ENSICAEN
2ème année Informatique

TP Optimisation stochastique
PageRank

1 Introduction

Dans la suite, on notre $\langle \cdot, \cdot \rangle$ le produit scalaire euclidien sur $\mathbb{R}^n$, $\|\cdot\|$ la norme euclidienne. Pour $x \in \mathbb{R}^n$, on notera $x[i]$ sa i-ème entrée.

Comme vu en 1A et en 21, PageRank est une approche pour calculer les scores d'importance des pages dans le web vu comme un graphe orienté non-valué. Le classement des pages suite à une requête sur le moteur de recherche suivra alors ces scores d'importance. PageRank prend en compte compte dans le calcul du score d'une page tous les scores des pages des pages incidentes et le nombre de liens sortant de chaque page. Calculer le vecteur des score dans PageRank revient à résoudre

$$\text{Trouver } x \geq 0 \text{ tel que } \quad Mx = x \quad \text{et} \quad \sum_{i=1}^{n} x[i] = 1, \tag{1}$$

où

$$M = (1-\alpha)A + \alpha/n, \quad \alpha \in]0, 1].$$

La valeur de α originellement utilisée par Google est 0.15. $A \in [0,1]^{n \times n}$ est la matrice des liens qui est stochastique en colonnes. Observons que M est aussi stochastique en colonnes. Le choix de remplacer A par M assure l'unicité des scores par le théorème de Perron-Frobenius, et que tous ces scores, une fois normalisés à une somme unitaire, seront eux-mêmes strictement positifs.

Nous avons vu en 2A que (1) est équivalent à résoudre le problème d'optimisation suivant

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Mx - x\|^2 + \frac{\lambda}{2} \left(\sum_{i=1}^{n} x[i] - 1 \right)^2, \tag{2}$$

où $\lambda > 0$ est un paramètre de pénalité pour la contrainte de normalisation $\sum_{i=1}^{n} x[i] = 1$, et $\|\cdot\|$ est la norme Euclidienne. La contrainte de positivité a été enlevée puisque par le théorème de Perron-Frobenius, il n'existe qu'une seule solution de $Mx = x$ avec $\sum_{i=1}^{n} x[i] = 1$ et ses entrées sont strictement positives. Ceci implique aussi l'unicité du minimiseur, qui est donc le vecteur de scores de Perron, et la valeur minimale est 0.

Dans la suite, on se propose d'appliquer une descente de gradient stochastique pour minimiser f. On s'appuiera pour ceci sur les résultats du TD 3.

2 Travail préparatoire

1. Montrer que résoudre (2) revient à minimiser l'objective f qui s'écrit sous la forme

$$f(x) = \frac{1}{2(n+1)} \|\mathcal{A}x - b\|^2 .$$

   Donner les expressions de $\mathcal{A} \in \mathbb{R}^{n \times n}$ et de $b \in \mathbb{R}^n$.
2. En déduire que $f(x) = \frac{1}{n+1} \sum_{i=1}^{n+1} f_i(x)$. Donner en particulier l'expression de $f_i(x)$ en fonction des colonnes a_i de $A^\top$.
3. Calculer $\nabla f_i(x)$ et sa constante de Lipschitz β_i. Calculer $\beta = \max_i \beta_i$.

On rappelle que l'itération du SGD avec minibatch de taille 1 est

$$\begin{cases} \text{Tirer } i_k \text{ aléatoirement uniformément avec remise dans } \{1, \dots, n+1\}, \\ x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k), \end{cases}$$

où $f_{i_k}(x)$ est celui calculé lors du travail préparatoire. On suppose que $x_0 \in \mathrm{Im}(\mathcal{A})$ (par exemple le vecteur 0).

3 Travail d'implémentation

Télécharger le notebook Jupyter `pagerank_sgd.ipynb` sur Moodle. Il fournit l'ossature du code sur lequel vous allez vous baser par la suite. Il contient plusieurs fonctions utilitaires ainsi qu'une fonction qui permet de générer aléatoirement A correspondant à un miniweb aléatoire. Ce code prend en argument la taille n, la densité dans $[0, 1]$, et un booléen pour dire si A est générée en mode `sparse` ou en mode `full`. Il utilise le package `sparse` dans `scipy`.

1. Reprendre votre code de la descente de gradient implémentée en 2A. Bien s'assurer que le gradient est calculé de manière efficace. Tester le sur le miniweb 1 par exemple et tracer $\|x_k - x^\star\|$ et $f(x_k)$ en fonction de k.

2. Implémenter le SGD ci-dessus qui prend comme arguments, entre autres, une matrice de liens A, les paramètres α et λ, et un choix de pas γ_k. Ce code retourne la suite $f(x_k)$ et $\|x_k - x^\star\|$. $x^\star$ est le vecteur de score trouvé précédemment, par exemple en utilisant `eig`. Prendre soin d'implémenter le calcul du gradient de manière efficace (pas d'auto-différentiation à ce stade) sans construire des variables intermédiaires (matrices, vecteurs) inutiles.
3. Nous avons vu en TD 3 que $\gamma_k \equiv \gamma \in]0, 2/\beta[$ assure la convergence p.s. de l'objective f et des itérées. Prendre les matrices A des miniwebs testés en 2A et tester votre code avec $\gamma = 1.99/\beta$, $\lambda = 1/n$ et $\lambda = 1/n^2$. Tracer $\|x_k - x^\star\|$ et $f(x_k)$ en fonction de k.
4. Commenter les résultats obtenus et comparer à la descente de gradient.
5. A quelle vitesse décroit $\|x_k - x^\star\|$. Y a-t-il un meilleur choix du pas γ pour accélérer la convergence (servez-vous du résultat du TD 3) ? Quel est le principal inconvénient de ce choix optimal ?
6. Reprendre les questions 2 à 4 avec le pas variable $\gamma_k = c/\sqrt{k+1}$ et $\gamma_k = c\log(k+1)/\sqrt{k+1}$, où $c > 0$ est une constante. Tester différentes valeurs de c et commenter les résultats obtenus. Est-ce un bon choix de pas ?
7. Quelle est la complexité du SGD à chaque itération ? Comparer à la descente de gradient.
8. Vous allez désormais tester votre code sur de larges miniwebs qui exploitent la structure creuse de A. En effet, stocker A devrait occuper seulement $O(|E|)$ en mémoire, où $|E| \ll n^2$ est le nombre d'arcs (de liens) dans le graphe. Par ailleurs, le coût en multiplications nécessaires après une passe sur tous les sommets (pages) sera en $O(|E| + n)$.
   a) Reprendre les questions 2-4 avec A de taille 100 générée aléatoirement en mode `full`.
   b) Pour n allant de 100 à 2000 avec un pas de 100, tracer le temps de calcul moyen par itération pour GD et SGD avec A générée en `sparse` et en `full`. Discuter et comparer à la complexité théorique.
   c) Quelle serait la complexité si A est dense (voir question 6) ?
9. Facultatif : Refaire les questions 1-3 en testant d'autres tailles de minibatch. Attention, il faudra dans ce cas ajuster le choix du pas de descente γ qui dépend de la taille des minibatch (il suffit pour cela de reprendre le calcul en TD).