera,

- le code source des algorithmes développé, ainsi que le code les utilisant ;
- les réponses aux questions posées dans ce sujet ;
- un exemple d'exécution.

Évidemment si vous utilisez un notebook pour réaliser ce TP, vous pouvez tout mettre dans celui-ci.

1 Cadre de travail

Dans ce travail pratique, nous nous intéressons à la résolution de systèmes d'équations linéaires de grandes tailles. Pour résoudre ces systèmes d'équations, nous allons nous focaliser sur la méthode des moindres carrés avec possibilité de régularisation. Le problème d'optimisation principal est donc le suivant,

$$\min_{x\in\mathbb{R}^d} \tfrac{1}{2}\|\mathbf{A}x - b\|^2 + \lambda\mathcal{R}(x) \ , \tag{P}$$

avec,

- $\mathbf{A} \in \mathbb{R}^{p\times d}$ la matrice contenant les coefficients du système d'équations ;
- $b \in \mathbb{R}^p$ le vecteur cible ;
- d la dimension des données (potentiellement grande) ;
- $\mathcal{R}$ un terme de régularisation ;
- λ le paramètre d'équilibre pour la gestion de la régularisation.

2 Adam : une méthode d'optimisation stochastique

Adam est un algorithme d'optimisation convexe stochastique très utilisé dans le cadre de l'apprentissage profond avec des réseaux de neurones. Il est donc adapté pour la résolution de problèmes de très grandes tailles. L'algorithme est donné par l'Algorithme 1 (page 2) dans l'article. Ici nous considérons un problème d'optimisation général,

$$\min_{x\in\mathbb{R}^d} \sum_{i=1}^{N} f_i(x) \ , \tag{Q}$$

avec $\{f_i\}_{i\in\{1\dots N\}}$ des fonctions à valeurs dans $\mathbb{R}$ convexes différentiables. Pour lier les deux problèmes (P) et (Q), il suffit de remarquer,

$$\|\mathbf{A}x - b\|^2 = \sum_{i=1}^{p} |\boldsymbol{a}_i^t x - b_i|^2 \ ,$$

avec $\boldsymbol{a}_i$ la i-ème ligne de $\mathbf{A}$.

2.1 Exercice 1 : première implémentation

Proposez une implémentation avec Python d'Adam en spécialisant le code pour le cadre du TP (gestion d'une matrice et d'une cible). Pour le calcul stochastique du gradient (g_t), nous pouvons par exemple prendre l'approche suivante : à chaque itération nous allons considérer uniquement un nombre fixé de lignes (donc un autre paramètre

de l'algorithme) de $\mathbf{A}$ tirées en considérant une distribution uniforme. N'oubliez pas d'expliquer le choix que vous avez fait (vous pouvez en tester plusieurs).

Faire des tests avec un nombre de lignes de plus en plus grand. Quelle est la différence avec les méthodes classiques proposées précédemment ?

2.2 Exercice 2 : régularisation de Tikhonov

Pour réduire l'espace des solutions, nous allons mettre en place une régularisation de Tikhonov qui consiste à utiliser le terme suivant,

$$\mathcal{R}_{\text{Tikhonov}}(x) = \tfrac{1}{2}\|x\|^2 \ .$$

Proposer une solution simple pour ajouter ce terme à l'algorithme précédent. Attention, avec la création d'un gradient par tirage aléatoire, le terme de régularisation peut n'être pris en compte que sporadiquement, proposez une solution pour éviter ce problème.

2.3 Exercice 3 : régularisation parcimonieuse

Le problème de la régularisation précédente est qu'elle a tendance à *lisser* la solution (autrement dit les valeurs de x ont tendances à être faibles). Pour cela, nous proposons d'utiliser la régularisation suivante,

$$\mathcal{R}_{\ell_1}(x) = \|x\|_1 = \sum_{i=1}^{d} |x_i| \ .$$

Cela correspond a la norme ℓ_1 du vecteur x, cependant ce terme n'est pas différentiable. A la place nous allons utiliser une version régularisée : la fonction de Huber,

$$\mathcal{R}_{\text{Huber}}(x) = \begin{cases} \frac{1}{2}\sum_{i=1}^{d} |x_i|^2 & \text{si } \|x\| < \delta, \\ \delta \sum_{i=1}^{d} |x_i| - \frac{1}{2}\delta^2 & \text{sinon,} \end{cases}$$

avec δ un paramètre de lissage (plus ce paramètre est grand plus on se rapproche de la régularisation précédente).

Introduire cette régularisation dans l'algorithme. Quels sont les résultats ?

2.4 Exercice 4 : moyenne de solutions

La nature de l'algorithme obtenu étant stochastique, le résultat est différent à chaque exécution. Pour éviter des problèmes d'extrema, nous allons lancer, pour des

mêmes entrées, plusieurs fois l'algorithme et moyenner les solutions obtenues. Maintenant nous pouvons comparer l'algorithme avec les autres méthodes vues en cours. Quels sont ses avantages et inconvénients ?

3 Descente de gradient par coordonnées

Nous allons maintenant comparer l'algorithme précédent avec un algorithme très récent proposant une formulation avec une régularisation ℓ_1 directe. L'algorithme est cette fois-ci proposé par l'Algorithme 1 (page 5).

3.1 Exercice 5 : une implémentation

Proposez une implémentation avec Python de l'algorithme en expliquant les choix que vous avez faits (par exemple, la règle de sélection de coordonnées choisie). Comparez avec l'algorithme précédent.

3.2 Exercice 6 : régularisation via Elastic-net

La régularisation par *elastic-net* correspond à un mélange des deux régularisations précédentes.,

$$\mathcal{R}_{\text{elastic-net}}(x) = \tfrac{1}{2}\|x\|^2 + \gamma\|x\|_1 \ ,$$

avec γ un paramètre d'équilibre en les deux termes de régularisation. Modifiez les deux algorithmes pour mettre en place une telle régularisation (en utilisant la fonction d'Huber pour Adam). Quels différences voyez-vous par rapport aux versions précédentes ? Comment choisir les valeurs des paramètres ?

3.3 Exercice 7 : comparaisons extensives

Proposez un moyen pour comparer équitablement les deux algorithmes que ce soit au niveau du temps de calcul que de la qualité des résultats. Pour éviter des effets de bords, nous conseillons fortement de comparer en faisant des moyennes de plusieurs lancer (pour les deux algorithmes).

Vous pouvez aussi regarder les courbes de convergences vers la solution en affichant la valeur moyenne (par rapport aux différents lancer) de la fonction objective selon les itérations. Quelle méthode choisissez-vous au final ?