Parallélisation de l'algorithme des k-médoïdes. Application au clustering de courbes.

Benjamin Auder [1] & Jairo Cugliari [2]

[1] *Laboratoire LMO. Université Paris-Sud. Bât 425. 91405 Orsay Cedex, France.*
benjamin.auder@math.u-psud.fr
[2] *Laboratoire ERIC. Université Lumière Lyon 2. Bât K. 69676 Bron Cedex, France.*
jairo.cugliari@univ-lyon2.fr

Résumé. Nous présentons une méthode de clustering adaptée à de grandes bases de données de courbes densément discrétisées, donc elle-mêmes en grande dimension. La classification obtenue peut être utilisée afin de choisir de meilleurs modèles de prédiction de courbes, plus spécifiques, dans chacun des groupes. La méthode consiste en deux principales étapes : réduire la dimension des courbes via une base d'ondelettes, puis effectuer le clustering en parallèle. Les ondelettes sont bien adaptées pour identifier des caractéristiques localisées en temps et échelle. De plus, l'aspect multi-résolution de la transformée en ondelettes permet de regrouper les coefficients selon leur contribution à l'énergie totale, fournissant ainsi des représentants compacts pour chaque courbe. L'algorithme des k-médoïdes est appliqué à plusieurs sous-échantillons de l'ensemble des représentations réduites, puis les centres finaux sont obtenus en recherchant la médiane (ou éventuellement la moyenne) des médoïdes des échantillons. Des applications sont présentées sur deux jeux de données, dont les consommations électriques irlandaises journalières.

Mots-clés. réduction de dimension, ondelettes, k-médoïdes, parallèle

Abstract. We present a clustering method adapted to large databases of densely sampled curves, hence themselves in high dimension. The resulting classification can help to better tune models to predict new curves, each in every group. The method consists of two main steps: use a wavelet basis to reduce curves dimensionality, and then run a parallel clustering algorithm. Wavelets are well suited to identify characteristics localized both in time and space. Moreover, the multiresolution nature of the wavelets transform allows to group coefficients according to their contribution to the total energy, thus providing compact representations for every curve. The k-medoids algorithm is applied to several subsamples of all the reduced representations, and then the final centers are obtained by computing the median (or mean) of the samples medoids. Applications are shown on two databases, including Irish daily electrical consumptions.

Keywords. dimensionality reduction, wavelets, k-medoids, parallel

1 Introduction

Récemment, le contexte du *Big Data* a rendu nécessaire le développement d'algorithmes spécifiques à de (très) grands volumes de données, éventuellement aussi en grande dimension comme c'est le cas dans notre étude. Ainsi ont vu le jour des algorithmes opérant sur de grands graphes (Kang et al. 2009) ou sur des flux de données haut débit (De Francisci Morales et Bifet 2013), entre entres. Le livre de Bekkerman et al. (2011) présente des algorithmes de Machine Learning s'exécutant en parallèle sur diverses architectures, et ce type de programmation est en plein essor.

La classification non supervisée (*clustering*) est une des branches de l'apprentissage non supervisé. L'objectif est de regrouper les données en *clusters* homogènes, suffisamment distincts deux à deux. Depuis la proposition originale de Tyron (1939) un grand nombre d'articles ont été publié à ce sujet (voir Berkhin 2006 pour une revue). Cependant, il n'y a pas de consensus sur la définition d'un cluster : celle-ci varie en fonction des données, du contexte et de l'algorithme utilisé. Malgré ce fait, la classification non supervisée reste une technique très populaire qui permet de réduire la taille des données en les résumant à quelques représentants ; c'est particulièrement intéressant lorsqu'on travaille sur de grosses bases de données.

Dans ce travail nous avons choisi d'adapter un algorithme de clustering classique pour une exécution parallèle, opérant sur des données de dimension réduite. La première partie présente la technique retenue afin d'abaisser la dimension via une base d'ondelettes. L'algorithme classant les données en parallèle est expliqué dans la seconde partie, puis nous montrons quelques résultats de tests.

2 Réduction de dimension

Dans cette section nous expliquons comment sont traitées les variables fonctionnelles (voir Antoniadis et al. (2013) pour plus de détails).

Chaque fonction z est d'abord discrétisée sur une fine grille (toutes les 30 minutes pour les consommations électriques), et la discrétisation est encore notée $z = \{z(t_i), i = 1, \ldots, N\}$. z est alors approchée (jusqu'à l'échelle J) par un développement tronqué sur une base d'ondelettes avec un coefficient d'échelle $c_{0,0}$ et les coefficients d'ondelettes $\{d_{j,k}\}_{j,k}$:

$$z_J(t) = c_{0,0}\phi_{0,0}(t) + \sum_{j=0}^{J-1}\sum_{k=0}^{2^j-1} d_{j,k}\psi_{j,k}(t). \tag{1}$$

Alors, l'énergie $\mathcal{E}_Z = \|z\|_{L_2}^2$ de chaque fonction z est égale à la somme des énergies de ses

coefficients en ondelettes distribuées à travers les échelles :

$$\mathcal{E}_z \approx c_{0,0}^2 + \sum_{j=0}^{J-1} \|\mathbf{W}_j\|_2^2, \tag{2}$$

où $\|\mathbf{W}_j\|_2^2 = (d_{j,0}, \dots d_{j,2^j-1})'$, l'approximation étant due à la troncature à l'échelle J. On ne s'intéresse qu'à l'information contenue dans la forme des fonctions, pas à celle de son niveau moyen. Ainsi, on définit la contribution absolue de l'échelle j à l'énergie globale de la fonction centrée comme

$$\text{cont}_j = \|\mathbf{W}_j\|_2^2, \qquad j = 0, \dots, J-1.$$

Finalement, chaque fonction est caractérisée par le vecteur de ses contributions absolues à l'énergie globale.

En pratique nous choisissons J de l'ordre de $\log_2(N)$: c'est la valeur maximale possible compte-tenu de la discrétisation.

3 k-médoïdes parallèle

Nous rappelons tout d'abord l'algorithme des k-médoïdes, puis indiquons comment s'est déroulée la parallélisation.

Algorithme des k-médoïdes
Comme dans l'algorithme des k-means, l'idée est de partitionner les données autour de k centres, ici appelés *médoïdes*. La fonction à minimiser (*distorsion*) est similaire :

$$D(x) = \min_{m_1,\dots,m_k \in \mathbb{R}^d} \sum_{i=1}^{n} \min_{j=1,\dots,k} \|x_i - m_j\|,$$

avec $x = (x_1, \dots, x_n)$, $\|\,.\,\|$ pouvant désigner n'importe quelle norme ; ici nous choisissons la norme euclidienne. Les distances n'apparaissant pas au carré, cet algorithme est moins sensible aux outliers que le k-means. Cependant, sa programmation est plus délicate car dans le cas de la médiane spatiale il n'existe pas de formule analogue à celle de la moyenne. La première implémentation de l'algorithme est celle de Kaufman et Rousseeuw (1987), PAM pour Partitioning Around Medoids. Elle est assez coûteuse, mais donne de bons résultats et reste simple à écrire. Diverses heuristiques ou accélérations autour de cette implémentation ont été développés (Chu et al. 2002, etc.), mais nous utilisons ici la version originale, qui se révèle suffisante.

Parallélisation avec MPI
MPI est une librarie permettant de faciliter l'écriture de programmes en parallèle. Nous

l'utilisons en mode master-slave, c'est à dire que le travail est divisé en deux types d'entités: un processus coordinateur, et des coeurs se contentant d'exécuter les demandes du processus maître. Voici le flot d'exécution du programme final :

1. le processus maître découpe le problème en sous-tâches sur des jeux de données plus petits, et les envoie à chaque esclave ;
2. pour une sous-tâche donnée, on réduit la dimension puis applique l'algorithme de clustering, et retourne les centres ;
3. le processus principal récupère et aggrège les résultats en un jeu de k centres.

Le code sera disponible prochainement sur github.

4 Applications

Deux jeux de données ont été utilisés ici : des évolutions de magnitudes d'étoiles, et de consommations électriques.

4.1 Magnitudes d'étoiles

Ce jeu de données récupéré via le site web de Keogh et al. (2011) consiste en un ensemble d'entraînement de 1000 courbes, et une base de test de 8236 courbes. Chacune d'entre elle a un label compris entre 1 et 3 ; la figure 4.1 les représente. On remarque que les groupes 1 et 3 sont assez similaires, contenant des courbes difficiles à classer.

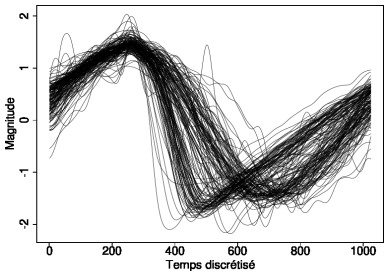

Figure 1: Groupe 1

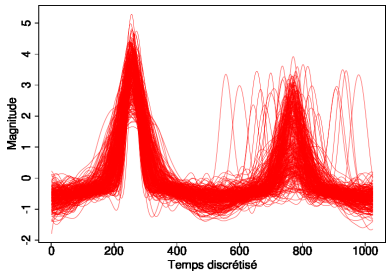

Figure 2: Groupe 2

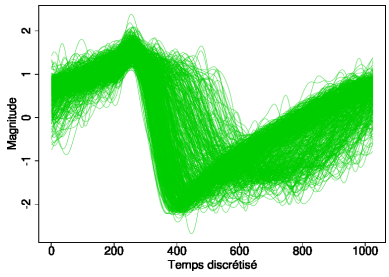

Figure 3: Groupe 3

Compte-tenu du relatif faible nombre d'échantillons nous pouvons lancer le programme sur tout le jeu de données ; cela permet de comparer aux résultats obtenus par la version parallèle, que l'on espère presque aussi bons. Le tableau 1 contient les distorsions empiriques calculées sur les ensemble d'entraînement et de test, ainsi que l'adéquation des deux partitions ("intra"), et leur adéquation à la partition réelle ("inter"). Cette dernière est mauvaise à cause de la remarque précitée : les courbes des clusters 1 et 3 se ressemblent trop pour appliquer un algorithme de partitionnement de type k-means.

	Distorsion	"Intra-adéquation"	"Inter-adéquation"
Entraînement seq.	1.31e4	0.79	0.77
Entraînement //	1.40e4	0.79	0.68
Test seq.	1.09e5	0.78	0.76
Test //	1.15e5	0.78	0.69

Table 1: Distorsions et indices d'adéquation des partitions

4.2 Consommations électriques irlandaises

Ce jeu de données consiste en 4621 séries temporelles représentant l'évolution de la consommation életrique d'autant de foyers irlandais. Celles-ci sont similaires aux courbes EDF sur lesquelles vise à être appliquée notre méthode. Nous avons choisi d'appliquer l'algorithme avec 3 et 5 classes. Compte-tenu du grand nombre de points de discrétisation (25k) et de la haute variabilité des données, celles-ci sont difficiles à représenter. Ainsi nous n'indiquons que les résultats numériques ; ils sont visibles dans le tableau 2, et sont cette fois plutôt bons.

	Distorsion	"Intra-adéquation" seq. VS //
3 clusters seq.	1.90e7	0.90
3 clusters //	2.15e7	0.90
5 clusters seq.	1.61e7	0.89
5 clusters //	1.84e7	0.89

Table 2: Distorsions et indices d'adéquation des partitions

5 Conclusion

Nous avons cherché à identifier des groupes de consommateurs à partir de données fonctionnelles pour bénéficier de l'information supplémentaire fournie par la forme des courbes, qui serait perdue si on les considérait comme de simples vecteurs. Cette forme est capturée − entre autres caractéristiques − par les coefficients d'ondelettes, regroupés par niveaux d'énergie. Les dissimilarités sont calculées ensuite via une distance L^p. Alternativement (ou complémentairement), il serait intéressant d'utiliser une mesure de dissimilarité plus directement basée sur la forme des courbes, comme le font Heckman et Zamar (2000).

Ensuite, la procédure de clustering mise en place est prévue pour passer à l'échelle pour plusieurs millions de courbes. En effet, elle a été conçue initialement pour être appliquée aux séries temporelles des clients EDF (plusieurs dizaines de millions). Elle consiste à appliquer l'algorithme des k-médoïdes sur des groupes de courbes, puis des groupes de

médoïdes jusqu'à obtenir un seul ensemble traité sur un processseur. Les résultats obtenus sur les deux jeux de données présentés sont assez encourageants, et permettent d'envisager une utilisation à plus grande échelle.

Bibliographie

[1] A. Antoniadis, X. Brossat, J. Cugliari et J.-M. Poggi (2013), Clustering Functional Data Using Wavelets, *International Journal of Wavelets, Multiresolution and Information Processing*, 11(1), 35–64.

[2] R. Bekkerman, M. Bilenko et J. Langford - éditeurs (2011), Scaling up Machine Learning: Parallel and Distributed Approaches, *Cambridge University Press.*

[3] P. Berkhin (2006), A Survey of Clustering Data Mining Techniques, *Grouping Multidimensional Data, éditeurs : J. Kogan, C. Nicholas, M. Teboulle.*

[4] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, et R.E. Gruber (2006), Bigtable: A Distributed Storage System for Structured Data, *Seventh Symposium on Operating System Design and Implementation.*

[5] S.-C. Chu, J.F. Roddick et J.S. Pan (2002), An Efficient K-Medoids-Based Algorithm Using Previous Medoid Index, Triangular Inequality Elimination Criteria, and Partial Distance Search, *Lecture Notes in Computer Science*, 2454, 63–72.

[6] J. Dean et S. Ghemawat (2004), MapReduce: Simplified Data Processing on Large Clusters, *Sixth Symposium on Operating System Design and Implementation.*

[7] G. De Francisci Morales et A. Bifet (2013), Introducing SAMOA, an open source platform for mining big data streams, *http://yahooeng.tumblr.com/post/65453012905/introducing-samoa-an-open-source-platform-for-mining.* [8] N.E. Heckman et R.H. Zamar (2000), Comparing the shapes of regression functions. *Biometrika*, 87(1), 135–144.

[9] U. Kang, C.E. Tsourakakis et C. Faloutsos (2009), PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations, *IEEE International Conference on Data Mining.*

[10] L. Kaufman et P.J. Rousseeuw (1987), Clustering by means of Medoids, *Statistical Data Analysis Based on the L_1-Norm and Related Methods, éditeur : Y. Dodge.*

[11] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei et C.A. Ratanamahatana (2011), The UCR Time Series Classification/Clustering, *http://www.cs.ucr.edu/~eamonn/time_series_data/.*

[12] R.C. Tryon (1939), Cluster analysis. *New York: McGraw Hill.*