SAX-DD : une nouvelle représentation symbolique pour séries temporelles

Sylvain W. COMBETTES[1], Charles TRUONG[1], Laurent OUDRE[1]

[1]Université Paris Saclay, Université Paris Cité, ENS Paris Saclay, CNRS, SSA, INSERM, Centre Borelli, F-91190, Gif-sur-Yvette, France.
{sylvain.combettes, charles.truong, laurent.oudre}@ens-paris-saclay.fr

Résumé – Nous proposons une nouvelle représentation symbolique pour séries temporelles basée sur SAX (*Symbolic Aggregate approXimation*), qui s'intitule SAX-DD (*SAX Dynamic Duo*). Nous introduisons également une nouvelle mesure de similarité sur les représentations symboliques basée sur la distance d'édition générale, appelée D-GED (*Dynamic General Edit Distance*). Nous montrons que la représentation SAX-DD performe mieux que SAX ainsi que 1d-SAX sur la classification par 1 plus proche voisin, pour un même taux de compression. Ces méthodes sont évaluées sur 85 jeux de données univariées et de longueur égale provenant de la *UCR Time Series Classification Archive*.

Abstract – We introduce SAX-DD (SAX Dynamic Duo), a symbolic representation of time series based on SAX (Symbolic Aggregate approXimation). We also introduce D-GED (Dynamic General Edit Distance), a novel similarity measure on symbolic representations based on the general edit distance. We demonstrate the superiority of the SAX-DD representation compared to both SAX and 1d-SAX on the 1-Nearest Neighbor classification task, for a same memory usage ratio. These algorithms are evaluated on 85 univariate equal-sized data sets from the UCR Time Series Classification Archive.

1 Introduction

L'une des questions fondamentales en traitement des données temporelles est celle du choix d'une représentation adaptée, à la fois concise et capable de prendre en compte l'information temporelle. Dans ce contexte, les méthodes de symbolisation proposent une solution simple et efficace. La symbolisation permet de transformer un signal à valeurs réelles $\boldsymbol{y}$ de longueur n en une séquence symbolique $\hat{\boldsymbol{y}}$ de longueur inférieure $w \leq n$ (typiquement une chaîne de caractères). L'atout majeur de ces approches est qu'elles permettent une compression double de la taille des données : réduction du nombre de caractères par rapport au nombre d'échantillons, et passage de valeurs réelles (typiquement 32 ou 64 bits) à un alphabet de plus petite taille. Travailler sur des séquences symboliques permet également d'utiliser les outils issus des techniques de la bioinformatique (par exemple sur les séquences de l'ADN) ou alors du traitement automatique du langage naturel.

Les représentations symboliques de séries temporelles sont utilisées pour de nombreuses tâches de fouille de données telles que la classification [1, 2], le partitionnement, la détection d'anomalies, ou encore la prédiction [3]. Certains auteurs obtiennent même de meilleurs résultats (par rapport aux données brutes) sur les tâches d'exploration de données ou de prédiction grâce à l'effet de lissage dû à la compression [1].

Dans cet article, nous proposons une nouvelle représentation symbolique pour séries temporelles qui s'intitule SAX-DD (*SAX Dynamic Duo*). Cette méthode de symbolisation offre la particularité d'être totalement adaptative, en proposant à la fois une segmentation temporelle, permettant d'adapter la taille de la représentation aux régimes observés dans le signal, mais aussi une quantification adaptative pour l'attribution des symboles. Nous introduisons également une nouvelle mesure de similarité D-GED (*Dynamic General Edit Distance*), spécifiquement adaptée aux séquences générées par SAX-DD, et basée sur la distance d'édition générale. Cette nouvelle mesure de similarité permet ainsi d'utiliser SAX-DD dans de nombreuses tâches de fouille de données, telles que la classification ou le partitionnement.

SAX (*Symbolic Aggregate approXimation*) [1] est probablement la méthode de symbolisation la plus utilisée pour les séries temporelles. La méthode SAX transforme un signal à valeurs réelles en une chaîne de caractères, par exemple `abbcaabc`, en utilisant une segmentation uniforme et une quantification basée sur des arguments probabilistes. SAX repose sur deux paramètres : la longueur w de la séquence symbolique que l'on souhaite former, et A le nombre de symboles possibles. Après normalisation de chaque signal (moyenne nulle et variance unitaire), la procédure de SAX comporte deux étapes principales.

1. L'axe des abscisses (celui du temps) est discrétisé en w segments de taille uniforme et on calcule la moyenne du signal sur chaque segment.
2. L'axe des ordonnées est discrétisé en A intervalles de quantification. Ces intervalles sont construits de sorte que tous les symboles aient la même probabilité selon une distribution $N(0, 1)$.

Un exemple de représentation SAX est donné en haut de la figure 1. Une mesure de similarité appelée MINDIST a été développée pour les séquences générées par SAX. Cette mesure de

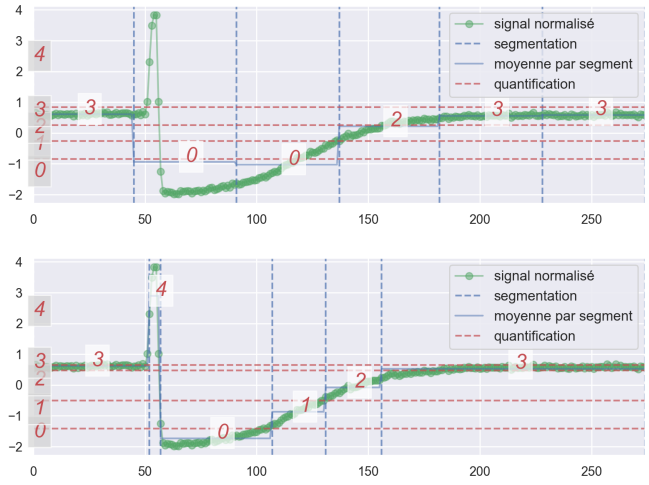

FIGURE 1 – Exemples de représentation symbolique d'un signal du jeu de données Trace (*UCR Time Series Classification Archive*. En haut : représentation SAX ($w = 6$ et $A = 5$) et en bas : représentation SAX-DD ($\lambda_0 = 0.5$ et $A = 5$).

similarité offre des garanties théoriques, notamment en étant une borne inférieure de la distance euclidienne sur les séries temporelles originales.

De nombreuses variantes de SAX ont été proposées afin de répondre aux limites de SAX et de l'améliorer. SAX ne prenant en compte que la moyenne par segment, certaines méthodes ont proposé d'augmenter le nombre de caractéristiques pour décrire les segments afin d'améliorer la qualité de la représentation. **1d-SAX** [4] utilise la régression linéaire pour représenter chaque segment par la moyenne et la pente, puis les combine en un seul symbole par segment. Le nombre total de symboles est $A = A_{\text{moy}} \cdot A_{\text{pente}}$, où A_{moy} et A_{pente} correspondent respectivement au nombre de symboles pour encoder la moyenne et la pente. Par ailleurs, la discrétisation uniforme selon l'axe du temps ne dépend que du choix de w et ne s'adapte pas aux changements dans le signal : on peut donc créer deux segments pour un même phénomène ou à l'inverse voir deux phénomènes différents comme appartenant à un même segment. Cet effet est visible sur la figure 1 où le pic autour de l'indice temporel 50 est mélangé avec le phénomène suivant au sein d'un symbole `0`. De plus, elle suppose que les signaux d'entrée soient de taille égale, ce qui exclut de nombreux jeux de données. Enfin, l'étape de quantification suppose une distribution normale centrée réduite pour les moyennes des segments, qui n'est pas toujours vérifiée dans des jeux de données réels.

2 Méthode

La méthode de symbolisation que nous proposons s'intitule **SAX-DD (*SAX Dynamic Duo*)**. Elle repose sur deux étapes : une étape de segmentation et une autre de construction des symboles. Chacune de ces étapes est contrôlée par un paramètre défini par l'utilisateur : la pénalité λ et le nombre de symboles A. Nous introduisons également la mesure de similarité **D-GED (*Dynamic General Edit Distance*)** qui est adaptée aux séquences symboliques générées par SAX-DD.

2.1 Segmentation des signaux

Après avoir été normalisés (moyenne nulle et variance unitaire), les signaux sont segmentés grâce à une méthode de détection de ruptures [5]. Soit un signal $\boldsymbol{y} = (y_1, \ldots, y_n)$ à valeurs réelles et composé de n échantillons. Le but d'une méthode de détection de ruptures est d'estimer les w^* instants inconnus $t_1^* < t_2^* < \ldots < t_{w^*+1}^*$ où certaines caractéristiques de $\boldsymbol{y}$ changent brusquement.

Dans le contexte de SAX-DD, nous nous intéressons aux ruptures de moyenne, et le nombre de ruptures w^* est inconnu (et doit donc également être estimé). Le problème d'optimisation estimant conjointement les instants de ruptures $\hat{t}_1, \ldots, \hat{t}_{\hat{w}+1}$ et le nombre de ruptures $\hat{w}$ est le suivant :

$$\underset{(w,t_1,\ldots,t_{w+1})}{\arg\min} \sum_{k=0}^{w+1} \sum_{t=t_k}^{t_{k+1}-1} \|y_t - \bar{y}_{t_k:t_{k+1}}\|^2 + \lambda(w+1) \quad (1)$$

où $\bar{y}_{t_k:t_{k+1}}$ est la moyenne empirique de $\{y_{t_k}, \ldots, y_{t_{k+1}-1}\}$ et $\lambda > 0$ est un paramètre de pénalisation. (Par convention, $t_0 := 0$ et $t_{w+1} := n$).

La formulation pénalisée (1) cherche un compromis entre l'erreur de reconstruction donnée par la somme des erreurs quadratiques et la complexité donnée par le nombre de ruptures. Le problème (1) peut être résolu de façon très efficace grâce à l'algorithme PELT (*Pruned Exact Linear Time*) [6], qui a une complexité en $\mathcal{O}(n)$.

Intuitivement, le paramètre λ pénalise l'introduction d'une nouvelle rupture : plus il est faible et plus le nombre de ruptures augmente. Le réglage de ce paramètre peut se révéler complexe car il dépend notamment de la taille initiale n du signal. Afin de faciliter la calibration, nous utilisons plutôt le facteur $\lambda_0 = \frac{\lambda}{\ln(n)}$ dans la suite. Il est important de mentionner que, contrairement à SAX, toutes les séquences symboliques sont susceptibles d'avoir une taille différente. En effet, la pénalité λ règle la granularité de l'algorithme de segmentation mais ne spécifie pas le nombre de ruptures.

2.2 Attribution des symboles

Une fois le signal segmenté, chaque intervalle est représenté par un symbole, pris dans un alphabet de taille A. Pour cela, nous réalisons une étape de quantification basée sur la valeur moyenne du signal sur le segment. Les moyennes de tous les segments extraits dans la base d'apprentissage sont partitionnés en différents intervalles de quantification, de façon à ce que chaque intervalle contienne le même nombre de points (approche par quantiles). On assigne ensuite un des A symboles à chaque segment, en fonction de l'intervalle dans lequel sa valeur moyenne se situe. Un exemple de représentation SAX-DD

est donné en bas de la figure 1. On observe au passage que plusieurs travers de la représentation SAX sont désormais gommés : en particulier, le pic autour d'indice temporel 50 est repéré en tant qu'un segment séparé. De plus, alors que le nombre de symboles est le même que pour SAX, on s'aperçoit qu'ils sont mieux utilisés et rendent mieux compte des phénomènes présents dans le signal.

2.3 Mesure de similarité entre séquences symboliques

L'intérêt principal des représentations symboliques est qu'elles permettent de réaliser des tâches (comme la classification) dans un espace de plus petite dimension, grâce à l'utilisation de mesures de similarités adaptées. En complément de la représentation SAX-DD, nous introduisons donc la D-GED (*Dynamic General Edit Distance*), une nouvelle mesure de similarité adaptée aux séquences symboliques générées par SAX-DD. Contrairement à la MINDIST associée à SAX, la D-GED est compatible avec des séquences symboliques de longueurs variables telles que celles obtenues avec SAX-DD.

La D-GED est basée sur la distance d'édition générale (également appelée distance de Levenshtein générale), qui a été introduite dans la communauté de l'algorithmique de texte. La distance d'édition générale entre deux chaînes de caractères `ch1` et `ch2` est le coût minimal d'une séquence d'opérations qui transforme `ch1` en `ch2`. Les opérations autorisées sont

— l'insertion d'un caractère dans une chaîne de caractères,
— la suppression d'un caractère dans une chaîne de caractères,
— la substitution de caractères (par un caractère différent) dans les deux chaînes de caractères.

À chaque opération correspond un coût, et ce coût dépend également des caractères impliqués. Notons que c'est le fait d'accepter des insertions et des suppressions qui permet à la D-GED de comparer des séquences symboliques de longueurs différentes.

La D-GED fixe les coûts de substitution de sorte à ce qu'ils intègrent la distance entre les différents symboles.

— Le coût de substitution $\mathrm{sub}(\mathtt{a},\mathtt{b})$ pour les symboles `a` et `b` est la distance euclidienne entre la moyenne $\mu_{\mathtt{a}}$ des valeurs moyennes par segment attribuées au symbole `a` et la moyenne $\mu_{\mathtt{b}}$ des valeurs moyennes par segment attribuées au symbole `b`

$$\mathrm{sub}(\mathtt{a},\mathtt{b}) = \|\mu_{\mathtt{a}} - \mu_{\mathtt{b}}\|_2 . \qquad (2)$$

— Pour tous les caractères, les coûts d'insertion et de suppression sont fixés à $\mathrm{sub}_{\max}$, où $\mathrm{sub}_{\max}$ est la valeur maximale des coûts de substitution donnés dans (2).

La taille mémoire de la représentation de chaque segment est, comme pour SAX, de $\lceil \log_2(A) \rceil$ bits. Cependant, comme notre étape de discrétisation selon x créé des segments de longueurs variables, nous proposons d'ajouter un bit supplémentaire à la représentation de chaque segment, qui encode si le segment a une petite ou une longue durée. Pour cela, nous calculons les longueurs minimales et maximales des segments pour tous les signaux du jeu de données étudié, et nous considérons que le segment est court ou long si sa durée est respectivement inférieure ou supérieure à $\frac{\min + \max}{2}$. Ainsi, on ne ne considère que deux longueurs de segments : une pour les segments courts et une pour les segments longs, et ceci nous permet de ne rajouter d'un bit par segment pour encoder cette information supplémentaire.

L'information de longueur est prise en compte dans la mesure de similarité D-GED de la façon suivante. Tout d'abord, les deux longueurs discrétisées sont divisées par le minimum des deux, puis arrondies à l'entier le plus proche : on obtient donc deux nouvelles longueurs, à savoir 1 et L. Ensuite, étant donnée une séquence symbolique, on réplique chaque symbole soit 1 fois (s'il s'agit d'un petit segment) soit L fois (s'il s'agit d'un grand segment). On calcule ensuite la distance d'édition générale décrite ci-dessus sur les nouvelles séquences symboliques redimensionnées. Cette astuce permet d'incorporer l'information de longueur dans la représentation SAX-DD et la mesure de similarité D-GED en ajoutant seulement un bit par segment, ce qui correspond au total à $\lceil \log_2(A) \rceil + 1$ bits par segment pour SAX-DD associé à la D-GED.

3 Expérimentations et résultats

Dans cette section, SAX-DD est comparé à deux approches de la littérature (SAX et 1d-SAX) sur une tâche de classification.

3.1 Configuration expérimentale

Tâche et concurrents Trois méthodes de représentation sont évaluées pour classifier des signaux : notre représentation SAX-DD associée à la mesure de similarité D-GED, SAX avec la mesure de similarité MINDIST, et 1d-SAX avec la mesure de similarité MINDIST. Ces représentations (et leur mesure de similarité correspondante) sont combinées avec l'algorithme du 1 plus proche voisin. La performance est calculée par validation croisée avec 5 plis.

Hyperparamètres Pour les trois méthodes, on fixe $A = 16$. Pour 1d-SAX, $A = 16$ correspond à $A_{\text{moy}} = 4$ et $A_{\text{pente}} = 4$. Pour SAX et 1d-SAX, w est choisi dans $\{2, 4, 8, 16, 32, 64\}$. Pour SAX-DD, λ_0 est choisi dans $\{0.01, 0.1, 1, 2, 5, 10\}$. Le choix du paramètre est fait par validation croisée.

Évaluation Pour mesurer la qualité de la classification, la métrique adoptée est le taux moyen de bonne classification sur la base de test, calculée sur les 5 plis de la validation croisée.

Taux de compression Le taux de compression est calculé de la manière suivante. On suppose que chaque échantillon à valeurs réelles est encodé sur 32 bits et on note n la longueur du

Méthode	Temps de calcul pour la symbolisation (s)	Temps de calcul pour la classification (s)
SAX	0.05	2.1
1d-SAX	0.8	2.2
SAX-DD	0.5	0.6

TABLE 1 – Temps de calcul pour la symbolisation et la classification avec 1 plus proche voisin sur le jeu de données ECG200, toutes les méthodes ayant un taux de compression proche de 96%.

signal original. Pour SAX and 1d-SAX, le taux de compression τ_{SAX} and $\tau_{\text{1d-SAX}}$ d'une représentation symbolique est alors

$$\tau_{\text{SAX}} = \tau_{\text{1d-SAX}} = 1 - \frac{w\,\lceil \log_2(A) \rceil}{32n}. \tag{3}$$

Pour SAX-DD, comme décrit dans 2.3, un bit supplémentaire est nécessaire pour chaque segment afin d'encoder sa longueur. Le taux de compression $\tau_{\text{SAX-DD}}$ est alors

$$\tau_{\text{SAX-DD}} = 1 - \frac{w\,(\lceil \log_2(A) \rceil + 1)}{32n}. \tag{4}$$

Notons qu'un taux de compression proche de 0% signifie que notre représentation symbolique a la même taille mémoire que le signal original. Nous visons un taux de compression supérieur à 95%.

Données Les méthodes de représentation sont comparées sur 85 jeux de données de la *UCR Times Series Classification Archive* [7] contenant des signaux univariés de même longueur. En effet, SAX et 1d-SAX ne peuvent être appliqués qu'aux signaux univariés et de longueurs égales. Ces jeux de données mêlent des séries temporelles réelles et synthétiques, pour un total de 19 178 126 échantillons.

3.2 Comparaison de SAX, 1d-SAX et SAX-DD

La figure 2 montre les scores de classification en fonction du taux de compression, pour les trois méthodes, pour une valeur fixe $A = 16$. Notons que des résultats similaires sont obtenus pour $A = 4$ et $A = 8$. Nos résultats montrent que SAX-DD est plus performant que SAX et 1d-SAX sur cette tâche de classification. En effet, pour chaque taux de compression, SAX-DD a un score plus élevé que celui de SAX et de 1d-SAX. Pour les trois méthodes, le score augmente lorsque le taux de compression diminue. En effet, la symbolisation est plus riche lorsqu'on peut l'encoder avec plus de bits.

Les temps de calcul des différentes méthodes sont comparés sur la tâche de classification appliquée au jeu de données ECG200 qui est composé de 200 signaux de longueur 96. Nous avons réalisé les expériences sur un ordinateur portable sous *macOS Monterey 12.1 with Apple M1 Chip 8-Core CPU and 7-Core GPU*. Le tableau 1 donne les temps de calcul des trois méthodes pour l'étape de symbolisation ainsi que de l'étape de classification (une fois la symbolisation effectuée). L'implémentation de SAX est optimisée (code en C du paquet `tslearn`

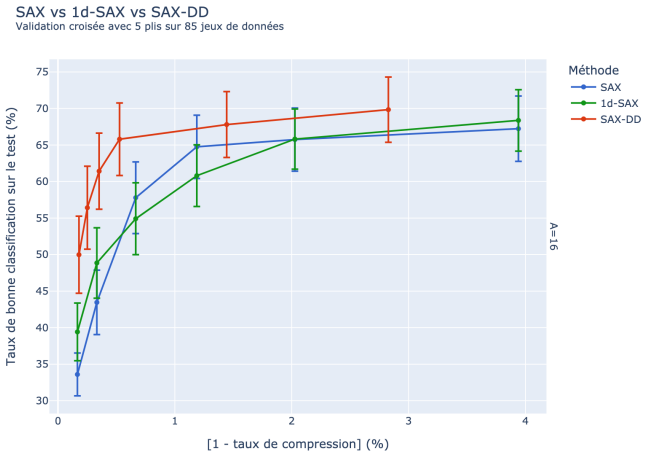

FIGURE 2 – Comparaison de SAX, 1d-SAX et SAX-DD sur le taux de bonne classification avec 1 plus proche voisin en fonction du taux de compression.

[8]). Le temps de calcul de la symbolisation de SAX-DD, actuellement implémenté en Python, reste raisonnable et pourrait faire l'objet du même type d'optimisation. En revanche, l'étape de classification est plus rapide pour SAX-DD du fait de l'utilisation de la distance de Levenshtein.

Références

[1] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi, "Experiencing sax : a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, pp. 107–144, 2007.

[2] T. L. Nguyen, S. Gsponer, I. Ilie, M. O'Reilly, and G. Ifrim, "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations," *Data Mining and Knowledge Discovery*, vol. 33, pp. 1183–1222, 2019.

[3] S. Elsworth and S. Güttel, "Time series forecasting using lstm networks : A symbolic approach," *ArXiv*, vol. abs/2003.05672, 2020.

[4] S. Malinowski, T. Guyet, R. Quiniou, and R. Tavenard, "1d-sax : A novel symbolic representation for time series," in *International Symposium on Intelligent Data Analysis*, pp. 273–284, Springer, 2013.

[5] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, Feb 2020.

[6] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[7] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[8] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, a machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020.