Clustering topologique pour le flux de données

Mohammed Ghesmoune*, Mustapha Lebbah*, Hanene Azzag*

*Université Paris 13, Sorbonne Paris Cité
LIPN-UMR 7030 - CNRS
99, av. J-B Clément – F-93430 Villetaneuse, France
prénom.nom@lipn.univ-paris13.fr

Résumé. Actuellement, le clustering de flux de données devient le moyen le plus efficace pour partitionner un très grand ensemble de données. Dans cet article, nous présentons une nouvelle approche topologique, appelée G-Stream, pour le clustering de flux de données évolutives. La méthode proposée est une extension de l'algorithme GNG (Growing Neural Gas) pour gérer le flux de données. G-Stream permet de découvrir de manière incrémentale des clusters de formes arbitraires en ne faisant qu'une seule passe sur les données. Les performances de l'algorithme proposé sont évaluées à la fois sur des données synthétiques et réelles.

1 Introduction

Un flux de données est une séquence, potentiellement infinie, non-stationnaire (la distribution de probabilité des données peut changer au fil du temps) de données arrivant en continu. Dans le cas d'un flux, l'accès aléatoire aux données n'est pas possible et le stockage de toutes les données arrivant est infaisable. Le clustering de flux de données nécessite un processus capable de partitionner des observations de façon continue avec des restrictions au niveau de la mémoire et du temps. Dans la littérature, de nombreux algorithmes de clustering de flux de données ont été adaptés à partir des algorithmes de clustering traditionnel, par exemple, la méthode DbScan (Cao et al. (2006); Isaksson et al. (2012)) basée sur la densité, la méthode de partitionnement *k*-means (Ackermann et al. (2012)), ou encore la méthode basée sur le passage de message AP (Affinity Propagation) (Zhang et al. (2008)). Dans cet article, nous proposons le modèle G-Stream, qui permet de découvrir des clusters de formes arbitraires dans un flux de données en constante évolution. Les caractéristiques et les principaux avantages de G-Stream sont décrits ci-dessous : (a) La structure topologique qui est représentée par un graphe dans lequel chaque nœud représente un cluster. Les nœuds (clusters) voisins sont reliés par des arêtes. La taille du graphe est évolutive. (b) L'utilisation d'une fonction d'oubli afin de réduire l'impact des anciennes données dont la pertinence diminue au fil du temps. Les liens entre les nœuds sont également pondérés. (c) Contrairement à de nombreux algorithmes qui utilisent un nombre important de données pour initialiser leur modèle, G-Stream utilise seulement deux nœuds au départ. (d) Toutes les fonctions de G-Stream sont effectuées en-ligne. (e) L'utilisation de la notion de réservoir pour maintenir, de façon temporaire, les données très éloignées

des prototypes courants. L'article est organisé comme suit : d'abord, la section 2 décrit plusieurs travaux liés au problème de clustering de flux de données. Ensuite, la section 3 présente notre nouvelle approche de clustering de flux de données, appelée G-Stream. Puis, dans la section 4, nous rapportons une évaluation expérimentale. Enfin, la section 5 conclut cet article et présente nos futurs travaux de recherche.

2 Travaux similaires

Cette section présente un bref état de l'art qui concerne les problèmes de clustering de flux de données. Nous mettons ainsi en évidence les algorithmes les plus pertinents proposés dans la littérature pour faire face à ce problème. La plupart des algorithmes existants (par exemple, *CluStream* proposé par (Aggarwal et al. (2003)), *DenStream* de (Cao et al. (2006)), *StreamKM++* de (Ackermann et al. (2012)) divisent le processus de clustering en deux phases : (a) *En-ligne*, dans cette phase, les données sont résumées, (b) *Hors-ligne*, dans cette phase, les clusters finaux sont calculés à partir de la quantification fournie par la phase *en-ligne*. Les deux algorithmes *CluStream* et *DenStream* utilisent une extension temporelle du *Clustering Feature vector* proposée par (Zhang et al. (1996)) (appelée *micro-clusters*) afin de maintenir des résumés statistiques sur les données ainsi que leur temps d'arrivée, ceci durant la phase en-ligne. En créant deux types de micro-clusters (*potentiel* et *outlier micro-clusters*), *DenStream* surmonte l'un des principaux inconvénients de *CluStream*, sa sensibilité au bruit. Dans la phase hors-ligne, les micro-clusters trouvés lors de la phase en-ligne sont considérés comme des *pseudo-points* et seront transmis à une variante de *k*-means dans l'algorithme *CluStream* (resp. une variante de DbScan dans l'algorithme *DenStream*), afin de déterminer les clusters finaux. *StreamKM++* est une extension de l'algorithme *k-means++* pour le flux de données. Les auteurs de (Isaksson et al. (2012)) ont proposé *SOStream*, qui est un algorithme de clustering de flux de données, basé sur la densité, inspiré à la fois du principe de l'algorithme DbScan et celui des cartes auto-organisatrices (SOM) de (Kohonen et al. (2001)). L'algorithme *E-Stream*, qui est proposé par (Udommanetanakit et al. (2007)), classe l'évolution des données en cinq catégories : apparition, disparition, auto-évolution, fusion et division. Il utilise une autre structure de données pour sauvegarder des statistiques sommaires, nommée α-bin histogramme. (Zhang et al. (2008)) présentent une extension de l'algorithme Affinity Propagation pour le flux de données, appelé *StrAP* et qui utilise un réservoir pour maintenir d'éventuels outliers. Les auteurs de (Bouguelia et al. (2013)) ont proposé une version incrémentale de l'algorithme GNG de (Fritzke (1994)), appelée *AING*.

3 Growing Neural Gas pour le flux de données

Dans cette section, nous introduisons un nouveau modèle, appelé G-Stream, à base de l'approche Growing Neural Gas (GNG) de (Fritzke (1994)) en adaptant ce dernier pour le flux de données. L'algorithme GNG est une approche auto-organisatrice incrémentale appartenant à la famille des cartes topologiques telles que les cartes auto-organisatrices (SOM) de (Kohonen et al. (2001)) ou Neural Gas (NG) de (Martinetz et Schulten (1991)). On suppose que le flux de données consiste en une séquence (potentiellement infinie) $\mathcal{DS} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ de n données arrivant en temps $t_1, t_2, ..., t_n$, où $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^d)$ est un vecteur dans l'espace

$\Re^d$. A tout moment, G-Stream est représenté par un graphe $\mathcal{C}$ où chaque nœud représente un cluster. Chaque nœud $c \in \mathcal{C}$ possède un prototype $\mathbf{w}_c = (w_c^1, w_c^2, ..., w_c^d)$ (resp. un seuil de distance δ_c) représentant sa position. A l'étape d'initialisation, uniquement deux nœuds sont créés. A l'arrivée d'une nouvelle donnée, le nœud le plus proche et le deuxième nœud le plus proche sont identifiés, liés par une arête, et le nœud le plus proche avec ses voisins topologiques sont déplacés vers la donnée. Chaque nœud a une variable calculant l'erreur cumulée. Il possède aussi une autre variable représentant son poids qui varie au fil du temps en utilisant la fonction d'oubli. En appliquant la procédure de gestion des arêtes, un, deux ou trois nœuds sont insérés dans le graphe entre les nœuds ayant les plus grandes valeurs de l'erreur cumulée. Les nœuds peuvent également être supprimés.

Fonction d'oubli : dans la plupart des scénarios de flux de données, les données les plus récentes peuvent refléter l'émergence de nouvelles tendances ou des changements dans la distribution des données (de Andrade Silva et al. (2013)). La fonction d'oubli $f_1(t) = 2^{-\lambda_1(t-t_0)}$ où $\lambda_1 > 0$, qui est une constante définissant le taux de décroissance du poids au fil du temps. t désigne le temps courant et t_0 est le temps d'arrivée de la donnée. Le poids d'un nœud est calculé à partir des poids des données qui lui sont affectées : $poids(c) = \sum_{i=1}^{m} 2^{-\lambda_1(t-t_{i_0})}$ où m est le nombre de données affectées au nœud c au temps courant t. Quand le poids d'un nœud est inférieur à une valeur donnée, alors ce nœud est considéré comme obsolète et sera supprimé (ainsi que ses liens).

Gestion des arêtes : la procédure de gestion des arêtes effectue des opérations liées à la mise à jour des arêtes du graphe (les étapes 14-19 de l'algorithme 1). Lors de l'incrémentation de l'âge des arêtes, l'instant de création d'une arête est pris en compte. Contrairement à la fonction d'oubli, l'âge des liens sera renforcé par la fonction exponentielle $f_2(t) = 2^{\lambda_2(t-t_0)}$ où $\lambda_2 > 0$, définit le taux de croissance au temps courant t, t_0 est le temps de création de l'arête. L'étape suivante consiste à ajouter une nouvelle arête reliant les deux nœuds les plus proches. La dernière étape consiste à supprimer chaque lien dépassant un âge maximum.

Gestion du réservoir : l'objectif de l'utilisation d'un réservoir est de maintenir, temporairement, les données éloignées. Comme nous l'avons mentionné précédemment, chaque nœud a un seuil de distance. Les premières données du flux sont affectées aux nœuds les plus proches sans prendre en considération les seuils de distances. Le seuil de distance de chaque nœud est mis-à-jour en prenant la distance maximale du nœud au point le plus éloigné qui lui est affecté. Lorsque le réservoir est plein, ses données sont re-transmises à l'apprentissage. Elles sont placées au début du flux de données, $\mathcal{DS}$, afin de les traiter en premier. Les seuils de distance des nœuds sont mis-à-jour en conséquence.

4 Évaluation expérimentale

Dans cette section, nous présentons une évaluation expérimentale de l'algorithme G-Stream. Nous avons comparé notre algorithme avec l'algorithme GNG ainsi qu'avec deux algorithmes pertinents de clustering de flux de données. Nos expériences ont été réalisées sur la plateforme MATLAB en utilisant des données réelles et synthétiques. Les bases de données réelles, Shuttle (43500x9) et KddCup1 (49402x34), ont été prises à partir du répertoire UCI. Les bases DS1 (9153x2) et DS2 (5458x2) sont générées à l'aide du programme disponible sur `http://impca.curtin.edu.au/local/software/synthetic-data-sets.tar.bz2`. Comme nous l'avons expliqué dans la section 3, les

Algorithme 1 : G-Stream

Données : $\mathcal{DS} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$
Résultat : ensemble de nœuds $\mathcal{C} = \{c_1, c_2, ...\}$ et leurs prototypes $\mathbf{W} = \{\mathbf{w_{c_1}}, \mathbf{w_{c_2}}, ...\}$

Initialisation de l'ensemble $\mathcal{C}$ en contenant deux nœuds, c_1 et c_2 : $\mathcal{C} = \{c_1, c_2\}$;
tant que *il y a une donnée à traiter* **faire**
    Soit $\mathbf{x_i}$ la prochaine donnée à traiter;
    Trouver le nœud le plus proche $bmu_1 \in \mathcal{C}$ le deuxième nœud le plus proche $bmu_2 \in \mathcal{C}$;
    si $\|\mathbf{x_i} - \mathbf{w_{bmu_1}}\| > \delta_{bmu_1}$ **alors**
        mettre $\mathbf{x_i}$ dans le réservoir;
        si *le réservoir est plein* **alors**
            Gestion du réservoir;
    sinon
        Incrémenter le nombre de données affectées à bmu_1;
        $error(bmu_1) = error(bmu_1) + \|\mathbf{x_i} - \mathbf{w_{bmu_1}}\|^2$;
        Déplacer bmu_1 ainsi que son voisinage topologique envers $\mathbf{x_i}$:
        $\mathbf{w_{bmu_1}} = \mathbf{w_{bmu_1}} + \alpha_1.\|\mathbf{x_i} - \mathbf{w_{bmu_1}}\|$;
        $\mathbf{w_c} = \mathbf{w_c} + \alpha_2.\|\mathbf{x_i} - \mathbf{w_c}\|$ pour chaque nœud voisin c du nœud bmu_1;
        Incrémenter l'âge de toutes les arêtes sortant de bmu_1 et les pondérer;
        si bmu_1 *et* bmu_2 *connectés* **alors**
            remettre l'âge du lien à zéro
        sinon
            créer une arête entre bmu_1 et bmu_2, et marquer son temps de création
        Supprimer les arêtes dont l'âge est supérieur à age_{max};
        si *le nombre de données passées est multiple d'un paramètre* β **alors**
            pour chaque *i=1 à 3 // création de 3 nœuds* **faire**
                Trouver le noeud q ayant la valeur maximale de l'erreur cumulée;
                Trouver, parmi les voisins de q, le nœud f ayant la valeur maximale de l'erreur cumulée;
                Ajouter un nouveau nœud, r, entre les nœuds q et f;
                Ajouter des arêtes reliant r avec q et f, et supprimer l'arête reliant q et f;
        Application de la fonction d'oubli;
        Suppression des nœuds obsolètes ainsi que les nœuds isolés;
        Décrémenter l'erreur de tous les nœuds;

TAB. 1: Comparaison de G-Stream avec différents algorithmes.

Base de données		GNG	**G-Stream**	StreamKM++	CluStream
DS1	Acc	0.511±0.251	**0.993±0.006**	0.675±0.018	0.701±0.028
	NMI	0.491±0.132	0.712 ±0.004	0.702±0.021	**0.723±0.022**
	Rand	0.621±0.122	**0.846±0.001**	0.844±0.004	0.845±0.007
DS2	Acc	0.438±0.163	**0.991±0.004**	0.626±0.036	0.699±0.055
	NMI	0.388±0.114	**0.684±0.003**	0.624±0.018	0.662±0.032
	Rand	0.623±0.089	0.865±0.005	0.853±0.007	**0.867±0.009**
Shuttle	Acc	0.963±0.002	**0.973±0.004**	0.822±0.003	0.899±0.017
	NMI	0.355±0	**0.362±0.007**	0.258±0.015	0.340±0.035
	Rand	0.378±0.001	0.376±0.001	**0.753±0.039**	0.559±0.059
KddCup1	Acc	0.929±0.085	**0.998±0.001**	0.768±0	0.998±0
	NMI	**0.655±0.319**	0.602±0.032	0.012±0.003	0.022±0.002
	Rand	**0.824±0.206**	0.655±0.045	0.623±0.003	0.369±0.083

deux algorithmes GNG et G-Stream créent deux nœuds à la phase d'initialisation. Pour cette expérimentation, nous avons utilisé une version en-ligne de GNG, mais sans les paramètres que nous avons ajoutés et ceci afin de montrer l'intérêt et la contribution de ces paramètres dans G-Stream. Par conséquent, nous avons réalisé des expériences en initialisant deux nœuds en les choisissant parmi les 20 premières données du flux, et nous avons répété ceci 10 fois. Nous avons utilisé la même initialisation pour les deux algorithmes (G-Stream et GNG) et la valeur moyenne et l'écart-type sont indiqués dans le tableau 1. Ces résultats montrent que l'algorithme G-Stream dépasse l'algorithme GNG sur presque toutes les bases de données. Nous avons utilisé le package **stream** de R pour exécuter l'algorithme CluStream. La comparaison est également effectuée avec StreamKM++. A partir du tableau 1, on constate que les puretés de G-Stream (Acc) sont plus fortes pour toutes les bases de données que celles de GNG, StreamKM++, et CluStream. Les valeurs de NMI de G-Stream sont également plus fortes que celles des autres algorithmes, à l'exception de CluStream sur la base DS1. Les valeurs de l'indice de Rand de G-Stream sont également plus fortes que celles des autres algorithmes, sauf sur les bases DS2 et Shuttle. Nous rappelons que G-Stream procède en une seule phase alors que CluStream et StreamKM++ utilisent deux phases (phase en-ligne et hors-ligne).

5 Conclusion

Dans ce papier, nous avons proposé, G-Stream, une méthode efficace pour le clustering topologique en-ligne de flux de données évolutives. Dans G-Stream, les nœuds ainsi que les arêtes composant la structure topologique sont pondérés. A partir de deux nœuds, G-Stream compare les données arrivant aux prototypes courants ; il sauvegarde celles très éloignées dans un réservoir ; il apprend les seuils de distance automatiquement ainsi que plusieurs nœuds sont créés à la fois. L'évaluation expérimentale sur des bases de données réelles et synthétiques a démontré l'efficacité de G-Stream à découvrir des clusters de formes arbitraires. Les résultats obtenus sont prometteurs. Nous envisageons à l'avenir d'appliquer le principe des fenêtres adaptatives, de rendre notre algorithme le plus autonome possible et de le développer en Spark.

Remerciements : ce travail a été soutenu et réalisé dans le cadre du projet "Square Predict" (investissement d'avenir – Big data).

Références

Ackermann, M. R., M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, et C. Sohler (2012). Streamkm++ : A clustering algorithm for data streams. *ACM Journal of Experimental Algorithmics 17*(1).

Aggarwal, C. C., T. J. Watson, R. Ctr, J. Han, J. Wang, et P. S. Yu (2003). A framework for clustering evolving data streams. In *In VLDB*, pp. 81–92.

Bouguelia, M.-R., Y. Belaïd, et A. Belaïd (2013). An adaptive incremental clustering method based on the growing neural gas algorithm. In *ICPRAM*, pp. 42–49.

Cao, F., M. Ester, W. Qian, et A. Zhou (2006). Density-based clustering over an evolving data stream with noise. In *SDM*, pp. 328–339.

de Andrade Silva, J., E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, et J. Gama (2013). Data stream clustering : A survey. *ACM Comput. Surv. 46*(1), 13.

Fritzke, B. (1994). A growing neural gas network learns topologies. In *NIPS*, pp. 625–632.

Isaksson, C., M. H. Dunham, et M. Hahsler (2012). Sostream : Self organizing density-based clustering over data stream. In *MLDM*, pp. 264–278.

Kohonen, T., M. R. Schroeder, et T. S. Huang (Eds.) (2001). *Self-Organizing Maps* (3rd ed.). Secaucus, NJ, USA : Springer-Verlag New York, Inc.

Martinetz, T. et K. Schulten (1991). A "Neural-Gas" Network Learns Topologies. *Artificial Neural Networks I*, 397–402.

Udommanetanakit, K., T. Rakthanmanon, et K. Waiyamai (2007). E-stream : Evolution-based technique for stream clustering. In *ADMA*, pp. 605–615.

Zhang, T., R. Ramakrishnan, et M. Livny (1996). Birch : An efficient data clustering method for very large databases. In *SIGMOD Conference*, pp. 103–114.

Zhang, X., C. Furtlehner, et M. Sebag (2008). Data streaming with affinity propagation. In *ECML/PKDD (2)*, pp. 628–643.

Summary

Clustering data streams is becoming the most efficient way to cluster very large data sets. In this paper, we present a new approach, called G-Stream, for topological clustering of evolving data streams. The proposed method is an extension of the GNG (Growing Neural Gas) algorithm specially designed to manage data streams. G-Stream allows to discover incrementally clusters of arbitrary shape by making one pass over the data. The performance of the proposed algorithm is evaluated on both synthetic and real-world data sets.