ENSAE

Geometric Methods in Machine Learning

SE314

Low rank factorization of kernel matrices

Auteur :
Julien Perrin

20 mai 2018

Introduction

A tout noyau $K(\cdot,\cdot)$ on associe une matrice $G \in \mathbb{R}^{n\times n}$ définie par $G_{ij} = K(x_i, x_j)$ où $x_i, x_j \in \mathbb{R}^d$ sont des données. Le principal problème que l'on rencontre lorsqu'on cherche à utiliser les noyaux à grande échelle est qu'il s'agit d'une méthode très gourmande en temps ($\mathcal{O}(n^2 d)$) et en mémoire ($\mathcal{O}(n^2)$). On cherche donc à approximer ces matrices de la façon la plus précise possible en économisant des ressources.
On s'intéresse particulièrement aux noyaux invariants par translation c'est à dire de la forme $K(x_i, x_j) = f(\eta(x_i - x_j))$ où f est une fonction de $\mathbb{R}^d$ dans $\mathbb{R}$ et η est une constante positive représentant "l'échelle" des données.
Il existe deux type de structures pour les matrices de noyaux : les matrices à faible rang (quand η est petit), dites *low-rank* et les matrices par blocs (η grand). Il existe des méthodes d'approximations performantes pour chaque type de structure.

1 Approximation *low-rank*

La méthode d'approximation low-rank la plus performante est la décomposition par valeurs singulières (**SVD**) qui consiste à approximer G par

$$\tilde{G} = U_k \Sigma_k U_k^T$$

où Σ_k est la matrice diagonale des k plus grandes valeurs singulières et U_k la matrice des vecteurs singuliers correspondants.
Néanmoins cette méthode n'est pas généralisable à grande échelle car elle est trop gourmande en temps et en mémoire.

Pour contourner ce problème, on utilise l'approximation de **Nyström**.

- On tire uniformément m points de données puis on sélectionne les m colonnes de G correspondantes pour construire une matrice C de dimensions $n \times m$. On appelle M la matrice du noyau baséee sur ces m points de données.
- On applique alors la décomposition SVD au rang k à la matrice M pour obtenir M_k dont on calcule la matrice pseudo-inverse M_k^+.
- On approxime alors G par :

$$\tilde{G} = C M_k^+ C^T$$

2 Approximation par bloc

A l'opposé, quand le paramètre d'échelle η du noyau est grand, les blocs non diagonaux de la matrice du noyau vont devenir petits et la majorité de l'information va se concentrer sur la diagonale, on va alors utiliser la méthode d'approximation du noyau par bloc (**BKA**).
On considère une partition $V_1, ..., V_c$ une partition des données. Alors la méthode BKA approxime la matrice G par la matrice diagonale par blocs dont les éléments sont les blocs diagonaux $G^{(s,s)}$ de G restreint à la partition V_s.

$$G \simeq \tilde{G} = \begin{pmatrix} G^{(1,1)} & 0 & \ddots \\ 0 & \ddots & \ddots \\ \ddots & \ddots & G^{(c,c)} \end{pmatrix}$$

Pour trouver la meilleur partition, on cherche à maximiser la quantité :

$$D^{Kernel}(\{V_s\}_{s=1}^{c} = \sum_{s=1}^{c} \frac{1}{|V_s|} \sum_{i,j \in V_s} K(x_I, x_j)^2$$

Cette méthode étant un peu gourmande en calculs on peut également procéder à un clustering *kmeans*.
La méthode BKA présente deux inconvénients : elle ignore les blocs non diagonaux ce qui la rend inefficace pour les matrices low-rank et elle coûte trop cher en calcul et en stockage lorsque l'échelle est trop grande.

3 Approximation du noyau efficace en mémoire

En remarquant que tous les blocs obtenus après une partition kmeans auront un range petit, on peut définir la méthode Memory Efficient Kernel Approximation (**MEKA**) qui consiste à

- Effectuer un clustering Kmeans sur les données et réarranger la matrice du noyau d'après les clusters identifiés.
- Calculer l'approximation low-rank uniquement pour les blocs diagonaux puis s'en servir pour approximer les blocs non-diagonaux.

Détaillons la deuxième étape.
L'approximation low-rank au rang k est donnée par $W^{(s)} L^{(s,s)} W^{(s)T}$.

L'approximation du noyau est : $\tilde{G} = WLW^T$ où

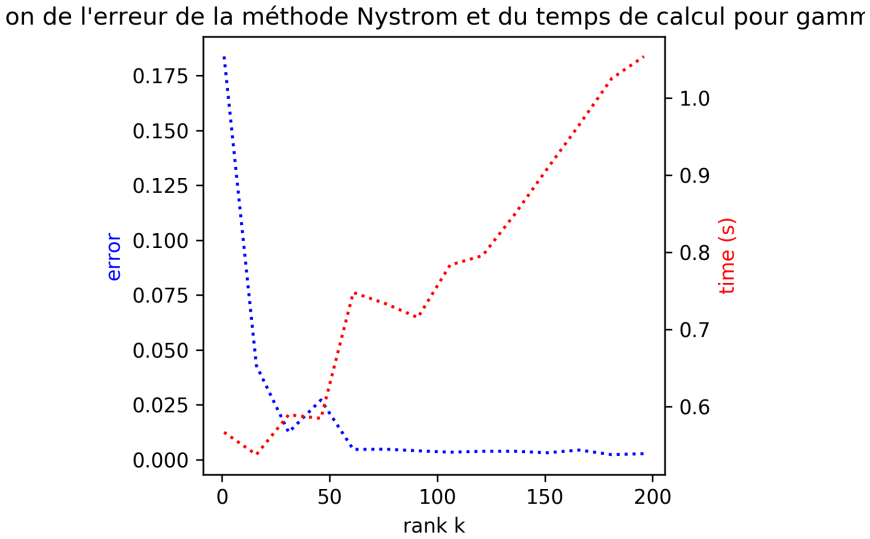
$$W = \begin{pmatrix} W^{(1)} & 0 & \ddots \\ 0 & \ddots & \ddots \\ \ddots & \ddots & W^{(c)} \end{pmatrix}$$

et L est une matrice de de c^2 blocs $L^{(s,t)}$ qui représentent les interactions entre les clusters s et t.
On commence par approximer Les blocs diagonaux à l'aide la méthode de Nyström ce qui permet aussi de calculer les $W^{(s)}$. On calcule les $L^{(s,t)}$ en minimisant l'erreur d'approxiation locale $||G^{(s,t)} - W^{(s)}L^{(s,s)}W^{(st)T}||$. Pour gagner du temps, on effectue la minimisation sur une sous matrice $\hat{G}^{(s,t)}$ de $G^{(s,t)}$ obtenue en sélectionnant aléatoirement v_s lignes et v_t colonnes, on a alors :

$$L^{(s,t)} = \left(W_{vs}^{(s)T} W_{vs}^{(s)}\right)^{-1} W_{vs}^{(s)T} \hat{G}^{(s,t)} W_{vt}^{(t)T} \left(W_{vt}^{(t)} W_{vt}^{(t)T}\right)^{-1}$$

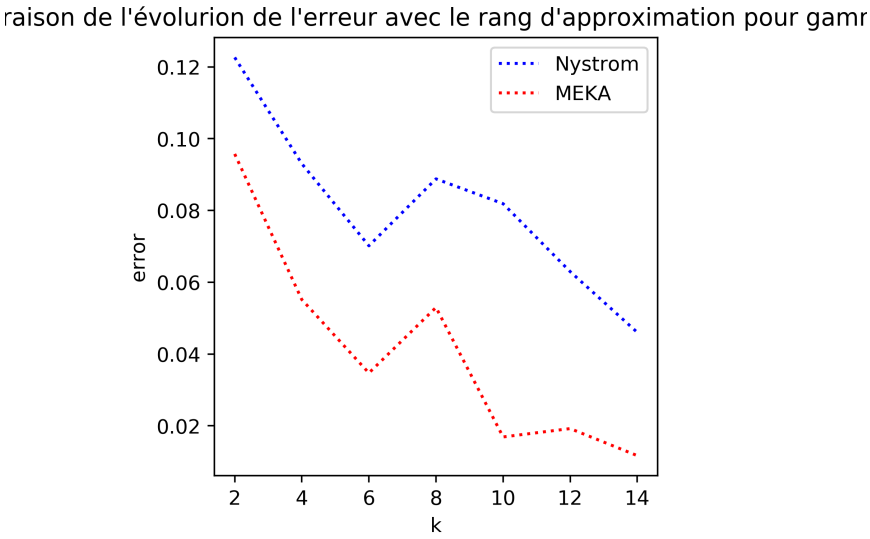
4 Résultats

On met en évidence le compromis à trouver entre performance et temps de calcul dans la méthode de Nyström.

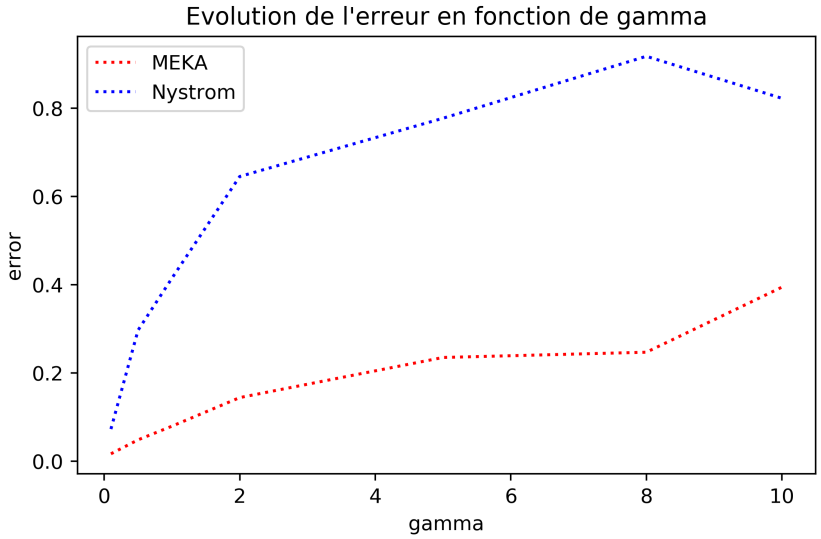


On compare la méthode de Nyström avec MEKA lorsqu'on fait évoluer le

rang d'approximation :



On remarque que logiquement plus on choisit un rang d'approximation élevé plus la performance est bonne pour les deux méthodes.
Enfin on fait varier le paramètre d'échelle du noyau pour pouvoir comparer les deux méthodes d'approximation.



La méthode de Nystrom étant une méthode low-rank, elle n'est efficace que pour les matrices de faible rang ce qui correspond aux cas où le paramètre d'échelle γ est petit comme on peut le voir sur la courbe. La méthode MEKA est plus construite comme une métohde hybride et donc parvient à obtenir de bonnes performances à la fois pour les γ faibles qu'importants.