

PFE

État des lieux sur le transport optimal/DTW et problème de rectangularisation

Abanish BASKARADEVAN et Félix HUSSON

MACS 3

Encadrant : LACAILLE Jérôme
Sup-Galilée
Université Paris 13

Table des matières

1	Avant propos	2
2	Introduction et présentation du problème de "rectangularisation des données"	2
	2.0.1 Point de vue continue	2
2.1	Un mot sur la normalisation	3
2.2	Problématique de la série ou distribution « cible »	3
2.3	Donnée structurée	3
3	Transport optimal et distance de Wasserstein	4
3.1	Le transport optimal	4
3.2	Exemple du cas discret	4
	3.2.1 Du continu au discret	5
4	La distance de Wasserstein	6
4.1	Comparaison avec la divergence de Kullback-Leibler	6
4.2	<i>Unbalanced</i> transport optimal	6
4.3	Les applications du transport optimal	6
4.4	Un petit mot sur les auto-encodeurs	7
4.5	Adaptation de domaine	7
4.6	Application naïve au problème de rectangularisation	7
5	Algorithme DTW	10
5.1	Présentation de l'algorithme	10
5.2	Application du DTW	11
5.3	Chemin DTW	12
5.4	Échantillonnage	13
5.5	La notion de Soft-DTW	14
5.6	Algorithme TOAT	14
5.7	Algorithme de Spatio-temporal alignment	15
6	Idées d'algorithme basé sur le DTW	16
7	Conclusion et perspectives	16

1 Avant propos

Le but de ce rapport est dans un premier temps de faire un modeste état des lieux, de donner une vision sur comment et pourquoi utiliser le transport optimal et l'algorithme DTW dans le domaine de la data et de la statistique. C'est pourquoi nous tenterons d'avoir une vision au plus proche des données et de l'interprétation physique. Enfin, nous présenterons des pistes de réflexion et des algorithmes. Ce rapport sera accompagné d'un GitHub où on essaiera de présenter et de mettre en œuvre de la façon la plus didactique possible les algorithmes et les outils mathématiques. De plus, une bonne ressource concernant le transport optimal est la librairie <https://PythonOT.github.io/>. Ensuite, pour avoir une bonne première idée de ce qu'est le transport optimal, je vous invite à regarder ces vidéos [Meetup ML Rennes(2019)] [Peyre(2017)].

2 Introduction et présentation du problème de "rectangularisation des données"

Avant de définir de nouveaux objets mathématiques, nous allons discuter en profondeur de notre problématique très concrète. En effet, rendre les données rectangulaires signifie que la mesure de chaque phénomène a les mêmes dimensions et la même forme. Autrement dit, lorsque l'on observe deux vols d'avions et particulièrement une donnée physique au cours du temps (une série temporelle) pour chacun des avions, alors la mesure des deux vols aura une dimension et une forme différentes. Par exemple, si on mesure l'altitude sur la durée du vol, chaque vol a une durée de croisière différente, une phase de décollage différente, le vol peut être long ou court... Cela rend toute analyse statistique classique difficile, on aimerait pouvoir rendre les vols comparables et que par exemple comparer les ACP de différents vols aient un sens. On ne cherche pas à se contenter de rééchantillonner ces données, mais on veut faire en sorte que des séries temporelles se "superposent" au mieux. Chacune de ces séries temporelles possèdent des phases de vols distinguables : le taxi au sol, la montée, la croisière, la descente, puis à nouveau le taxi au sol. On veut donc rééchantillonner en assurant que les phases de vols soient aux mieux superposées.

"Rendre les données comparables" revient-il à trouver un espace tel que la projection de la donnée devienne comparable ? Un des objectifs de ce problème est de trouver une dilatation ou une contraction de la courbe qui permette de réduire ou allonger la taille (temporelle) tout en conservant l'allure. Il y a plusieurs façons de définir le problème, certaines sont plus judicieuses que d'autres. Au début, on voyait le problème d'un point de vue discret, i.e qu'on considérait les séries temporelles comme des vecteurs, donc on cherchait des matrices de dilatation qui prenait en entrée une taille m (de la série source) et en sortie une taille n (de la série cible).

Définition 1. Soit $t_1 \rightarrow f(t_1)$ et $t_2 \rightarrow g(t_2)$ deux séries temporelles (qui peuvent être multivariés) où t_1 et t_2 sont les temps associés. On cherche deux fonctions de dilatation ϕ_1, ϕ_2 telles que $t'_1 = \phi_1(t_1)$ et $t'_2 = \phi_2(t_2)$ pour ainsi écrire $f(t'_1)$ et $g(t'_2)$ qui sont les séries temporelles de même taille et de même allure(en effet t'_1 et t'_2 sont de même longueur).

À noter qu'il y a plusieurs manières de déterminer les fonctions ϕ_1, ϕ_2 , la plus évidente est de trouver un algorithme qui en fonction de f, g, t_1, t_2 trouve $t'_{1,2}$ de la longueur souhaitée.

Premièrement, nous avons essayé d'appliquer le transport optimal à ce problème, en utilisant différentes manières de définir le problème, mais sans succès. En effet, le transport optimal dans l'état actuel, en considérant les variantes connues à ce jour, n'est pas adapté pour conserver la structure de courbe. C'est pourquoi nous verrons une méthode utilisant l'algorithme de DTW où on va finalement construire un chemin entre deux séries temporelles, ce dernier permet de faire la correspondance entre les séries.

Définition 2 (Point de vue discret : Variante liée à l'algorithme DTW). Soit $t_1 \rightarrow f(t_1)$ et $t_2 \rightarrow g(t_2)$ deux séries temporelles (qui peuvent être multivariés) où t_1 et t_2 sont les vecteurs de temps. On cherche un chemin c entre f et g tels que $c(f, g) = T$ où T est un couple $T = (t_1, t_2)$. On a $f(T[0])$ et $g(T[1])$ qui sont de même taille et où $T[0]$ désigne la première composante et $T[1]$ deuxième composante.

Voir la partie qui explique plus en détails l'algorithme de DTW. La grande différence avec l'algorithme précédent est que la longueur du chemin est toujours supérieure à la longueur des séries temporelles. Le plus grand défaut de cette méthode est que l'on ne peut pas convertir plusieurs séries sources en une série cible, mais seulement qu'on convertisse à la fois une série source et une série cible dans une taille qui dépend du chemin.

2.0.1 Point de vue continue

On a longtemps pensé avec une vision discrète, mais cette dernière est trop rigide et est trop spécifique. Par exemple, cette méthode n'est pas adaptée à la comparaison de plusieurs séries et encore moins capable d'appliquer une transformation sur la série. Par contre, la définition 1 reste vrai en temps discret comme en temps continu. Finalement, on considère la série temporelle comme étant un phénomène continu qu'on a échantillonné et donc par interpolation du vecteur des mesures, on peut s'affranchir de la contrainte de la taille de l'échantillon, voir partie 5.4 .

2.1 Un mot sur la normalisation

Dans la suite, on va essayer de trouver des distances entre des points ou des distributions, il est donc impératif de normaliser la donnée pour éviter qu'une dimension soit trop grande par rapport à l'autre, ce qui modifierait la matrice de coût. Lorsque l'on souhaite garder l'amplitude de la donnée, on réalise une Z normalisation, i.e., on soustrait la moyenne et on divise par l'écart-type.

La généralisation de cette normalisation en dimension supérieure, appelée normalisation de Mahalanobis consiste à multiplier par l'inverse de la matrice de variance covariance. Une remarque sur ce point est qu'il faut fixer les valeurs propres très petites à zéro pour éviter de diviser par une valeur nulle. Dans le notebook, cette normalisation est implémentée parce qu'on avait l'idée de faire du transport optimal sur l'entièreté du dataset. En effet, cette normalisation prend en compte les corrélations entre variables, d'autant plus que notre cas d'applications, les vols d'avions, sont des séries temporelles très fortement corrélées les unes aux autres.

2.2 Problématique de la série ou distribution « cible »

Lorsque l'on a plusieurs séries temporelles et que l'on veut **rectangulariser** par rapport à une série temporelle, cela signifie qu'on veut modifier les autres séries temporelles pour qu'elles soient de même allure et de même taille que la série cible.

Un bon candidat, d'un point de vue mathématique, est la moyenne ou le barycentre, mais dans le contexte d'ingénierie, c'en est un mauvais. En effet, la moyenne ou le barycentre ne sont pas des distributions réelles, mais une création, donc on peut perdre le comportement d'une série (voir Figure 1 et 2). C'est pourquoi la médiane / médiane géométrique (on pourrait même la pondérer) sont plus robustes d'une et de deux représentent une réelle courbe, ce qui est meilleur dans le but d'une application industrielle ou l'expliquabilité est la clef.

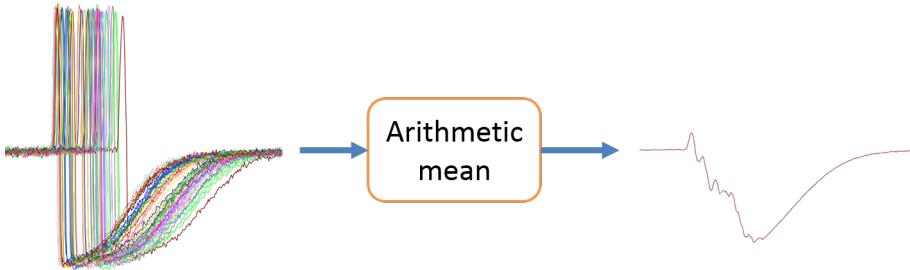


FIGURE 1 – Moyenne arithmétique

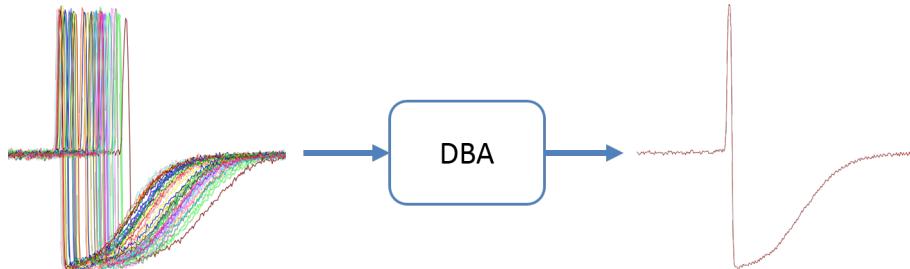


FIGURE 2 – DBA [François-Petitjean(2014)]

2.3 Donnée structurée

Dans la volonté de présenter à la fois l'état de l'art sur le transport optimal, mais aussi de présenter une variante qu'on avait pensé utile pour notre problème de *rectangularisation*.

Quand on parle de données structurées, on fait mention de structure de graphe, i.e des noeuds sont reliés par des arêtes. Concrètement, on dispose d'une matrice d'adjacence qui répertorie et quantifie les interactions entre les noeuds. On peut rajouter un label pour chaque noeud et ainsi appliquer un transport optimal sur cette structure de graphe labellisée. À noter que si les labels sont différents entre le graphe que l'on souhaite transporter et le graphe cible, alors, on devra utiliser la distance de **Gromov-Wasserstein**[Salmona et al.(2021)Salmona, Delon, and Desolneux].

Exemple : Si les espaces euclidiens des distributions ne sont pas de même dimension, i.e $\mathcal{R}^{25 \times 25}$ non comparable avec $\mathcal{R}^{10 \times 10}$. Si l'on considère des graphes, alors comparer directement deux noeuds qui appartiennent à différents graphes n'a pas de sens. Pour remédier à ce problème, on introduit la distance de **Gromov-Wasserstein** qui mesure la distance au sein de la distribution ou entre les noeuds d'un même graphe. Autrement dit, cette distance mesure la géométrie, la forme des distributions ou des graphes. Ainsi, le transport optimal avec cette distance revient à trouver un alignement entre les matrices de coût. (voir page 172 [Peyré and Cuturi(2020)])

On voulait appliquer cette distance de **Gromov-Wasserstein** aux séries temporelles après avoir lu [Kolouri et al.(2021)] et [Vayer et al.(2020)]. Cependant, les séries temporelles ne sont pas des graphes tels quels. Pour forcer la conversion en graphe orienté, on interprète chaque point de la série comme des nœuds relié uniquement entre le point précédent et points suivant dans le sens du temps. Cette vision en collier de perle nous paraissait intéressante pour garantir la structure de courbe. Cependant, on pense que cette méthode ne garantie pas que le transport de deux points ne se croise pas.

3 Transport optimal et distance de Wasserstein

La théorie du transport optimal (TO) trouve son origine dans le défi de déplacer des tas de sables de manière efficace, comme l'a illustré Gaspard Monge. Elle compare deux distributions de probabilité représentant des configurations des tas de sable, visant à minimiser le coût global du transport. Cette approche définit une distance entre distributions et confère une structure géométrique à l'espace des distributions. Le TO, redécouvert à maintes reprises dans l'histoire, a trouvé des applications variées, de la logistique à l'apprentissage automatique, grâce à des avancées récentes dans les solveurs approximatifs.

3.1 Le transport optimal

La première chose à préciser, c'est que le transport optimal se fait sur les distributions, qu'elles soient discrètes ou non. Deuxièmement, si vous voulez une ressource qui redémontre bien les objets mathématiques du transport optimal, on vous conseille Computational Optimal Transport de Gabriel Peyré et Marco Cuturi [Peyré and Cuturi(2020)]. Enfin, on présentera les grandes lignes du transport optimal et spécifiquement la formulation de Kantorovich. Cette dernière est moins restrictive que la formulation de Monge qui cherche une bijection entre les points, qui n'est pas toujours possible, en effet la formulation de Kantorovich permet de séparer la masse de probabilité.

3.2 Exemple du cas discret

On considère deux distributions, μ_0 et μ_1 .

$$\forall k = 0, 1, \quad \mu_k = \sum_{i=1}^{n_k} p_{k,i} \delta_{x_{k,i}}$$

où n_0 et n_1 sont le nombre de points, δ_x est une distribution de Dirac en $x \in \mathbb{R}^d$, et $X_k = (x_{k,i})_{i=1}^{n_k} \subset \mathbb{R}^d$ pour $k = 0, 1$ sont deux nuages de points. Les $p_{k,i}$ sont des probabilités associées à chaque point de la distribution.

On définit une matrice de couplage entre μ_0 et μ_1 telle que

$$\mathcal{P} = \left\{ (\Pi_{i,j})_{i,j} \in (\mathbb{R}^+)^{n_0 \times n_1} : \forall i, \sum_j \Pi_{i,j} = p_{0,i}, \forall j, \sum_i \Pi_{i,j} = p_{1,j} \right\}$$

La formulation de Kantorovich est donnée par

$$\Pi^* \in \arg \min_{\Pi \in \mathcal{P}} \sum_{i,j} \Pi_{i,j} C_{i,j}$$

où $C_{i,j} \geq 0$ est le coût de déplacer une masse unitaire de $x_{0,i}$ à $x_{1,j}$.

On montre que le couplage optimal Π^* peut être obtenu par une matrice sparse avec moins de $n_0 + n_1 - 1$ entrées non nulles. Une entrée $\Pi_{i,j}^* \neq 0$ devrait être comprise comme un lien entre $x_{0,i}$ et $x_{1,j}$ où une quantité de masse égale à $\Pi_{i,j}^*$ est transférée.

Dans la suite, nous nous concentrerons sur la distance de **Wasserstein** L^2 .

$$C_{i,j} = \|\mathbf{x}_{0,i} - \mathbf{x}_{1,j}\|^2$$

La distance de **Wasserstein** L^2 est alors définie comme

$$W_2(\mu_0, \mu_1)^2 = \sum_{i,j} \Pi_{i,j}^* C_{i,j}.$$

En pratique, on prendra soit la distance L^2 ou L^1 qui dépendra si on veut mettre plus d'importance dans les points qui sont proches ou non. En pratique, on pourrait prendre n'importe quelle distance, d'ailleurs une grande variété sont déjà implémentées dans la librairie POT.

Conservation des masses des distributions d'entrée et de sortie (la somme sur les lignes et la somme sur les colonnes de la matrice de transport Π).

$$\Sigma(n_0, n_1)\Pi = [\mathbf{p}_0; \mathbf{p}_1]$$

où $\Sigma(n_0, n_1) \in \mathbb{R}^{(n_0+n_1) \times (n_0n_1)}$.

Cet exemple est tiré des TDs de Gabriel Peyré [TD₍₀₎].

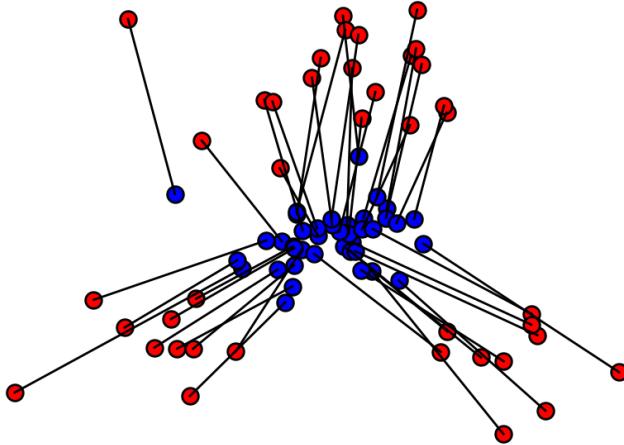


FIGURE 3 – Représentation d'un transport entre la mesure rouge et la bleue [TD₍₀₎]

Il faut bien remarquer que dans la Figure 3, on traite de distributions et même de réalisation de distribution sous forme de nuage de points, on est dans le cadre applicatif du transport optimal.

3.2.1 Du continu au discret

Cette représentation nous montre les différents plans de transport entre les distributions, toutes définitions discrètes ont leur pendant en continu, toutefois lors de l'implémentations numérique et algorithmique les distributions sont discrètes et il s'agit de problème de programmation linéaire.

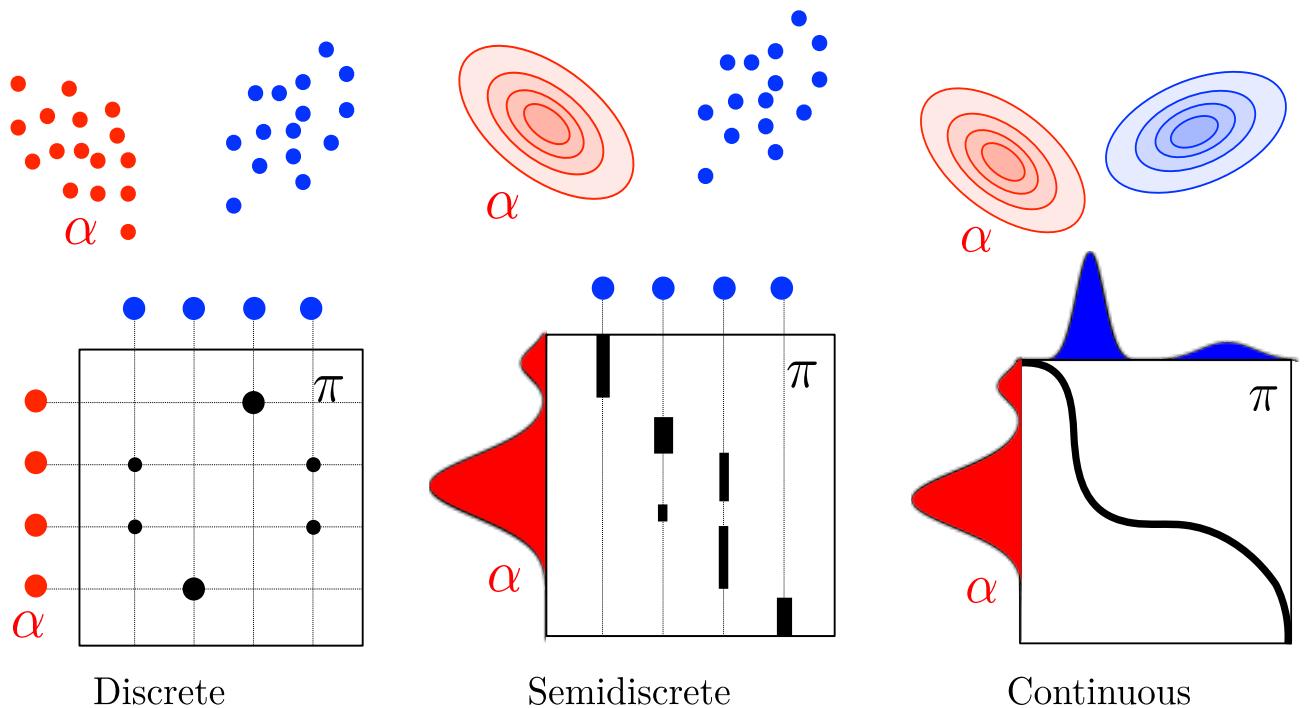


FIGURE 4 – Représentation de fonction de couplage (tiré de [Peyré and Cuturi(2020)])

L'application π donne deux informations, l'une est le couplage entre des distributions et l'autre est la quantité de masse transportée (représenté par l'épaisseur du point dans le cas discret).

4 La distance de Wasserstein

L'une des manières intuitives pour comprendre cette distance est le schéma (Figure 3) qui montre que la distance de **Wasserstein** compare les distributions par rapport à leur support, i.e de manière horizontale sur la Figure 5. On peut dire qu'elle est géométrique ou encore que la distance à des propriétés de convergence faible, de convergence en loi.

4.1 Comparaison avec la divergence de Kullback-Leibler

On rappelle sa définition, P et Q sont les distributions que l'on souhaite comparer. $D_{\text{KL}}(P\|Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right)$

La divergence de Kullback-Leibler compare deux distributions sur un même support alors que la distance de **Wasserstein** compare justement l'éloignement géométrique entre deux distributions. Autrement dit, si on applique la divergence de Kullback-Leibler sur des distributions à support disjoint, alors la divergence est égale à $-\infty$.

4.2 Unbalanced transport optimal

L'*unbalanced* optimal transport consiste à relâcher la contrainte sur les colonnes et les lignes de la matrice de transport, i.e. que la somme des poids ne doit plus être égal à 1. Dans une application en data ou par exemple, lorsque les distributions sont bruitées par exemple, cette application peut être utile. Concrètement, on déplace toujours les points, mais on autorise aussi la création et la diminution de masse, cette application a été prouvée comme plus robuste.

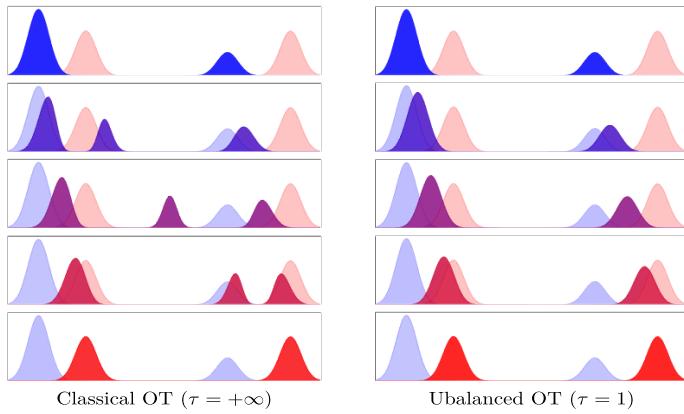


FIGURE 5 – Comparaison entre le transport optimal et l'unbalanced optimal transport [Séjourné et al.(2023)]

Sur cette Figure 5, on compare le transport optimal classique et l'*unbalanced* transport optimal sur des gaussiennes en 1D. Les gaussiennes bleues sont la source tandis que les roses sont les cibles et on représente un découpage temporel du transport. Ce qu'on remarque clairement, c'est que pour ce problème, le transport optimal classique sépare la gaussienne de gauche et envoie de la masse faire le trajet jusqu'à la seconde cible. Alors que l'*unbalanced* transport optimal se contente de transporter le maximum de masse pour les distributions qui sont proches et autorise une création et suppressions de masse de part et d'autre pour s'ajuster à la cible.

4.3 Les applications du transport optimal

Les applications du transport optimal sont diverses : l'adaptation de domaines, l'apprentissage visuel, l'alignement de structures 3D [Shen et al.(2021b)], le transport de graphes [Vayer et al.(2020)]. D'autant plus que, dans la théorie du transport optimal, on définit une distance de **Wasserstein** qui permet de mesurer la distance (longitudinale) entre distribution et elle est différentiable ! Cela est très utile pour les applications de machine learning où, par exemple, la distance de **Wasserstein** est la fonction *loss* qui pourra être minimisée par un algorithme de descente de gradients. Elle est aussi intéressante pour comparer des distributions après avoir transporté les données dans le même espace. Attention toutefois, si les distributions sont au départ dans des espaces différents, cela rend la distance de **Wasserstein** telle quelle caduque.

En pratique, on résout rarement le problème de transport optimal tel qui l'est écrit, en effet la résolution classique est de complexité $o(n^3 \log(n))$. Pour remédier à ce problème, on résout un problème dit relaxé ou on rend le problème convexe en introduisant un terme d'entropie (voir algorithme de **Sinkhorn** [Cuturi(2013)]) cela permet de passer à une complexité algorithmique de $o(n^2)$. De même, la distance de **Sinkhorn** à l'avantage d'être rapide à calculer.

Il existe de multiples variantes pour le transport optimal (Sinkhorn, *Unbalanced,fused-gromov,slice Wasserstein, low rank,...*)

4.4 Un petit mot sur les auto-encodeurs

Les auto-encodeurs sont à la mode pour encoder de l'information et générer de nouvelles informations. Le transport optimal se retrouve aussi dans certain auto-encodeur comme les **Wasserstein Auto Encoder** (WAE) reposant sur la distance de **Wasserstein** [Tolstikhin et al.(2019b)]. Lorsqu'on utilisait les auto-encodeurs type *Variational Auto Encoders* (VAE) sur des distributions comme une image par exemple, l'un des problèmes est que l'image générée était floutée. Ce problème est sûrement dû au fait que les VAE sont construits sur la divergence de Kullback et comme mentionné précédemment, cette divergence ne rend pas compte de l'éloignement géométrique des distributions. Ce problème est corrigé dans les WAE sans compromettre la stabilité de l'apprentissage ni la qualité de la génération.

Après avoir "vu" qu'on pouvait encoder et générer des distributions, il existe aussi le pendant pour encoder et générer des séries temporelles. Il s'agit de l'auto-encodeur TimeVAE qui comporte même une variante plus interprétable. Si l'on encode différentes séries temporelles, on ne pourra pas comparer directement son encodage dans l'espace latent alors que si on les rectangulise, on espère que leurs encodages soient comparables. Ce problème est récurrent avec les réseaux convolutifs qui traitent une courbe sans prendre en compte qu'il s'agisse du début ou de la fin ; c'est-à-dire que les réseaux convolutifs ne garde pas la structure de courbe. Peut-être que dans l'encodage de série temporelle, on donnera en plus de la série un paramètre de **rectangularisation** qui sera peut-être le chemin de la correspondance des temps. L'algorithme sera forcément difficilement robuste puisque la **rectangularisation** est pour le moment stochastique, si on lance deux fois le même code alors, on n'obtient pas le même résultat. Peut-être qu'il existe une façon déterministe optimal résolvant le problème de rectangularisation.

4.5 Adaptation de domaine

On peut faire de l'adaptation de domaine sur des séries temporelles, vu dans [S9E10(2023)] dans le but de modifier la série source pour la rendre comparable à la série cible. On a essayé de reprendre leurs méthodes qui consistent à stationnariser la série temporelle puis à transporter la série source sur une série cible. Ce transport est une transformation qui dépend des distributions de probabilités associées aux points.

La Figure 6 représente une EGT, qui est la température des gaz d'échappement, au cours d'un vol (source), l'EGT aux cours d'un autre vol (target) et enfin une EGT de la source transportée. On a donc entraîné un modèle d'adaptation de modèle basé sur **Sinkhorn** pour permettre de trouver la matrice de couplage entre la source et la cible. Ensuite, grâce à ce couplage, on peut transporter la source sur la cible appelé transport.

Le résultat n'est pas satisfaisant pour notre cas, le comportement de la série transporté est bien trop ressemblant à la série source. Cette méthode repose sur la forte hypothèse que les données de notre série temporelles, puissent être vues comme une distribution. Le seul cas où cette méthode pourrait être utile, c'est quand un signal est bruité et que l'on cherche à le débruiter. Dans ce cas, le signal bruité serait la source et le signal débruité la cible.

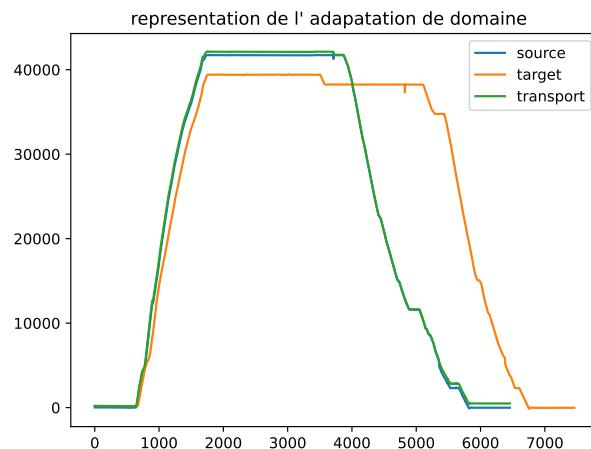


FIGURE 6 – Représentation de l'altitude au cours du vol

4.6 Application naïve au problème de rectangularisation

Pourquoi ne peut-on pas appliquer le transport optimal à des séries temporelles ?

La principale raison est que le temps n'est pas une variable aléatoire suivant une distribution ! Voyons voir si on tente d'appliquer le transport optimal, pour cela, on construit un vecteur en dimension deux avec la variable et son temps associé. Pour la figure 7 on a normalisé en temps et en espace.

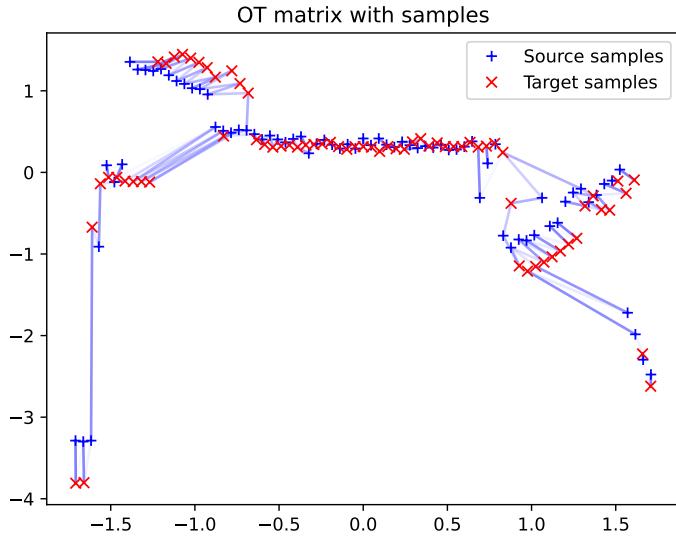


FIGURE 7 – Représentation de deux EGT normalisé et de leur couplage

La Figure 7 est une représentation du couplage entre deux séries temporelles. Plus le trait est foncé, plus il y a de la masse qui est déplacée de la source à la cible. Comme les séries temporelles sont différentes tailles et qu'à chaque point on a associé un poids uniforme, alors chaque point source a nécessairement plusieurs couplages si le poids sur la source est plus grand (et vice versa).

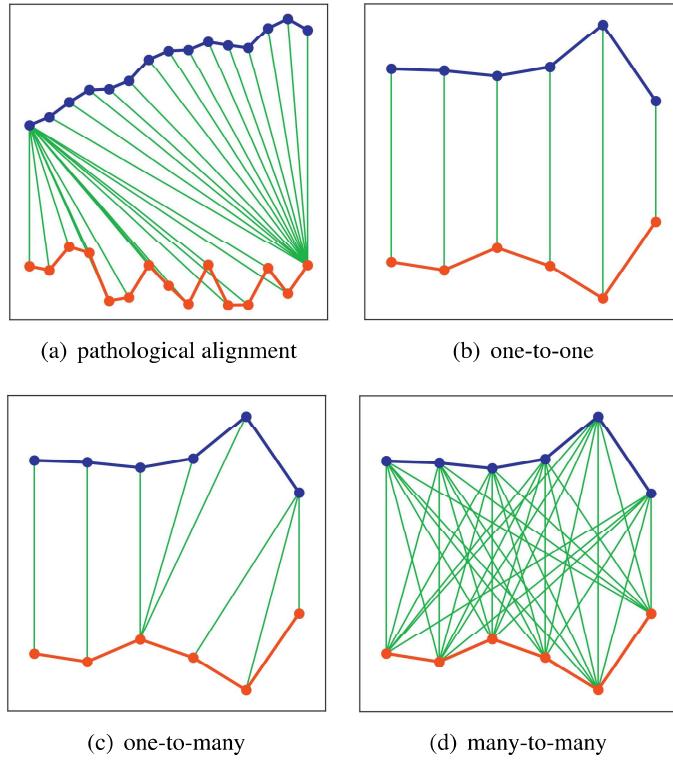


FIGURE 8 – Représente les différentes catégories d’alignement de série temporelle [Zhang et al.(2020)]

Les données de type série temporelle comportent une structure de courbe, i.e. les données sont ordonnées suivant le temps. Cette structure de courbe doit être maintenue après **rectangularisation** des données sinon on perd en interprétabilité. D'un point de vue plus physique/ ou proche de la donnée, "une structure de courbe" revient à dire que pour deux points qui se suivent dans le temps, leur couplage (symboliser par un trait vert) ne se croise pas. Comme dans le transport optimal, les couplages sont de type *many-to-many* alors, il est impossible en l'état de garder une structure cohérente.

Il est important de remarquer que les données sont rarement sous la forme de distribution discrète ou continue, de plus le passage naïf en fréquentiel n'est pas intéressant pour notre problème de **rectangularisation** parce qu'on perd la dépendance en temps dans nos séries temporelles. Mais si ce qui nous intéresse sont les spectres fréquentiels des séries temporelles, alors là, il n'y a pas de problèmes d'applications, voir notebook.

D'ailleurs, au début, on voulait passer des séries temporelles en fréquentiel, perdant ainsi la dépendance en temps, faire le transport et revenir à une série temporelle avec un noyau de covariance temporel des séries initiales, donnant une série temporelle modifiée. Dans l'esprit, on a fait en partie cette démarche dans la partie adaptation de domaine 4.5.

5 Algorithme DTW

Le DTW (Dynamic Time Warping) est un algorithme qui permet de comparer deux séries temporelles. Il est souvent utilisé dans des applications de traitement numérique du langage, telles que la reconnaissance de la parole ou détecter des anomalies. En effet, cet algorithme, permet de détecter les similarités entre deux séries temporelles. Cet algorithme revient à résoudre un problème d'optimisation, en essayant de minimiser la distance choisie par l'utilisateur. Cela revient donc à faire correspondre un point de la série temporelle A avec un ou plusieurs points de la série temporelle B. L'avantage incontestable vient du fait que les deux séries temporelles ne sont pas forcément de même taille. C'est un phénomène que nous allons rencontrer dans l'aviation avec des données de vols d'avions de durées différentes. Il est à noter que le choix de la distance influence les correspondances entre les deux séries temporelles.

5.1 Présentation de l'algorithme

La distance usuelle utilisée dans les librairies utilisant DTW est la suivante :

$$D(i, j) = \begin{cases} |x(i) - y(j)|, & i = j = 1 \\ |x(i) - y(j)| + \min(D(i-1, j), D(i-1, j-1), D(i, j-1)), & \text{sinon} \end{cases}$$

Remarque : en Python, les indices d'une liste ou d'une matrice commencent à 0.

Or, pour calculer les coefficients de la matrice, le temps d'exécution est important.

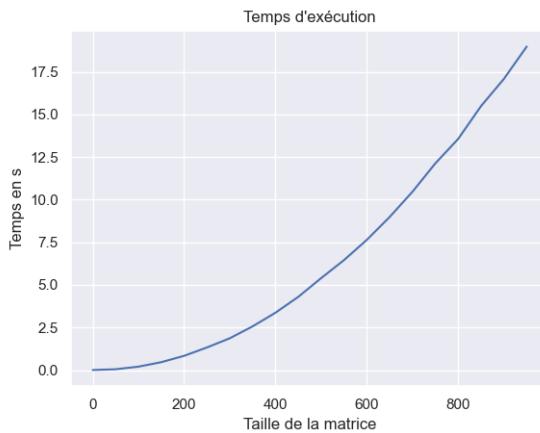


FIGURE 9 – Temps d'exécution pour calculer tous les coefficients de la matrice D

Ainsi, nous remarquons que la Figure 9 a une forme parabolique. On peut conclure que plus la taille augmente, plus le temps de calcul est considérable. En effet, la complexité de l'algorithme est proportionnelle au nombre de coefficients de la matrice $o(n^2)$. Cela pose un défi majeur compte tenu du nombre important de données et des dimensions conséquentes que l'on peut rencontrer dans le monde industriel.

Nous allons essayer de calculer seulement une partie des coefficients. Ainsi, nous allons uniquement calculer les coefficients autour de la diagonale de la matrice selon une largeur de « bande ». C'est la méthode de Sakoe-Chiba qui nous ramène à une complexité linéaire.

En effet, il est intéressant de calculer une distance entre le premier point de la première série temporelle et le dernier point de la seconde série, car nous n'aurons aucune correspondance. Par extension, les correspondances entre deux séries temporelles se font sur un laps de temps proche autour des points.

L'idée est de calculer une certaine largeur de bande autour de la diagonale. Cela permettrait d'avoir une matrice creuse avec des coefficients non nuls autour de la diagonale. Ainsi, le temps de calcul devient plus simple, c'est-à-dire que la complexité est linéaire.

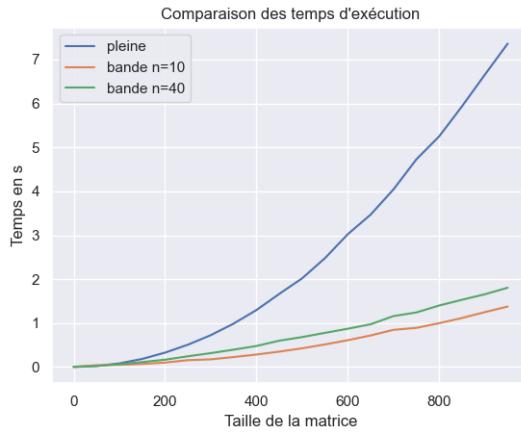


FIGURE 10 – Comparaisons des temps de calculs

On remarque bien que le temps de calcul est inférieur et plus abordable dans un contexte industriel.

Il existe également d'autres méthodes pour calculer rapidement la matrice, comme la méthode d'Itakura. Cette méthode permet de calculer les coefficients de la matrice selon une forme de parallélogramme.

5.2 Application du DTW

Nous allons utiliser les altitudes de deux vols pour comparer deux séries temporelles de longueurs différentes. Nous avons calculé la matrice de coût telle que la distance DTW soit respectée. Ensuite, il est nécessaire de « trouver » le chemin optimal pour minimiser le coût. Pour trouver ce chemin, nous devons utiliser le minimum entre les 3 cases voisines de la case initiale. Il existe deux méthodes, soit en partant du début ou en partant de la fin.

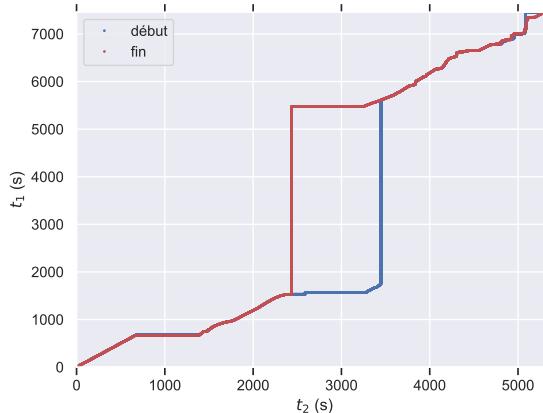


FIGURE 11 – Représentation du chemin

On remarque, que le chemin n'est pas unique selon la méthode utilisée. Il faut essayer de rendre ce chemin « unique », pour permettre le passage d'une série temporelle à une autre.

Le DTW permet l'analyse de deux séries temporelles pour faire correspondre les points entre eux grâce au chemin optimal calculé.

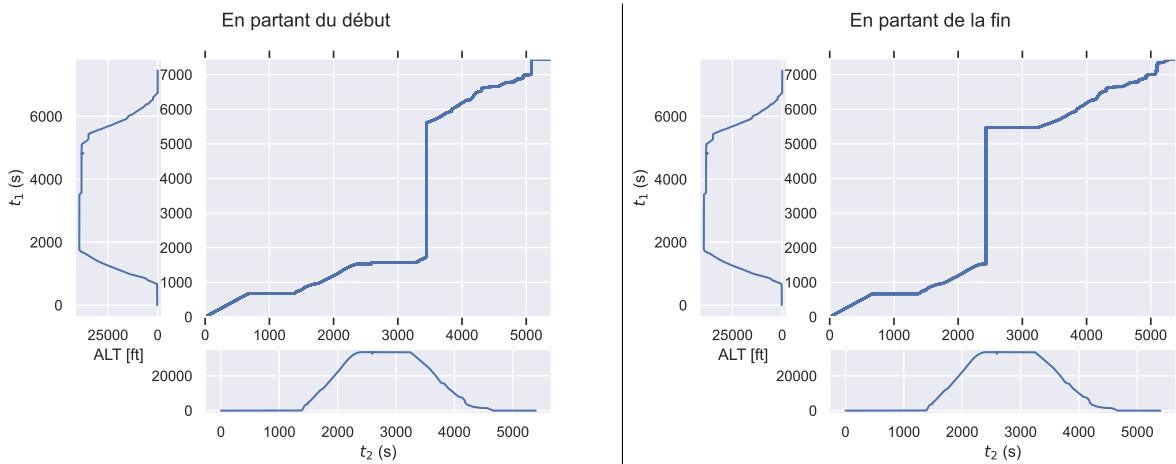


FIGURE 12 – Calcul du chemin selon le départ initial du chemin

On remarque qu'il est plus judicieux de prendre un chemin en partant de la fin plutôt qu'au début. Sur la première représentation graphique de la Figure 12, nous remarquons que lorsque t_1 est entre 2000 et 5 000, cela correspond à t_2 qui est d'environ 3500. Cela nous montre qu'une phase de croisière peut être considérée comme une phase d'atterrissage dans l'autre série temporelle. Ceci est plutôt absurde dans notre cas, car nous voulons garder une certaine cohérence entre les phases de vol des deux séries temporales.

5.3 Chemin DTW

[Morel et al.(2018)] En utilisant le chemin trouvé entre les deux séries temporales, nous pouvons retracer une nouvelle série en utilisant ce chemin. Ce chemin nous permet effectivement d'établir des correspondances entre les indices temporels des deux séries. Nous allons donc utiliser ces indices pour les remettre dans les données de la série temporelle. Cela nous donne donc une liste de données dont la taille de la série temporelle est égale à la taille du chemin pour les deux séries temporales. Les questions qu'on pourrait se poser sont les suivantes : 'Les nouvelles séries sont-elles légitimes ?' ou 'Les nouvelles séries permettent-elles de faire une analyse statistique sans perte d'information ?'. Il convient de noter que nous avons réussi à transformer deux séries temporales de longueurs différentes en une série de taille unique.

Nous avons testé cette méthode pour calculer la moyenne point à point sur les deux séries temporales.

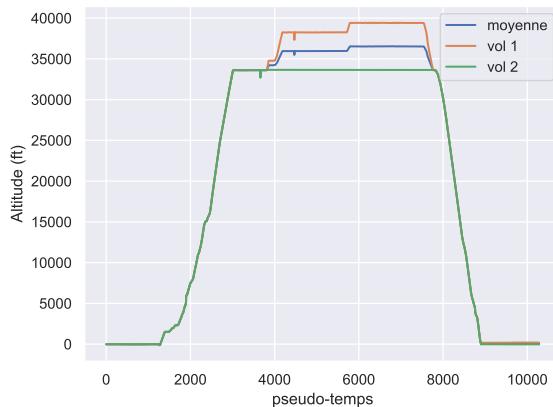


FIGURE 13 – Exemple de calcul statistique d'une moyenne de 2 vols de durées différentes

En abscisse, nous notons que c'est un pseudo-temps. En effet, nous ne pouvons pas le considérer comme un temps de la série temporelle, car nous traçons les correspondances entre les deux séries temporales grâce au chemin trouvé avec le DTW. Plus précisément, nous traçons deux vecteurs de même taille qui nous permettent de passer d'une série à une autre grâce au chemin.

À noter qu'une moyenne entre plusieurs séries temporales ne donne pas une bonne « estimation ». Pour palier à ce problème, nous pouvons utiliser le DBA (Dynamic Time Warping Barycenter Averaging) qui est plus robuste pour trouver une moyenne entre plusieurs séries temporales. Le DBA peut aussi être utilisé pour le clustering.

5.4 Échantillonnage

L'échantillonnage permet d'obtenir un nombre de points que nous pouvons choisir. L'idée est de sélectionner la taille de la série temporelle à comparer. Sauf qu'en l'état, nous avons des vecteurs "discrets" de mesures de phénomènes temporels continus que nous rendons "continus" par interpolation des points de mesure. Une fois cette interpolation faite, on peut choisir le nombre de points qu'on veut échantillonner pour être de même taille que la série cible.

Nous allons prendre une série temporelle cible "Vol 1" et on transforme les autres séries temporelles vers la série "Vol 1".

Idée n°1 : Échantillonner le temps dans l'intervalle initial de la série temporelle.

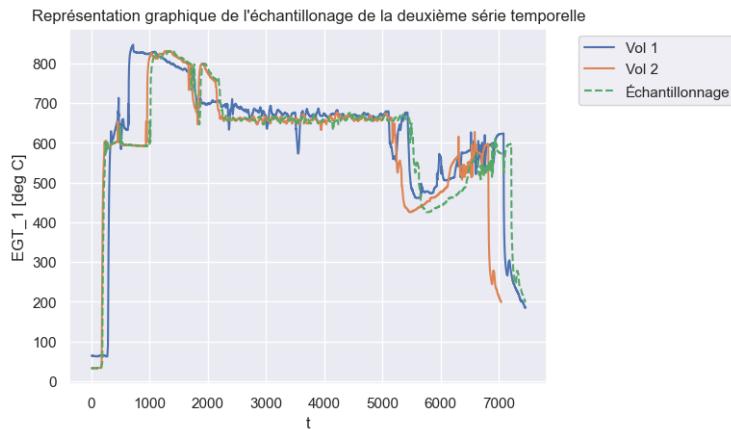


FIGURE 14 – Échantillonnage du temps

L'inconvénient vient que nous avons échantillonné la seconde série sans être proche de la première et sans respecter l'environnement et le contexte de cette dernière. Nous avons en réalité contracté la seconde série temporelle.

Idée n°2 : Échantillonner le chemin DTW puis reconstruire la série temporelle à l'aide du chemin et des données.

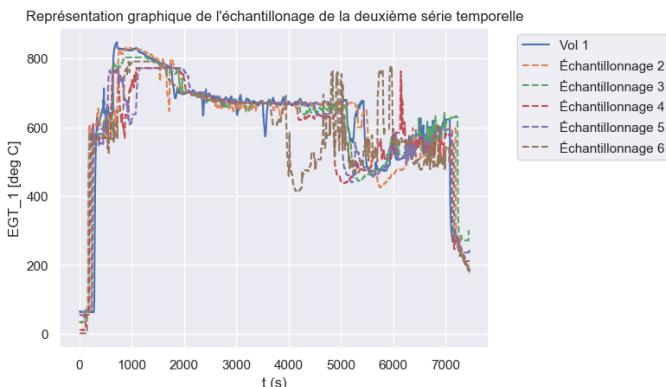


FIGURE 15 – Échantillonnage de plusieurs séries temporelles avec le chemin DTW

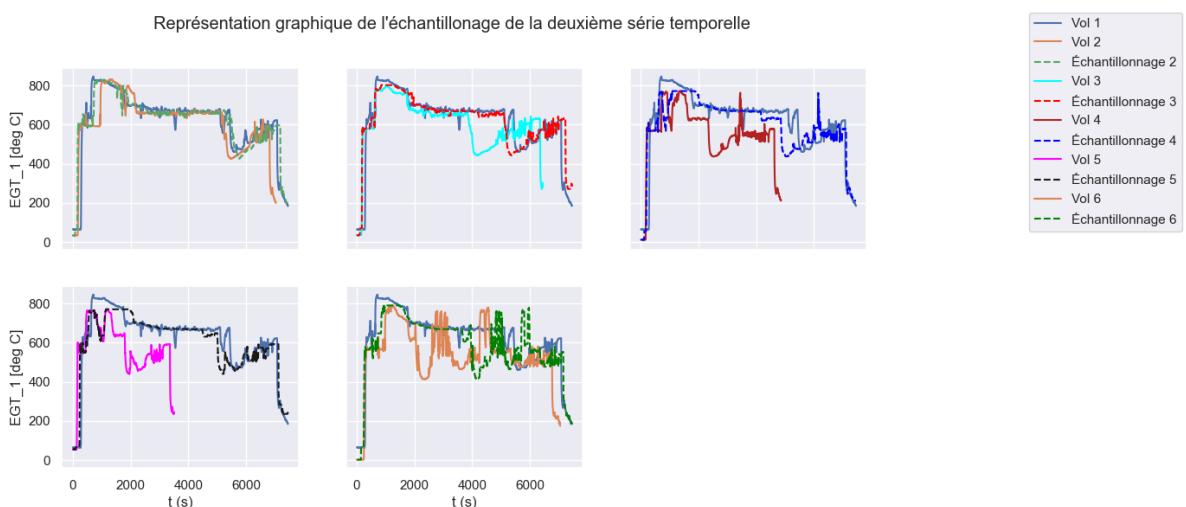


FIGURE 16 – Zoom sur chaque échantillon avec le chemin DTW

On remarque que la transformation entre deux séries temporelles est plutôt bien gérée avec l'échantillonnage du chemin de DTW. Cependant, il existe des cas extrêmes où la série à transformer ne ressemble pas à la série cible (voir la 5ème figure dans Figure 16). La question que nous devons nous poser ici est : est-ce que cette transformation correspond à notre attente ? La réponse à cette question peut être positive. En effet, même si l'allure de la série ne ressemble pas à la série cible, on peut constater que la nouvelle série échantillonnée garde l'allure de la série de base. Nous souhaitons tout de même conserver le contexte du vol dans un espace-temps différent de celui enregistré, ce qui est plutôt réussi.

Pour voir, si on arrive à reconstruire la série initiale en faisant le processus inverse. Regardons sur deux exemples la reconstruction de la série initiale.

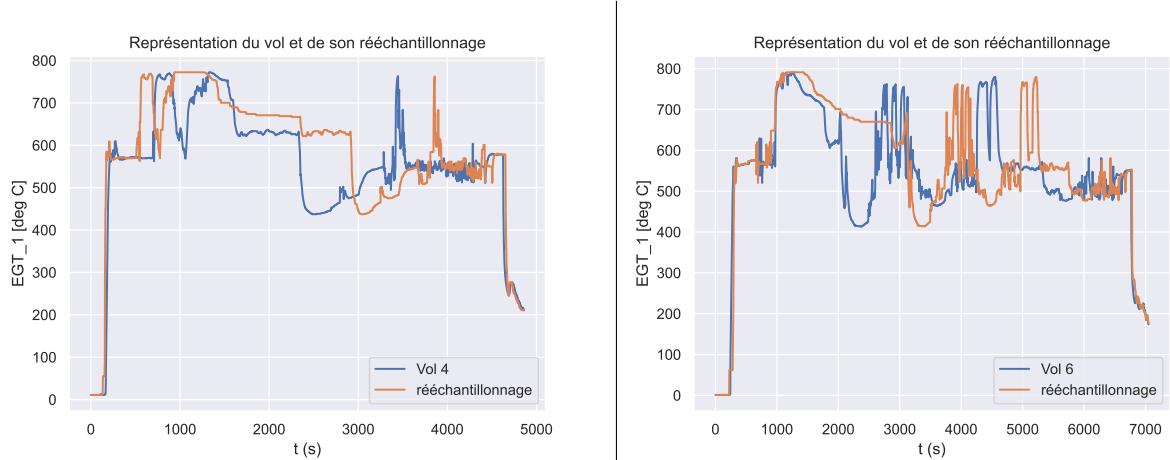


FIGURE 17 – Rééchantillonnage/Erreur de reconstruction

Le rééchantillonnage de la Figure 17 montre que la série temporelle est déplacée sur certains laps de temps. Ainsi, la série rééchantillonnée conserve la même allure, mais ne se retrouve pas aux bons endroits. La raison de cette translation vers la droite ou la gauche peut être attribuée à l'aléatoire. En effet, lorsqu'on échantillonne, on effectue un choix uniforme des points pour ajuster la série à la taille de la série cible. Ainsi, la série peut être dilatée ou contractée avec l'échantillonnage.

Cette seconde idée semble être une réponse possible à notre problématique, permettant de répéter le processus sur plusieurs séries temporelles en ayant une série "cible". Cependant, le problème de l'aléa implique que nous ne pouvons pas contrôler le choix des points. Nous pourrions plutôt attribuer une certaine importance à chaque période de temps. Par exemple, au lieu de choisir les points au hasard avec une probabilité uniforme, nous pourrions décider de choisir les points de telle sorte que chaque intervalle de temps entre les points soit égal. Notons qu'on ne peut pas utiliser une correspondance *many to many* comme chemin parce que non bijectif ; cela concerne le transport optimal, mais aussi TAOT.

5.5 La notion de Soft-DTW

L'un des intérêts dans le machine learning est de pouvoir appliquer l'algorithme de descente de gradient. La fonction coût (ou Loss function) doit être différentiable, ce qui n'est pas le cas pour la fonction coût DTW classique. Cette problématique a motivé M.Cuturi et M.Blondel à créer une fonction coût différentiable en relâchant la contrainte du minimum par un Soft-min [Cuturi and Blondel(2018)].

5.6 Algorithme TOAT

Vouloir comparer des séries temporelles n'est pas nouveau, c'est pourquoi il y a plusieurs variantes de l'algorithme de DTW. L'algorithme TOAT [Zhang et al.(2020)] est un algorithme de DTW utilisant le transport optimal (la variante Sinkhorn précisément). Cette méthode est meilleure que la méthode de DTW classique, surtout pour éviter certains alignements pathologiques. Si l'on reprend la figure 8 l'algorithme DTW est un alignement *one-to-many* (Figure 8) rigide en temps, mais a l'avantage de garder l'ordonnancement temporel (la structure de courbe). L'avantage de l'algorithme TAOT est de tirer parti du meilleur de DTW et OT, il produit un alignement *many-to-many* (Figure 8) grâce au transport optimal, le rendant plus flexible, tout en gardant l'ordonnancement temporel de la méthode DTW. En somme, cette méthode permet d'aligner des séries en minimisant les coûts et en prenant compte la forme globale de la courbe. De plus, la complexité algorithmique est de $O(N^2 \log(N))$. Mieux comparer des séries temporelles peut nous permettre de mieux catégoriser ces dernières.

Algorithm 1 Computation of Time Adaptive Optimal Transport

Input: $A, B, \lambda, w, maxIter = 5000, tolerance = 0.005$

Output: $distance, plan$

```

1:  $d = |A|$ 
2:  $t = zscore(linspace(1, d, d))$  #Régularisation du temps  $\mu=0$  et  $\sigma=1$ 
3: for  $i = 1 : |A|$  do
4:   for  $j = 1 : |B|$  do
5:      $M(i, j) = (a_i - b_j)^2 + w * (t_i - t_j)^2$  #Construction de la matrice de coût
6:   end for
7: end for
8:  $M = M / median(M(:))$  #Normalisation de la matrice par une valeur médiane
9:  $K = exp(-\lambda * M)$  #Noyaux de Gibbs
10:  $curIter = 0$ 
11:  $u = ones(d, 1)/d$ 
12: while  $curIter < maxIter$  do
13:    $v = 1. / (d * K' * u)$  #(u,v) solution du problème de Sinkhorn
14:    $u = 1. / (d * K * v)$ 
15:    $curIter = curIter + 1$ 
16:   if  $mod(curIter, 20) == 1$  then # le calcul du critère se fait toute les 20 itérations
17:      $criterion = sum(abs(v. * (K' * u) - 1/d))$  # calcul de la somme des erreurs absolue
18:     if  $criterion < tolerance$  then de la conservation de masse
19:        $break$ 
20:     end if
21:   end if
22: end while
23:  $distance = sum(u. * ((K. * M) * v))$  # calcul de la distance TAOT
24:  $plan = bsxfun(@times, v', (bsxfun(@times, u, K)))$  #matrice de transport

```

FIGURE 18 – Algorithme de la méthode TOAT [Zhang et al.(2020)]

Remarque : *bsxfun* est une fonction Matlab qui permet d'effectuer des calculs composante par composante sans que la taille des deux matrices ou vecteurs ne soit nécessairement de même dimension.

Explication des données : A et B sont deux séries temporelles ; λ coefficient de régularisation qui se trouve dans la distance de Sinkhorn ; w est le poids qu'on souhaite donner au temps. K est le noyau de Gibbs voir formulation du problème de Sinkhorn page 66 [Peyré and Cuturi(2020)].

Le point intéressant de cet algorithme est la ligne 5 où on reconnaît la matrice de coûts définie dans la partie transport optimal (3.1 et 3.2), cependant la grande différence est la prise en compte d'une distance en temps.

5.7 Algorithme de Spatio-temporal alignment

Comparer des données définies dans l'espace et le temps est notoirement difficile. Cela implique de quantifier à la fois la variabilité spatiale et temporelle tout en tenant compte de la structure chronologique des données. Le Dynamic Time Warping (DTW) calcule un alignement de coût minimal entre des séries temporelles qui préservent l'ordre chronologique, mais est intrinsèquement aveugle aux décalages spatio-temporels.

L'algorithme STA commence par calculer le transport optimal entre les distributions spatiales à chaque instant. Cela permet d'aligner spatialement les différentes images ou distributions, capturant ainsi les variations spatiales au fil du temps. Ensuite, l'algorithme utilise la distance (DTW) pour aligner les distributions temporelles résultantes obtenues à partir de l'étape précédente. Ainsi, en combinant le transport optimal pour la dimension spatiale et la DTW pour la dimension temporelle, l'algorithme STA permet une analyse complète des distributions spatio-temporelles, telles que celles rencontrées dans les vidéos ou d'autres séquences temporelles de données. L'article conclut que STA est capable d'identifier des clusters spatio-temporels significatifs et qu'il constitue une approche prometteuse pour l'analyse de données spatio-temporelles.[Janati et al.(2019)].

6 Idées d'algorithme basé sur le DTW

But de l'algorithme : cet algorithme permet de transformer n'importe quelle série temporelle de la dimension et forme que l'on veut à partir seulement d'une matrice de dilatation-contraction.

Entrée : -La série temporelle en entrée

- la dimension souhaitée de la série temporelle de sortie (se retrouve dans la dimension de la matrice) - les matrices de dilatation-contraction suivant chaque cluster - la série temporelle cible qui permet de faire le clustering

Sortie : une série temporelle de la dimension souhaitée et de la forme souhaitée

Description de l'algorithme Une des idées qu'on retrouve dans la littérature est d'utiliser l'algorithme de DTW pour catégoriser les séries temporelles. En somme, on choisit une série temporelle dite "cible" qui sera capable de discriminer au mieux les différents comportements du phénomène mesuré. Qu'est-ce que cela veut dire discriminer au mieux ? Selon quelle fonction d'objectif ? Sans parler des différentes variantes d'algorithme qui peuvent relaxer le problème afin de gagner en rapidité. Là encore, ce problème peut nous amener loin, mais comme il ne s'agit pas de notre objectif, nous nous limiterons à une perspective du problème. On fait une étape de clustering le coût total (ou de la distance entre les deux séries temporelles, qui là aussi à différents choix) fournit par l'algorithme DTW. En fonction du cluster, on a une matrice de dilatation-contraction qui lui est associée et qui va nous permettre de dilater et de contracter la série temporelle en entrée.

Critiques et amélioration : on voudrait bien simplifier le nombre de paramètres en entrée de notre algorithme et le rendre plus non supervisé. Par exemple, trouver une méthode mathématique qui nous trouve le nombre de clusters pertinent et qui pour chaque cluster construit une matrice de contraction-dilatation adaptée. Remarque, trouver le nombre de clusters pertinent est plus facile.

Une autre idée consistait à sélectionner une série cible parmi notre ensemble de séries temporelles. Cette série cible aurait permis d'utiliser l'algorithme de DTW sur les autres séries temporelles de l'ensemble de données et d'obtenir plusieurs chemins. Nous aurions ensuite pu utiliser ces chemins pour déterminer les correspondances temporelles entre les différentes séries temporelles. Cependant, un problème se pose quant à la taille du chemin. En effet, les différents chemins auront des longueurs différentes, ce qui rendra impossible le recalage des séries temporelles selon la taille du chemin. Pour illustrer ce problème, imaginez la Figure 13 appliquée à plusieurs séries temporelles où les séries ont été étirées pour que leurs longueurs correspondent à celle du chemin.

7 Conclusion et perspectives

On a vu de multiples applications au transport optimal et de l'algorithme de DTW. Pour résumer le transport optimal s'intéresse aux distributions et répond à la question de déplacer des masses de probabilité de la source vers la cible en minimisant les coûts (matrices des coûts sur le support) mais en assignant toute la masse de l'entrée sur celle de la sortie (matrices de transport sur les densités). Suivant ce qui nous intéresse, on va utiliser le transport optimal de différentes manières, par exemple, on peut s'intéresser aux distances entre deux distributions et faire du *clustering*. Cet outil est donc utilisé pour transformer des densités de probabilités en d'autres densités de probabilité, qu'on soit en discret ou en continu.

Dans le cadre de l'analyse de série temporelle, la méthode la plus adaptée est celle du DTW qui comporte aujourd'hui de multiples variantes pour répondre à une variété de problèmes. On rappelle que l'algorithme de DTW cherche la somme des distances parcourue par le chemin entre deux séries temporelles. Pour résoudre partiellement notre problème de **rectangularisation**, on a réécrit les séries temporelles en fonction de ce chemin. Le problème majeur de cette méthode est que l'on ne peut pas comparer plusieurs séries temporelles, en effet, on aurait un chemin différent de taille différente pour chacune des séries temporelles. Ajoutez à cela que le chemin n'est pas dans la plupart des cas bijectifs.

Pour reprendre notre démarche dans un ordre chronologique, on a pensé que la matrice de transport qui définissait une fonction de transfert d'une distribution à une autre pouvait nous être utile pour transporter une série temporelle à une autre. En effet, à première vue, le transport avait de bonnes propriétés pour rectangulariser : celui de modifier la taille de la série en transportant la distribution de façon à garder le plus d'information.

Cependant, les plus grands freins à l'utilisation du transport optimal est que le temps ne peut pas être traité comme les autres variables, ce n'est pas une variable aléatoire et l'autre grand problème est la discontinuité du transport optimal, i.e le non-respect de la structure de courbe. On a montré que l'application du transport à des problèmes temporels n'était qu'effleurée par la littérature et que son application était souvent délicate. L'une des pistes que l'on propose est de voir la série temporelle non plus comme une courbe, mais comme une série de points où chacun serait lié à son suivant et précédent comme une chaîne. C'est-à-dire qu'on aurait une matrice d'adjacence en plus des données de notre dataset. Avec cela, on espère pouvoir utiliser le transport optimal (peut-être ici la distance de **gromov-wasserstein**) et garder la structure de courbe. Ou alors dans l'optique de trouver un couplage entre des séries temporelles, il faudrait trouver une matrice de transport dans un espace sous la contrainte de respecter une structure de courbe.

Sinon l'une des manières de voir le problème de **rectangularisation** est indirecte, c'est-à-dire qu'on va chercher à poser le problème sous forme de minimisation du coût. Et ensuite, grâce l'interpolation de série temporelle, on peut échantillonner suivant un chemin qui va pondérer plus ou moins fortement les phases de vols.

Bibliographie

- [TD()] Jupyter Notebook Viewer. URL https://nbviewer.org/github/gpeyre/numerical-tours/blob/master/python/optimaltransp_1_linprog.ipynb.
- [Al-Naymat et al.(2012)] G. Al-Naymat et al. Sparsedtw : A novel approach to speed up dynamic time warping. *arXiv*, 2012. URL <http://arxiv.org/abs/1201.2969>.
- [Cuturi(2013)] M. Cuturi. Sinkhorn distances : Lightspeed computation of optimal transportation distances. *arXiv*, 2013. URL <http://arxiv.org/abs/1306.0895>.
- [Cuturi and Blondel(2018)] M. Cuturi and M. Blondel. Soft-dtw : a differentiable loss function for time-series. *arXiv*, 2018. URL <http://arxiv.org/abs/1703.01541>.
- [Desai et al.(2021)] A. Desai et al. Timevae : A variational auto-encoder for multivariate time series generation. *arXiv*, 2021. URL <http://arxiv.org/abs/2111.08095>.
- [François-Poitjean(2014)] François-Poitjean. Dba, 2014. URL <https://github.com/fpoitjean/DBA>. Consulté le 14 avril 2024.
- [Graph Diffusion Wasserstein - CAp 2020(2020)] Graph Diffusion Wasserstein - CAp 2020. Graph diffusion wasserstein - cap 2020, 2020. URL <https://www.youtube.com/watch?v=yI-rwY4PUIE>. Consulté le 20 janvier 2024.
- [Janati et al.(2019)] H. Janati et al. Spatio-temporal alignments : Optimal transport through space and time. *arXiv*, 2019. URL <http://arxiv.org/abs/1910.03860>.
- [Kolouri(2022)] S. Kolouri. Wasserstein embeddings in the deep learning era, 2022. URL <https://www.youtube.com/watch?v=xs9uibP0DGk>. Consulté le 20 janvier 2024.
- [Kolouri et al.(2021)] S. Kolouri et al. Wasserstein embedding for graph learning. *arXiv*, 2021. URL <http://arxiv.org/abs/2006.09430>.
- [Latorre et al.(2023)] Latorre, Liu, Sahoo, and Hoi] F. Latorre, C. Liu, D. Sahoo, and S. C. H. Hoi. OTW : Optimal Transport Warping for Time Series, June 2023. URL <http://arxiv.org/abs/2306.00620>. arXiv :2306.00620 [cs].
- [Meetup ML Rennes(2019)] Meetup ML Rennes. Représentations complexes en séries temporelles pour du ml riemannien, 2019. URL <https://www.youtube.com/watch?v=nHvVjUkNihU>. Consulté le 20 janvier 2024.
- [Mei et al.(2022)] Mei, Huang, Yu, Zhang, and Bennamoun] G. Mei, X. Huang, L. Yu, J. Zhang, and M. Bennamoun. COTReg :Coupled Optimal Transport based Point Cloud Registration, Oct. 2022. URL <http://arxiv.org/abs/2112.14381>. arXiv :2112.14381 [cs].
- [Morel et al.(2018)] M. Morel et al. Time-series averaging using constrained dynamic time warping with tolerance. *Pattern Recognition*, 74 :77–89, 2018. doi : 10.1016/j.patcog.2017.08.015.
- [Peyre(2017)] G. Peyre. Le transport optimal numérique et ses applications, 2017. URL <https://www.youtube.com/watch?v=4FtamHah29M>. Consulté le 20 janvier 2024.
- [Peyré and Cuturi(2020)] G. Peyré and M. Cuturi. Computational optimal transport. *arXiv*, 2020. URL <http://arxiv.org/abs/1803.00567>.
- [Rioux et al.(2024)] Rioux, Goldfeld, and Kato] G. Rioux, Z. Goldfeld, and K. Kato. Entropic Gromov-Wasserstein Distances : Stability and Algorithms, Jan. 2024. URL <http://arxiv.org/abs/2306.00182>. arXiv :2306.00182 [math, stat].
- [S9E10(2023)] S9E10. Transport optimal et machine learning, 2023. URL <https://www.youtube.com/watch?v=3QDDbqr7Tao>. Consulté le 12 février 2024.
- [Salmona et al.(2021)] Salmona, Delon, and Desolneux] A. Salmona, J. Delon, and A. Desolneux. Gromov-Wasserstein Distances between Gaussian Distributions, Apr. 2021. URL <http://arxiv.org/abs/2104.07970>. arXiv :2104.07970 [math].
- [Shen et al.(2021a)] Shen, Feydy, Liu, Curiale, Estepar, Estepar, and Niethammer] Z. Shen, J. Feydy, P. Liu, A. H. Curiale, R. S. J. Estepar, R. S. J. Estepar, and M. Niethammer. Accurate Point Cloud Registration with Robust Optimal Transport, Oct. 2021a. URL <http://arxiv.org/abs/2111.00648>. arXiv :2111.00648 [cs].
- [Shen et al.(2021b)] Z. Shen et al. Accurate point cloud registration with robust optimal transport. *arXiv*, 2021b. URL <http://arxiv.org/abs/2111.00648>.
- [Séjourné et al.(2023)] T. Séjourné et al. Unbalanced optimal transport, from theory to numerics. *arXiv*, 2023. URL <http://arxiv.org/abs/2211.08775>.

- [Tolstikhin et al.(2019a)] Tolstikhin, O., Bousquet, S., Gelly, A., and Schoelkopf, B. Wasserstein Auto-Encoders, Dec. 2019a. URL <http://arxiv.org/abs/1711.01558>. arXiv :1711.01558 [cs, stat].
- [Tolstikhin et al.(2019b)] I. Tolstikhin et al. Wasserstein auto-encoders. *arXiv*, 2019b. URL <http://arxiv.org/abs/1711.01558>.
- [Vayer et al.(2020)] T. Vayer et al. Optimal transport for structured data with application on graphs. 2020.
- [Zhang et al.(2019)] Zhang, O., Lin, R.-S., and Gou, Y. Optimal Transport Based Generative Autoencoders, Oct. 2019. URL <http://arxiv.org/abs/1910.07636>. arXiv :1910.07636 [cs].
- [Zhang et al.(2020)] Z. Zhang et al. Time adaptive optimal transport : A framework of time series similarity measure. *IEEE Access*, 8 :149764–74, 2020. doi : 10.1109/ACCESS.2020.3016529. URL <https://doi.org/10.1109/ACCESS.2020.3016529>.