Transfer Learning

Prince Mezui et Hadrien Bigo-Balland

Contents

1 Introduction **1**

2 État de l'art **2**

3 Méthode proposée **5**
3.1 Méthode DAN 5
3.1.1 Extraction de caractéristiques 6
3.1.2 Couches d'adaptation 6
3.1.3 Classification 7
3.2 Méthode GBRFF 7

4 Expériences et résultats **8**
4.1 Jeux de données utilisés 8
4.1.1 Jeu de données Moons 8
4.1.2 Jeu de données MNIST 10
4.2 Résultats des différentes méthodes 10
4.2.1 Méthode DAN 10
4.2.2 Méthode GBRFF 12

5 Conclusion **13**

A Affichage des résultats pour GBRFF **15**

Abstract

Dans ce rapport, nous parlerons d'une méthode d'apprentissage automatique nommée l'apprentissage par transfert. Dans un premier temps, nous présenterons ce qu'il y a à savoir à son propos : son but, ses spécificités et ses moyens d'utilisation. Par la suite, nous présenterons les expériences que nous avons pu réaliser pour mettre en jeu l'adaptation de domaine ainsi que leurs résultats avant de conclure en les comparant aux travaux déjà effectués sur ce sujet.

1 Introduction

Afin de pouvoir induire des conclusions générales dans différents domaines, il existe une famille de méthodes permettant d'apprendre à partir de quelques données un modèle qui se fera généralisant, on parle d'apprentissage automatique. Cette méthode permet d'automatiser le développement de modèles analytiques au moyen d'algorithmes

d'optimisation. C'est une forme d'intelligence artificielle qui permet, grâce à différents algorithmes, à un système d'apprendre de manière itérative des données pour les décrire et prévoir des résultats en apprenant de données de formation qui génèrent des modèles précis.

Il existe un grand nombre de méthodes d'apprentissage automatique qui ont toutes leurs spécificités et leur but. Cependant, la majorité supposent que les modèles sont formés et testés à l'aide de données tirées d'une distribution fixe. Cette hypothèse bien pratique permet d'induire des conclusions à partir de données sources et de les tester sur d'autres jeux de données en ayant la garantie que l'erreur d'apprentissage empirique d'un modèle est proche de son erreur réelle.

En pratique, cette hypothèse peut être handicapante. On pourrait reprendre l'exemple du filtre anti-spam formé à partir d'une grande collection de courriels reçus sur une boîte pro (le domaine source) que nous souhaitons adapter à une boîte mail personnelle (le domaine cible). Sachant que les boîtes mails reçoivent leurs emails de façon unique et indépendante, l'hypothèse selon laquelle les 2 domaines ont une distribution identique n'est pas respectée.

Un autre exemple connu est celui d'une voiture autonome entraînée à New York qui excellera à détecter des piétons dans les rues new-yorkaises. Mais qui ne pourra pas être aussi performante à Paris par exemple car, bien que rien ne différencie un new-yorkais d'un parisien le changement d'environnement perturbera l'environnement. En effet, l'architecture des bâtiments, la signalisation routière, et l'agencement des routes et trottoirs sont bien différents dans les deux métropoles.

Des techniques d'apprentissage par transfert ont été proposées pour traiter spécifiquement ce type de situations. Les algorithmes d'apprentissage par transfert cherchent à appliquer les connaissances acquises lors d'une tâche précédente à une nouvelle tâche, mais liée. L'intuition derrière l'apprentissage par transfert découle de la capacité des humains à étendre ce qui a été appris dans un contexte à un nouveau contexte. Dans le domaine de l'apprentissage automatique, les bénéfices de l'apprentissage par transfert sont nombreux ; moins de temps est consacré à l'apprentissage de nouvelles tâches, moins d'informations sont requises des experts (généralement humains) et davantage de situations peuvent être gérées efficacement, ce qui rend le modèle appris plus robuste. Ces avantages potentiels ont conduit les chercheurs à appliquer des techniques d'apprentissage par transfert à de nombreux domaines avec plus ou moins de succès.

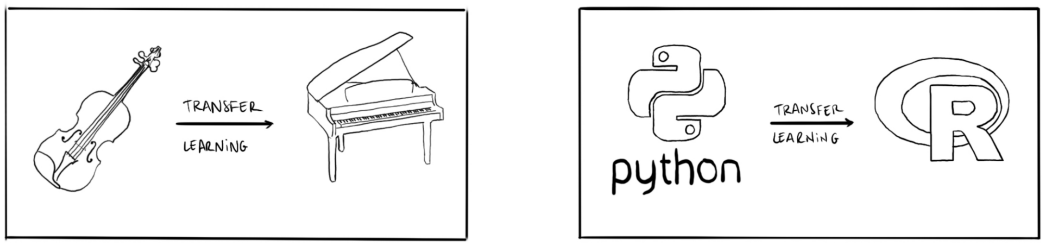

Figure 1: Exemples de transferts d'apprentissage naturels.

2 État de l'art

Définition 2.1. Un domaine $\mathcal{D} = \{\mathcal{X}, P(X)\}$ est composé de deux parties, un espace de features $\mathcal{X}$ et une distribution marginale $P(X)$, où $X = \{x_1, \ldots, x_n\}$ désigne un jeu de

n observations $x_i \in \mathcal{X}$ décrites dans l'espace de features.

Remarque 2.1. Les features peuvent être quantitatifs (la valeur d'un pixel, le prix d'une maison,...) ou qualitatifs (la forme d'un visage, le sentiment derrière un texte,...).

Définition 2.2. Pour un domaine $\mathcal{D}$ donné, une tâche $\mathcal{T} = \{\mathcal{Y}, f\}$ est composée d'un espace d'étiquettes $\mathcal{Y}$ et d'une fonction de décision $f : \mathcal{X} \to \mathcal{Y}$ apprise à partir des observations et de leurs étiquettes correspondantes.

Remarque 2.2. Souvent, pour entraîner le modèle on part d'observations dans l'espace de features munies de leurs étiquettes correspondantes, i.e. $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ où $x_i \in \mathcal{X}$ et $y_i \in \mathcal{Y}$. On peut voir les étiquettes comme des classes dans un problème de classification par exemple.

Remarque 2.3. Typiquement, trouver une fonction de décision revient au même que de trouver les probabilités conditionnelles $P(y \mid x)$ où $x \in \mathcal{X}$ et $y \in \mathcal{Y}$.

Définition 2.3. Étant données des observations correspondants à $N^S \in \mathbb{N}^+$ domaines et tâches sources (i.e., $\{(\mathcal{D}_{S_i}, \mathcal{T}_{S_i}) \mid i = 1, \ldots, N^S\}$), et des observations correspondant à $N^T \in \mathbb{N}^+$ domaines et tâches cibles (i.e., $\{(\mathcal{D}_{T_j}, \mathcal{T}_{T_j}) \mid j = 1, \ldots, N^T\}$), les méthodes de *transfer learning* utilisent les connaissances sur les domaines sources pour améliorer la performance des fonctions de décisions apprises $f^{T_j}, j = 1, \ldots, N^S$ sur les domaines cibles.

Remarque 2.4. Dans la suite, on va considérer le cas où $N^S = N^T = 1$ (apprentissage à une source).

Intuitivement, plus le domaine source et le domaine cible ont de points communs, plus la généralisation d'un à l'autre semble réalisable. Par exemple, savoir jouer du violon semble inutile pour apprendre à conduire.

Définition 2.4. Une méthode de transfer learning est dite homogène lorsque les domaines source et cible ont les même espaces de features et d'étiquettes, i.e. $\mathcal{X}^S = \mathcal{X}^T$ et $\mathcal{Y}^S = \mathcal{Y}^T$. Dans le cas contraire, elle est dite hétérogène, et on peut par exemple faire intervenir des procédures d'adaptation d'espace de features pour y palier.

Dans la suite, on va s'intéresser seulement aux transferts homogènes.

Les méthodes de transfer learning que l'on va explorer dans cette partie utilisent un espace latent de plus faible dimension pour faire le lien entre deux domaines.

Principe 2.1. *Partant d'observations X^S munies d'étiquettes Y^S, on cherche à obtenir les étiquettes Y^T correspondantes à un vecteur d'observation X^T qui suit une distribution marginale différente de celle de X^S. L'objectif est de trouver un espace de features, dit espace latent, qui sera de plus faible dimension, et tel que la distance entre les distributions marginales $F(X^S)$ et $F(X^T)$ dans cet espace soit minimale. Si les probabilités conditionnelles $P(Y^S \mid F(X^S))$ et $P(Y^T \mid F(X^T))$ sont proches, alors la fonction de décision apprise avec $F(X^S)$ et Y^S pourra être utilisée pour prédire les étiquettes Y^T directement à partir de $F(X^T)$.*

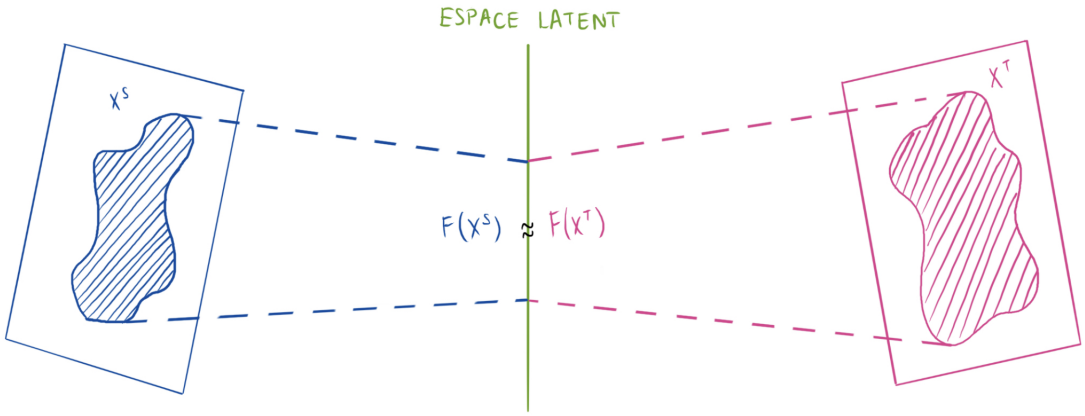

Figure 2: L'espace latent est tel que la distance entre les distributions marginales $F(X^S)$ et $F(X^T)$ dans cet espace soit minimale.

Les méthodes traditionnelles de réduction dimensionnelle utilisent une projection des données sur un espace latent mais ne garantissent pas que les distributions des différentes données soient similaires dans l'espace latent. [3] propose une nouvelle méthode de réduction dimensionnelle adaptée au transfer learning.

Dans la suite on prend (X, d) un espace métrique.

Définition 2.5. Soit (X, d) un espace métrique. On appelle noyau reproductif d'un espace de Hilbert $\mathcal{H}$ une fonction

$$\begin{aligned} k : X \times X &\to \mathbb{C} \\ (s,t) &\mapsto K(s,t) \end{aligned}$$

telle que

1. $\forall x \in X, K(\cdot, x) \in \mathcal{H}$
2. $\forall x \in X, \forall f \in \mathcal{H}, \langle f, K(\cdot, x)\rangle_{\mathcal{H}} = f(x)$ (propriété de reproduction).

De ces deux propriétés s'ensuit que

$$\forall (x,y) \in X \times X, K(x,y) = \langle K(\cdot, y), K(\cdot, x)\rangle_{\mathcal{H}}.$$

Définition 2.6. Un noyau reproductif k sur (X, d) est dit universel si l'espace de fonctions induites par k est dense dans l'ensembles des fonctions continues sur X.

Définition 2.7. Un espace de Hilbert de fonctions complexes qui possède un noyau reproductif (resp. reproductif universel) est appelé un *reproducing kernel Hilbert space (RKHS)* (resp. *universal RKHS*).

Définition 2.8. [3] Soit $X = \{x_1, \ldots, x_{n_1}\}$ et $Y = \{y_1, \ldots, y_{n_2}\}$ des ensembles de variables aléatoires de distributions $\mathcal{P}$ et $\mathcal{Q}$. On estime la distance entre $\mathcal{P}$ et $\mathcal{Q}$ par *maximum mean discrepancy* de la manière suivante :

$$\text{MMD(X,Y)} = \sup_{\|f\|_{\mathcal{H}} \leq 1} \left(\frac{1}{n_1} \sum_{i=1}^{n_1} f(x_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} f(y_i) \right)$$

qui est équivalent à

$$\mathrm{MMD(X,Y)} = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} k(\cdot, x_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} k(\cdot, y_i) \right\|_{\mathcal{H}}$$

où $\mathcal{H}$ est un *universal RKHS* de noyau k.

Propriété 2.1. *$Dist(X, Y)$ tend vers 0 quand n_1, n_2 tendent vers l'infini si, et seulement si $\mathcal{P} = \mathcal{Q}$.*

3 Méthode proposée

L'adaptation de domaine non adaptée et adaptée sont deux approches bien différentes de l'adaptation de domaine :

Adaptation de domaine non adaptée : dans l'adaptation de domaine non adaptée, le modèle est formé sur le domaine source et directement appliqué au domaine cible sans aucune technique d'adaptation de domaine. Cette approche est simple et facile à mettre en œuvre, mais peut ne pas fonctionner correctement dans le domaine cible en raison du changement de domaine.

Adaptation de domaine adaptée : dans l'adaptation de domaine adaptée, le modèle est entraîné sur le domaine source et adapté au domaine cible à l'aide de diverses techniques telles que l'entraînement contradictoire de domaine, l'inversion de gradient ou l'utilisation d'une perte de confusion de domaine. Ces techniques visent à minimiser l'écart de domaine entre les domaines source et cible afin d'améliorer les performances du modèle dans le domaine cible. L'adaptation de domaine adaptée est plus complexe que l'adaptation de domaine non adaptée mais peut conduire à de meilleurs résultats dans le domaine cible.

On va s'intéresser ici à deux méthodes d'adaptation de domaine adaptée, DAN et GBRFF.

3.1 Méthode DAN

La Deep Adaptation Network (DAN) est une méthode mettant en jeu un type de réseau neuronal profond conçu pour l'adaptation non supervisée à un domaine. L'idée de DAN est d'apprendre une représentation partagée des caractéristiques qui est invariante aux différences entre les domaines source et cible, tout en préservant l'information discriminante pour la classification. On voit tout de suite aisément l'idée d'espace latent sur lequel notre algorithme se basera. Nous tenterons donc de l'implémenter grâce à la bibliothèque Pytorch de Python et nous verrons de par ces résultats si elle peut être une bonne approche pour répondre aux problématiques de l'adaptation de domaine.

Nous allons présenter l'architecture globale de l'implémentation de notre algorithme DAN avec des pseudo codes et/ou les expressions des fonctions de calculs utilisées au besoin. Les 3 principales étapes de notre algorithme sont :

1. L'extraction de caractéristiques qui sera partagée entre les domaines source et cible, de sorte que les caractéristiques extraites des deux domaines sont alignées.

2. Les couches d'adaptation utilisées pour apprendre les caractéristiques invariantes du domaine. Elles sont conçues pour minimiser la différence entre les caractéristiques

des domaines source et cible, tout en préservant les informations discriminantes pour la classification.

3. La classification qui sert de dernière couche de l'algorithme et qui se sert des caractéristiques adaptées comme entrée et fait des prédictions basées sur l'information discriminante apprise.

L'implémentation de ces 3 étapes a nécessité des choix de méthodes et pour se faire, nous avons dû nous baser sur tout ce que nous avons pu trouver dans la littérature et faire des choix en fonction de la facilité d'implémentation et des résultats obtenus.

3.1.1 Extraction de caractéristiques

Pour réaliser cette tâche, nous nous sommes servi de 2 classes implémentées grâce à la bibliothèque PyTorch et à ses classes et modules. La première classe qui définit un réseau de neurones avec deux couches convolutives, deux couches de regroupement maximal et deux couches entièrement connectées (linéaires) à pour but d'extraire des caractéristiques d'une image d'entrée et de les retourner ensuite. La seconde définit quant à elle un réseau de neurones avec une seule couche entièrement connectée (linéaire). Son but est de classer une image d'entrée en fonction des caractéristiques extraites par le réseau précédent et de renvoyer les logits (sorties brutes du réseau) avant qu'ils ne soient transformés en probabilités par une fonction softmax.

3.1.2 Couches d'adaptation

Pour minimiser la différence entre les domaines source et cible, tout en préservant les informations discriminantes pour la classification, nous avons choisi d'implémenter une fonction mettant en jeu la version kernelisée de l'écart moyen maximal encore appelée Maximum Mean Discrepancy (MMD).

```
> def pairwise_distance(x, y)
```

Cette fonction calcule la distance euclidienne au carré entre chaque paire de vecteurs le long de la première dimension du tenseur de sortie et renvoie le tenseur de forme résultant (batch size, batch size).

```
> def gaussian_kernel_matrix(x, y, sigmas)
```

La fonction noyau gaussienn est définie comme suit :

$$K(x, y) = \exp(-\|x - y\|^2/2\sigma^2) \text{ et } K(x) = \exp(-\|x\|^2/2\sigma^2) \tag{1}$$

où σ est la largeur du noyau et $\|x - y\|^2$ est le carré de la distance euclidienne entre x et y.

```
> def maximum_mean_discrepancy(x, y, kernel= gaussian_kernel_matrix)
```

La MMD est calculé à l'aide de la fonction de matrice du noyau gaussien par défaut et vaut :

$$\text{MMD}(x, y) = \sum_x K(x) + \sum_y K(y) - 2\sum_x \sum_y K(x, y) \tag{2}$$

```
> def mmd_loss(source_features, target_features)
    loss_value=mmd(source_features,target_features,kernel=
    ↪ gaussian_kernel)
```

3.1.3 Classification

La tâche de classification est réalisée grâce aux deux précédentes. On part des modèles de réseau neuronal pour l'extracteur de caractéristiques communes et du classificateur source définis lors de notre première étape ensuite, on parcourt les chargeurs de données source et cible (MNIST et MNIST-M) et, on évalue le modèle sur chaque lot ; le nombre de lots étant un paramètre préfixé à 'epochs = 6' sur lequel on ne fera pas de tuning au même titre que tous nos autres hyperparamètres qu'on préfixe simplement en fonction de nos tests et des indications sur les standards recommandés dans la littérature.

Pour le jeu de données source, la fonction applique l'extracteur de caractéristiques communes et le classificateur source aux données d'entrée, calcule les étiquettes prédites et les compare aux étiquettes de vérité terrain pour déterminer le nombre de prédictions correctes. Pour le jeu de données cible, la fonction fait la même chose en utilisant l'extracteur de caractéristiques communes et le classificateur source.

On voit donc que l'algorithme DAN est entraîné en minimisant la combinaison de la perte de classification sur le domaine source et la perte d'adaptation de domaine entre les domaines source et cible. L'objectif est de trouver un équilibre entre ces deux pertes, de sorte que le classificateur final soit à la fois discriminant et invariant aux différences entre les domaines source et cible. DAN est un moyen de résoudre le problème du décalage de domaine dans l'adaptation de domaine, en apprenant une représentation partagée des caractéristiques qui est invariante aux différences entre les domaines source et cible, tout en préservant l'information discriminante pour la classification.

3.2 Méthode GBRFF

La méthode GBRFF (Gradient Boosting with Random Fourier Features) combine les deux méthodes de Gradient Boosting (GB) et les kernel Random Fourier Features (RFF) pour apprendre un classifieur de type somme de noyaux convenant à une situation spécifique. L'idée est de construire plusieurs noyaux de la forme

$$k_{q^t}(\mathbf{x}^t - \mathbf{x}) = \cos(\omega \cdot (\mathbf{x}^t - \mathbf{x})), \tag{3}$$

puis d'apprendre un classifieur de la forme

$$\forall \mathbf{x} \in \mathbb{R}^d, \operatorname{sign}\left(H^0(x) + \sum_{t=1}^{T} \alpha^t k_{q^t}(\mathbf{x}) \right), \tag{4}$$

où H^0 est un classifieur initial. Ici, on se propose de personnaliser l'algorithme de GBRFF donné par Metzler, Gautheron et al dans [6] pour des problèmes d'adaptation non supervisée à un domaine.

Nous allons maintenant présenter l'architecture de notre algorithme GBRFF. L'algorithme de base est celui de [6], auquel on a rajouté un terme de distance entre les distributions, ici en rouge :

où d est la dimension des données utilisées ($d = 2$ pour Moons), et MMD est la fonction définie dans une partie précédente.

Algorithm 1 GBRFF

Entrées : Ensemble d'entraînement source $(\mathbf{x}^S, y^S) = \left\{(\mathbf{x}_i^S, y_i^S)\right\}_{i=1}^{n^S}$; Ensemble d'entraînement cible $\mathbf{x}^S = \left\{\mathbf{x}_i^T\right\}_{i=1}^{n^T}$, Nombre d'itérations T; Paramètres γ, λ, λ_b et λ_ω

Sortie : sign $\left(H^0(\mathbf{x}) + \sum_{t=1}^{T} \alpha^t k_{q^t}(\mathbf{x})\right)$

for $t = 1, \ldots, T$ **do**

$\forall i = 1 \ldots n,\ w_i = \exp\left(-y_i H^{t-1}(\mathbf{x}_i)\right)$

$\forall i = 1 \ldots n,\ \tilde{y}_i = y_i w_i$

Draw $\omega \sim \mathcal{N}(0, 2\gamma)^d$

$b^t = \text{argmin}_{b\in[-\pi,\pi]} \dfrac{1}{n}\sum_{i=1}^{n} \exp\left(-\tilde{y}_i \cos(\omega \cdot \mathbf{x}_i - b)\right) + \lambda_b \text{ MMD}\left(\mathbf{x}^S, \mathbf{x}^T\right)$

$\omega^t = \text{argmin}_{\omega\in\mathbb{R}^d}\ \lambda\|\omega\|_2^2 + \dfrac{1}{n}\sum_{i=1}^{n} \exp\left(-\tilde{y}_i \cos(\omega \cdot \mathbf{x}_i - b^t)\right) + \lambda_\omega \text{ MMD}\left(\mathbf{x}^S, \mathbf{x}^T\right)$

$\alpha^t = \dfrac{1}{2} \ln \dfrac{\sum_{i=1}^{n}(1 + y_i \cos(\omega^t \cdot \mathbf{x}_i - b^t) w_i)}{\sum_{i=1}^{n}(1 - y_i \cos(\omega^t \cdot \mathbf{x}_i - b^t) w_i)}$

$\forall i = 1, \ldots, n,\ H^t(\mathbf{x}_i) = H^{t-1}(\mathbf{x}_i) + \alpha^t \cos(\omega \cdot \mathbf{x}_i - b^t)$

end for

On a testé l'algorithme avec trois versions de la MMD :

$$\text{MMD}_1(\mathbf{x}^S, \mathbf{x}^T) = \left(\frac{1}{n^S}\sum_{i=1}^{n^S} \cos(\omega \cdot \mathbf{x}_i^S - b) - \frac{1}{n^T}\sum_{i=1}^{n^T} \cos(\omega \cdot \mathbf{x}_i^T - b)\right)^2, \tag{5}$$

$$\text{MMD}_2(\mathbf{x}^S, \mathbf{x}^T) = \left(\frac{1}{n^S}\sum_{i=1}^{n^S} \cos(\omega \cdot \mathbf{x}_i^S - b)^2 - \frac{1}{n^T}\sum_{i=1}^{n^T} \cos(\omega \cdot \mathbf{x}_i^T - b)^2\right)^2, \tag{6}$$

$$\text{MMD}_3(\mathbf{x}^S, \mathbf{x}^T) = \text{MMD}_1(\mathbf{x}^S, \mathbf{x}^T) + \text{MMD}_2(\mathbf{x}^S, \mathbf{x}^T), \tag{7}$$

qui représentent en quelque sorte les distances "en espérance", "en variance" et une combinaison des deux.

Quand aux paramètres, on a "tuné" γ et λ par cross-validation, et on a tuné λ_ω et λ_b à la main (mais on aurait pu le faire avec le code implémenté si on disposait de plus de puissance de calcul).

4 Expériences et résultats

4.1 Jeux de données utilisés

Pour commencer les expérimentations, nous avons cherché à trouver des jeux de données appropriés à notre problématique d'adaptation de domaine.

4.1.1 Jeu de données Moons

Nous avons choisi premièrement la base de données "Moons", qui est simple à mettre en place et à customiser, en plus d'être couramment utilisé pour évaluer les performances

d'algorithmes de classification. Cette base de données est fournie par la bibliothèque Scikit-learn, et est composée de points bidimensionnels appartenant à deux demi-cercles entrelacés qui définissent les deux classes du jeu de données.

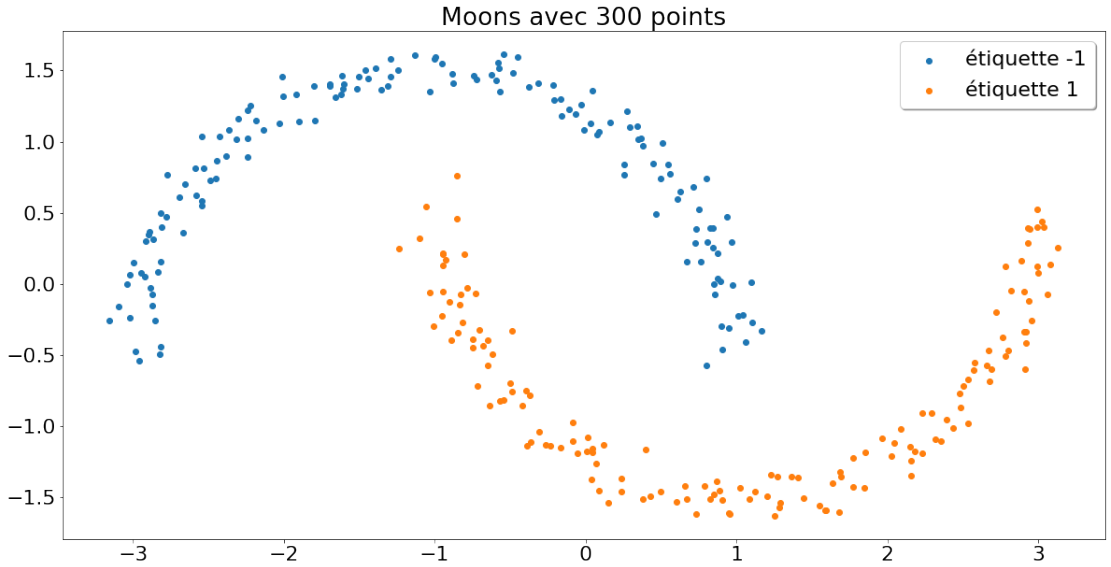

Figure 3: Un échantillon du jeu de données source Moons avec 300 points.

Pour notre étude sur l'adaptation de domaines, nous avons modifié la distribution des données Moons sources en effectuant une rotation pour construire les données cibles.

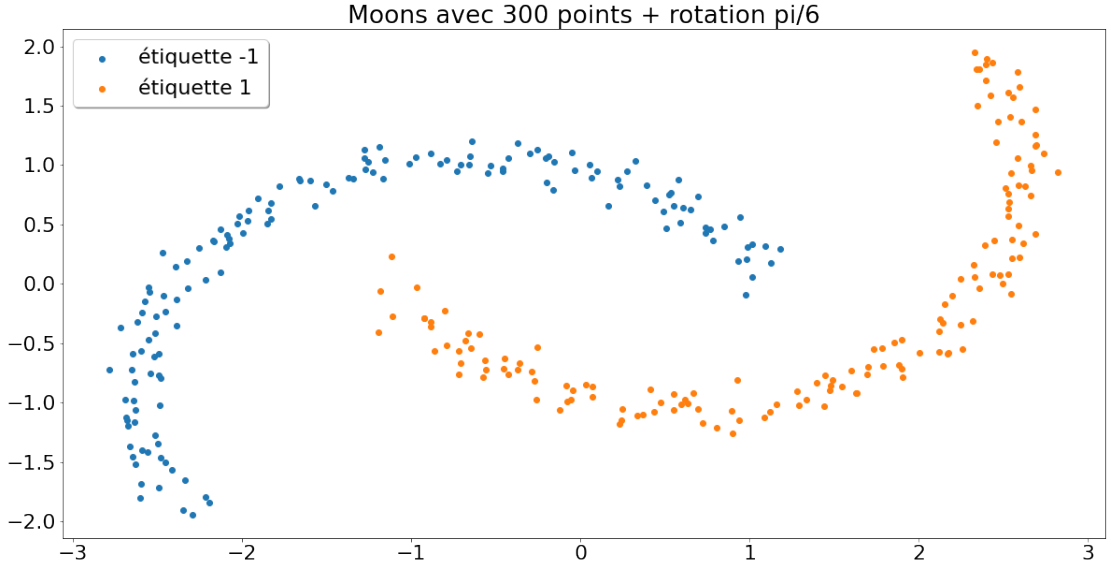

Figure 4: Un échantillon du jeu de données cible Moons auquel on a appliqué une rotation de $\pi/6$ avec 300 points.

L'angle de la rotation est un paramètre qui peut être modifié, et il sera intéressant de regarder les performances de nos modèles en fonction de l'angle de la rotation. Cette modification nous permet de simuler des changements dans les propriétés des données, tout en maintenant la structure générale de la base de données originale, ce qui en fait une base de données convenant à l'adaptation de domaines.

4.1.2 Jeu de données MNIST

Ensuite, nous avons sélectionné la base de données MNIST (pour 'Mixed National Institute of Standards and Technology'), qui est une base de données de chiffres écrits à la main qui est très utilisée en apprentissage automatique. Elle regroupe 60000 images d'apprentissage et 10000 images de test. Ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté.

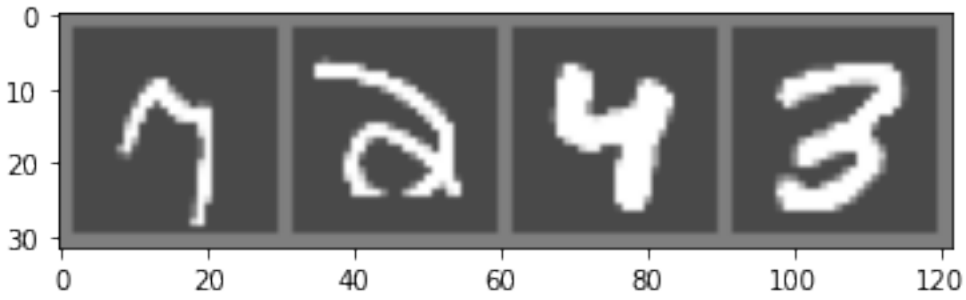

Figure 5: Un échantillon du jeu de données source MNIST.

Cette base constitue notre domaine source et a également permis de générer la base MNIST-M ; notre domaine cible. Elle est créée par Iaroslav Ganin (ganin.net) en combinant les chiffres de la précédente base avec en arrière plan les patchs extraits aléatoirement de BSDS500 ; un grand jeu de données de photos en couleurs d'images naturelles. Elle contient 29001 images d'apprentissage et 90001 images de test.

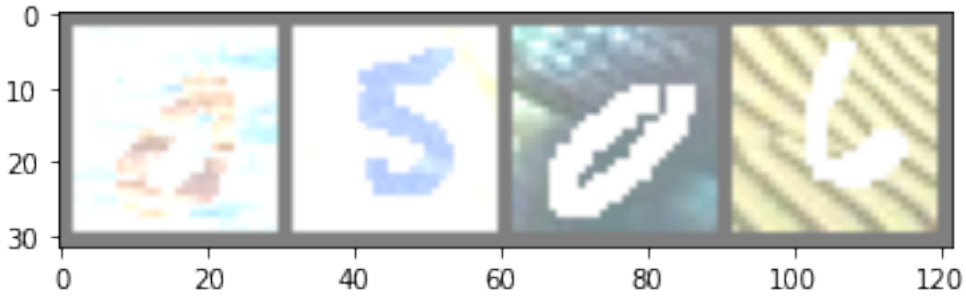

Figure 6: Un échantillon du jeu de données cible MNIST-M.

4.2 Résultats des différentes méthodes

4.2.1 Méthode DAN

Pour ce qui est des résultats obtenus, nous pouvons être plus ou moins satisfaits bien que la marge d'amélioration de ceux-ci reste très grande.

L'implémentation de notre algorithme DAN a été une réussite et nous pouvons dans un premier temps présenter à quel point notre algorithme a été en mesure de former cet espace latent aux domaines source et cible à travers ce graphique qui présente les représentations des caractéristiques des données des domaines source et cible dans un espace 2D. Idéalement, après l'adaptation du domaine, les données du domaine cible doivent être bien alignées avec les données du domaine source dans l'espace des fonctionnalités. Cela signifierait que le modèle s'est adapté avec succès au domaine cible et peut

bien fonctionner sur celui-ci. Obtenir cet idéal sera évidemment une tâche complexe à mettre en œuvre mais, du fait de l'adaptation entre source et cible qui établit dans notre algorithme on pourrait avoir des résultats "viables". En effet, cette phase d'adaptation justifie que la méthode DAN appartienne bien à la branche de l'adaptation de domaine adaptée.

Avec les graphiques suivants, nous présenterons 2 choses :

- La qualité de notre algorithme a créer cet espace latent ;
- La comparaison qu'on pourrait faire entre notre algorithme et la représentation qu'on obtiendrait avec une méthode qui ne mettrait pas en jeu d'adaptation entre les domaines source et cible. Ici, nous proposons par exemple la technique t-SNE (t-Distributed Stochastic Neighbor Embedding) pour réduire les représentations de caractéristiques de haute dimension des ensembles de données MNIST et MNIST-M en représentations 2D. On aura donc ici aucun rapprochement entre les 2 domaines et on se classera donc parfaitement dans le cadre d'une méthode d'adaptation de domaine non adaptée.

On peut afficher ce qu'on obtient avec DAN et avec t-SNE. Les points rouges représentent les données du domaine source, tandis que les points bleus représentent les données du domaine cible.

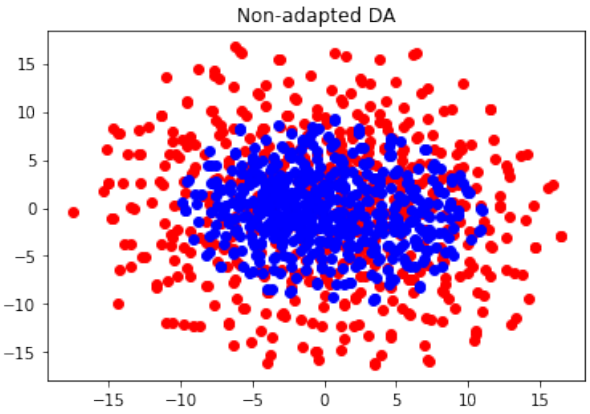

Figure 7: Résultats avec t-SNE.

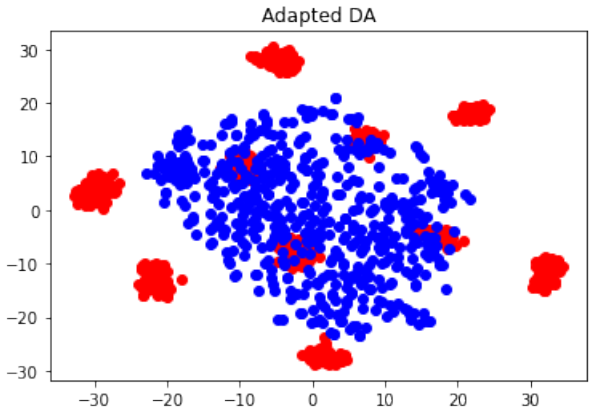

Figure 8: Résultats avec DAN.

- Figure 7 : On peine à tirer des conclusions remarquables. Nous n'observons aucun trait particulier tant sur le source que sur le cible. Cependant, il apparaît bien qu'aucune superposition particulière n'est observable.
- Figure 8 : Avant d'interpréter plus en profondeur ce graphique, nous pouvons nous rendre compte que le domaine source (représenté en rouge) présente clairement les 10 classes d'études ; les classes ici étant représentées par tous les chiffres de 0 à 9. L'idéal auquel nous faisions allusion plus tôt impliquerait que la représentation du domaine source présente la même observation avec une superposition notable avec les classes du source. Mais, ces résultats restent bel et bien satisfaisants car ils permettent des résultats bien meilleurs que ceux donnés par une approche sans adaptation entre les domaines. D'ailleurs on pourrait dire que la représentation du domaine cible est plus disparate et moins regroupée en bloc en comparaison avec la figure 7.

Ici nous voyons que la perfection recherchée est loin d'avoir été atteinte. On peut donc conclure que le modèle aurait besoin d'un réglage supplémentaire. En prenant en compte un tuning des hyperparamètres et un approfondissement du rapprochement des domaines effectué lors de l'étape d'extraction des caractéristiques de l'algorithme nous serions assurément en mesure d'obtenir un meilleur rapprochement des domaines et de ce fait, une meilleure précision de prédiction sur le domaine cible.

Avec les hyper paramètres choisis, nous obtenons une précision de 0.99 sur le dataset cible et de 0.55 sur le dataset cible. En ce qui concerne la loss de notre modèle, nous avons pu la minimiser jusqu'à obtenir une loss de notre fonction MMD avoisinant 0.08 ce qui est également satisfaisant.

Les graphiques suivants présentent l'évolution de la précision de prédiction lors de notre étude d'un epoch à l'autre et l'évolution des pertes (loss) de notre modèle.

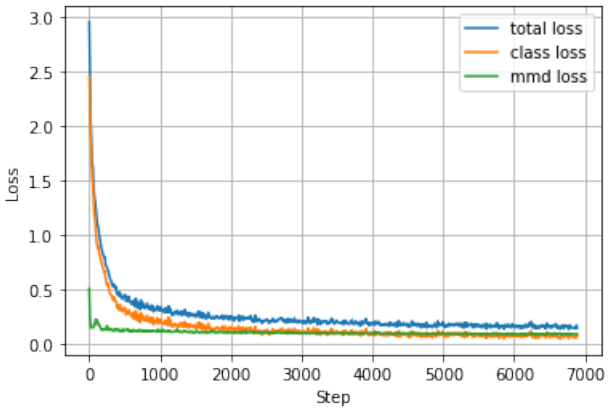

Figure 9: Loss

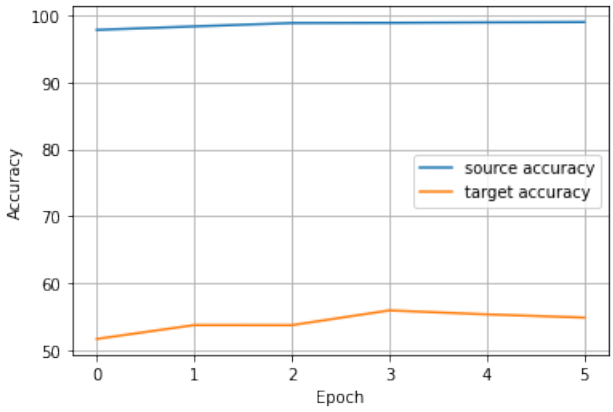

Figure 10: Accuracy

4.2.2 Méthode GBRFF

Les résultats obtenus avec la méthode GBRFF sont encourageants. L'implémentation, basée sur les codes de Léo Gautheron, a ici aussi été une réussite et nous voulons souligner à quel point l'expérimentation est encore possible avec les codes fournis.

Quoique nous soyons enthousiastes des résultats obtenus, il y a encore une nette marge de progression. On s'est vite aperçu que les paramètres λ_b et λ_ω influaient davantage sur différents angles en fonction de leurs valeurs. Une approche par cross-validation est donc ici de mise, permettant de sélectionner à chaque fois les bons paramètres correspondant à l'angle voulu. Ici, par faute de puissance de calcul, nous n'avons pu faire qu'une cross-validation limitée sur ces deux paramètres.

Avec les hyper-paramètres choisis, pour le domaine cible des Moons tourné de $\pi/5$ radians, on obtient une précision de 78.6 sans adaptation de domaine et de 97.7 avec adaptation de domaine MMD_3. C'est plus que satisfaisant. DALC, une autre méthode d'adaptation de domaines sans deep learning, donne une accuracy de 90.7 dans cette configuration.

Le graphique en annexe présente l'évolution de la précision de la prédiction (accuracy) en fonction de l'angle de rotation du dataset cible pour les méthodes considérées. On est loin de doubler l'accuracy, mais on voit que les différentes méthodes correspondantes aux différentes MMD utilisées présentent des "points forts" améliorant les méthodes déjà existantes, et aussi des "points faibles". Visualisons cela. Sur le graphique suivant, on

a représenté les accuracy en fonction de l'angle de rotation du dataset cible pour trois méthodes : DALC, GBRFF avec la MMD_2 et GBRFF avec la MMD_3.

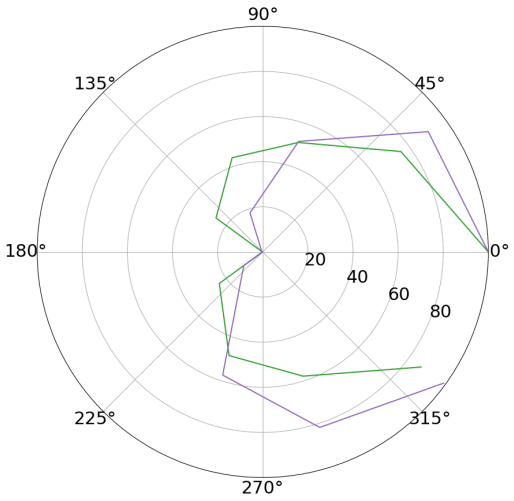

Figure 11: Accuracy de DALC et GBRFF avec la MMD_2

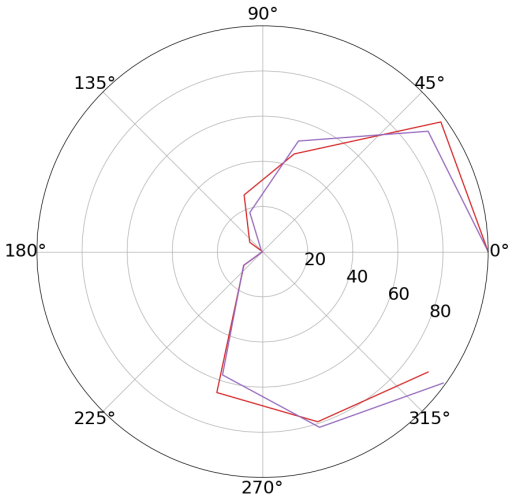

Figure 12: Accuracy de DALC et GBRFF avec la MMD_3

Force est de constater que la MMD_2 donne de meilleurs résultats autour de 120 degrés, mais des résultats médiocres autour de 35 degrés, alors que MMD_3 donne de bons résultats autour de 35 degrés mais des résultats moins bons autour de 120 degrés. On pourrait donc imaginer, si l'on dispose de la puissance de calcul adéquat, rajouter des paramètres qui sélectionnerait à chaque fois par cross-validation la MMD qui donne les meilleurs résultats.

En couplant cela avec une plus puissante cross-validation sur les paramètres λ_b et λ_ω, on peut réalistiquement s'attendre à des résultats bien meilleurs.

5 Conclusion

En conclusion, nous sommes satisfaits des résultats obtenus dans ce travail qui a permis de mettre en perspective les deux modèles d'adaptation de domaines en machine learning, à savoir GBRFF et DAN. Les résultats montrent que les deux modèles peuvent être relativement efficaces dans certaines situations.

Néanmoins, il reste encore du travail à faire pour améliorer ces modèles. Les pistes évoquées dans le rapport peuvent conduire à des améliorations significatives.

De plus, nous pensons que combiner les deux méthodes, GBRFF et DAN, pourrait également être une piste intéressante à explorer pour améliorer les performances de l'adaptation de domaines en machine learning. En utilisant le meilleur des deux modèles, nous pourrions peut-être obtenir des résultats encore meilleurs et plus fiables.

En somme, nous pensons que ce travail peut aider à ouvrir de nouvelles voies de recherche pour améliorer ces modèles.

Bibliographie

[1] Fuzhen Zhuang et al. *A Comprehensive Survey on Transfer Learning*. 2019. DOI: 10.48550/ARXIV.1911.02685. URL: https://arxiv.org/abs/1911.02685.

[2] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[3] Sinno Jialin Pan, James T Kwok, Qiang Yang, et al. "Transfer learning via dimensionality reduction." In: *AAAI*. Vol. 8. 2008, pp. 677–682.

[4] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[5] Ingo Steinwart. "On the influence of the kernel on the consistency of support vector machines". In: *Journal of machine learning research* 2.Nov (2001), pp. 67–93.

[6] Léo Gautheron et al. "Landmark-based ensemble learning with random Fourier features and gradient boosting". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2021, pp. 141–157.

[7] Pascal Germain et al. "A PAC-Bayesian approach for domain adaptation with specialization to linear classifiers". In: *International conference on machine learning*. PMLR. 2013, pp. 738–746.

[8] Pascal Germain et al. "PAC-Bayes and domain adaptation". In: *Neurocomputing* 379 (2020), pp. 379–397. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.10.105. URL: https://www.sciencedirect.com/science/article/pii/S0925231219315486.

[9] Ievgen Redko et al. "A survey on domain adaptation theory: learning bounds and theoretical guarantees". In: *arXiv preprint arXiv:2004.11829* (2020).

[10] Walter Rudin. *Fourier analysis on groups*. Courier Dover Publications, 2017.

[11] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems* 20 (2007).

[12] Ekin Tiu. *Understanding Latent Space in Machine Learning*. https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d. [Online]. 2020.

[13] Feuz and Cook. *Transfer Learning across Feature-Rich Heterogeneous Feature Spaces via Feature-Space Remapping (FSR)*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4804893/. [Online]. 2023.

[14] Cuthbert Cai. *GitHub*. https://github.com/CuthbertCai/pytorch_DAN. [Online]. 2023.

[15] Mingsheng Long and Jianmin Wang. "Learning Transferable Features with Deep Adaptation Networks". In: *CoRR* abs/1502.02791 (2015). arXiv: 1502.02791. URL: http://arxiv.org/abs/1502.02791.

A Affichage des résultats pour GBRFF

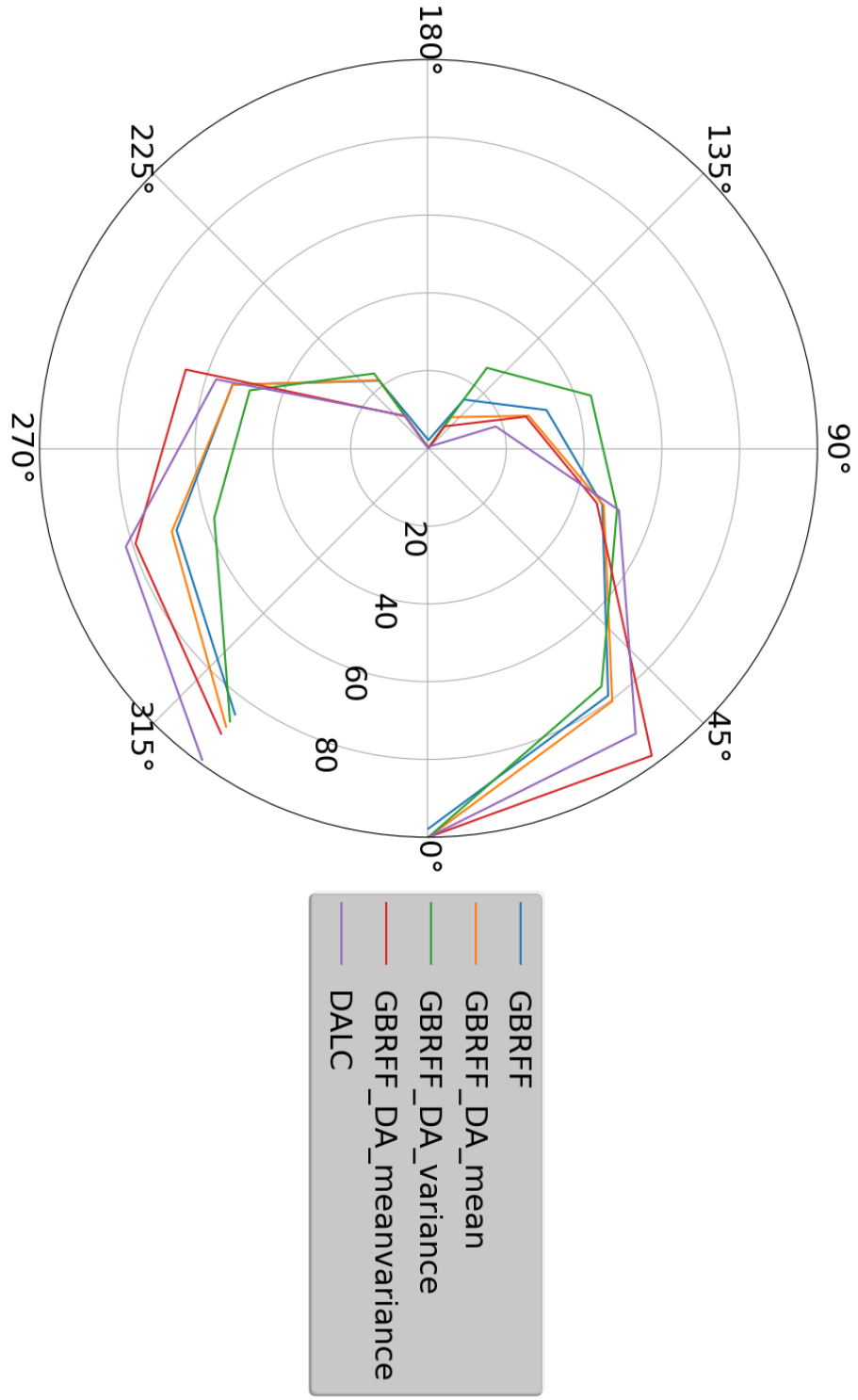

Figure 13: Évolution de l'accuracy en fonction de l'angle de rotation du dataset cible. DA mean : MMD_1, DA variance : MMD_2, DA meanvariance : MMD_3.