Clustering semi-supervisé de séries temporelles multivariées par apprentissage profond

Dino Ienco*, Roberto Interdonato**

*INRAE, UMR TETIS, Univ. Montpellier
dino.ienco@inrae.fr
**CIRAD, UMR TETIS, Univ. Montpellier
roberto.interdonato@cirad.fr

Résumé. De grands volumes de données sont aujourd'hui produits par différents capteurs qui mesurent, généralement, plusieurs variables au cours du temps. Ces informations peuvent être organisées sous forme de séries temporelles multivariées. Recueillir suffisamment d'échantillons étiquetés, pour mettre en place une analyse d'apprentissage automatique supervisée pour ce type de données, est quasiment impossible. Dans ce contexte, les méthodes de clustering semi-supervisé représentent un outil particulièrement adapté pour exploiter au mieux la quantité réduite de connaissances à disposition. Dans le but d'améliorer l'analyse de séries temporelles multivariées dans ce contexte, nous proposons une approche de clustering semi-supervisé (sous contrainte) de séries temporelles basée sur des méthodes d'apprentissage profond utilisant des contraintes de type must- et cannot-link comme forme de supervision faible. L'évaluation expérimentale de cette méthode sur différents jeux de données a mis en évidence la plus value de notre proposition.

1 Introduction

De nos jours, l'acquisition continue de données massives est devenue un élément essentiel des processus dans plusieurs domaines d'application, tels que l'agriculture, la biochimie et la santé humaine. Ces données sont acquises par l'utilisation de capteurs de natures différentes (p.ex., des capteurs à distance, biochimiques ou portables), capables de produire des flux de données intégrant le suivi de plusieurs caractéristiques dans le temps. Lorsqu'il est nécessaire d'analyser ces données de manière automatique, la façon habituelle de les modéliser est de les organiser en séries temporelles multivariées. Ces structures de données suscitent un intérêt croissant, et plusieurs méthodes ont été proposées ces dernières années pour des tâches telles que la classification (Zhang et al., 2020) et la prévision de valeurs (Xie et al., 2021) d'une série temporelle multivariée. Cependant, le clustering de séries temporelles multivariées reste un problème ouvert. Dans les cas réels, étiqueter les données est une tâche coûteuse en termes de temps et de ressources, quel que soit le domaine d'application. C'est pourquoi le développement d'approches avancées de clustering non supervisé et semi-supervisé est devenu une priorité (Lampert et al., 2018), d'autant plus que ces méthodes permettent de caractériser un

ensemble de séries temporelles grâce à un faible nombre de données de référence, voir aucune. Dans ce domaine, les approches d'apprentissage profond ont récemment fait preuve d'être plus efficaces que celles de la science des données classiques. La capacité des architectures neuronales à apprendre une représentation appropriée des données permet une meilleure résolution de la tâche à accomplir, et cela est encore plus évident dans le contexte des séries temporelles multivariées, où la nécessité de traiter la dimension temporelle ajoute une complexité supplémentaire au problème à résoudre (Fawaz et al., 2019). Une méthode de clustering non supervisée basée sur l'apprentissage profond pour les séries temporelles multivariées a récemment été proposée dans (Ienco et Interdonato, 2020). Celle-ci exploite un autoencodeur récurrent intégrant des mécanismes d'attention et de portail d'accès afin de produire des plongements représentatifs des données d'entrée. Cependant, dans des cas opérationnels, il serait bénéfique d'exploiter les connaissances disponibles y compris si elles sont réduites. Bien que cette quantité soit trop faible pour être utilisée pour l'apprentissage d'un modèle de classification supervisé, elle serait tout de même utile pour guider le processus de clustering, ce qui est impossible en utilisant une méthode complètement non supervisée. Pour cette raison, nous proposons une nouvelle méthode spécialement conçue pour le clustering sous contrainte pour les séries temporelles multivariées, à savoir conDetSEC (constrained DEep Time Series Embedding Clustering). Celle-ci prend en compte les faibles volumes de connaissances disponibles sous la forme de contraintes *must-link* (ML) et *cannot-link* (CL).

À cette fin, nous proposons une architecture de réseau neuronale basée sur des GRUs (Gated Recurrent Units), dans laquelle le processus de clustering est basé sur une procédure en deux étapes impliquant une génération de plongements et une étape de raffinement du clustering, toutes deux exploitent le faible volume de connaissances disponibles fournies par les contraintes ML et CL. Au cours de la génération des plongements, les contraintes sont utilisées pour optimiser conjointement les paramètres du réseau via des tâches non supervisées et semi-supervisées. À l'étape de raffinement, elles sont utilisées conjointement avec l'objectif de rapprocher les plongements aux centroïdes de la solution de clustering pour améliorer le résultat final. Dans le but d'évaluer le comportement de la méthode proposée, nous fournissons une analyse expérimentale avec six jeux de données de séries temporelles provenant de différents domaines, qui montre l'efficacité et la flexibilité de notre approche. Les résultats de cette analyse montrent clairement comment conDetSEC est capable d'atteindre des bonnes performances avec des faibles quantités de données de référence. De plus, les résultats de notre méthode s'améliorent systématiquement quand la quantité de données étiquetées augmentent.

2 Method

Dans cette section, nous présentons une nouvelle méthode de clustering sous contrainte spécialement adaptée à l'analyse de séries temporelles multivariées, appelée conDetSEC (constrained DEep Time Series Embedding Clustering). Avec $X = \{X_i\}_{i=1}^{n}$ nous indiquons une série temporelle multivariée où X_i est une série temporelle et $X_{ij} \in R^d$ est le vecteur multidimensionnelle de la série temporelle X_i au temps j. Nous rappelons que les contraintes must-link (resp. cannot-link) sont définies sur des paires de séries temporelles et indiquent que deux séries temporelles multivariées devraient (resp. ne devraient pas) appartenir à la même partition. Le cœur de notre proposition est une architecture basée sur un réseau neuronal. Nous utilisons des architectures de réseaux neuronaux récurrents (Bengio et al., 2013), et plus parti-

culièrement une unité récurrente à déclenchement (Gated Recurrent Unit - GRU) (Cho et al., 2014), pour traiter à la fois l'information séquentielle ainsi que l'information multivariée qui sont contenues dans les séries temporelles acquises par différents capteurs en situation opérationnelles.

conDetSEC comprend deux étapes principales : la génération du plongement (embedding) et le raffinement du clustering. Les informations sur les contraintes (ou la semi-supervision) sont intégrées dans les deux étapes : pendant la génération des plongements des données, en optimisant conjointement les paramètres du réseau par des tâches non supervisées et semi-supervisées, et lors de l'étape de raffinement du clustering, où les plongements sont étirés vers les centres des clusters. Les centres des clusters (centroïdes) peuvent être dérivés en appliquant n'importe quel algorithme de clustering (par exemple, K-means) sur la nouvelle représentation des données. Le résultat final est dérivé en appliquant l'algorithme de clustering K-means sur les plongements produits par conDetSEC.

Algorithm 1 conDetSEC

Require: $X, ML, CL, N\_PRET\_EPOQUES, N\_REFINE\_EPOQUES, nClust.$
Ensure: $plongements.$
1: i = 0
2: **while** i $< N\_PRET\_EPOQUES$ **do**
3: Mise à jour Θ_1, Θ_2 and Θ_3 à travers la descente du gradient :
4: $\nabla_{\Theta_1,\Theta_2,\Theta_3} \frac{1}{|X|} \sum_{i=1}^{|X|} ||X_i - AE_f(X_i|\Theta_1,\Theta_2)||_2^2 + \frac{1}{|X|} \sum_{x_i \in X} ||rev(X_i) - AE_b(X_i|\Theta_1,\Theta_3)||_2^2 + \lambda L_{Contr}(ML, CL|\Theta_1)$
5: i = i + 1
6: **end while**
7: $plongements$ = extractEmbedding(Θ_1, X)
8: δ, C = runKMeans($plongements$, nClust)
9: i = 0
10: **while** $i < N\_REFINE\_EPOQUES$ **do**
11: Mise à jour Θ_1, Θ_2 and Θ_3 à travers la descente du gradient :
12: $\nabla_{\Theta_1,\Theta_2,\Theta_3} \frac{1}{|X|} \sum_{i=1}^{|X|} ||X_i - AE_f(X_i|\Theta_1,\Theta_2)||_2^2 + \frac{1}{|X|} \sum_{x_i \in X} ||rev(X_i) - AE_b(X_i|\Theta_1,\Theta_3)||_2^2 + L_{stretch}(plongements, \delta, C|\Theta_1) + \lambda L_{Contr}(ML, CL|\Theta_1)$
13: i = i + 1
14: **end while**
15: $plongements$ = extractEmbedding(Θ_1, X)
16: **return** $plongements$

L'algorithme 1 résume l'approche conDetSEC. La première étape (lignes 1-6) est consacrée à l'apprentissage de la représentation de séries temporelles multivariées (plongement) en exploitant la supervision fournie par les contraintes disponibles. À cette fin, nous mettons en place une tâche de reconstruction dans laquelle des autoencodeurs GRU sont employés pour encoder/décoder l'information séquentielle. Nous utilisons ici deux autoencodeurs GRU AE_f et AE_b. Chaque autoencodeur a deux composantes, un encodeur (enc_f et enc_b) et un décodeur (dec_f et dec_b). Plus précisément, AE_f encode (enc_f) et décode (dec_f) la série temporelle originale en considérant l'ordre naturel (en avant) tandis que AE_b encode (enc_b) et décode (dec_b) la série temporelle en ordre inverse (en arrière) par rapport à l'axe temporel. Les deux modèles d'autoencodeur interagissent l'un avec l'autre puisque la même représentation ou plongement, que nous pouvons nommer emb_i ($emb_i = enc_f(X_i) + enc_b(rev(X_i))$), est fournie en entrée aux deux décodeurs. Ici, X_i est la série temporelle i et $rev(X_i)$ est la même série temporelle en ordre inverse par rapport à l'axe temporel. smb_i désigne le plongement de la série temporelle

X_i. En outre, θ_1, θ_2 et θ_3 sont les paramètres du modèle récurrent où θ_1 sont les paramètres associés aux deux encodeurs (enc_f et enc_b) et θ_2 (resp. θ_3) sont les paramètres associés au décodeur avant dec_f (resp. arrière dec_b). L'erreur de reconstruction est évaluée à travers une fonction de perte standard telle que l'erreur quadratique moyenne (ou la norme L2 au carré) entre la série temporelle originale et la série reconstruite (ligne 4). En outre, les deux fonctions de perte de reconstruction (avant et arrière) sont complétées par la fonction de perte L_{Contr} qui a pour objectif de prendre en compte la semi-supervision apportée par l'ensemble des contraintes ML et CL. A cette fin, nous utilisons une fonction de perte contrastive (Chopra et al., 2005) dans le but d'injecter les connaissances disponibles dans la représentation apprise par le processus d'apprentissage. Plus formellement, nous pouvons définir la fonction de perte contrastive, dans notre contexte, comme suit :

$$L_{Contr} = \sum_{(X_i,X_j)\in ML} \frac{1}{2} Dist_{ij} + \sum_{(X_i,X_j)\in CL} \frac{1}{2}[m - Dist_{ij}]_+ \quad (1)$$

où $Dist_{ij}$ est la distance euclidienne au carré entre les plongements des séries temporelles X_i et X_j ($Dist_{ij} = ||emb_i - emb_j||_2^2$), $[c]_+$ est définie comme max(0,c) et m est l'hyperparamètre de marge. L'idée derrière cette fonction de perte contrastive est de minimiser la distance entre deux séries temporelles multivariées qui devraient appartenir (ML) à la même partition (premier terme de l'équation) et de maximiser la distance (jusqu'à une certaine valeur de marge m) entre deux séries temporelles multivariées qui ne devraient pas appartenir (CL) à la même partition (deuxième terme de l'équation). En considérant la ligne 4 de l'Algorithme 1, l'hyperparamètre λ pondère l'importance de la fonction de perte semi-supervisé par rapport la fonction de perte globale. Ensuite, les encodeurs pré-entraînés sont utilisés pour extraire la représentation de la série temporelle multivariée dans le but de calculer une solution de clustering initiale, à partir de laquelle les centroïdes sont dérivés (lignes 7-8). Successivement (lignes 10-14), la solution de clustering est raffinée à travers un terme supplémentaire dans la fonction de perte défini comme suit :

$$L_{stretch}(plongements, \delta, C|\Theta_1) = \frac{1}{|X|} \sum_{emb_i \in plongements} \sum_{j=1}^{nClust} \delta_{ij} ||C_j - emb_i||_2^2 \quad (2)$$

où $nClust$ est le nombre de clusters, δ_{ij} est une fonction qui indique si une série temporelle i appartient au cluster j et C_j est le centroïde du cluster j. Un tel terme permet d'étirer davantage l'espace de représentation de données dans le but de rapprocher, le plongement d'une séries temporelle, du centroïde le plus proche. L'objectif est d'induire une structure de clustering plus claire. Enfin, la nouvelle représentation ($plongements$) est extraite (ligne 15) et restituée par la procédure. Le partitionnement finale de données est obtenue en appliquant l'algorithme de clustering K-Means sur cette nouvelle représentation.

3 Experimental Setting

Dans cette section, nous allons décrire les expériences que nous avons menée pour évaluer les performances de conDetSEC. Pour l'étude comparative, nous considérons les méthodes

suivants : 1) L'algorithme de clustering *K-means* équipé de la mesure de distance Dynamic Time Warping (Dau et al., 2018) (*DTW*) ; 2) Une autre version de l'algorithme *K-means* équipé de la mesure de distance Soft Dynamic Time Warping (Cuturi et Blondel, 2017) (*SOFTDTW*) ; 3) L'approche de clustering par contrainte basé sur la théorie spectrale des graphes (Lampert et al., 2018) (*Spec*) ; 4) L'algorithme COP *K-means* couplé avec DTW (Lampert et al., 2018) (*COPK-Means*) et ; 5) Une variante entièrement non supervisée de notre approche qui ne tient pas compte des contraintes d'entrée (conDetSEC$_{noC}$).

L'évaluation a été réalisée sur six jeux de données (Fawaz et al., 2019) provenant de domaines d'application différents avec des caractéristiques très variées en termes de nombre d'échantillons, de nombre d'attributs (dimensions) et de longueur des séries temporelles.

Jeu de données	**# Exemples**	**# Variables**	**Taille Min/Max**	**Taille Moyenne**	**# Classes**
ArabicDigits	8 800	13	4/93	39	10
JapVowel	640	12	7/29	15	9
Dordogne	9 919	6	23/23	23	7
ECG5000	4 686	1	140/140	140	5
HAR	10 299	9	128/128	128	6
PenDigits	10 992	2	8/8	8	10

TAB. 1 – *Caractéristiques des benchmarks*

Pour mesurer les performances des méthodes de clustering, nous utilisons la NMI (Normalized Mutual Information) (Strehl et Ghosh, 2002). La NMI, est une mesure comprise entre $[0, 1]$ qui prend sa valeur maximale lorsque la partition de clustering correspond complètement à celle originale, c'est-à-dire la partition induite par les étiquettes de classe. Nous simulons la supervision en termes de contraintes, en sélectionnant un nombre croissant d'exemples étiquetés par classe, dans l'ensemble {5, 10, 15, 20, 25}, qui correspondent à des niveaux croissants de supervision. Pour chaque valeur, nous induisons l'ensemble complet de contraintes correspondant. En raison du processus de sélection aléatoire des échantillons et de la nature non déterministe des algorithmes de clustering, nous répétons l'étape de sélection des échantillons 5 fois pour chaque nombre d'étiquettes par classe. Successivement, à partir de l'ensemble des échantillons étiquetés par classe, nous dérivons l'ensemble des contraintes Must-Link et Cannot-Link. Enfin, pour chaque niveau de supervision, nous présentons les valeurs moyennes de NMI. Pour toutes les méthodes, le nombre de clusters est égal au nombre de classes. conDetSEC est implémenté via la bibliothèque python *Tensorflow*. Les paramètres du modèle sont appris à l'aide de l'optimiseur Adam (Kingma et Ba, 2014) avec un taux d'apprentissage égal à 5×10^{-4} pour les deux étapes de notre méthodologie (génération des plongements et raffinement du clustering). Nous fixons la valeur de λ à 1, une taille de lot de 32 et le nombre d'époques de pré-entraînement ($N\_PRET\_EPOQUES$) et de raffinage ($N\_REFINE\_EPOQUES$) à 40 et 60, respectivement. Pour toutes les méthodes concurrentes, nous utilisons l'implémentation de la mesure Dynamic Time Warping et Soft Dynamic Time Warping de la bibliothèque python TSLEARN (Tavenard et al., 2020). Les expériences sont réalisées sur une station de travail équipée d'un processeur Intel(R) Xeon(R) W-2133, 3.6Ghz, avec 64Gb de RAM et un GPU GTX1080 Ti.

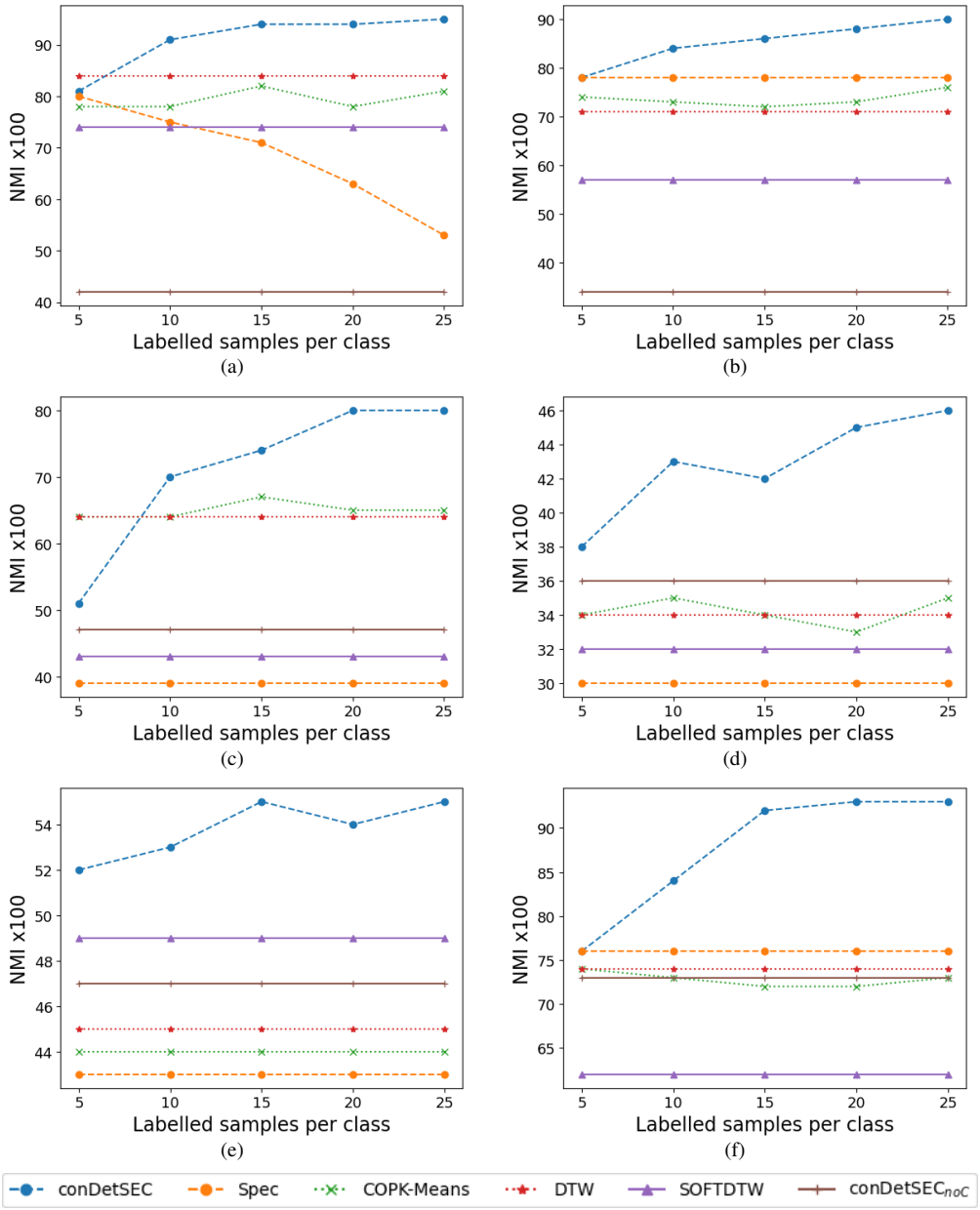

FIG. 1 – *Résultats (en termes de NMI) des différentes approches faisant varier la quantité d'échantillons étiquetés par classe pour générer les contraintes Must-Link et Cannot-Link sur : (a) JapVowel (b) ArabDigit (c) HAR (d) Dordogne (e) ECG500 et (f) PenDigits.*

4 Results

La figure 1 présente les résultats obtenu par les méthodes considérées, en termes de NMI moyen, sur les différents jeux de données de séries temporelles multivariées. Cela en faisant varier la quantité d'échantillons étiquetés à partir desquels les contraintes sont dérivées. À partir de dix échantillons par classe, pour générer les contraintes de type must-link et cannot-link, conDetSEC surpasse systématiquement toutes les autres approches testées. Nous pouvons également noter que notre méthode tend à fournir de meilleures solutions de regroupement lorsque le nombre d'échantillons étiquetés disponibles par classe augmente. À l'inverse, les autres méthodes de clustering semi-supervisé ($Spec$ et COP-KMeans) ont clairement du mal à tirer parti d'une quantité croissante de connaissances, sous la forme de contraintes. Nous observons que seule l'approche COP-KMeans semble sensible à l'augmentation du nombre de contraintes, tandis que l'approche $Spec$ présente un comportement extrêmement stable quelle que soit la quantité de contraintes injectée dans le processus semi-supervisé. En ce qui concerne les approches entièrement non supervisées DTW et $SOFTDTW$, nous remarquons que la première surpasse la seconde dans la majorité des cas. En outre, la stratégie K-means basée sur la mesure DTW présente des performances compétitives par rapport à ses homologues semi-supervisés. Une comparaison directe entre conDetSEC et DTW indique également que sur trois jeux de données sur six ($JapVowel$, $ArabDigit$ et HAR), lorsque la plus faible quantité d'échantillons étiquetés par classe est considérée (5), l'approche entièrement non supervisée DTW obtient des performances compétitives par rapport à conDetSEC. Ensuite, lorsqu'une quantité raisonnable de contraintes est intégrée dans le processus d'apprentissage, notre approche exploite clairement ces informations pour améliorer les performances de clustering en réalisant des gains évidents par rapport à la méthode DTW. Enfin, la comparaison entre conDetSEC et son ablation entièrement non supervisée (conDetSEC$_{noC}$) met en évidence le fait que l'algorithme proposé exploite efficacement la quantité de supervision à laquelle il a accès.

5 Conclusion

Dans ce travail, nous avons présenté conDetSEC, un nouvel algorithme de clustering semi-supervisé (sous contrainte) spécialement adapté à l'analyse de séries temporelles multivariées. Le cadre proposé est basé sur des modèles de Gated Recurrent Unit (GRU) et comprend deux étapes : la génération des plongements et le raffinement du clustering. Les informations issues des contraintes sont intégrées dans les deux étapes de l'algorithme. L'évaluation sur six jeux de données a démontré l'efficacité de conDetSEC et sa flexibilité sur des données provenant de différents domaines d'application. Les résultats obtenus mettent clairement en évidence que conDetSEC a la capacité d'exploiter efficacement des faibles quantités de données de références quand elles sont disponibles.

Références

Bengio, Y., A. C. Courville, et P. Vincent (2013). Representation learning : A review and new perspectives. *IEEE TPAMI 35*(8), 1798–1828.

Cho, K., B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, et Y. Bengio (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pp. 1724–1734.

Chopra, S., R. Hadsell, et Y. LeCun (2005). Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pp. 539–546.

Cuturi, M. et M. Blondel (2017). Soft-dtw : a differentiable loss function for time-series. In *ICML*, pp. 894–903.

Dau, H. A., D. F. Silva, F. Petitjean, G. Forestier, A. J. Bagnall, A. Mueen, et E. J. Keogh (2018). Optimizing dynamic time warping's window width for time series data mining applications. *Data Min. Knowl. Discov. 32*(4), 1074–1120.

Fawaz, H. I., G. Forestier, J. Weber, L. Idoumghar, et P. Muller (2019). Deep learning for time series classification : a review. *Data Min. Knowl. Discov. 33*(4), 917–963.

Ienco, D. et R. Interdonato (2020). Deep multivariate time series embedding clustering via attentive-gated autoencoder. In *PAKDD*, pp. 318–329.

Kingma, D. P. et J. Ba (2014). Adam : A method for stochastic optimization. *CoRR abs/1412.6980*.

Lampert, T. A., T. Dao, B. Lafabregue, N. Serrette, G. Forestier, B. Crémilleux, C. Vrain, et P. Gançarski (2018). Constrained distance based clustering for time-series : a comparative and experimental study. *Data Min. Knowl. Discov. 32*(6), 1663–1707.

Strehl, A. et J. Ghosh (2002). Cluster ensembles — A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research 3*, 583–617.

Tavenard, R., J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, et E. Woods (2020). Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research 21*(118), 1–6.

Xie, Z., H. Hu, Q. Wang, et R. Li (2021). Algenet : Adaptive log-euclidean gaussian embedding network for time series forecasting. *Neurocomputing 423*, 353–361.

Zhang, X., Y. Gao, J. Lin, et C. Lu (2020). Tapnet : Multivariate time series classification with attentional prototypical network. In *AAAI*, pp. 6845–6852.

Summary

Huge amounts of data are nowadays produced by a large family of sensors, which typically measure multiple variables over time. Such information can be profitably organized as multivariate time-series. Collecting enough labelled samples to set up a supervised analysis for such kind of data is challenging. In this context, semi-supervised clustering methods represent a well suited tool to get the most out of such reduced amount of knowledge. With the aim to deal with the semi-supervised analysis of multivariate time-series data, we propose a semi-supervised (constrained) deep embedding time-series clustering framework to cope with weakly supervision in form of must- and cannot-link constraints. Experimental evaluation on real-world benchmarks has highlighted the effectiveness of our proposal.