Évaluation optimisation convexe II

L'évaluation est un travail d'analyse, à faire en groupe d'un ***maximum de 3 personnes et d'un minimum de 2****. Ce travail doit donner lieu à un rapport, dans le format de votre choix, ainsi qu'à une soutenance. Les dates de soutenances seront communiquées ultérieurement après discussion avec les délégués et l'ADM ; elles auront lieux courant octobre 2019. La première section de cette évaluation n'est pas optionnelle. Pour les sections suivantes il vous faut faire au moins une section. Plus vous attaquez de problèmes meilleure est votre note*[*].

Rendu

Le rendu doit contenir :
— un rapport ;
— l'ensemble des implémentations ;
— les datasets utilisés et les problèmes d'optimisations générés, si applicable ;
— les slides de la soutenance.

Les dates de soutenances seront fixés le 09 septembre. Les contenus hors slides seront à rendre 48 heures avant l'heure de soutenance. Vous êtes libres en ce qui concerne le format de rendu. Votre choix sera jugé quant à son adéquation avec les contraintes de ***clarté***, de ***compréhensibilité*** et d'***éxhaustivité***.

Contraintes techniques

Les implémentations seront à faire en `python`. L'environement `python` permet un prototypage agréable et les bibliothèques de ML qui y sont disponibles vous seront utiles pour faire des comparatifs.

Vous pouvez utiliser les fonctions de `numpy`, `scipy` et de `pandas` liées au calcul différentiel, à l'algèbre linéaire et au pré-traitement et traitement des datasets que vous seriez amenés à manipuler. Tout ce qui n'est pas explicitement mentionné est à proscrire [1].

Attendus

Des ***dessins*** pour résumer les études de performances. On attend de vous des tests fiables, donc en nombre suffisamment important et de préférence avec une estimation du risque. Aucun comparatif ou analyse de performance d'un algo n'a de sens sinon. À vous de vous mettre dans la peau de quelqu'un qui participe par son expertise à une decision importante sur un choix de techno.

Quand on parle de comparaison, on entend batch de tests et résumé des résultats en sortie.

1 Méthode de Newton

Cette section ***n'est pas optionnelle***, ce qui n'est pas en bonus doit être traité par chaque groupe.

[*] Quand c'est bien fait ...

1. Seul les chargés de cours ont droit de vous libérer de cette contrainte. Il vous revient de nous tenir informé de toute contravention à cette règle. Une argumentation sera attendue

1.1 Méthode de Newton

Dans les implémentations que vous abordez il n'est pas nécessaire de traiter le cas d'un point initial non-admissible. Son traitement sera comptabilisé s'il est fait [2]. Sans contraintes cette remarque est bien entendu sans intérêt.

Il vous est autorisé d'utiliser les méthodes de calcul numérique ou symbolique des gradients, hessiennes et les solveurs de systèmes linéaires disponibles dans les bibliothèques `numpy` et `scipy`, si vous en éprouvez le besoin.

1.1.1 Cas sans contraintes

Question 1-1. Générer des problèmes d'optimisations sans contraintes.

Question 1-2. Implémenter et tester une méthode de Newton contre le test set que vous avez généré.

Question 1-3. Comparer votre méthode de Newton aux implémentation de descente de gradient vu en TP.

1.1.2 Cas avec contraintes d'égalités

Question 1-4. Étendre le test set du cas sans contraintes pour générer des problèmes d'optimisation convexes sous contraintes d'égalités.

Question 1-5. Implémenter une méthode de résolution des problèmes générés basées sur une descente de gradient après élémination des contraintes d'égalités et la tester.

Question 1-6. Implémenter la méthode de Newton sous contraintes d'égalités et la tester sur le test set généré.

Question 1-7. Comparer les deux méthodes précédentes.

1.1.3 Bibliographie

Question 1-8. Qu'est-ce qu'une méthode de Quasi-Newton ?

2 Régularisation de problèmes mal posés

Dans cette thématique, on s'intéresse aux méthodes classiques de régularisation de problèmes mal posés (par exemple lorsqu'un problème admet une infinité de solutions). Les problèmes mal posés sont fréquents en *machine learning* (sur-apprentissage) et en traitement du signal/des images (déconvolution, débruitage, in-painting).

2. Tout travail mérite salaire.

Dans le cas d'un problème mal posé, il convient de faire des hypothèses supplémentaires pour *régulariser* la solution (imposer que la solution à un problème de débruitage soit continue par morceaux par exemple)

Le but de cette section est de vous faire manipuler les deux méthodes de régularisation les plus classiques :

— la régularisation de Tikhonov (aussi appelée régularisation ridge en *machine learning*).
— la régularisation LASSO (pour Least Absolute Shrinkage and Selection Operator)

Question 2-9. Rappelez la méthode de résolution analytique des moindres carrés.

Question 2-10. Effectuez une recherche bibliographique sur les méthodes de régularisation de Tikhonov et LASSO. Vous trouverez une multitude de références sur le net. À vous de les synthétiser pour faire ressortir l'idée générale de ces méthodes de régularisation et leurs principales différences.
En particulier, expliquez :

— comment l'ajout des pénalités de Tikhonov et du LASSO modifient la résolution analytique des moindres carrés. Déroulez (et commentez) cette résolution analytique.
— l'influence du choix de la norme et l'interprétation géométrique qui en découle lorsque le problème de régularisation est vu comme un problème de minimisation sous contrainte.

Question 2-11. Implémenter ces deux méthodes de régularisation sur un problème de moindre carrés mal posé de votre choix (originaire du *machine learning* ou du traitement du signal/d'images). Comparer leurs performances et évaluer l'impact du paramètre de régularisation sur la régularité de la solution.

3 Thématiques plus avancées

Les thématiques de cette section sont aux choix, il vous revient de choisir lesquelles et combien d'entre elles vous souhaitez traiter.

3.1 SVM et SMO

Dans cette thématique on cherche à implémenter la ***Sequential Minimal Optimisation*** concernant le traitement des SVMs.

Question 3-12. Expliqer le problème d'optimisation sous-jacent à un ***Support Vector Classifier*** ?

Question 3-13. Implémenter la SMO. Laisser la possibilité de passer le noyau souhaité en argument.

Question 3-14. Tester l'implémentation précédente sur le problème de classification proposé par le dataset ***MNIST***. Dans le but de traiter un problème de classification binaire on se limite au repérage d'un chiffre.
Garder le noyau qui vous donne les meilleurs résultats suivant la métrique de votre choix.

Question 3-15. Implémenter une méthode de résolution par barrière logarithmique d'un ***SVM***. Comparer cette démarche à la SMO.

3.2 Méthode du point intérieur

Cette thématique est plus théorique que les précédentes. La méthode du point intérieur n'a pas été abordée en cours ; elle est évaluée en conséquence.

Le but est de comparer la performance de la méthode du point intérieur à celle du simplexe.

Question 3-16. Expliquer la méthode primal-dual du point intérieur exposée dans [BV04, 11.7].

Question 3-17. En donner une implémentation en présupposant que l'origine est toujours un point admissible [a].

a. Il y a des méthodes standards pour se ramener à ce cas.

Question 3-18. Comparer la performance de la méthode précédente à une méthode du simplexe de votre cru. [a].

a. Bien entendu, dans le cas des programmes linéaires.

Question 3-19. Appliquer les méthodes de résolution précédentes au cas des flots maximaux sur les graphes.

Références

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.