Domain-Adversarial Neural Networks

Hana Ajakan[1] HANA.AJAKAN.1@ULAVAL.CA
Pascal Germain[1] PASCAL.GERMAIN@IFT.ULAVAL.CA
Hugo Larochelle[2] HUGO.LAROCHELLE@USHERBROOKE.CA
François Laviolette[1] FRANCOIS.LAVIOLETTE@IFT.ULAVAL.CA
Mario Marchand[1] MARIO.MARCHAND@IFT.ULAVAL.CA

[1] Département d'informatique et de génie logiciel, Université Laval, Québec, Canada
[2] Département d'informatique, Université de Sherbrooke, Québec, Canada
All authors contributed equally to this work.

Abstract

We introduce a new representation learning algorithm suited to the context of domain adaptation, in which data at training and test time come from similar but different distributions. Our algorithm is directly inspired by theory on domain adaptation suggesting that, for effective domain transfer to be achieved, predictions must be made based on a data representation that cannot discriminate between the training (source) and test (target) domains. We propose a training objective that implements this idea in the context of a neural network, whose hidden layer is trained to be predictive of the classification task, but uninformative as to the domain of the input. Our experiments on a sentiment analysis classification benchmark, where the target domain data available at training time is unlabeled, show that our neural network for domain adaption algorithm has better performance than either a standard neural network or an SVM, even if trained on input features extracted with the state-of-the-art marginalized stacked denoising autoencoders of Chen et al. (2012).

1. Introduction

The cost of generating labeled data for a new learning task is often an obstacle for applying machine learning methods. There is thus great incentive to develop ways of exploiting data from one problem that generalizes to another. Domain adaptation focuses on the situation where we have data generated from two different, but somehow similar, distributions. One example is in the context of sentiment analysis in written reviews, where we might want to distinguish between the positive from the negative ones. While we might have labeled data for reviews of one type of products (*e.g.*, movies), we might want to be able to generalize to reviews of other products (*e.g.*, books). Domain adaptation tries to achieve such a transfer by exploiting an extra set of unlabeled training data for the new problem to which we wish to generalize (*e.g.*, unlabeled reviews of books).

One of the main approach to achieve such a transfer is to learn a classifier *and* a representation which will favor the transfer. A large body of work exists on training both a classifier and a representation that are linear (Bruzzone & Marconcini, 2010; Germain et al., 2013; Cortes & Mohri, 2014). However, recent research has shown that non-linear neural networks can also be successful (Glorot et al., 2011). Specifically, a variant of the denoising autoencoder (Vincent et al., 2008), known as marginalized stacked denoising autoencoders (mSDA) (Chen et al., 2012), has demonstrated state-of-the-art performance on this problem. By learning a representation which is robust to input corruption noise, they have been able to learn a representation which is also more stable across changes of domain and can thus allow cross-domain transfer.

In this paper, we propose to control the stability of representation between domains explicitly into a neural network learning algorithm. This approach is motivated by theory on domain adaptation (Ben-David et al., 2006; 2010) that suggests that a good representation for cross-domain transfer is one for which an algorithm cannot learn to identify the domain of origin of the input observation. We show that this principle can be implemented into a neural network learning objective that includes a term where the network's hidden layer is working adversarially towards output connections predicting domain membership. The neural network is then simply trained by gradient descent on this objective. The success of this domain-adversarial neural network (DANN) is confirmed by extensive experiments on both toy and real world datasets. In particular, we show that DANN achieves better performances than a regular neu-

ral network and a SVM on a sentiment analysis classification benchmark. Moreover, we show that DANN can reach state-of-the-art performance by taking as input the representation learned by mSDA, confirming that minimizing domain discriminability explicitly improves over on only relying on a representation which is robust to noise.

2. Domain Adaptation

We consider binary classification tasks where $\mathcal{X} \subseteq \mathbb{R}^n$ is the input space and $\mathcal{Y} = \{0, 1\}$ is the label set. Moreover, we have two different distributions over $\mathcal{X} \times \mathcal{Y}$, called the *source domain* $\mathcal{D}_\text{S}$ and the *target domain* $\mathcal{D}_\text{T}$. A *domain adaptation* learning algorithm is then provided with a *labeled source sample* S drawn *i.i.d.* from $\mathcal{D}_\text{S}$, and an *unlabeled target sample* T drawn *i.i.d.* from $\mathcal{D}_\text{T}^x$, where $\mathcal{D}_\text{T}^x$ is the marginal distribution of $\mathcal{D}_\text{T}$ over $\mathcal{X}$.

independent and identically distributed

$$S = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^m \sim (\mathcal{D}_\text{S})^m\,; \quad T = \{\mathbf{x}_i^t\}_{i=1}^{m'} \sim (\mathcal{D}_\text{T}^x)^{m'}.$$

The goal of the learning algorithm is to build a classifier $\eta : \mathcal{X} \to \mathcal{Y}$ with a low *target risk*

$$R_{\mathcal{D}_\text{T}}(\eta) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x}^t, y^t) \sim \mathcal{D}_\text{T}} \left(\eta(\mathbf{x}^t) \neq y^t\right),$$

while having no information about the labels of $\mathcal{D}_\text{T}$.

2.1. Domain Divergence

To tackle the challenging domain adaptation task, many approaches bound the target error by the sum of the source error and a notion of distance between the source and the target distributions. These methods are intuitively justified by a simple assumption: the source risk is expected to be a good indicator of the target risk when both distributions are similar. Several notions of distance have been proposed for domain adaptation (Ben-David et al., 2006; 2010; Mansour et al., 2009a;b; Germain et al., 2013). In this paper, we focus on the $\mathcal{H}$-divergence used by Ben-David et al. (2006; 2010), and based on the earlier work of Kifer et al. (2004).

Definition 1 (Ben-David et al. (2006; 2010); Kifer et al. (2004))**.** Given two domain distributions $\mathcal{D}_\text{S}^x$ and $\mathcal{D}_\text{T}^x$ over $\mathcal{X}$, and a hypothesis class $\mathcal{H}$, the *$\mathcal{H}$-divergence* between $\mathcal{D}_\text{S}^x$ and $\mathcal{D}_\text{T}^x$ is

$$d_\mathcal{H}(\mathcal{D}_\text{S}^x, \mathcal{D}_\text{T}^x) \stackrel{\text{def}}{=} 2 \sup_{\eta \in \mathcal{H}} \left| \Pr_{\mathbf{x}^s \sim \mathcal{D}_\text{S}^x} \left[\eta(\mathbf{x}^s) = 1\right] - \Pr_{\mathbf{x}^t \sim \mathcal{D}_\text{T}^x} \left[\eta(\mathbf{x}^t) = 1\right] \right|.$$

That is, the $\mathcal{H}$-divergence relies on the capacity of the hypothesis class $\mathcal{H}$ to distinguish between examples generated by $\mathcal{D}_\text{S}^x$ from examples generated by $\mathcal{D}_\text{T}^x$. Ben-David et al. (2006; 2010) proved that, for a symmetric hypothesis class $\mathcal{H}$, one can compute the *empirical $\mathcal{H}$-divergence* between two samples $S \sim (\mathcal{D}_\text{S}^x)^m$ and $T \sim (\mathcal{D}_\text{T}^x)^{m'}$ by computing

$$\hat{d}_\mathcal{H}(S, T) \stackrel{\text{def}}{=} 2\left(1 - \min_{\eta \in \mathcal{H}} \left[\frac{1}{m} \sum_{i=1}^m I[\eta(\mathbf{x}_i^s) = 1] + \frac{1}{m'} \sum_{i=1}^{m'} I[\eta(\mathbf{x}_i^t) = 0]\right]\right), \tag{1}$$

where $I[a]$ is the indicator function which is 1 if predicate a is true, and 0 otherwise.

2.2. Proxy Distance

Ben-David et al. (2006; 2010) suggested that, even if it is generally hard to compute $\hat{d}_\mathcal{H}(S, T)$ exactly (*e.g.*, when $\mathcal{H}$ is the space of linear classifiers on $\mathcal{X}$), we can easily approximate it by running a learning algorithm on the problem of discriminating between source and target examples. To do so, we construct a new dataset

$$U = \{(\mathbf{x}_i^s, 1)\}_{i=1}^m \cup \{(\mathbf{x}_i^t, 0)\}_{i=1}^{m'}, \tag{2}$$

where the examples of the source sample are labeled 1 and the examples of the target sample are labeled 0. Then, the risk of the classifier trained on new dataset U approximates the "min" part of Equation (1). Thus, given a test error ϵ on the problem of discriminating between source and target examples, this *Proxy A-distance* (PAD) is given by

$$\hat{d}_A = 2(1 - 2\epsilon). \tag{3}$$

In the experiments section of this paper, we compute the PAD value following the approach of Glorot et al. (2011); Chen et al. (2012), *i.e.*, we train a linear SVM on a subset of dataset U (Equation (2)), and we use the obtained classifier error on the other subset as the value of ϵ in Equation (3).

2.3. Generalization Bound on the Target Risk

The work of Ben-David et al. (2006; 2010) also showed that the $\mathcal{H}$-divergence $d_\mathcal{H}(\mathcal{D}_\text{S}^x, \mathcal{D}_\text{T}^x)$ is upper bounded by its empirical estimate $\hat{d}_\mathcal{H}(S, T)$ plus a constant complexity term that depends on the *VC dimension* of $\mathcal{H}$ and the size of samples S and T. By combining this result with a similar bound on the source risk, the following theorem is obtained.

Theorem 2 (Ben-David et al. (2006))**.** *Let $\mathcal{H}$ be a hypothesis class of VC dimension d. With probability $1 - \delta$ over the choice of samples $S \sim (\mathcal{D}_\text{S})^m$ and $T \sim (\mathcal{D}_\text{T}^x)^m$, for every $\eta \in \mathcal{H}$:*

$$R_{\mathcal{D}_\text{T}}(\eta) \leq R_S(\eta) + \sqrt{\frac{4}{m}\left(d \log \tfrac{2em}{d} + \log \tfrac{4}{\delta}\right)} + \hat{d}_\mathcal{H}(S, T) + 4\sqrt{\frac{1}{m}\left(d \log \tfrac{2m}{d} + \log \tfrac{4}{\delta}\right)} + \beta,$$

with $\beta \geq \inf_{\eta^* \in \mathcal{H}} \left[R_{\mathcal{D}_\mathrm{S}}(\eta^*) + R_{\mathcal{D}_\mathrm{T}}(\eta^*) \right]$, *and*

$$R_S(\eta) \;=\; \frac{1}{m} \sum_{i=1}^{m} I\left[\eta(\mathbf{x}_i^s) \neq y_i^s\right]$$

is the empirical source risk.

The previous result tells us that $R_{\mathcal{D}_\mathrm{T}}(\eta)$ can be low only when the β term is low, *i.e.*, only when there exists a classifier that can achieve a low risk on both distributions. It also tells us that, to find a classifier with a small $R_{\mathcal{D}_\mathrm{T}}(\eta)$ in a given class of fixed VC dimension, the learning algorithm should minimize (in that class) a trade-off between the source risk $R_S(\eta)$ and the empirical $\mathcal{H}$-divergence $\hat{d}_{\mathcal{H}}(S,T)$. As pointed-out by Ben-David et al. (2006), a strategy to control the $\mathcal{H}$-divergence is to find a representation of the examples where both the source and the target domain are as indistinguishable as possible. Under such a representation, a hypothesis with a low source risk will, according to Theorem 2, perform well on the target data. In this paper, we present an algorithm that directly exploits this idea.

3. A Domain-Adversarial Neural Network

The originality of our approach is to explicitly implement the idea exhibited by Theorem 2 into a neural network classifier. That is, to learn a model that can generalize well from one domain to another, we ensure that the internal representation of the neural network contains no discriminative information about the origin of the input (source or target), while preserving a low risk on the source (labeled) examples.

3.1. Source Risk Minimization (Standard NN)

Let us consider the following standard neural network (NN) architecture with one hidden layer:

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= \mathrm{sigm}\big(\mathbf{b} + \mathbf{W}\mathbf{x}\big)\,, \\ \mathbf{f}(\mathbf{x}) &= \mathrm{softmax}\big(\mathbf{c} + \mathbf{V}\mathbf{h}(\mathbf{x})\big)\,, \end{aligned} \tag{4}$$

with

$$\begin{aligned} \mathrm{sigm}(\mathbf{a}) &\stackrel{\mathrm{def}}{=} \left[\frac{1}{1+\exp(-a_i)}\right]_{i=1}^{|\mathbf{a}|}, \\ \mathrm{softmax}(\mathbf{a}) &\stackrel{\mathrm{def}}{=} \left[\frac{\exp(a_i)}{\sum_{j=1}^{|\mathbf{a}|}\exp(a_j)}\right]_{i=1}^{|\mathbf{a}|}. \end{aligned}$$

Note that each component $f_y(\mathbf{x})$ of $\mathbf{f}(\mathbf{x})$ denotes the conditional probability that the neural network assigns $\mathbf{x}$ to class y. Given a training source sample $S = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^m$, the natural classification loss to use is the negative log-probability of the correct label:

$$\mathcal{L}\big(\mathbf{f}(\mathbf{x}), y\big) \;\stackrel{\mathrm{def}}{=}\; \log \frac{1}{f_y(\mathbf{x})}\,.$$

This leads to the following learning problem on the source domain.

$$\min_{\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c}} \left[\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\big(\mathbf{f}(\mathbf{x}_i^s), y_i^s\big) \right]. \tag{5}$$

We view the output of the hidden layer $\mathbf{h}(\cdot)$ (Equation (4)) as the internal representation of the neural network. Thus, we denote the source sample representations as

$$\mathbf{h}(S) \;\stackrel{\mathrm{def}}{=}\; \big\{\mathbf{h}(\mathbf{x}_i^s)\big\}_{i=1}^m\,.$$

3.2. A Domain Adaptation Regularizer

Now, consider an unlabeled sample from the target domain $T = \{\mathbf{x}_i^t\}_{i=1}^{m'}$ and the corresponding representations $\mathbf{h}(T) = \{\mathbf{h}(\mathbf{x}_i^t)\}_{i=1}^{m'}$. Based on Equation (1), the empirical $\mathcal{H}$-divergence of a symmetric hypothesis class $\mathcal{H}$ between samples $\mathbf{h}(S)$ and $\mathbf{h}(T)$ is given by

$$\hat{d}_{\mathcal{H}}\big(\mathbf{h}(S), \mathbf{h}(T)\big) \;= \tag{6}$$

$$2\left(1 - \min_{\eta \in \mathcal{H}} \left[\frac{1}{m} \sum_{i=1}^{m} I\big[\eta(\mathbf{h}(\mathbf{x}_i^s)) = 1\big] + \frac{1}{m'} \sum_{i=1}^{m'} I\big[\eta(\mathbf{h}(\mathbf{x}_i^t)) = 0\big] \right]\right).$$

Let us consider $\mathcal{H}$ as the class of hyperplanes in the representation space. Inspired by the Proxy A-distance (see Section 2.2), we suggest estimating the "min" part of Equation (6) by a logistic regressor that model the probability that a given input (either $\mathbf{x}^s$ or $\mathbf{x}^t$) is from the source domain $\mathcal{D}_\mathrm{S}^x$ (denoted $z = 1$) or the target domain $\mathcal{D}_\mathrm{T}^x$ (denoted $z = 0$):

$$p(z = 1 \mid \boldsymbol{\phi}) \;=\; o(\boldsymbol{\phi}) \;\stackrel{\mathrm{def}}{=}\; \mathrm{sigm}(d + \mathbf{u}^\top \boldsymbol{\phi})\,, \tag{7}$$

where $\boldsymbol{\phi}$ is either $\mathbf{h}(\mathbf{x}^s)$ or $\mathbf{h}(\mathbf{x}^t)$. Hence, the function $o(\cdot)$ is a *domain regressor*.

This enables us to add a domain adaptation term to the objective of Equation (5), giving the following problem to solve:

$$\min_{\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c}} \left[\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\big(\mathbf{f}(\mathbf{x}_i^s), y_i^s\big) \right. \tag{8}$$

$$\left. + \lambda \max_{\mathbf{u},d} \left(-\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^d\big(o(\mathbf{x}_i^s), 1\big) - \frac{1}{m'} \sum_{i=1}^{m'} \mathcal{L}^d\big(o(\mathbf{x}_i^t), 0\big) \right) \right],$$

where the hyper-parameter $\lambda > 0$ weights the domain adaptation regularization term and

$$\mathcal{L}^d\big(o(\mathbf{x}), z\big) \;=\; -z \log\big(o(\mathbf{x})\big) - (1-z) \log\big(1 - o(\mathbf{x})\big)\,.$$

In line with Theorem 2, this optimization problem implements a trade-off between the minimization of the source risk $R_S(\cdot)$ and the divergence $\hat{d}_{\mathcal{H}}(\cdot,\cdot)$. The hyper-parameter λ is then used to tune the trade-off between these two quantities during the learning process.

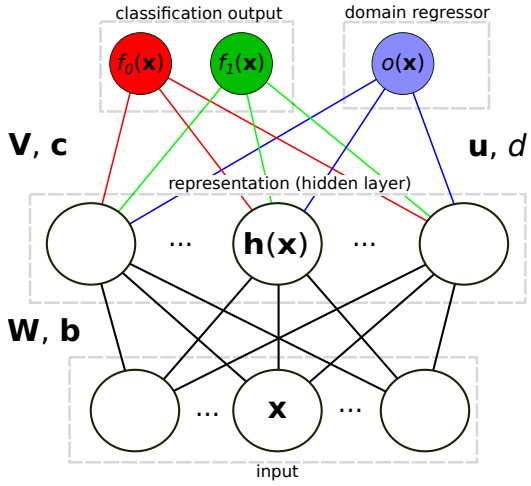


Figure 1. DANN architecture

3.3. Learning Algorithm (DANN)

We see that Equation (8) involves a maximization operation. Hence, the neural network (parametrized by $\{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$) and the domain regressor (parametrized by $\{\mathbf{u}, d\}$) are competing against each other, in an adversarial way, for that term. The obtained *domain adversarial neural network* (DANN) is illustrated by Figure 1. In DANN, the hidden layer $\mathbf{h}(\cdot)$ maps an example (either source or target) into a representation in which the output layer $\mathbf{f}(\cdot)$ accurately classifies the source sample, while the domain regressor $o(\cdot)$ is unable to detect if an example belongs to the source sample or the target sample.

To optimize Equation (8), one option would be to follow a hard-EM approach, where we would alternate between optimizing until convergence the adversarial parameters $\mathbf{u}, d$ and the other regular neural network parameters $\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}$. However, we've found that a simpler stochastic gradient descent (SGD) approach is sufficient and works well in practice. Here, an SGD approach consists in sampling a pair of source and target example $\mathbf{x}_i^s, \mathbf{x}_j^t$ and updating a gradient step update of all parameters of DANN. Crucially, while the update of the regular parameters follows as usual the opposite direction of the gradient, for the adversarial parameters $\mathbf{u}, d$ the step must follow the gradient's direction (since we maximize with respect to them, instead of minimizing). The algorithm is detailed in Algorithm 1. In the pseudocode, we use $\mathbf{e}(y)$ to represent a "one-hot" vector, consisting of all 0s except for a 1 at position y. Also, $\odot$ is the element-wise product.

For each experiment described in this paper, we used *early stopping* as the stopping criteria: we split the source labeled sample to use 90% as the training set S and the remaining 10% as a validation set S_V. We stop the learning process when the risk on S_V is minimal.

Algorithm 1 DANN

Input: samples $S = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^m$ and $T = \{\mathbf{x}_i^t\}_{i=1}^{m'}$,
hidden layer size l, adaptation parameter λ, learning rate α.
Output: neural network $\{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$

$\mathbf{W}, \mathbf{V} \leftarrow \text{random\_init}(l)$
$\mathbf{b}, \mathbf{c}, \mathbf{u}, d \leftarrow 0$
while stopping criteria is not met **do**
  for i from 1 to m **do**
    `# Forward propagation`
    $\mathbf{h}(\mathbf{x}_i^s) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_i^s)$
    $\mathbf{f}(\mathbf{x}_i^s) \leftarrow \text{softmax}(\mathbf{c} + \mathbf{V}\mathbf{h}(\mathbf{x}_i^s))$
    `# Backpropagation`
    $\Delta_\mathbf{c} \leftarrow -(\mathbf{e}(y_i^s) - \mathbf{f}(\mathbf{x}_i^s))$
    $\Delta_\mathbf{V} \leftarrow \Delta_\mathbf{c}\, \mathbf{h}(\mathbf{x}_i^s)^\top$
    $\Delta_\mathbf{b} \leftarrow \left(\mathbf{V}^\top \Delta_\mathbf{c}\right) \odot \mathbf{h}(\mathbf{x}_i^s) \odot (1 - \mathbf{h}(\mathbf{x}_i^s))$
    $\Delta_\mathbf{W} \leftarrow \Delta_\mathbf{b} \cdot (\mathbf{x}_i^s)^\top$
    `# Domain adaptation regularizer...`
    `# ...from current domain`
    $o(\mathbf{x}_i^s) \leftarrow \text{sigm}(d + \mathbf{u}^\top \mathbf{h}(\mathbf{x}_i^s))$
    $\Delta_d \leftarrow \lambda(1 - o(\mathbf{x}_i^s))$; $\Delta_\mathbf{u} \leftarrow \lambda(1 - o(\mathbf{x}_i^s))\mathbf{h}(\mathbf{x}_i^s)$
    $\text{tmp} \leftarrow \lambda(1 - o(\mathbf{x}_i^s))\mathbf{u} \odot \mathbf{h}(\mathbf{x}_i^s) \odot (1 - \mathbf{h}(\mathbf{x}_i^s))$
    $\Delta_\mathbf{b} \leftarrow \Delta_\mathbf{b} + \text{tmp}$; $\Delta_\mathbf{W} \leftarrow \Delta_\mathbf{W} + \text{tmp} \cdot (\mathbf{x}_i^s)^\top$
    `# ...from other domain`
    $j \leftarrow \text{uniform\_integer}(1, \ldots, m')$
    $\mathbf{h}(\mathbf{x}_j^t) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_j^t)$
    $o(\mathbf{x}_j^t) \leftarrow \text{sigm}(d + \mathbf{u}^\top \mathbf{h}(\mathbf{x}_j^t))$
    $\Delta_d \leftarrow \Delta_d - \lambda o(\mathbf{x}_j^t)$; $\Delta_\mathbf{u} \leftarrow \Delta_\mathbf{u} - \lambda o(\mathbf{x}_j^t)\mathbf{h}(\mathbf{x}_j^t)$
    $\text{tmp} \leftarrow -\lambda o(\mathbf{x}_j^t)\mathbf{u} \odot \mathbf{h}(\mathbf{x}_j^t) \odot (1 - \mathbf{h}(\mathbf{x}_j^t))$
    $\Delta_\mathbf{b} \leftarrow \Delta_\mathbf{b} + \text{tmp}$; $\Delta_\mathbf{W} \leftarrow \Delta_\mathbf{W} + \text{tmp} \cdot (\mathbf{x}_j^t)^\top$
    `# Update neural network parameters`
    $\mathbf{W} \leftarrow \mathbf{W} - \alpha\Delta_\mathbf{W}$; $\mathbf{V} \leftarrow \mathbf{V} - \alpha\Delta_\mathbf{V}$
    $\mathbf{b} \leftarrow \mathbf{b} - \alpha\Delta_\mathbf{b}$; $\mathbf{c} \leftarrow \mathbf{c} - \alpha\Delta_\mathbf{c}$
    `# Update domain classifier parameters`
    $\mathbf{u} \leftarrow \mathbf{u} + \alpha\Delta_\mathbf{u}$; $d \leftarrow d + \alpha\Delta_d$
  end for
end while

4. Related Work

As mentioned previously, the general approach of achieving domain adaptation by learning a new data representation has been explored under many facets. A large part of the literature however has focused mainly on linear hypothesis (see for instance Blitzer et al., 2006; Bruzzone & Marconcini, 2010; Germain et al., 2013; Baktashmotlagh et al., 2013; Cortes & Mohri, 2014). More recently, non-linear representations have become increasingly studied, including neural network representations (Glorot et al., 2011; Li et al., 2014) and most notably the state-of-the-art mSDA (Chen et al., 2012). That literature has mostly focused on exploiting the principle of robust representations, based on the denoising autoencoder paradigm (Vincent et al., 2008). One of the contribution of this work is to show that domain discriminability is another principle that is complimentary to robustness and can improve cross-domain adaptation.

What distinguishes this work from most of the domain adaptation literature is DANN's inspiration from the theoretical work of Ben-David et al. (2006; 2010). Indeed, DANN directly optimizes the notion of $\mathcal{H}$-divergence. We do note the work of Huang & Yates (2012), in which HMM representations are learned for word tagging using a posterior regularizer that is also inspired by Ben-David et al.'s work. In addition to the tasks being different (word tagging versus sentiment classification), we would argue that DANN learning objective more closely optimizes the $\mathcal{H}$-divergence, with Huang & Yates (2012) relying on cruder approximations for efficiency reasons.

The idea of learning representations that are indiscriminate of some auxiliary label has also been explored in other contexts. For instance, Zemel et al. (2013) proposes the notion of fair representations, which are indiscriminate to whether an example belongs to some identified set of groups. The resulting algorithm is different from DANN and not directly derived from the $\mathcal{H}$-divergence.

Finally, we mention some related research on using an adversarial (minimax) formulation to learn a model, such as a classifier or a neural network, from data. There has been work on learning linear classifiers that are robust to changes in the input distribution, based on a minimax formulation (Bagnell, 2005; Liu & Ziebart, 2014). This work however assumes that a good feature representation of the input for a linear classifier is available and doesn't address the problem of learning it. We also note the work of Goodfellow et al. (2014), who propose generative adversarial networks to learn a good generative model of the true data distribution. This work shares with DANN the use of an adversarial objective, applying it instead to the unsupervised problem of generative modeling.

5. Experiments

5.1. Toy Problem

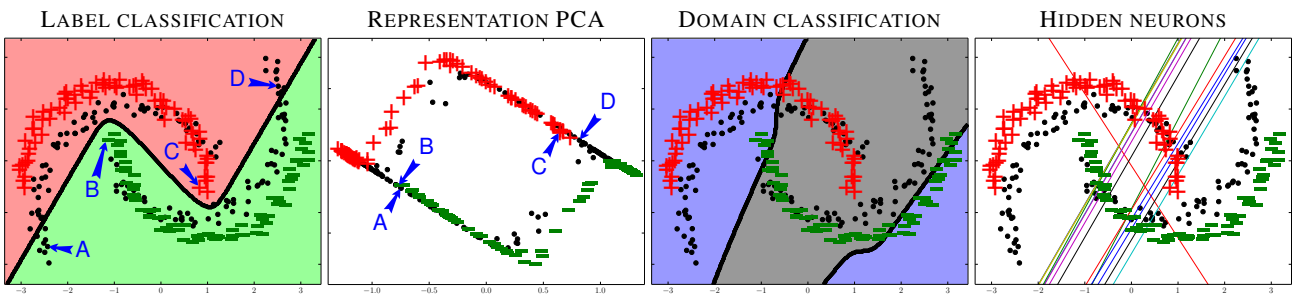
As a first experiment, we study the behavior of the proposed DANN algorithm on a variant of the *inter-twinning moons* 2D problem, where the target distribution is a rotation of the source distribution. For the source sample S, we generate a lower moon and an upper moon labeled 0 and 1 respectively, each of which containing 150 examples. The target sample T is obtained by generating a sample in the same way as S (without keeping the labels) and then by rotating each example by 35°. Thus, T contains 300 unlabeled examples. In Figure 2, the examples from S are represented by "+"and "−", and the examples from T are represented by black dots.

We study the adaptation capability of DANN by comparing it to the standard NN. In our experiments, both algorithms share the same network architecture, with a hidden layer size of 15 neurons. We even train NN using the same procedure as DANN. That is, we keep updating the domain regressor component using target sample T (with a hyperparameter $\lambda = 6$; the same value used for DANN), but we disable the *adversarial* back-propagation into the hidden layer. To do so, we execute Algorithm 1 by omitting the lines numbered 21 and 28. In this way, we obtain a NN learning algorithm – based on the source risk minimization of Equation (5) – and simultaneously train the domain regressor of Equation (7) to discriminate between source and target domains. Using this toy experiment, we will first illustrate how DANN adapts its decision boundary compared to NN. Moreover, we will also illustrate how the representation given by the hidden layer is less adapted to the domain task with DANN than it is with NN. The results are illustrated in Figure 2, where the graphs in part (a) relate to the standard NN, and the graphs in part (b) relate to DANN. By looking at the corresponding (a) and (b) graphs in each column, we compare NN and DANN from four different perspectives, described in detail below.

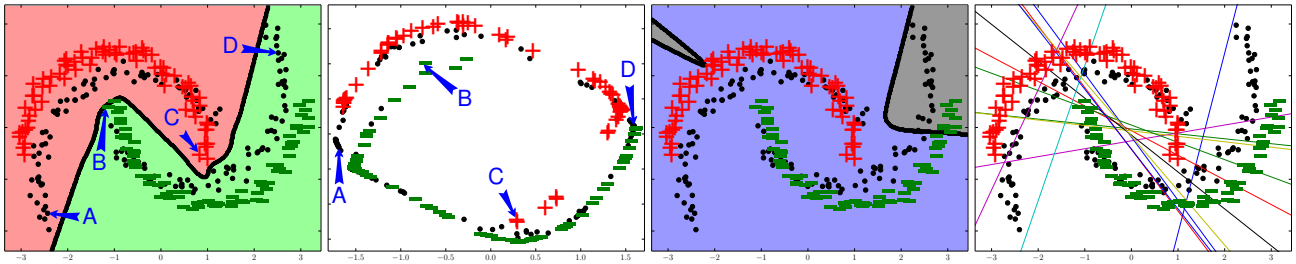
Label classification. The first column of Figure 2 shows the decision boundaries of DANN and NN on the problem of predicting the labels of both source and the target examples. As expected, NN accurately classifies the two classes of the source sample S, but is *not fully adapted* to the target sample T. On the contrary, the decision boundary of DANN perfectly classifies examples from both source and target samples. DANN clearly adapts here to the target distribution.

Representation PCA. To analyze how he domain adaptation regularizer affects the representation $\mathbf{h}(\cdot)$ provided by the hidden layer, the second column of Figure 2 presents a principal component analysis (PCA) on the set of all representations of source and target data points, *i.e.*, $\mathbf{h}(S) \cup \mathbf{h}(T)$. Thus, given the trained network (NN or DANN), every point from S and T is mapped into a 15-dimensional feature space through the hidden layer, and projected back into a two-dimensional plane defined by the first two principal components. In the DANN-PCA representation, we observe that target points are homogeneously spread out among the source points. In the NN-PCA representation, clusters of target points containing very few source points are clearly visible. Hence, the task of labeling the target points seems easier to perform on the DANN-PCA representation.

To push the analysis further, four crucial data points identified by A, B, C and D in the graphs of the first column (which correspond to the moon extremities in the original space) are represented again on the graphs of the second column. We observe that points A and B are very close to each other in the NN-PCA representation, while they clearly belong to different classes. The same hap-



(a) Standard NN and a *non adverserial* domain regressor (Algorithm 1, without Lines 21 and 28)



(b) DANN (Algorithm 1)

Figure 2. The *inter-twinning moons* toy problem. We adopt the colors of Figure 1 for classification output and domain regressor value.

pens to points C and D. Conversely, these four points are located at opposite corners in the DANN-PCA representation. Note also that the target point A (resp. D) – which is difficult to classify in the original space – is located in the "+"cluster (resp. "−"cluster) in the DANN-PCA representation. Therefore, the representation promoted by DANN is more suited for the domain adaptation task.

Domain classification. The third column of Figure 2 shows the decision boundary on the domain classification problem, which is given by the domain regressor $o(\cdot)$ of Equation (7). More precisely, $\mathbf{x}$ is classified as a source example when $o(\mathbf{x}) \geq 0.5$, and is classified as a domain example otherwise. Remember that, during the learning process of DANN, the $o(\cdot)$ regressor struggles to discriminate between source and target domains, while the hidden representation $\mathbf{h}(\cdot)$ is *adversarially* updated to prevent it to succeed. As explained above, we trained the domain regressor during the learning process of NN, but without allowing it to influence the learned representation $\mathbf{h}(\cdot)$.

On one hand, the DANN domain regressor utterly fails to discriminate the source and target distributions. On the other hand, the NN domain regressor shows a better (although imperfect) discriminant. This again corroborates that the DANN representation doesn't allow discriminating between domains.

Hidden neurons. In the plot of the last column of Figure 2, the lines show the decision surfaces of the hidden layer neurons (defined by Equation (4)). In other words, each of the fifteen plot line corresponds to the points $\mathbf{x} \in \mathbb{R}^2$ for which the ith component of $\mathbf{h}(\mathbf{x})$ equals $\frac{1}{2}$, for $i \in \{1, \ldots, 15\}$.

We observe that the neurons of NN are grouped in three clusters, each one allowing to generate a straight line part of the curved decision boundary for the label classification problem. However, most of these neurons are also able to (roughly) capture the rotation angle of the domain classification problem. Hence, we observe that the adaptation regularizer of DANN prevents these kinds of neurons to be produced. It is indeed striking to see that the two predominant patterns in the NN neurons (*i.e.*, the two parallel lines crossing the plane from the lower left corner to the upper right corner) are absent among DANN neurons.

5.2. Sentiment Analysis Dataset

In this section, we compare the performance of our proposed DANN algorithm to a standard neural network with one hidden layer (NN) described by Equation (5), and a Support Vector Machine (SVM) with a linear kernel. To select the hyper-parameters of each of these algorithms, we use grid search and a very small validation set which consists in 100 labeled examples from the target domain. Finally, we select the classifiers having the lowest target validation risk.

We compare the algorithms on the *Amazon reviews* dataset, as pre-processed by Chen et al. (2012). This dataset includes four domains, each one composed of reviews of a

Table 1. Error rates on the Amazon reviews dataset (left), and Pairwise Poisson binomial test (right).

(a) Error rates on the Amazon reviews dataset

	Original data			mSDA representation		
source name → *target name*	DANN	NN	SVM	DANN	NN	SVM
books → dvd	0.201	**0.199**	0.206	0.176	**0.171**	0.175
books → electronics	**0.246**	0.251	0.256	**0.197**	0.228	0.244
books → kitchen	0.230	0.235	**0.229**	0.169	**0.166**	0.172
dvd → books	**0.247**	0.261	0.269	0.176	**0.173**	0.176
dvd → electronics	**0.247**	0.256	0.249	**0.181**	0.234	0.220
dvd → kitchen	**0.227**	**0.227**	0.233	**0.151**	0.153	0.178
electronics → books	**0.280**	0.281	0.290	0.237	0.241	**0.229**
electronics → dvd	**0.273**	0.277	0.278	**0.216**	0.228	0.261
electronics → kitchen	**0.148**	0.149	0.163	**0.118**	0.126	0.137
kitchen → books	**0.283**	0.288	0.325	**0.222**	0.226	0.234
kitchen → dvd	**0.261**	**0.261**	0.274	**0.208**	0.214	0.209
kitchen → electronics	0.161	0.161	**0.158**	0.141	**0.136**	0.138

(b) Pairwise Poisson binomial test

Original data			
	DANN	NN	SVM
DANN	0.50	**0.90**	**0.97**
NN	0.10	0.50	0.87
SVM	0.03	0.13	0.50

mSDA representations			
	DANN	NN	SVM
DANN	0.50	**0.82**	**0.88**
NN	0.18	0.50	0.90
SVM	0.12	0.10	0.50

specific kind of product (books, dvd disks, electronics, and kitchen appliances). Reviews are encoded in $5\,000$ dimensional feature vectors of unigrams and bigrams, and labels are binary: "0" if the product is ranked up to 3 stars, and "1" if the product is ranked 4 or 5 stars.

We perform twelve domain adaptation tasks. For example, "books → dvd" corresponds to the task for which books is the source domain and dvd disks the target one. All learning algorithms are given $2\,000$ labeled source examples and $2\,000$ unlabeled target examples. Then, we evaluate them on separate target test sets (between $3\,000$ and $6\,000$ examples). Note that NN and SVM don't use the unlabeled target sample for learning. Here are more details about the procedure used for each learning algorithms.

DANN. The adaptation parameter λ is chosen among 9 values between 10^{-2} and 1 on a logarithmic scale. The hidden layer size l is either $1, 5, 12, 25, 50, 75, 100, 150,$ or 200. Finally, the learning rate α is fixed at 10^{-3}.

NN. We use exactly the same hyper-parameters and training procedure as DANN above, except that we don't need an adaptation parameter. Note that one can train NN by using the DANN implementation (Algorithm 1) with $\lambda = 0$.

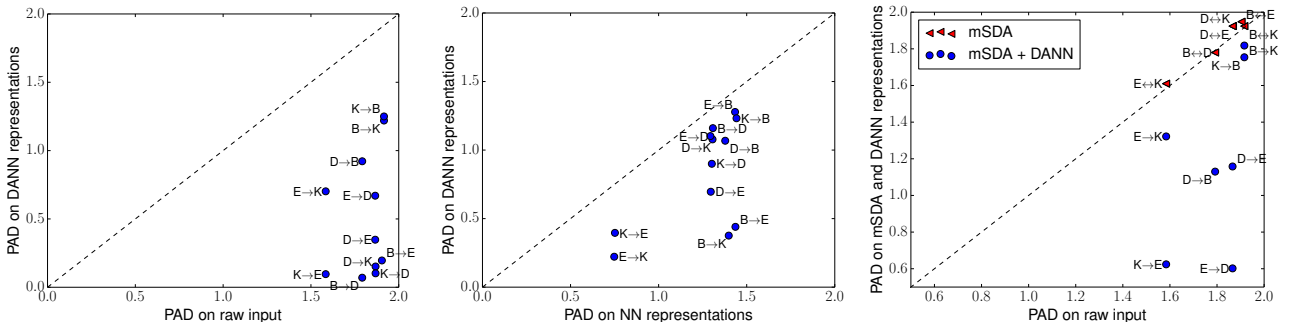
SVM. The hyper-parameter C of the SVM is chosen among 10 values between 10^{-5} and 1 on a logarithmic scale. This range of values is the same used by Chen et al. (2012) in their experiments.

The "Original data" part of Table 1(a) shows the target test risk of all algorithms, and Table 1(b) reports the probability that one algorithm is significantly better than another according to the Poisson binomial test (Lacoste et al., 2012). We note that DANN has a significantly better performance than NN and SVM, with respective probabilities **0.90** and **0.97**. As the only difference between DANN and NN is the domain adaptation regularizer, we conclude that our approach successfully helps to find a representation suitable for the target domain.

5.3. Combining DANN with Autoencoders

We now wonder whether our DANN algorithm can improve on the representation learned by the state-of-the-art *Marginalized Stacked Denoising Autoencoders* (mSDA) proposed by Chen et al. (2012). In brief, mSDA is an unsupervised algorithm that learns a new robust feature representation of the training samples. It takes the unlabeled parts of both source and target samples to learn a feature map from the input space $\mathcal{X}$ to a new representation space. As a *denoising autoencoder*, it finds a feature representation from which one can (approximately) reconstruct the original features of an example from its noisy counterpart. Chen et al. (2012) showed that using mSDA with a linear SVM classifier gives state-of-the-art performance on the *Amazon reviews* datasets. As an alternative to the SVM, we propose to apply our DANN algorithm on the same representations generated by mSDA (using representations of both source and target samples). Note that, even if mSDA and DANN are two representation learning approaches, they optimize different objectives, which can be complementary.

We perform this experiment on the same *amazon reviews* dataset described in the previous subsection. For each pair source-target, we generate the mSDA representations using a corruption probability of 50% and a number of layers of 5. We then execute the three learning algorithms (DANN, NN, and SVM) on these representations. More precisely, following the experimental procedure of Chen et al. (2012), we use the concatenation of the output of the 5 layers and the original input as the new representation. Thus, each example is now encoded in a vector of



(a) DANN on *Original data*. (b) DANN & NN with 100 hidden neurons. (c) DANN on *mSDA representations*.

Figure 3. Proxy A-distances (PAD). Note that the PAD values of mSDA representations are symmetric when swapping source and target samples. Also, some DANN results, when used on top of mSDA, have PAD values lower than 0.5 and doesn't appear on Figure (c).

30 000 dimensions. Note that we use the same grid search as in Subsection 5.2, but with a learning rate α of 10^{-4} for both DANN and NN. The results of "mSDA representation" columns in Table 1(a) confirm that combining mSDA and DANN is a sound approach. Indeed, the Poisson binomial test shows that DANN has a better performance than NN and SVM with probabilities **0.82** and **0.88** respectively, as reported in Table 1(b).

5.4. Proxy A-distance

The theoretical foundation of DANN is the domain adaptation theory of Ben-David et al. (2006; 2010). We claimed that DANN finds a representation in which the source and the target example are hardly distinguishable. Our toy experiment of Section 5.1 already points out some evidences, but we want to confirm it on real data. To do so, we compare the Proxy A-distance (PAD) on various representations of the *Amazon Reviews* dataset. These representations are obtained by running either NN, DANN, mSDA, or mSDA and DANN combined. Recall that PAD, as described in Section 2.2, is a metric estimating the similarity of the source and the target representations. More precisely, to obtain a PAD value, we use the following procedure: (1) we construct the dataset U of Equation (2) using both source and target representations of the training samples; (2) we randomly split U in two subsets of equal size; (3) we train linear SVMs on the first subset of U using a large range of C values; (4) we compute the error of all obtained classifiers on the second subset of U; and (5) we use the lowest error to compute the PAD value of Equation (3).

Firstly, Figure 3(a) compares the PAD of DANN representations obtained in the experiments of Section 5.2 (using the hyper-parameters values leading to the results of Table 1) to the PAD computed on raw data. As expected, the PAD values are driven down by the DANN representations.

Secondly, Figure 3(b) compares the PAD of DANN representations to the PAD of standard NN representations. As the PAD is influenced by the hidden layer size (the discriminating power tends to increase with the dimension of the representation), we fix here the size to 100 neurons for both algorithms. We also fix the adaptation parameter of DANN to $\lambda \simeq 0.31$ as it was the value that has been selected most of the time during our preceding experiments on the *Amazon Reviews* dataset. Again, DANN is clearly leading to the lowest PAD values.

Lastly, Figure 3(c) presents two sets of results related to Section 5.3 experiments. On one hand, we reproduce the results of Chen et al. (2012), which noticed that the mSDA representations gave greater PAD values than those obtained with the original (raw) data. Although the mSDA approach clearly helps to adapt to the target task, it seems to contradict the theory of Ben-David et al.. On the other hand, we observe that, when running DANN on top of mSDA (using the hyper-parameters values leading to the results of Table 1), the obtained representations have much lower PAD values. These observations might explain the improvements provided by DANN when combined with the mSDA procedure.

6. Conclusion and Future Work

In this paper, we have proposed a neural network algorithm, named DANN, that is strongly inspired by the domain adaptation theory of Ben-David et al. (2006; 2010). The main idea behind DANN is to encourage the network's hidden layer to learn a representation which is predictive of the source example labels, but uninformative about the domain of the input (source or target). Extensive experiments on the *inter-twinning moons* toy problem and *Amazon reviews* sentiment analysis dataset have shown the effectiveness of this strategy. Notably, we achieved state-of-the-art performances when combining DANN with the mSDA autoencoders of Chen et al. (2012), which turned out to be

two complementary representation learning approaches.

We believe that our domain adaptation regularizer that we develop for the DANN algorithm can be incorporated into many other learning algorithms. Natural extensions of our work would be deeper network architectures, multi-source adaptation problems and other learning tasks beyond the basic binary classification setting. We also intend to meld the DANN approach with denoising autoencoders, to potentially improve on the two steps procedure of Section 5.3.

References

Bagnell, J. Andrew. Robust supervised learning. In *Proceedings of 20th National Conference on Artificial Intelligence*, pp. 714–719. AAAI Press / The MIT Press, 2005.

Baktashmotlagh, Mahsa, Harandi, Mehrtash, Lovell, Brian, and Salzmann, Mathieu. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, pp. 769–776, 2013.

Ben-David, Shai, Blitzer, John, Crammer, Koby, and Pereira, Fernando. Analysis of representations for domain adaptation. In *NIPS*, pp. 137–144, 2006.

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

Blitzer, John, McDonald, Ryan T., and Pereira, Fernando. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, pp. 120–128, 2006.

Bruzzone, Lorenzo and Marconcini, Mattia. Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *Transaction Pattern Analysis and Machine Intelligence*, 32(5):770–787, 2010.

Chen, Minmin, Xu, Zhixiang Eddie, Weinberger, Kilian Q., and Sha, Fei. Marginalized denoising autoencoders for domain adaptation. In *ICML*, pp. 767–774, 2012.

Cortes, Corinna and Mohri, Mehryar. Domain adaptation and sample bias correction theory and algorithm for regression. *Theor. Comput. Sci.*, 519:103–126, 2014.

Germain, Pascal, Habrard, Amaury, Laviolette, François, and Morvant, Emilie. A PAC-Bayesian approach for domain adaptation with specialization to linear classifiers. In *ICML*, pp. 738–746, 2013.

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pp. 513–520, 2011.

Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets. In *NIPS*, pp. 2672–2680, 2014.

Huang, Fei and Yates, Alexander. Biased representation learning for domain adaptation. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1313–1323, 2012.

Kifer, Daniel, Ben-David, Shai, and Gehrke, Johannes. Detecting change in data streams. In *Very Large Data Bases*, pp. 180–191, 2004.

Lacoste, Alexandre, Laviolette, François, and Marchand, Mario. Bayesian comparison of machine learning algorithms on single and multiple datasets. In *AISTATS*, pp. 665–675, 2012.

Li, Yujia, Swersky, Kevin, and Zemel, Richard. Unsupervised domain adaptation by domain invariant projection. In *NIPS 2014 Workshop on Transfer and Multitask Learning*, 2014.

Liu, Anqi and Ziebart, Brian D. Robust classification under sample selection bias. In *NIPS*, pp. 37–45, 2014.

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

Mansour, Yishay, Mohri, Mehryar, and Rostamizadeh, Afshin. Multiple source adaptation and the rényi divergence. In *UAI*, pp. 367–374, 2009b.

Vincent, Pascal, Larochelle, Hugo, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Extracting and composing robust features with denoising autoencoders. In *ICML*, pp. 1096–1103, 2008.

Zemel, Richard S., Wu, Yu, Swersky, Kevin, Pitassi, Toniann, and Dwork, Cynthia. Learning fair representations. In *ICML*, pp. 325–333, 2013.