Adversarial Weighting for Domain Adaptation in Regression

Antoine de Mathelin, Guillaume Richard, Mathilde Mougeot, Nicolas Vayatis

▶ **To cite this version:**

Antoine de Mathelin, Guillaume Richard, Mathilde Mougeot, Nicolas Vayatis. Adversarial Weighting for Domain Adaptation in Regression. 2020. hal-02867802v1

HAL Id: hal-02867802

https://hal.science/hal-02867802v1

Preprint submitted on 15 Jun 2020 (v1), last revised 15 Sep 2021 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Adversarial Weighting for Domain Adaptation in Regression

Antoine de Mathelin
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, Michelin
91190, Gif-sur-Yvette, France
antoine.demathelin@cmla.ens-cachan.fr

Guillaume Richard
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, EDF
91190, Gif-sur-Yvette, France
guillaume.richard@cmla.ens-cachan.fr

Mathilde Mougeot
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, ENSIIE
91190, Gif-sur-Yvette, France
mathilde.mougeot@ensiie.fr

Nicolas Vayatis
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli
91190, Gif-sur-Yvette, France
nicolas.vayatis@ens-paris-saclay.fr

Abstract

We present a novel instance based approach to handle regression tasks in the context of supervised domain adaptation. The approach developed in this paper relies on the assumption that the task on the target domain can be efficiently learned by adequately reweighting the source instances during training phase. We introduce a novel formulation of the optimization objective for domain adaptation which relies on a discrepancy distance characterizing the difference between domains according to a specific task and a class of hypotheses. To solve this problem, we develop an adversarial network algorithm which learns both the source weighting scheme and the task in one feed-forward gradient descent. We provide numerical evidence of the relevance of the method on public datasets for domain adaptation through reproducible experiments accessible via an online demo interface.

1 Introduction

Many applications of machine learning methods require to learn a regression task, for instance, estimation of manufactured products performance, sentiment analysis of customer reactions, forecasting of supply and demand or prediction of the time spent by a patient in a hospital. In most of these applications, groups of products or patients define several domains with different distributions. Acquiring a sufficient amount of labeled data to provide a model performing well on all of these domains is often difficult and expensive. In practical cases, only a few labeled data are available for the *target* domain of interest whereas a large amount of labeled data are available from other *source*

Preprint. Under review.

domains. One then seeks to leverage information from these source domains to learn efficiently the task on the target one through supervised training with a small sample of labeled target data.

Most of previous works on domain adaptation have focused on the unsupervised scenario where no target labels are available. Unlabeled target data are then used to correct the difference between source and target distributions by either creating a new feature space (feature-based methods [9], [25], [5]) or reweighting the training instance losses (instance-based methods [13], [24], [6]).

Available domain adaptation methods for regression are essentially instance-based methods which reweights the loss of source instances in order to minimize a distance between source and target distributions, such as the KL-divergence [24], [10], the MMD [13], [26], or the discrepancy [16], [6], [19], [1]. The latter offers the advantage, of being adapted to the underlying task and the particular class of hypotheses chosen to learn this task. An extension of the discrepancy, known as the $\mathcal{Y}$-discrepancy, introduced in [19], defined as the maximal difference between source and target risk over a set of hypotheses, presents tighter theoretical bounds of the target risk than the discrepancy [17]. However, as far as we know, previous discrepancy minimization algorithms do not estimate the $\mathcal{Y}$-discrepancy directly. As they focus on the use of unlabeled target data, they choose instead to consider unsupervised approximation of this distance as the discrepancy [19] or the *generalized discrepancy* [7].

Most instance-based methods rely on the use of functions induced by positive semi-definite kernels and their weighting strategy consists in general in solving a quadratic problem [6], [7], [13], [26]. Thus, these methods present a computational burden when the number of data is important. The work of [20] presents a boosting method for domain adaptation with regression tasks which scales better with large datasets by using neural networks or decision trees as base learner.

In this paper, we present a novel instance-based method for supervised domain adaptation for regression tasks using a few labeled target data. We propose the Weighting Adversarial Neural Network (WANN) algorithm to learn the optimal weights to correct the difference between source and target distributions. WANN proceeds to the minimization, in one feed-forward gradient descent, of an original objective function composed of the empirical $\mathcal{Y}$-discrepancy between source and target domains and the task risk on these same domains (section 2). Compared to other discrepancy minimization algorithms, we use adversarial neural networks to estimate at each gradient step the importance weights of source instances and the empirical $\mathcal{Y}$-discrepancy. We thus propose an efficient way to extend adversarial domain adaptation to regression tasks. After presenting related work in section 3, we show on several experiments that the novel weighting strategy of WANN leads to results which outperform state of the art methods for domain adaptation in regression and provides a method which scales better with large datasets. All the code for the experiments presented in this paper is available on GitHub. We also implement an online demo of our algorithm which can be found via the links provided in section 4.

2 Weighting Adversarial Neural Network

2.1 Notations

Given $X \subset \mathbb{R}^p$ and $Y \subset \mathbb{R}$, we consider the supervised domain adaptation regression setting where $\mathcal{S} = \{(x_1, y_1), ..., (x_m, y_m)\} \in (X \times Y)^m$ and $\mathcal{T} = \{(x_1, y_1), ..., (x_n, y_n)\} \in (X \times Y)^n$ are respectively the labeled source and target datasets. In this setting, the sample size n of $\mathcal{T}$ is typically much smaller than m, the one of $\mathcal{S}$ ($n << m$).

We consider two classes of neural networks H, H' of a given architecture and activation functions. We introduce three networks $h_t, h_d \in H$ and $W \in H'$ called respectively the *task*, the *discrepancy* and the *weighting* network.

2.2 Objective function

Our approach is based on the instance-based assumption that an optimal *task* hypothesis $h_t \in H$ for the regression task on the target domain can be computed by optimally weighting the loss of the source instances during the training phase. We suppose in addition that the optimal source weights can be computed by a *weighting network* $W \in H'$ such that $W : X \to \mathbb{R}_+$ learns the relationship between the input space and the source weights. A reweighting of the source instances is considered

to be optimal if it minimizes the empirical $\mathcal{Y}$-discrepancy between the reweighted source and the target distributions. However, computing the empirical $\mathcal{Y}$-discrepancy requires finding a maximum over the set of hypotheses H which is difficult in general. We then introduce a *discrepancy* hypothesis $h_d \in H$ to approximate the empirical $\mathcal{Y}$-discrepancy using adversarial techniques introduced by [9].

This leads to the objective function G provided below (equation 1). The function G can be understood as a regularization of the target risk (second term of G) by the weighted source risk (first term of G). Since the sample size of $\mathcal{T}$ is small, adding a selection of labeled source data will prevent from overfitting. The weighting network W is trained to "select" the most informative source instances which are "close" to the target instances in term of the estimated $\mathcal{Y}$-discrepancy (third term of G).

$$\begin{aligned} G(W, h_t, h_d) =& \frac{1}{m} \sum_{(x_i, y_i) \in \mathcal{S}} W(x_i)(h_t(x_i) - y_i)^2 + \frac{1}{n} \sum_{(x_j, y_j) \in \mathcal{T}} (h_t(x_j) - y_j)^2 \\ &+ \left| \frac{1}{m} \sum_{(x_i, y_i) \in \mathcal{S}} W(x_i)(h_d(x_i) - y_i)^2 - \frac{1}{n} \sum_{(x_j, y_j) \in \mathcal{T}} (h_d(x_j) - y_j)^2 \right| . \end{aligned} \tag{1}$$

In order to approximate the $\mathcal{Y}$-discrepancy which consists of the maximal difference between source and target risks, the network h_d is trained to maximize the third term of G, i.e h_d seeks to provide antagonist performances on the two distributions. Thus, by looking for the source weights minimizing the $\mathcal{Y}$-discrepancy, the network W learns a new source distribution on which any hypothesis h_d in H will perform as well as on the target distribution.

The purpose of using a weighting neural network W is to capture the underlying dependence between source instances. Indeed, the weights of dependent source instances should increase or decrease altogether as these instances will be similarly related to the target data. Using a neural network for this purpose provides a way to preserve the spatial structure in the source weighting scheme such that the weights of source instances close to each other in the input space will be similar (section 4.1).

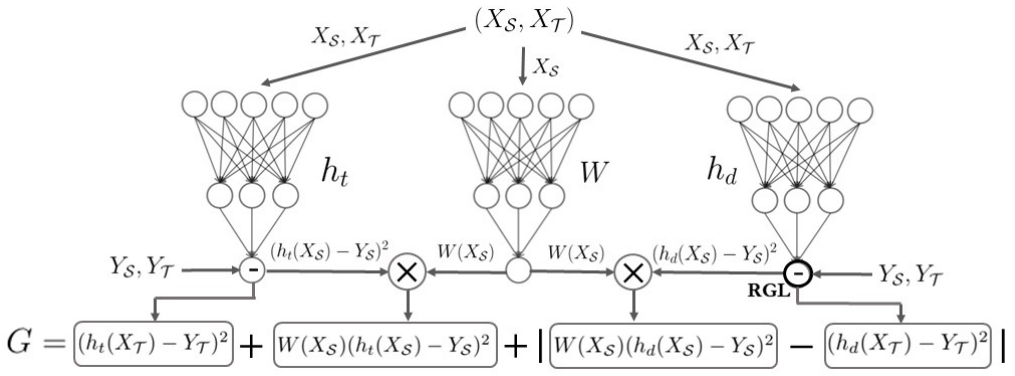

Figure 1: The WANN algorithm trains three networks in parallel in the same gradient descent. W learns the source instance weights which are multiplied to the source losses of networks h_t and h_d. The last network, which estimates the $\mathcal{Y}$-discrepancy between the reweighted source and target instances, is trained with an opposite objective function ($-G$). This is done by using a Reversal Gradient Layer (RGL) in bold on the Figure. Source and target data are denoted respectively $(X_\mathcal{S}, Y_\mathcal{S})$ and $(X_\mathcal{T}, Y_\mathcal{T})$.

2.3 Weighting Adversarial Neural Network Algorithm

We define here $\beta_t, \beta_d, \beta_w$ the parameters of the respective networks h_t, h_d and W which will be denoted in this section $h_t^{\beta_t}, h_d^{\beta_d}$ and W^{β_w} for the sake of clarity. In the same way, the objective function G from equation 1 will be denoted as a function of the parameters.

The details of WANN gradient descent are presented in Algorithm 1. The goal of this algorithm is to approximate the saddle point $(\beta_w^*, \beta_t^*, \beta_d^*)$ verifying:

$$\begin{aligned}(\beta_w^*, \beta_t^*) &= \underset{\beta_w, \beta_t}{\operatorname{argmin}} G(\beta_w, \beta_t, \beta_d^*) \\ \beta_d^* &= \underset{\beta_d}{\operatorname{argmax}} G(\beta_w^*, \beta_t^*, \beta_d)\,.\end{aligned} \tag{2}$$

A *reversal gradient* layer, as defined in [9], is used to change the sign of the gradient in back-propagation. Thus, in the same gradient step, the current objective function $G(\beta_w, \beta_t, \beta_d)$ is back-propagated through W^{β_w} and $h_t^{\beta_t}$ whereas its opposite value is returned to $h_d^{\beta_d}$ (Figure 1).

There are no theoretical guarantees that G admits this saddle point. However, in practice, we observe numerical convergence of WANN and improved performance on the target task compared to other methods. (section 4)

A constraint is applied on each network by projecting the weights of their layers at each gradient step on the Euclidean ball of radius C [23]. This constraint is used in several adversarial algorithms such as WGAN [2] and DANN [9]. Furthermore, mini-batch gradient descent is also used. Notice finally that we consider, in the algorithm, the squared $\mathcal{Y}$-discrepancy in order to make G differentiable.

Algorithm 1 WANN algorithm

Input:
- $\mathcal{S} = \{(x_i, y_i)\}_{i<m}$, $\mathcal{T} = \{(x_j, y_j)\}_{j<n}$
- D, D' dimensions of neural networks
- E number of epochs, B batch size, μ learning rate, C projecting constant

Output: neural networks W^{β_w}, $h_t^{\beta_t}$ and $h_d^{\beta_d}$

Initialization
$\beta_t \leftarrow$ *Glorot uniform*$^* \in \mathbb{R}^D$
$\beta_w \leftarrow$ *Glorot uniform* $\in \mathbb{R}^{D'}$
$\beta_d \leftarrow$ *Glorot uniform* $\in \mathbb{R}^D$
Reshape: $\mathcal{T} \leftarrow \mathcal{T} \times \lfloor m/n \rfloor$
for e from 1 to E **do**
  Split $\mathcal{S}, \mathcal{T}$ in $\{\mathcal{S}_k, \mathcal{T}_k\}_{k<\lfloor m/B \rfloor}$ batches
  for k from 1 to $\lfloor m/B \rfloor$ **do**
    Forward propagation
    $\Delta S_t \leftarrow \sum_{\mathcal{S}_k} W^{\beta_w}(x_i)(h_t^{\beta_t}(x_i) - y_i)^2$
    $\Delta T_t \leftarrow \sum_{\mathcal{T}_k} (h_t^{\beta_t}(x_j) - y_j)^2$
    $\Delta S_d \leftarrow \sum_{\mathcal{S}_k} W^{\beta_w}(x_i)(h_d^{\beta_d}(x_i) - y_i)^2$
    $\Delta T_d \leftarrow \sum_{\mathcal{T}_k} (h_d^{\beta_d}(x_j) - y_j)^2$
    Back propagation
    $\beta_t \leftarrow P_C\left(\beta_t - \mu\left(\frac{\partial \Delta S_t}{\partial \beta_t} + \frac{\partial \Delta T_t}{\partial \beta_t}\right)\right)$
    $\beta_w \leftarrow P_C\left(\beta_w - \mu\left(\frac{\partial \Delta S_t}{\partial \beta_w} + \frac{\partial |\Delta S_d - \Delta T_d|^2}{\partial \beta_w}\right)\right)$
    $\beta_d \leftarrow P_C\left(\beta_d + \mu\left(\frac{\partial |\Delta S_d - \Delta T_d|^2}{\partial \beta_d}\right)\right)$
  end for
end for

Note: In this pseudo-code, P_C refers to the projection of the weights of each layer of a neural network on the ball of radius C. * *Glorot uniform* is the initialization procedure proposed in [11].

3 Related work

3.1 Discrepancy Minimization

The present work is in line with discrepancy minimization methods, which were first introduced in [16] and further developed in [6], [19], [15], [28] and [7]. More specifically, the WANN algorithm aims at minimizing the empirical $\mathcal{Y}$-discrepancy introduced in [19].

Definition and theoretical results for the $\mathcal{Y}$-discrepancy are provided in [17] (Definition 5, Proposition 1) where theoretical bounds of the average loss over the target domain can be found. Considering the empirical source and target distributions $\widehat{Q}$ and $\widehat{P}$ and labeling functions f_Q, f_P, it is showed that the task risk on the target distribution can be upper bounded by the empirical risk on any reweighted source distribution plus the empirical $\mathcal{Y}$-discrepancy. Besides, the target risk is also upper bounded by its empirical estimation plus a *Rademacher* complexity term.

Following these considerations, it appears that to minimize the target risk, one should minimize the $\mathcal{Y}$-discrepancy and the task risk on the empirical distributions $\widehat{Q}$ and $\widehat{P}$. This is the purpose of WANN

algorithm which aims at solving the following optimization formulation:

$$\min_{h_t \in H, W \in H'} \left(\mathcal{L}_{W(\widehat{Q})}(h_t, f_Q) + \mathcal{L}_{\widehat{P}}(h_t, f_P) + \max_{h_d \in H} |\mathcal{L}_{W(\widehat{Q})}(h_d, f_Q) - \mathcal{L}_{\widehat{P}}(h_d, f_P)| \right), \quad (3)$$

where $\mathcal{L}_{W(\widehat{Q})}(h, f_Q) = \mathrm{E}_{x \sim \widehat{Q}}[W(x)L(h(x), f_Q)]$, and $\mathcal{L}_{\widehat{P}}(h, f_P) = \mathrm{E}_{x \sim \widehat{P}}[L(h(x), f_P)]$ are respectively the reweighted source risk and the target risk with L a loss function over pairs of labels.

The optimization formulation (3) is a min-max optimization problem. The algorithms in [6] and [7] solve a related problem for respectively the discrepancy and the *generalized discrepancy* on the class of functions induced by PSD kernels using quadratic programming. In this paper, we choose instead to estimate the "max" part of the equation with a discrepancy network h_d trained with adversarial techniques.

3.2 Adversarial neural networks

As far as we know, our WANN algorithm is the first application of adversarial techniques to domain adaptation for supervised regression tasks. Indeed, adversarial techniques, originally introduced for domain adaptation in [9], are essentially used in unsupervised feature-based methods for classification tasks. DANN [9] and ADDA [27] algorithms, focus on finding a new representation of the input features where source and target instances cannot be distinguished by any discriminative hypothesis. This process aims at minimizing the $\mathcal{H}$-divergence introduced by [3]. Considering other distances, the adversarial methods MCD [22] and MDD [28] learn a new features representation by minimizing respectively the absolute difference between the predictions of two classifiers and the *disparity discrepancy* between source and target domains. Similarly, in [1], the discrepancy distance is considered for the training of GANs.

4 Experiments

In this section, we report the results of WANN algorithm compared to other domain adaptation methods for regression. The experiments are conducted on one synthetic and three public datasets: *Superconductivity* [12], *Kin-familly* [21] and *Amazon review* [4]. Following the standards of reproducible experiments, the source code of the used methods and all the scripts to obtain the presented results are available on GitHub [1] with an online demo. GDM code used is the one provided by the authors of [7] [2]. All results presented in this section have been computed on a (1.8 GHz, 8 G RAM) computer. The following competitors are selected to compare the performance of the WANN algorithm:

- **TrAdaBoostR2** [20] is based on a reverse-boosting principle where the weight of source instances poorly predicted are decreased at each boosting iteration. We choose the two-stage version of TrAdaBoostR2 with 10 first stage and 5 second stage iterations. A 5 fold cross-validation is performed at each first stage and the best hypothesis is returned.
- **Generalized Discrepancy Minimization (GDM)** [7] is an adaptation of DM algorithm [6] to the supervised scenario. The GDM hyper-parameter λ is selected from the set $\{2^i, -10 \leq i \leq 10\}$ and the r and σ hyper-parameters from the set: , $\{2^i, -5 \leq i \leq 5\}$, the selection is made with cross-validation on the few available labeled target data.
- **Kullback-Leibler Importance Estimation Procedure (KLIEP)** [24] is a sample bias correction method minimizing the KL-divergence between a reweighted source and target distributions. We choose the KLIEP likelihood cross validation (LCV) version with selection of Gaussian kernel bandwidth in the set $\sigma = \{2^i, -5 \leq i \leq 5\}$.
- **Kernel Mean Matching (KMM)** [13] reweights source instances in order to minimize the MMD between domains. A Gaussian kernel is used with $\sigma \in \{2^i, -5 \leq i \leq 5\}$ selected with cross-validation on labeled target data. Parameters B and ϵ are set to 1000 and $\frac{\sqrt{m}}{\sqrt{m}-1}$.
- **Discriminative Adversarial Neural Network (DANN)** [9] is used here for regression tasks by considering the mean squared error as task loss instead of the binary cross-entropy

[1] `https://github.com/AnonymousAccount0/WANN`

[2] `https://cims.nyu.edu/~munoz/`

proposed in the original algorithm. In the following DANN uses all labeled data to learn the task and all available training target data (including unlabeled ones) to find a common feature space. The trade-off parameter λ is selected on 7 values between 0.01 and 1 with cross-validation on the training labeled target data.

To compare only the adaptation effect of each method, we use, for all of them, the same class of functions H to learn the task which is the class of fully-connected neural networks with *ReLU* activation functions and a static architecture. All networks implement a projecting regularization of parameter C. Adam optimizer [14] is used in all experiments for the training of neural networks. For WANN algorithm, the two networks h_t, h_d are chosen in the specified class H. For DANN algorithm, a linear discriminative network is placed at the last hidden layer of the task network, thus DANN uses the same hypothesis as other compared methods to learn the task.

4.1 Synthetic Experiment

We first propose to give an intuitive understanding of WANN behavior through a one-dimensional dataset. For this purpose, we consider the synthetic experiment where source and target input instances are drawn uniformly on $[0, 1]$. Source instances follow (with equal probability) one of these five labeling functions: $f^k(x) = \sin(20x) + kx^3 + \epsilon$ for $k \in [-2, -1, 0, 1, 2]$, with $\epsilon = \mathcal{N}(0, 0.1)$. Target instances follow the labeling function $f(x) = \sin(20x) - 0.75x^3 + \epsilon$. As presented in Figure 2.A, we thus model a domain adaptation scenario where target and source data have fairly the same behaviour on the first half of the distribution but differ on the second. We consider 10 labeled target data equally separated along the domain with additional noise (black squares).

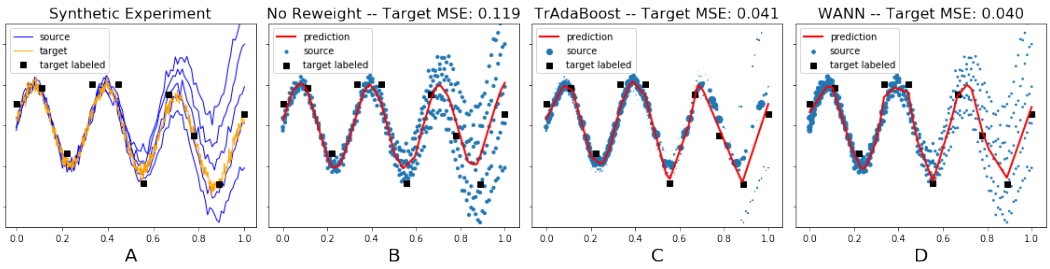

Figure 2: Visualization of synthetic experiment results.

Figure 2.B displays the predictions computed with the "No reweight" method which attributes uniform weights to all training instances. It appears that the "No reweight" strategy fails to provide a suitable hypothesis for the target task by following the mean of source tasks on the second half of the domain. In the contrary, the two domain adaptation methods TrAdaBoostR2 (Figure 2.C) and WANN (Figure 2.D) are able to "select", with an appropriate reweighting, the source instances which present similar behaviour than the target data and to discard the others. WANN, however, presents a more continuous reweighting than TrAdaBoostR2 due to the use of a weighting network which conserves some spatial structure. In this case, a slight benefit is observed for WANN in terms of target risk. We notice however for some cases in this synthetic setup (in particular for high learning rate), suboptimal convergence of WANN which may be due to the instability of adversarial training [18].

4.2 Experiments on a large dataset

A major advantage of WANN algorithm over previous instance-based methods for domain adaptation in regression is to propose a weighting strategy suited for neural networks. Thus our method scales better to large datasets than other methods involving kernels and quadratic programming. We propose here to demonstrate the efficiency of WANN on the UCI dataset *Superconductivity* [12], [8], against "No Reweight", TrAdaBoostR2 and DANN which can also handle large datasets.

The goal is to predict the critical temperature of superconductors based on features extracted from their chemical formula. This is a common regression problem in industry, as industrials are particularly interested to model the relationship between a material and its properties.

Table 1: Superconductivity experiments MSE ($\times$ 1000)

Expe.	l $\rightarrow$ ml	l $\rightarrow$ mh	l $\rightarrow$ h	ml $\rightarrow$ l	ml $\rightarrow$ mh	ml $\rightarrow$ h	mh $\rightarrow$ l
Tgt Only	432 (54)	412 (68)	385 (63)	726 (46)	488 (60)	527 (76)	875 (82)
Src Only	563 (35)	512 (50)	883 (176)	525 (34)	316 (23)	513 (55)	1996 (85)
No Re.	444 (28)	436 (35)	498 (56)	499 (12)	317 (19)	477 (63)	1167 (105)
DANN	597 (39)	589 (66)	755 (90)	484 (21)	343 (20)	465 (62)	1526 (254)
TrAdaB.	313 (17)	340 (39)	**383 (62)**	483 (44)	281 (6)	383 (35)	659 (22)
WANN	**235 (10)**	**324 (18)**	392 (40)	**430 (9)**	**261 (13)**	**352 (26)**	**626 (25)**
Expe.	mh $\rightarrow$ ml	mh $\rightarrow$ h	h $\rightarrow$ l	h $\rightarrow$ ml	h $\rightarrow$ mh	Avg MSE	Avg rank
Tgt Only	669 (77)	599 (59)	982 (106)	725 (81)	602 (80)	618 (71)	4.67
Src Only	629 (29)	353 (14)	3092 (531)	1101 (209)	372 (16)	904 (105)	5.00
No Re.	566 (34)	344 (13)	740 (41)	**493 (40)**	345 (17)	527 (39)	3.42
DANN	503 (29)	387 (39)	1808 (553)	527 (65)	355 (23)	695 (105)	4.33
TrAdaB.	458 (36)	**338 (9)**	656 (23)	555 (61)	404 (23)	438 (31)	2.25
WANN	**392 (12)**	339 (10)	**625 (30)**	503 (28)	**331 (23)**	**401 (20)**	**1.33**

We divide this dataset in separate domains following the setup of [20]. We select an input feature with a moderate correlation factor with the output (~ 0.3). We then sort the set according to this feature and split it in four parts: low (l), midle-low (ml), midle-high (mh), high (h). Each part defining a domain with around 5000 instances. The considered feature is then withdrawn from the dataset. We conduct an experiment for each pair of domains which leads to 12 experiments. All source and 10 target labeled instances are used in the training phase, the other target data are used to compute the results reported in Table 1. We also report the average MSE as well as the average rank over the 12 experiments. Notice that each experiment is repeated 10 times to obtain the standard deviation in brackets. Here, the networks from H used by all methods to learn the task is composed of a layer of 100 neurons, with a projecting parameter $C = 1$. WANN weighting network W is taken in H with a projecting constant equal to 0.1. The learning rate is set to 0.001, the number of epochs to 200 and the batch size to 1000. A standard scaling preprocessing is performed with the training data on both input and output features. We also consider the two basic methods: "Src Only", trained on source data only and "Tgt Only", trained on the few labeled target data only.

The results of Table 1 underlined the ability of WANN to efficiently adapt between domains. In particular, we observe significant gains against DANN, "No Reweight", "Src Only" and "Tgt Only", when the source and target domains are less related, for instance when adapting from "midle-high" to "low" (mh $\rightarrow$ l). TrAdaboostR2 shows competitive results to WANN in some experiments. It should be mentioned however, that this method requires to train 100 networks where the others only have to train one. Besides, as boosting iterations need to be executed successively, the training of these networks cannot be parallelized. The fact that WANN algorithm is based on the minimization of a theoretically well founded objective function may explain its better performances over TrAdaBoostR2.

4.3 Experiments on small datasets

In order to compare our method against KMM, KLIEP and GDM, we consider several experiments on smaller datasets extracted from respectively *Kin-8xy* [21] and *Amazon review* [4]. We choose the same experimental setups than GDM in [7] for the choice of training and testing data. Notice that the training set in all experiments is composed of the source and a few labeled target instances as well as unlabeled target instances. However, WANN and TrAdaBoostR2 do not use the unlabeled ones. It should also be underlined that KMM and KLIEP are two stage methods which first reweight the training instances and then learn the task hypothesis. Gaussian kernels are used only in the first stage. To learn the task, the same class of hypotheses H is used for all methods. Exception is made for GDM algorithm which is a one stage algorithm implemented for hypotheses induced by PSD kernels.

The first experiments are conducted on *Kin-8xy* [21] which is a family of datasets synthetically generated from a realistic simulation of the forward kinematics of an 8 link all-revolute robot arm. The task consists in predicting the distance of the end-effector from a target. The task for each dataset has a specific degree of noise (moderate "m" or high "h") and linearity (fairly-linear "f", non-linear "n"). We conduct one experiment on each of the 12 pairs of domains defined by these 4 datasets. We

Table 2: Sentiment analysis experiments summary (MSE $\times 1000$).

	WANN	TrAdaBoost	DANN	No Rew.	KLIEP	KMM	GDM
1^{st} rank	**8**	2	0	0	0	1	1
Avg MSE	**970 (24)**	1002 (33)	1127 (36)	1017 (19)	1019 (18)	1020 (18)	1039 (2)
Avg rank	**1.42**	2.92	6.83	3.92	4.08	4.17	4.67

pick 200 source, 200 target unlabeled and 10 target labeled instances. 400 other target instances are used to compute the MSE scores reported in Figure 3.

We conduct the next experiments on the cross-domain sentiment analysis dataset of *Amazon review* [4] where reviews from four domains: dvd, kitchen, electronics and books are rated between 1 and 5. The task consists in predicting the rating given one review. For pre-processing, we select the top 1000 uni-grams and bi-grams. Here, 700 labeled source and unlabeled target data are given as well as 50 labeled target data. The results are computed on 1000 target data.

For both datasets, the networks of H used to learn the task are composed of 2 layers of respectively 100 and 10 neurons, parameter C is set to 1. Dropouts are added at the end of each layer with the respective rates $(0.5, 0.2)$. A learning rate of 0.001, 300 epochs and a batch size of 32 are used in the optimization for the experiments on *Kin-8xy*. For the ones on *Amazon review* the number of epochs is set to 200 and the batch size to 64. All experiments are run 10 times to compute standard deviations.

The choice of WANN hyper-parameters lies in the choice of W network architecture. In the experiments, we arbitrarily choose the same architecture as task networks from H. For each dataset, we choose the same C_w projecting parameter of W in all experiments, the choice of C_w is done using cross-validation on one of the 12 experiments using the few training labeled target data. The constant selected here is 1 for *Kin-8xy* and 0.2 for *Amazon review*. We try other architectures and choices of parameter C_w for W and also notice leading results for WANN algorithm.

Figure 3.A presents the results of kin experiments. WANN provides the best MSE in a majority of experiments, in particular when labeling functions differ between source and target domains. Again, our algorithm presents better performance than other methods on the sentiment analysis experiments (Table 2). As the only difference between WANN and the other domain adaptation methods (at the exception of GDM and DANN) is the weighting strategy, these results underline the efficiency of using a neural network to learn the source instances weights. Notice that our method and TradaboostR2 do not take advantage of unlabeled target data, however the two methods present the best score in almost all experiments. These considerations highlight the difficulty to make consistent unsupervised adaptation on a regression task. This fact can also be observed on Figure 3.B presenting the impact of the number of labeled target data on the target risk. We observe that significant decreases of MSE are due to the presence of labeled target data more than to the method used, in particular when labeling functions differ between domains (*fm* $\rightarrow$ *nh* or *nh* $\rightarrow$ *fm*). In these cases, it appears that it is better to use a "No reweight" strategy with a few labeled target data than to use an unsupervised algorithm. However, we observe that the "No Reweight" strategy needs between 30 to 100 labeled target data to obtain the same level of MSE obtained with WANN algorithm using only 10 of them.

5 Conclusion

In this work, we present a novel instance based approach for regression tasks in the context of supervised domain adaptation. We show that the weights accorded to source instance losses during the training phase can be optimally adjusted with a neural network in order to learn efficiently the target task. We propose the WANN algorithm which minimizes with adversarial techniques an original objective function involving the $\mathcal{Y}$-discrepancy. WANN algorithm provides, on various experiments, results which outperform baselines for regression domain adaptation and proposes a weighting strategy able to handle large datasets. We show that using a weighting network for instance-based domain adaptation provides an efficient way to conserve spatial structure in the weighting scheme. Our work also reveals the importance of labeled target data to obtain performing models in the context of domain adaptation with regression tasks.

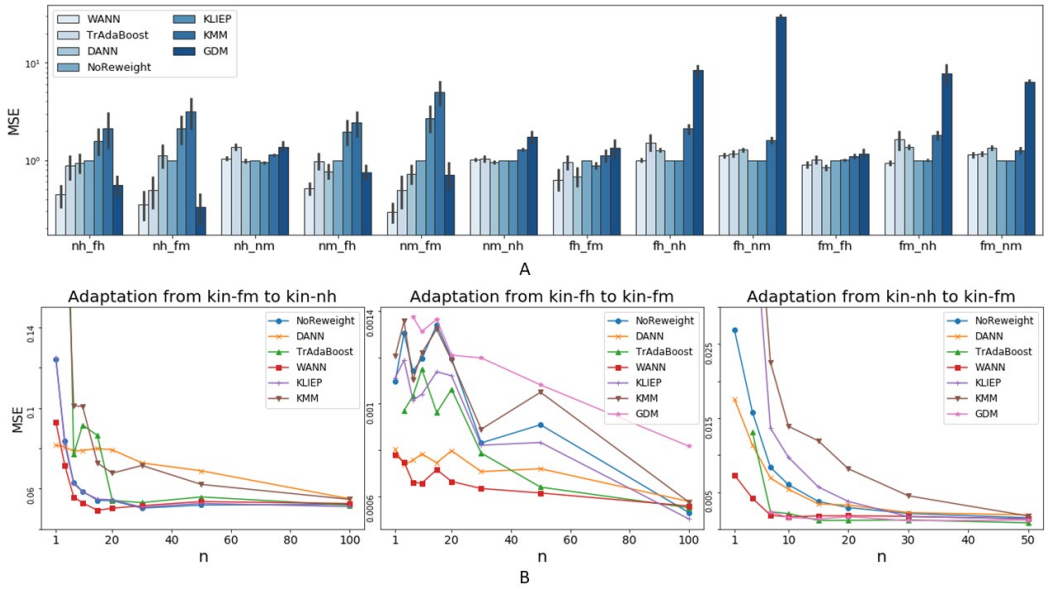

Figure 3: Kin experiments results: (A) presents the comparison of MSE for each experiment normalized in the way that "No reweight" score is always 1. (B) presents the evolution of MSE depending on the number of labeled target instances in the training set for three experiments.

References

[1] Ben Adlam, Corinna Cortes, Mehryar Mohri, and Ningshan Zhang. Learning gans and ensembles using discrepancy. In *Advances in Neural Information Processing Systems*, pages 5788–5799, 2019.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[3] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 137–144. MIT Press, 2007.

[4] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. https://www.cs.jhu.edu/~mdredze/datasets/sentiment/index2.html. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447, 2007.

[5] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, page 1627–1634, Madison, WI, USA, 2012. Omnipress.

[6] Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519, 2014.

[7] Corinna Cortes, Mehryar Mohri, and Andrés Muñoz Medina. Adaptation based on generalized discrepancy. *J. Mach. Learn. Res.*, 20(1):1–30, January 2019.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2017.

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January 2016.

[10] Jochen Garcke and Thomas Vanck. Importance weighted inductive transfer learning for regression. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 466–481, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[12] Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. https://archive.ics.uci.edu/ml/datasets/superconductivty+data#. *Computational Materials Science*, 154:346–354, 2018.

[13] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex J. Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 601–608. MIT Press, 2007.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[15] Seiichi Kuroki, Nontawat Charoenphakdee, Han Bao, Junya Honda, Issei Sato, and Masashi Sugiyama. Unsupervised domain adaptation based on source-guided discrepancy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4122–4129, 07 2019.

[16] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[17] Andrés Munoz Medina. *Learning Theory and Algorithms for Auctioning and Adaptation Problems*. PhD thesis, PhD thesis, New York University, 2015.

[18] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3481–3490, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[19] Mehryar Mohri and Andres Muñoz Medina. New analysis and algorithm for learning with drifting distributions. In Nader H. Bshouty, Gilles Stoltz, Nicolas Vayatis, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 124–138, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[20] David Pardoe and Peter Stone. Boosting for regression transfer. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, June 2010.

[21] Carl Edward Rasmussen, Radford M Neal, Geoffrey Hinton, Drew van Camp, Michael Revow Zoubin Ghahramani, Rafal Kustra, and Rob Tibshirani. The delve project. http://www.cs.toronto.edu/~delve/data/datasets.html, 1996.

[22] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[23] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[24] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1433–1440, Red Hook, NY, USA, 2007. Curran Associates Inc.

[25] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[26] Qian Sun, Rita Chattopadhyay, Sethuraman Panchanathan, and Jieping Ye. A two-stage weighting framework for multi-source domain adaptation. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 505–513. Curran Associates, Inc., 2011.

[27] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[28] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7404–7413, Long Beach, California, USA, 09–15 Jun 2019. PMLR.