Adaptation de domaine en régression par alignement de décompositions non-négatives

Mohamad DHAINI[1,2], Maxime BERAR[1], Paul HONEINE[1], Antonin VAN EXEM[2]

[1]Université de Rouen Normandie, LITIS, 76000 Rouen, France

[2]Tellux, 76650 Petit-Couronne, France

mohamad.dhaini@univ-rouen.fr

Résumé – Les approches d'adaptation de domaine cherchent à généraliser les connaissances acquises dans un domaine source étiqueté à un autre domaine cible non étiqueté. La plupart des approches d'apprentissage profond avec adaptation de domaine visent la tâche de classification. Parmi les modèles profonds existant pour les tâches de régression, l'adaptation de domaine peut être réalisée en alignant les vecteurs propres des données source et cible. Ce processus, bien qu'il donne des résultats acceptables, est cependant instable et n'est pas favorable à la rétro-propagation du gradient. Dans cet article, nous présentons un nouveau modèle d'adaptation de domaine pour réseau de neurones profond basé sur l'alignement des sous-espaces non négatifs dérivés des domaines source et cible. L'élimination des contraintes d'orthogonalité rend le modèle plus stable en apprentissage. Notre approche est évaluée sur un jeu de données de référence pour l'adaptation de domaine en régression. Les résultats montrent des performances compétitives par rapport aux autres modèles de pointe.

Abstract – Domain Adaptation methods seek to generalize the knowledge learned on a labeled source domain across another unlabeled target domain. Most of the deep learning methods for domain adaptation address the classification task, while regression models are still one step behind with some positive results in a shallow framework. Existing deep models for regression adaptation tasks rely on aligning the eigenvectors of both source and target data. This process, although providing satisfying results, is however unstable and not a backpropagation-friendly process. In this paper, we present a novel deep adaptation model based on aligning the non-negative sub-spaces derived from source and target domains. Removing the orthogonality constraints makes the model more stable for training. The proposed method is evaluated on a domain adaptation regression benchmark. Results show competitive performance compared to state-of-the-art models.

1 Introduction

L'apprentissage profond a connu un succès remarquable dans l'ingénierie de la connaissance, comme la classification, la régression et le clustering. Cependant, il existe encore plusieurs problèmes que les utilisateurs rencontrent lors de sa mise en œuvre. Tout d'abord, les modèles d'apprentissage profond nécessitent des jeux de données étiquetées à grande échelle, ce qui, dans les applications apprises sur des données réelles, peut être coûteux et chronophage. Un autre problème majeur, imposé par les hypothèses de la théorie de l'apprentissage, est que les données d'entraînement et de test doivent être tirées du même espace et de la même distribution. Une solution à ces deux problèmes consiste à utiliser des données étiquetées d'un domaine pertinent (appelé domaine source) pour apprendre des modèles statistiques qui fonctionnent bien dans le nouveau domaine (généralement appelé domaine cible), tout en appliquant des approches d'adaptation qui minimisent le changement de domaine et le biais entre ces deux domaines. Cette stratégie en apprentissage statistique est appelée adaptation de domaine.

Les méthodes d'adaptation de domaines peuvent être regroupées en quatre grandes catégories ([1]). La première regroupe les approches par auto-étiquetage qui sont basées sur une première estimation des labels du domaine cibles et leur ajustement pendant l'apprentissage. La deuxième se base sur les clusters qui donnent le même label aux instances appartenant aux mêmes régions denses. Dans la troisième catégorie, on trouve les approches basées sur la pondération des instances, où les instances correspondantes entre deux domaines sont repondérées. La dernière classe comprend les approches d'apprentissage de représentation qui cherchent à trouver un espace commun de représentation avec des composantes invariantes entre les deux domaines. Ces dernières années, la communauté de l'apprentissage statistique s'est concentrée sur cette dernière classe d'approches. Dans [2, 3, 4] la divergence moyenne maximale est utilisée pour entraîner un classifieur sur des données sources qui sera utilisé ultérieurement sur des données cibles. De la même manière, les méthodes adversarielles telles que [5] peuvent être utilisées pour extraire des caractéristiques cachées qui sont discriminantes pour la tâche de classification.

Le choix d'une de ces approches dépend de la tâche à accomplir, qu'il s'agisse de classification ou de régression. L'adaptation de domaine pour la classification donne de bons résultats avec presque toutes les classes d'approches, tandis que dans l'adaptation de domaine pour la régression, la plupart des méthodes sont basées sur des approches de pondération des instances ou sur l'apprentissage de représentations invariantes qui

reposent sur des données étiquetées dans le domaine cible. L'adaptation non supervisée de domaine dans un cadre de régression reste donc un problème actuel.

La différence dans le processus d'apprentissage entre les tâches de classification et de régression repose sur la fonction de perte associée à chaque tâche. En classification, la fonction de perte d'entropie croisée avec des fonctions d'activation *softmax* est souvent utilisée, avec une robustesse au changement des échelles des caractéristiques. La perte principalement associée à la régression est plus habituellement la perte L2. Toutefois, cette perte est sensible au changement d'échelle [6]. Pour compenser cela, les auteurs de [6] proposent d'utiliser la décomposition en valeurs singulières (SVD) pour aligner les vecteurs propres orthogonaux de chaque domaine. Cependant, le comportement de la SVD dans un réseau neuronal n'est pas stable car les gradients dépendent des valeurs singulières des caractéristiques où la SVD est appliquée. Des valeurs singulières répétées ou proches de zéro peuvent ainsi provoquer une explosion du gradient. Une décomposition plus favorable au mécanisme de rétropropagation serait donc préférable pour réaliser l'adaptation de domaine.

Dans cet article, nous proposons une nouvelle méthode d'adaptation de domaine par apprentissage profond dans un cadre de régression. La méthode proposée est basée sur la factorisation en matrices non négatives (NMF) de caractéristiques extraites d'un réseau de neurones profond. L'idée est d'extraire deux ensembles de matrices de base de chaque domaine via NMF, puis d'aligner ces deux bases par une étape de codage parcimonieux. Le modèle sera entraîné pour minimiser le résidu de l'étape de codage parcimonieux. Les résultats expérimentaux obtenus sur le jeu de données dSprites montrent la pertinence de la méthode proposée.

2 Méthode proposée

Dans cette section, nous présentons la méthode proposée pour l'adaptation de domaine via les sous-espaces extraits de la NMF (Section 2.1), l'alignement des décompositions dans les domaines source et cible (Section 2.2) ainsi que l'algorithme d'optimisation (Section 2.3). L'architecture du système est présentée dans la figure 1.

2.1 Génération de décompositions non-négatives

Les données de source et de cible sont désignées respectivement par $x_i^{\mathcal{S}}$ et $x_i^{\mathcal{T}}$. Soient $\{x_1^{\mathcal{S}}, x_2^{\mathcal{S}}, \cdots x_n^{\mathcal{S}}\}$ et $\{x_1^{\mathcal{T}}, x_2^{\mathcal{T}}, \cdots x_n^{\mathcal{T}}\}$ les lots (i.e., *batch*) de n données source et cible, respectivement, pour une itération donnée.

Soit un réseau de neurones ϕ_w, de paramètres w, réalisant l'apprentissage de représentation de l'espace d'entrée vers $\mathbb{R}^p$. Tout type de réseau extracteur de caractéristiques peut être utilisé dans notre méthode. Les caractéristiques extraites des données source et cible de chaque lot sont respectivement

$$\mathbf{f}_{\mathcal{S}} = [\phi_w(x_1^{\mathcal{S}}) \;\; \phi_w(x_2^{\mathcal{S}}) \cdots \phi_w(x_n^{\mathcal{S}})],$$
$$\mathbf{f}_{\mathcal{T}} = [\phi_w(x_1^{\mathcal{T}}) \;\; \phi_w(x_2^{\mathcal{T}}) \cdots \phi_w(x_n^{\mathcal{T}})].$$

Nous proposons la factorisation en matrices non négatives des deux matrices $\mathbf{f}_{\mathcal{S}}$ et $\mathbf{f}_{\mathcal{T}}$. Cette factorisation vise à décomposer chacune de ces deux matrices en deux matrices non négatives W et H, par la résolution du problème d'optimisation

$$\min_{W,H} \|\mathbf{f} - WH\|_F^2 \quad \text{sous contraintes } W, H \geq 0 \qquad (1)$$

où $\mathbf{f}$ est l'une des deux matrices, $\mathbf{f}_{\mathcal{S}}$ ou $\mathbf{f}_{\mathcal{T}}$, $W \in \mathbb{R}^{p \times k}$ peut être considéré comme contenant k vecteurs de base (i.e., dictionnaires), et la matrice $H \in \mathbb{R}^{k \times N_b}$ contient les coefficients associés à chaque vecteur de base. Une stratégie très connue pour résoudre l'équation 1 est celle des moindres carrés alternés non négatifs (ANLS). La procédure est basée sur une optimisation alternée, par une minimisation sous contrainte par rapport à une matrice tout en maintenant l'autre matrice fixe. En particulier, les règles de mise à jour multiplicatives constituent un bon compromis entre la vitesse et la facilité de mise en œuvre. Les règles de mise à jour associées à cette méthode sont :

$$H \leftarrow H \odot \frac{W^T\mathbf{f}}{W^TWH} \qquad W \leftarrow W \odot \frac{\mathbf{f}H^T}{WHH^T} \qquad (2)$$

où le produit $\odot$ et la division sont de Hadamard (i.e., terme à terme). L'approche itérative du nmf rend le processus plus pratique à mettre en œuvre dans un réseau neuronal. Les matrices W d'un lot serviront de point de départ des mises à jour du lot suivant, de cette manière la base va être entraînée sur l'ensemble des exemples.

2.2 Alignement des décompositions non-négatives

Une fois les matrices de base de chaque domaine obtenues, c'est-à-dire $W_{\mathcal{S}}$ et $W_{\mathcal{T}}$, nous cherchons à les aligner. Toutefois, comme les dictionnaires ne sont pas orthonormés, un simple calcul d'angles n'est pas suffisant. Nous proposons de considérer que les 2 dictionnaires soient alignés quand ils se redécrivent parfaitement. Ceci consiste à appliquer un codage parcimonieux joint à chacun des dictionnaires obtenus, c'est à dire l'estimation des matrices de codage $A_{\mathcal{ST}}$ et $A_{\mathcal{TS}}$ qui minimisent

$$\mathcal{L}_{Domain} = \|W_{\mathcal{S}} - A_{\mathcal{ST}}W_{\mathcal{T}}\|_F^2 + \lambda_{\mathcal{ST}} \; \|A_{\mathcal{ST}}\|_1 + \|W_{\mathcal{T}} - A_{\mathcal{TS}}W_{\mathcal{S}}\|_F^2 + \lambda_{\mathcal{TS}} \; \|A_{\mathcal{TS}}\|_1 ,$$

où $\lambda_{\mathcal{ST}}$ et $\lambda_{\mathcal{TS}}$ contrôlent le compromis entre l'erreur de reconstruction et la dispersion, mesurée par la norme L_1.

Nous cherchons alors à résoudre le problème d'optimisation

$$\min_{A_{\mathcal{ST}}, A_{\mathcal{TS}}} \mathcal{L}_{Domain}. \qquad (3)$$

Selon [7], la règle de mise à jour multiplicative suivante peut être utilisée :

$$A_{\mathcal{ST}} \leftarrow A_{\mathcal{ST}} \odot \frac{W_{\mathcal{T}}^\top W_{\mathcal{S}}}{W_{\mathcal{T}}^\top W_{\mathcal{T}} A_{\mathcal{ST}} + \lambda_{\mathcal{ST}}} \qquad (4)$$

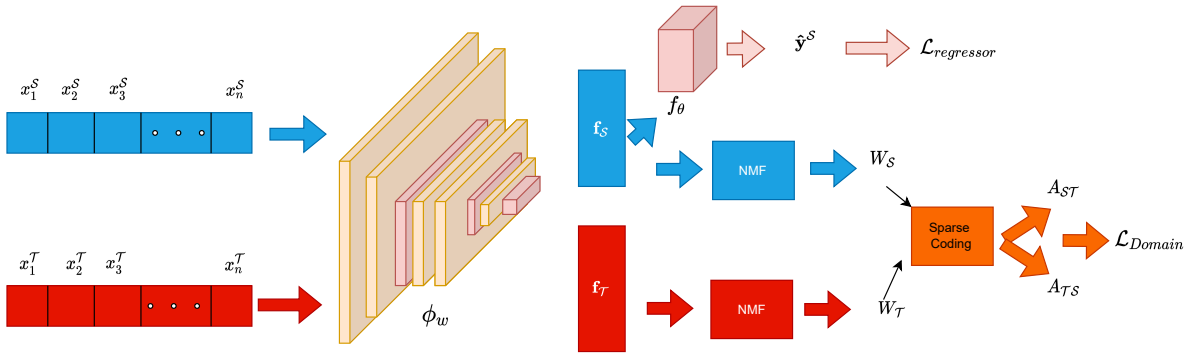


FIGURE 1 – Architecture de la méthode proposée.

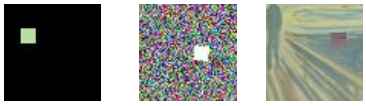


FIGURE 2 – Exemples d'images dSprites.

TABLE 1 – Paramètres du dSprites Dataset

Factor	Parameters	Task
Shape	square, ellipse, heart	recognition
Scale	$\in [0.5, 1]$	regression
Orientation	$\in [0, 2\pi]$	regression
Position X	$\in [0, 1]$	regression
Position Y	$\in [0, 1]$	regression

où la division de Hadamard est utilisée, ainsi que la multiplication $\odot$. Le même processus est effectué sur $A_{\mathcal{TS}}$. Lors de l'étape de rétropropagation du gradient stochastique, nous voulons minimiser le résidu de l'équation 3 par rapport à $\mathbf{f}_\mathcal{S}$ et $\mathbf{f}_\mathcal{T}$. Cette opération sera réalisée systématiquement par le biais du module de différenciation automatique *autograd* in *PyTorch*, qui calcule automatiquement les gradients associés à des codes et est communément utilisé lors de l'apprentissage de réseau de neurones profonds.

2.3 Régression par adaptation au domaine

L'objectif final est d'apprendre un régresseur capable de généraliser ses connaissances dans différents domaines. Soit f_θ le module de réseau de neurones régresseur de paramètre θ attaché à la sortie de l'extracteur de caractéristiques ϕ_w qui produit les valeurs de régression de prédiction, à savoir

$$\hat{\mathbf{y}}_i^\mathcal{S} = f_\theta(\phi_w(x_i^\mathcal{S}))$$

Le régresseur ainsi que l'extracteur de caractéristiques seront entraînés à l'aide de la fonction de perte d'erreur quadratique moyenne sur les données connues $\mathbf{y}$ du domaine source comme suit :

$$\mathcal{L}_{regressor} = \frac{1}{n}\sum_{i=1}^{n} \left\|\mathbf{y}_i^\mathcal{S} - \hat{\mathbf{y}}_i^\mathcal{S}\right\|^2 \quad (5)$$

Par conséquent, le processus d'optimisation complet sera conforme à l'équation suivante :

$$\mathcal{L} = \mathcal{L}_{regressor} + \lambda_{tradeoff} \times \mathcal{L}_{Domain} \quad (6)$$

où $\lambda_{tradeoff} > 0$ est un hyperparamètre de compromis.

3 Expériences

Dans cette section, nous évaluons les performances du modèle et le comparons avec d'autres modèles. Pour cela, nous utilisons un jeu de données synthétiques 2D très connu (dSprites [1]) souvent utilisé pour les tâches de régression avec adaptation de domaine.

Le jeu de données se compose de trois domaines comprenant chacun 737 280 images. Les trois domaines sont Color (**C**), Noisy (**N**) et Scream (**S**). Un exemple d'images est présenté dans la figure 2 et les propriétés associées à chaque image sont données dans le tableau 1. Pour notre expérience, nous allons essayer de prédire les trois paramètres de régression qui sont l'échelle, la position en X et en Y. Le modèle sera ensuite évalué dans le cadre des 6 possibilités d'adaptation de domaines : $\mathbf{C} \to \mathbf{N}, \mathbf{C} \to \mathbf{S}, \mathbf{N} \to \mathbf{C}, \mathbf{N} \to \mathbf{S}, \mathbf{S} \to \mathbf{C}$, et $\mathbf{S} \to \mathbf{N}$.

Le modèle a été implémenté sous PyTorch [2] et les calculs ont été effectués au moyen d'un processeur Tesla P100 de Google Colab. Pour le réseau extracteur de caractéristiques, nous utilisons le réseau pré-entraîné ResNet-18. Les données à prédire ont été normalisées à [0,1] et les images ont été redimensionnées à 224×224 pixels chacune.

La taille du lot a été fixée à 36. Le modèle a été entraîné par descente de gradient stochastique, le taux d'apprentissage étant fixé à 10 fois plus élevé pour f_θ que pour ϕ_w. Pour les hyperparamètres, un réglage manuel a été effectué et $\lambda_{\mathcal{TS}}$ et $\lambda_{\mathcal{ST}}$ ont été fixés à 1. Pour $\lambda_{tradeoff}$, nous avons choisi une règle de mise à jour adaptative basé sur [11] selon la formule

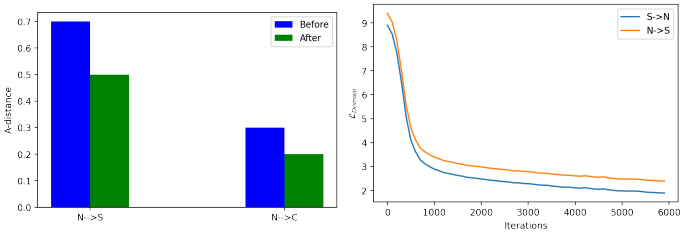
$$\lambda_{tradeoff} = \lambda_{initial} \times (1 + 0.001 \times iterations)^{-0.75}, \quad (7)$$

où $\lambda_{initial}$ a été fixé à 10^{-3} et $iterations$ correspond au nombre de lots passés. Le nombre de dictionnaires a été fixé à $k = 18$. La figure 3c montre la sensibilité du modèle par rapport à k et $\lambda_{tradeoff}$. Le modèle montre une performance robuste sur une large gamme de paramètres.

1. https ://github.com/deepmind/dsprites-dataset
2. github.com/mohamaddhaini/nmf-adaptation

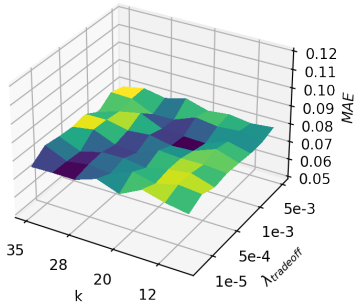
TABLE 2 – MAE des trois tâches de régression sur dSprites

Method	N → C	N → S	S → C	S → N	C → N	C → S
ResNet-18 [8]	0.16 ± 0.02	0.65 ± 0.02	0.08 ± 0.01	0.26 ± 0.03	0.94 ± 0.06	0.90 ± 0.08
TCA [2]	0.19 ± 0.02	0.66 ± 0.05	0.10 ± 0.02	0.23 ± 0.04	0.94 ± 0.03	0.87 ± 0.02
DAN [3]	0.12 ± 0.03	0.50 ± 0.05	0.06 ± 0.02	0.11 ± 0.04	0.70 ± 0.05	0.77 ± 0.09
DANN [5]	0.16 ± 0.02	0.65 ± 0.05	0.05 ± 0.00	0.10 ± 0.01	0.47 ± 0.07	0.46 ± 0.07
JDOT [9]	0.19 ± 0.02	0.64 ± 0.05	0.10 ± 0.02	0.23 ± 0.04	0.86 ± 0.03	0.79 ± 0.02
MCD [4]	0.17 ± 0.12	0.65 ± 0.03	0.07 ± 0.02	0.19 ± 0.04	0.81 ± 0.09	0.81 ± 0.12
AFN [10]	0.16 ± 0.03	0.62 ± 0.04	0.08 ± 0.01	0.32 ± 0.06	1.00 ± 0.04	0.96 ± 0.05
RSD [6]	0.16 ± 0.02	0.57 ± 0.01	0.08 ± 0.01	0.09 ± 0.02	$\mathbf{0.32 \pm 0.02}$	$\mathbf{0.35 \pm 0.02}$
Méthode proposée	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.45 \pm 0.02}$	$\mathbf{0.04 \pm 0.01}$	$\mathbf{0.08 \pm 0.02}$	0.39 ± 0.02	0.43 ± 0.02



(a) A-distance.

(b) $\mathcal{L}_{Domain}$



(c) Sensibilité aux hyperparamètres.

FIGURE 3 – Performances du modèle sur les tâches $\mathbf{N} \rightarrow \mathbf{S}$ et $\mathbf{N} \rightarrow \mathbf{C}$, avec l'évolution de $\mathcal{L}_{Domain}$.

Pour évaluer les performances du modèle proposé, la figure 3a montre la A-distance de deux tâches de transfert de l'ensemble de données dSprites avant et après l'application du modèle. Les résultats prouvent que le transfert de domaine diminue l'adaptation. La tendance à la perte dans la figure 3b montre une diminution régulière, donc une convergence stable et régulière. Le tableau 2 montre une comparaison avec différentes méthodes de pointe. Les résultats indiqués sont la moyenne et l'écart-type de 5 essais de train/test. Le modèle présente de meilleures performances dans presque toutes les tâches.

En ce qui concerne la complexité temporelle, le comportement du modèle dépend du nombre d'étapes nécessaires à la convergence du bloc NMF et du codage parcimonieux. Pour une taille de lot de 36, 100 étapes ont été suffisantes pour atteindre la convergence. Le temps d'apprentissage du ResNet-18 seul est de 0,3 seconde pour chaque itération alors qu'il est de 0,57 seconde après application du processus d'adaptation.

4 Conclusion

Dans cet article, nous avons présenté une nouvelle architecture pour les réseaux profonds en adaptation de domaine en régression. L'écart entre les domaines a été minimisé par le biais d'une mesure de l'alignement des bases non négatives extraites par factorisation en matrices non négatives. Il est prouvé que cette méthode est compatible avec la rétropropagation et peut être mise en œuvre dans des réseaux profonds de bout en bout. Les résultats ont montré de meilleures performances que les modèles existants de l'état de l'art.

Références

[1] A. Margolis, "A literature review of domain adaptation with unlabeled data," *Tec. Report*, pp. 1–42, 2011.

[2] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE transactions on neural networks*, vol. 22, no. 2, pp. 199–210, 2010.

[3] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, pp. 97–105, PMLR, 2015.

[4] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3723–3732, 2018.

[5] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[6] X. Chen, S. Wang, J. Wang, and M. Long, "Representation subspace distance for domain adaptation regression," in *International Conference on Machine Learning*, pp. 1749–1759, PMLR, 2021.

[7] P. O. Hoyer, "Non-negative sparse coding," in *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, pp. 557–565, IEEE, 2002.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[9] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[10] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable : An adaptive feature norm approach for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1426–1435, 2019.

[11] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang, "Demystifying learning rate policies for high accuracy training of deep neural networks," in *2019 IEEE International conference on big data (Big Data)*, pp. 1971–1980, IEEE, 2019.