CONCEPTUAL DOMAIN ADAPTATION USING DEEP LEARNING

A Dissertation Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Behrang Mehrparvar

August 2017

CONCEPTUAL DOMAIN ADAPTATION

USING DEEP LEARNING

Behrang Mehrparvar

APPROVED:

Ricardo Vilalta, Chairman
Dept. of Computer Science

Albert Cheng
Dept. of Computer Science

Carlos Ordonez
Dept. of Computer Science

Klaus Kaiser
Dept. of Mathematics

Dean, College of Natural Sciences and Mathematics

CONCEPTUAL DOMAIN ADAPTATION USING DEEP LEARNING

An Abstract of a Dissertation Presented to
the Faculty of the Department of Computer Science
University of Houston

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

By
Behrang Mehrparvar
August 2017

Abstract

Domain adaptation scenarios have been successively gaining attention in practical applications of machine learning. Here, the source distribution in which the classifier is trained, differs from the target distribution in which the classifier will ultimately be applied. The discrepancy between source and target distributions often results in poor predictive performance.

Recent work shows several approaches providing a solution to alleviate the problem of distribution discrepancy. Deep neural networks extract high-level abstract representations of data, and have been used to transform the source and target data into a new common space, such that the aforementioned discrepancy is minimized.

In this document, we introduce conceptual domain adaptation where semantic information hidden in high-level concepts –as opposed to information from low-level representational properties of the data– is directly used for adaptation. We investigate current deep learning-based domain adaptation approaches and argue that due to their reliance on representational properties of data, using them for domain adaptation is prone to failure under certain scenarios. These scenarios are investigated as cases with inherently lower-level representational discrepancy.

In this project, we introduce an adjustment approach as a solution towards conceptual domain adaptation. Accordingly, we propose a search framework to adjust high-level representation of target data along with basic supervised, graph-based and PCA-based fitness evaluation. Based on experimental results, we contend that the proposed solution is beneficial for domain adaptation problems with lower-level representational in supervised scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In many practical machine learning applications, we have access to labeled data from one domain, while another *similar* domain happens to be unlabeled. Consider the case of sentiment classification, where we have a labeled dataset on Amazon book reviews, but the kitchen appliances dataset is unlabeled. Another practical example would be spam filtering, where the first user has already labeled his emails as spam or non-spam (and induced a model on that data), while the data for the second user is unlabeled. Similarly, we can think of labeled images of high-quality SLR cameras versus unlabeled low-quality webcam images.

In all the cases above, the main idea is to apply the source model (dataset with labels) on the target domain (dataset with no labels). Although we do not have sufficient labeled samples in the target domain to train the model, we do know that the labeling function in both domains is similar; this suggests using the source model on the target domain.

Despite the motivation above, the main problem that emerges in domain adaptation is domain discrepancy; the data-generating distributions for the source and target domain are different. Such discrepancy prevents the developer from using the source model directly on the target data. Domain adaptation techniques are defined to alleviate the discrepancy between source and target domain; one recent approach to domain adaptation uses deep learning architectures.

Deep learning has recently been instrumental in transferring source and target data into a common space with the hope that the discrepancy between the two domains is minimized. Accordingly, through the use of multiple nonlinear layers, common high-level concepts between two domains are extracted.

Although deep learning is able to extract high-level concepts, however, only the representation of these concepts are naively used as an input to the classifier learned on the source model. In other worlds, the nature of a concept, which is carrying specific semantic meaning is not really exploited in domain adaptation solutions.

Furthermore, in order to truly benefit from the semantics of concepts for domain adaptation, one would require the network to extract correspondent concepts between two domains in the same hidden units. Therefore, the range of the problems that can be tackled by these approaches would rely on representational similarities between the correspondent concepts in two domains.

In this dissertation, I propose *conceptual domain adaptation* (CDA) as a novel solution for domain adaptation techniques that fail to exploit conceptual information in data. My proposed solution is specific to problems that suffer from inherently

lower-level representational discrepancy.

The specific approach followed in this dissertation is to separate concept extraction from concept alignment as different consecutive steps in conceptual domain adaptation; such approach prevents the alignment step to affect the quality of the concepts extracted during learning.

This document proposes conceptual domain adaptation and divides into the following chapters. The dissertation starts with background information in Chapter 2 and related work in Chapter 3. Later on, in Chapter 4 the main problem of applying deep learning on domain adaptation is introduced; this refers to data with lower-level representational discrepancy. The proposed solution is described in Chapter 5; experiments and analysis of the results are provided in Chapter 6. Contributions and future work are listed in Chapter 7. Finally, Chapter 8 gives a summary and conclusions.

Chapter 2

Background

2.1 Domain adaptation

Although in many machine learning applications it is assumed that the source and target data should be drawn from the same distributions, in real-world applications this assumption is often not realistic; hence, there is a need to transfer knowledge from one domain to the other. Accordingly, *transfer learning* [29] as a general subfield of machine learning, has been introduced as a group of methods that enable the transfer of knowledge whenever we observe differences between domains, such as in the feature spaces and the sample distributions.

Transfer learning is divided into three main categories: *inductive*, *transductive*, and *unsupervised* transfer learning. For more information about the three categories, please see [29]. Domain adaptation is consider as transductive transfer learning with

the following properties:

- Only the source data is labeled and the target is unlabeled,
- The domains are different between source and target,
- The AI task is the same for both source and target domains.

More specifically in [1], the difference between domains is defined as the difference between the marginal probability distributions of the input data ($P(X_s) \neq P(X_t)$) where unlike the more general transductive transfer learning categorization, the feature spaces are assumed to be the same ($X_s = X_t$).

Several approaches have been proposed to perform domain adaptation in the literature. Accordingly these approaches can be categorized into three main categories:

- **Instance-based methods** - The main idea in instance-based methods is to weight each source instance with $\frac{P_T(X_i^t, Y_i^t)}{P_S(X_i^s, Y_i^s)}$ while computing the loss for learning the source model parameters. Weighting the samples would potentially reduce the discrepancy between the source and target domains. More information about the instance-based methods is provided in [30].
- **Iterative methods** - In the iterative methods, an initial model is trained on the source model and further iteratively adjusted by providing more information from the target domain. An example of iterative approaches is DASVM proposed in [7] where the support vectors of the model are adjusted based on the closest target samples to the margins in the source SVM.

- **Feature-based methods** - The main idea of feature-based methods in domain adaptation is to transfer the source and target data into a new common representation such that the discrepancy between the two domains is minimized. Structured correspondence learning (SCL) [6] as an example, tries to find the new representation by defining common pivot features between two domains an trying to predict the pivots using other features.

In this document, we narrow the scope of domain adaptations into feature-based methods that specifically apply deep learning techniques to transfer the data into the new representation. Chapter 3 covers related work in this area.

2.2 Deep learning

Deep learning corresponds to learning multiple layers of non-linear transformation of the data. Each layer provides a set of abstract features composed of the features from the previous layer.

Although according to the universal approximation theory [10], a feed-forward neural network with only one hidden layer and sufficient hidden units is able to approximate any continuous function, the main benefits of deep architectures compared to shallow ones can be listed as follows [5]:

1. Feature re-use: Deep architectures promote re-use of the features detected in the previous layers which has two benefits:

(a) Computational: Re-use of nodes in previous layers reduces the number of computational nodes.

(b) Statistical: Re-use results in reduction in number of parameters which need less number of examples to learn the parameters

2. Abstraction: More abstract concepts are captured in deeper layers of networks. According to [20] features in deeper layers of networks are more invariant to other data variations.

According to [5] single layer representation learning algorithms can be categorized in two categories as follows:

1. Probabilistic models: including different variations of restricted Boltzmann machines (RBMs) [21].

2. Neural network models: including different variations of Auto-encoders (AEs) [3], convolutional neural networks (CNNs) [24], and recurrent neural networks (RNNs) [31].

Furthermore, deep learning algorithms can be categorized into two general approaches:

1. With pre-training: Due to the problem of vanishing gradient [22], in [21] the authors proposed a new method for learning deep networks. In this approach, each layer is trained separately in an unsupervised manner and then the whole network would be trained with supervised fine-tuning.

2. Without pre-training: Some architectures do not suffer from the problem of vanishing gradient. Therefore, the whole network can be easily trained using back-propagation of error. Convolutional neural networks [24], recurrent neural networks [31] and some networks built with rectified linear units incorporating dropout are trained without pre-training.

2.2.1 Autoencoders

An autoencoder as defined in [3] is a three layer neural network consisting of input layer, middle layer and output layer. The network performs the encoding and decoding of the data so as to capture hidden data representations (middle layer).
The encoding of the input of the autoencoder is defined as follows:

$$H = \sigma(W_1 X + b_1) \tag{2.1}$$

where H is the hidden layer, X is the input, W_1 is a set of weights and b_1 is the biases. The sigmoid function $\sigma(.)$ is defined as follows:

$$\sigma(x) = \frac{1}{1 + exp(-x)} \tag{2.2}$$

The decoder part of an autoencoder similarly maps the hidden layer back to the output layer as follows:

$$Z = \sigma(W_2 H + b_2) \tag{2.3}$$

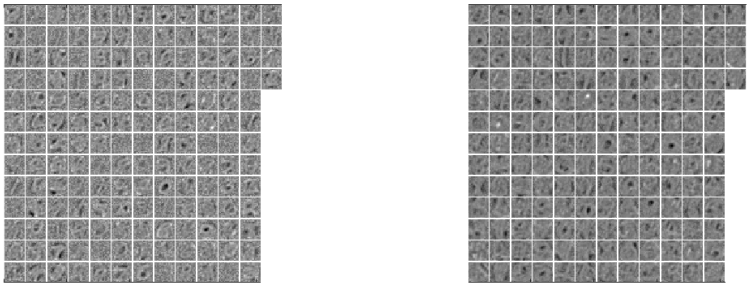
The goal of an autoencoder is to learn the hidden layer corresponding to the underlying factors of the data by reconstructing the input on the output. As a result the cost function defined for autoencoders is defined as the average reconstruction error over the data samples.

$$Loss(X, Z) = ||X - Z|| \tag{2.4}$$

By forcing the length of the hidden unit to be less than the input size or by forcing some sort of sparsity, we would be able to learn meaningful filters that capture existing variations in the data. Figure 2.1 illustrates the filters (templates) learned over MNIST and USPS datasets.

By designing an autoencoder with number of hidden units equal or more than the number of inputs, the network will potentially learn identity function and do not learn meaningful filters. In order to help this issue, denoising autoencoders are designed.

In denoising autoencoder [33], the network encodes a noisy version of the input and tries to reconstruct the original version. Accordingly, the network would be forced to capture the statistical dependencies between features in the input, in order to undo the noise. Figure 2.2 illustrates the architecture of a denoising autoencoder.



(a) MNIST filters (b) USPS filters

Figure 2.1: Filters trained using denoising autoencoder

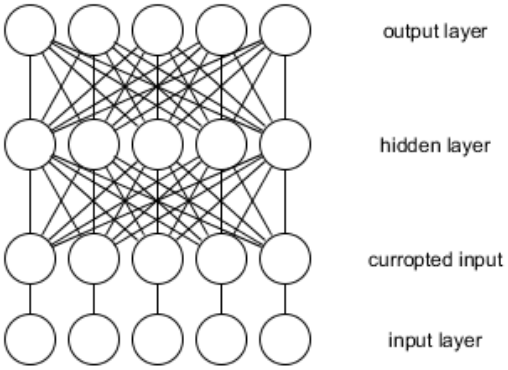


Figure 2.2: Denoising autoencoder

2.2.1.1 Stacked denoising autoencoders

Although denoising autoencoders help extract meaningful dependencies in the data, however, in order to capture more complex variations, multiple layers of nonlinearity are required. Accordingly, stacked denoising autoencoderes [34] have been introduced.

The main idea in stacked denoising autoencoders is to build an architecture of

multiple layers of autoencoders, each receiving the input from the hidden layer of the autoencoder in the layer below. The training of stacked denoising autoencoder is simply done by layer-wise training of the autoencoders from the lowest level to the highest level. The weights of the encoder in each level can be later used as the weights of a deep neural network which is fine-tuned during training. Figure 2.3 shows the architecture of a stacked denoising autoencoder.

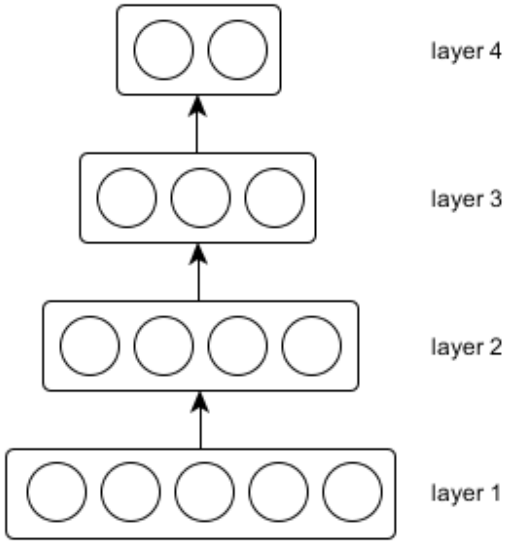


Figure 2.3: Stacked denoising autoencoder

2.3 Genetic algorithm

Genetic algorithm is an instance of evolutionary algorithms [12] inspired by biological evolution. This algorithm provides optimization and search solutions by performing biological operations e.g., mutation and crossover over a pool of solution instances.

In order to perform genetic algorithm, one has to initially encode (represent) the solutions of the problem in forms of genetic coding (chromosome). Each chromosome consists of a set of genes. The chromosomes in computers are represented using specific data structures such as binary arrays, integer arrays, character arrays, or trees depending on the problem.

After choosing the encoding scheme for chromosomes, one would require to define a fitness function that evaluates the goodness of the individual solution represented in forms of a chromosome. The goodness of a solution provides the ability to perform the optimization or search for the best solution.

A genetic algorithm performs two main operations on the pool of solutions listed as follows:

- **Crossover** - The idea of crossover operation is to swap part of the chromosomes of two parent individuals such that two new child solutions are generated. In each crossover, a subset of genes from one parent is replaced with the corresponding genes of the other parent and vice versa.
- **Mutation** - Similar to crossover, mutation also replaces part of the parent genes but this time with random genes. Therefore only one parent participates in each mutation operation.

Genetic algorithm is run in multiple iterations, each iteration with a new pool of solution individuals. Accordingly, the next generation for each iteration is generated either from selecting elite individuals with highest fitness directly from the previous generation or by performing crossover and/or mutation operations.

By performing genetic algorithm using sufficient iterations and performing the genetic operations with appropriate proportions, one often attains an optimal solution by sorting the final pool of individuals based on their fitness values.

Chapter 3

Related Works

As introduced in Section 2.2, deep networks provide the ability to transform raw data into high-level representations such that the underlying factors of variation are disentangled. As a result, in highest layer of the deep network, abstract concepts are extracted from the data that convey implicit semantic meanings.

Consider feature-based domain adaptation as introduced in Section 2.1; the method transfers source and target data into a common space. It would be promising to equip deep networks to perform this transformation. In this section, we provide a brief summary of current work that applies deep networks to domain adaptation.

In [19] the authors propose learning intermediate representation to remove the gap between the source and target domains. The higher level representation is learned using information from all available domains. Then the classifier is trained in the new space using the data from source domain. They used stacked denoising autoencoders

as deep architecture. They realize that in the new space, unlike [2], the domain discrepancy is higher than in the raw space. However they reported a good transfer of the model. They claim that disentangling domain-specific information would be the reason for high transfer quality despite domain discrepancy.

Similar to [19], in [8] the authors train a common space using the data from all domains and then train the classifier on the source domain. However, they use marginalized stacked denoising autoencoders as an alternative architecture with lower computational costs and scalability to higher-dimensional features.

In [9], the authors propose using the information about the distribution shift from source to target in domain adaptation. The path from source to target is interpolated by generating intermediate sample datasets where the proportion of source samples gradually decrease. They train one deep network on each sample dataset and then concatenate the representations for each input sample for classification. They claim that their approach bridges the gap between the source and target distributions and decreasing the A-distance between the two domains. The main difference between the proposed approach and other methods is considering the distribution of the interpolated datasets.

In [27], the authors propose using a sparse and hierarchical network (DASH-N) for domain adaptation. Their main contribution is jointly learning the hierarchy of features and addressing the domain mismatch between source and target. In their approach, adaptation is performed in multiple levels of hierarchy. At each level, they obtain sparse codes by using common dictionaries between domains which are used for discovering common structures between domains.

In [11], the authors initially train a model on the target domain and then use it as a regularizer to train the source model. The regularizer reduces the distance between the weights of the source data and the target model, therefore, the weights of the target model are used as prior knowledge for training the source model.

In [14], the authors propose an architecture based on backpropagation that jointly learns a common representation space, learns a classifier and minimizes the domain discrepancy between source and target domain. They measure the disparity between distributions based on their separability by a deep discriminatively trained classifier. The learned representation is discriminative for the main learning task on source domain while also remains invariant with respect to the domain shift. The architecture is composed of feature extractor followed by label predictor and also a gradient reversal layer that connects the domain classifier. The role of the gradient reversal layer leaves the input unchanged during forward propagation and reverses the gradient by multiplying it by a negative scalar during the backpropagation. It ensures that the feature distributions over the two domains are made similar.

In [16], the authors propose using human knowledge as an auxiliary information to leverage the domain mismatch between source and target. The architecture they proposed (DHN) is composed of a sparse autoencoder followed by stacked restricted Boltzmann machine. They perform their experiments on MNIST as source domain and noisy (local or global) MNIST or USPS as target domain. A variant of their architecture imposes sparse locally connected weights in the bottom layer in addition to the sparsity regularizer. They train the network entirely on source domain and test it on target domain. They have achieved high performance both in in-domain

and cross-domain experiments. The key contribution of their approach is the human knowledge about the noise in the target data, which is not usually available in real world domain adaptation problems.

In [17] the authors propose an architecture (DaNN) that minimizes the distribution difference between source and the target using a new regularizer. Maximum Mean Discrepancy (MMD) is used as a measure of distance between distributions and the network is trained to minimize this measure along with the classification error on the source domain. Although they achieved good results on hand-crafted features and raw features, they realized that deeper architectures do not necessarily result in improving the performance against distribution mismatch.

In [15] the authors propose an architecture for domain generalization. The architecture MTAE is composed of an encoder followed by multiple decoders. Using this architecture they learn a shared representation, across all domains by reconstructing the input from one domain into the analogs in all other domains.

A different direction is to alleviate the distribution discrepancy between source and target by shifting domains. [23] proposed an architecture composed of one encoder followed by two decoders one for each domain; the encoder captures both source and target data into a common space, while each decoder is responsible to decode the data on the corresponding domain.

In [25] the authors base their proposed approach on the fact that feature transferability drops in the higher layers of deep networks while domain discrepancy increases. In their approach, they explicitly increase the transferability in higher layers

by decreasing domain discrepancy using mean embedding matching of multi-layer representations across domains in a reproducing kernel Hilbert space.

In [18] the authors propose a multi-task learning architecture for domain adaptation composed of an encoder followed by a source class predictor and a target reconstruction. The target reconstruction network works as a regularizer to prevent the source classifier from over-fitting the source data. They realized that using only target data for reconstruction is sufficient for training. They also studied the domain correspondence and realized that the reconstructed image of the source data more looks like the target data. They also theoretically showed that their training is approximately equivalent to solving a semi-supervised learning problem on the target domain.

Chapter 4

Problem Specification

By considering related work in Chapter 3, we can conclude that deep networks can be beneficial to domain adaptation by providing two main functionalities; (i) extraction of common concepts, which transforms the data into a new common space and (ii) minimization of discrepancy, which forces the distribution of source and target data to be similar in the new common space.

In this section, we introduce a type of discrepancy between the domains and explain that existing deep learning solutions providing the main functionalities listed above are not able to solve domain adaptation problems with inherent introduced discrepancy. Accordingly, we argue that common concept extraction would fail in presence of *inherently low-level representational discrepancy*. As a result, we provide our intuitions respectively in Section 4.1 and later investigate it experimentally in Section 6.2.

The failure of common concept extraction due to low-level representational discrepancy had led us to introduce a new domain adaptation approach using deep learning: *conceptual domain adaptation*, which will be introduced in Chapter 5.

4.1 Low-level representational discrepancy

A high-level concept in domain D is captured by a pattern of output values at the top level of a deep network; it is an abstract entity that is assumed to convey a clear semantic meaning in D. An ideal network in future should be able to capture high level concepts such as *man*, *sitting*, *slowness*, *adjacency*, and *table* from data (in text, video or any other modality) conveying information about a man sitting slowly next to a table. Concepts can be represented in various low-level representations in the data. In a more simple example, in the domain of hand-written digits, *seven* is a high-level concept that carries the same meaning regardless of the writing form or style.

Concept c_i^s in domain D^s is *correspondent* to concept c_j^t in domain D^t, $c_i^s \leftrightarrow c_j^t$, if and only if they carry the same semantic meaning.

Concepts can be represented in mainly two ways: local representations and distributed representations. In local representation, activation of one hidden unit is necessary to represent the concept while in distributed representation, the concept is represented by an activation pattern over more than one hidden unit [35]. In higher layers of deep networks, the representations tend to be more local, rather than distributed. In this document, we consider (strictly) local representations of high-level

concepts defined as follows:

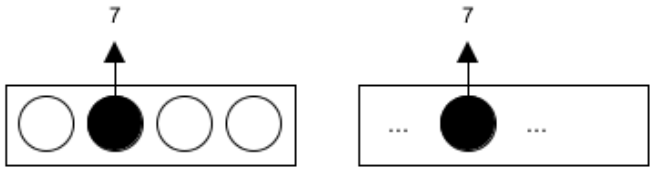
Definition 1 ((strictly) local representation)**.** *A representation is local [32] if for each concept only one (output) unit is active (Figure 4.1(a)).*

$$\forall r \in c \exists h_j : r(h_j) = 1, (\forall h_k, j \neq k : r(h_k) = 0) \tag{4.1}$$

where $r \in c$ is the representation of a pattern from concept c and $r(h_j)$ is the jth hidden unit of representation r.

In case of strictly local representation, activation of other units does not affect the representation of the concept [35](Figure 4.1(b)).

$$\forall r \in c \exists h_j : r(h_j) = 1 \tag{4.2}$$



(a) local (b) strictly local

Figure 4.1: Local (a) and strictly local (b) representations of concept *7*

Definition 2 (Aligned local representations)**.** *Assuming (strictly) local representations, representation r^s of concept c^s in domain D^s is aligned to representation r^t of*

its correspondent concept c^t in domain D^t, if and only if the activating unit of r^s is also activated in r^t.

$$\forall c_i^s \in D^s, c_j^t \in D^t : c_i^s \leftrightarrow c_j^t \iff (\forall r^s \in c_i^s, r^t \in c_j^t : r^s = r^t) \tag{4.3}$$

where

$$\forall r^s \in c^s, r^t \in c^t : r^s = r^t \iff (r^s(h) = 1 \iff r^t(h) = 1) \tag{4.4}$$

In previous approaches to domain adaptation using solely single deep network such as [19], the source and target data are fed to the same network. The alignment of high-level representations is performed during the learning process assuming that the hidden unit capturing concept $c^s \in D^s$ is also able to capture its corresponding concept $c^t \in D^t$ at the same time.

A feature in an autoencoder is captured by updating its corresponding weights when patterns having similar variations are sufficiently frequently observed in the data during the learning process. Accordingly, the formation of high-level concepts are dependent on the similarity of the low-level representation of their corresponding patterns.

Based on the dependency of concept extraction on low-level representations, in order to align correspondent concepts in same hidden units using solely joint training, we would require the correspondent concepts to exhibit lower-level representational similarities.

As investigated in this document, in real-world scenarios, the source and target

data might exhibit *inherently low-level representational discrepancy* where the representation of correspondent concepts in source and target domain in lower levels are not similar. In such cases, the joint training approach in domain adaptation would potentially capture correspondent concepts in two different hidden units in their high-level local representations and results in misalignment between the representation of the two concepts as follows:

$$\forall r_i^s \in c^s, r_j^t \in c^t, c^s = c^t \exists h : r_i^s(h) = 1 \iff r_j^t(h) = 1 \tag{4.5}$$

The misalignment of local representation of correspondent concepts between two domains trained using joint training directly affects domain adaptation performance. Consider concept c^s in source domain represented by activating hidden unit h in its local representation r^s so $r^s(h) = 1$. The corresponding concept c^t in target domain is also trained using the same network while they are not locally aligned. The misalignment will result in deactivation of hidden unit h in r^t (the local representation of c^t) so $r^t(h) = 0$.

Assume a classifier trained on source data that is supposed to discriminate c^s from non-c^s. The learning algorithm for the classifier would find a hyperplane that separates the space into two regions based on the activation of h and concept c^s would fall into the region where h is activated. However, when the target data is fed into the source classifier, based on the deactivation of h for c^t, the source classifier would classify c^t as non-c^s while they were correspondent concepts carrying same semantic meanings.

Figure 4.2 illustrates the problem of alignment for local representations of high-level concepts for hand-written digits, using joint training approach in presence of inherently low-level representational discrepancy.

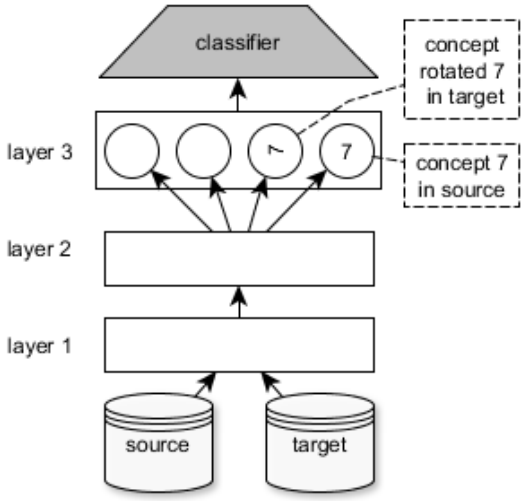


Figure 4.2: The problem of alignment of source and target concepts using joint training approach

Chapter 5

Methods

5.1 Conceptual domain adaptation

In order to define conceptual domain adaptation, we consider each domain consisting of a set of high-level abstract underlying concepts $c_i^\bullet \in D^\bullet$. Each concept itself is represented in multiple low-level representations of instances in the dataset $r^\bullet \in c_i^\bullet$. In this document, for each instance of data, we assume two types of properties: representational and conceptual.

Representational property of a data item tells us about the shape, style and form of the data item. For instance, digit 7 can be written in several forms and shapes which are considered as *representational properties*. Similarly, for example, datasets containing Huskies have representational similarities between them. On the other hand, in all forms and styles, 7 has a single semantic meaning which gives us the

ability to distinguish it from other digits. Similarly, the concept of *sitting* or *running* is common between all domains of different dogs. A property that is not related to how the raw data is represented, is called a *conceptual property* of the data.

Deep learning provides the ability to transform low-level representations of data into their high-level concepts. The transformation is performed by disentangling factors of variations and furthermore, performing a sort of abstraction in each layer of the network. Ideally, we would expect the network to extract underlying abstract concepts of the data in their local representations in the top layer of the network.

In this document, we propose conceptual domain adaptation as follows:

Definition 3 (Conceptual domain adaptation)**.** *Conceptual domain adaptation is a new approach in domain adaptation in which:*

- ***Correspondences*** *between high-level concepts in two domains are considered as direct evidence of the* ***relatedness*** *between the domains,*
- ***Conceptual information*** *is used alone to perform the* ***alignment*** *of source and target.*

Accordingly we perform conceptual domain adaptation in three main steps:

1. *Extracting* high-level concepts from source and target domain using deep network,
2. *Aligning* the concepts in source and target by adjusting their representations,
3. Performing *AI task* e.g. classification on aligned target representations.

According to definition 3, considering hierarchical representations of the data in two domains, a domain adaptation solution is considered conceptual (as opposed to representational) if the alignment between high-level correspondent concepts in the two domains does not rely on their lower-level representations.

Relying on representational domain adaptation solutions, where the alignment is dependent on the lower-level representations of the concepts as mentioned in section 4.1, limits the range of solvable problems into those with low-level similarities between the correspondent concepts in two domains. Therefore, by relaxing this dependency in conceptual domain adaptation, we would potentially be able to solve a rich variety of domain adaptation problems.

5.2 Supervised approach

Considering the three main steps to conceptual domain adaptation as mentioned in Section 5.1 and the definition of conceptual domain adaptation in definition 3, defining the alignment step is a basic step to clearly propose the new approach.

As mentioned before, alignment of local representations is essential for conceptual domain adaptation (definition 2). However, joint training of source and target data is not able to align correspondent concepts without representational relationships and it would result in separation of the hidden units representing the correspondent concepts (equation 4.5). In this section we propose an alignment method that does not rely on low-level representational relationship between the concepts and provides the basis for a conceptual domain adaptation solution.

5.2.1 Mapping matrix

Assuming local representations trained by deep network, we propose the adjustment approach for alignment of high-level concepts. As illustrated in figure 5.1(a), the target representation is adjusted in a way that correspondent concepts from target and source fall into the same hidden units in the representation.

The data in the target is adjusted into the source representation by defining a function that maps the target units into each source hidden unit. Accordingly, the mapping function corresponding to each hidden unit in the new source-adjusted representation is defined as follows:

$$\begin{aligned} r^n(h_j) &= \sum_i v_{ij} r^t(h_i) \\ \text{s.t.} \;\; v_{ij} \in \{0,1\}, \; \forall j : \sum_i v_{ij} &\leq 1 \end{aligned} \tag{5.1}$$

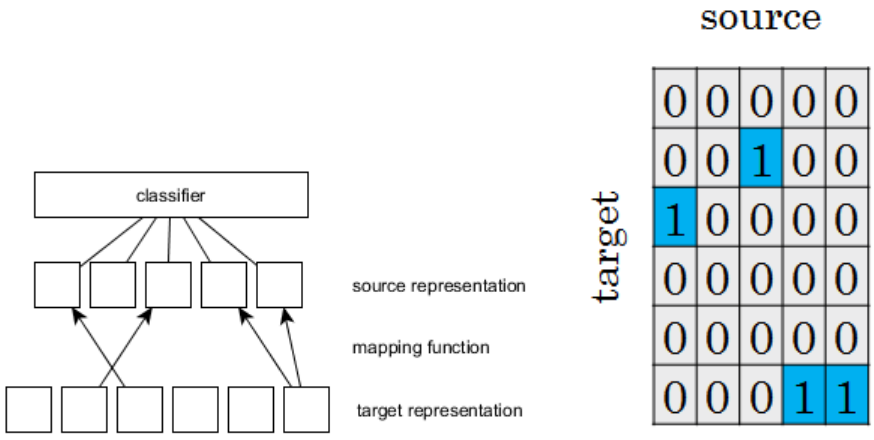
such that the new source-adjusted target representation is aligned with the source representation:

$$\forall c_i^s \in D^s, c_j^t \in D^t : c_i^s \leftrightarrow c_j^t \Rightarrow (\forall r^s \in c_i^s, r^n \in c_j^t : r^s = r^n) \tag{5.2}$$

and $r^s = r^t$ have been defined as in equation 4.4.

Using v_{ij}'s, each hidden unit in source-adjusted representation will reflect either one or none of the original hidden units in the initial target representation.

We implement the mapping functions by defining a mapping matrix (figure 5.1(b)) with h_t rows, and h_s columns, equivalent to number of units in target and source



(a) adjustment of target (b) Mapping matrix

Figure 5.1: aligning target to source in adjustment approach

representations, respectively. Activation of each element in the mapping matrix corresponds to a mapping from the specified target unit to the source unit. Each source-adjusted unit can only be activated by either one or none of the units in target.

Using the mapping matrix M^*, the aligned target representation T_{new} is calculated as follows:

$$T_{new} = T \times M^* \mid M^*_{i,j} \in \{0,1\}, \forall j : \sum_i M^*_{i,j} \leq 1 \tag{5.3}$$

The proposed adjustment approach directly maps one or none of the initial target units into a hidden unit in the source-adjusted representation. The intuition behind this type of adjustment is based on the assumption that the deep learning process in the previous step had already been able to extract meaningful concepts from the data

in both source and target domains. As a result, a simple adjustment, as proposed in this section, would be sufficient to align the correspondent concepts with same semantic meanings.

The main question in conceptual domain adaptation using adjustment approach is finding the mapping matrix M^*, without relying on the lower-level representations of the concepts. Accordingly, search framework is proposed to find the mapping matrix between source and target representations with maximum measured score as follows:

$$M^* = \arg\max_{M} Score(S, T; M) \tag{5.4}$$

Accordingly, the score of a mapping matrix indicates the goodness of the matrix in depicting the correspondences between the concepts in two domains.

5.2.2 Search framework

In order to find the best mapping matrix, one requires to provide measurements to evaluate the score of the mapping without relying on low-level representational properties of the concepts.

The total number of possible solutions for a $h_t \times h_s$ mapping matrix is $h_t^{h_s}$. Due to the large space complexity of the problem, we chose genetic algorithms [12] to perform the search on the space of mapping matrix solutions. Accordingly, each mapping matrix M is encoded as an offspring as follows:

$$E(M) = V \Rightarrow (\forall i : V(i) = j \iff M(j, i) = 1) \tag{5.5}$$

Figure 5.2 illustrates a sample encoding of a mapping matrix.

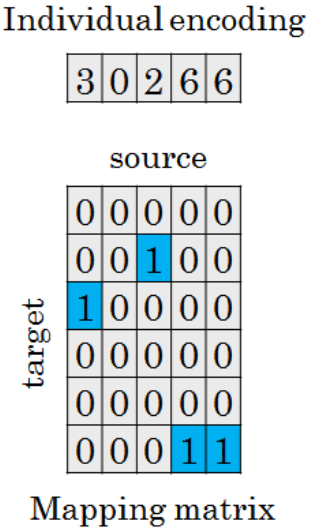


Figure 5.2: The encoding of a mapping matrix into its corresponding offspring

We performed three main operations of elite selection, crossover and mutation to generate the next population in each iteration.

Using the search framework for adjustment of high-level representations, one might be able to define different fitness functions for conceptual domain adaptation. We propose basic supervised and context-based fitness evaluation as introduced in Sections 5.2.3 and 5.3.1.

5.2.3 Basic supervised alignment

In order to evaluate the fitness of the offspring, we can train a classifier on high-level representation of source data and perform classification on the mapped high-level representations of target data. Then we use the domain adaptation accuracy as the fitness value. The algorithm to compute the score of a mapping M is defined as in algorithm 1.

Data: source data S, target data T, mapping matrix M

Result: score of mapping matrix M

$T_{new} = T \times M$;

$\theta^* = train(S^X, S^Y)$;

$score = classify(T^X_{new}, T^Y_{new}, \theta^*)$;

retrun $score$;

Algorithm 1: Computing the score of mapping matrix M using graph-based evaluation

5.3 Unsupervised approach

Although the basic supervised approach to conceptual domain adaptation has been proposed as a solution to correspondent concept misalignment in presence of lower-level representational discrepancy, reliance of the evaluation function on labeled target data can be impractical for real-world applications. Therefore, in this section, we introduce unsupervised approaches to conceptual domain adaptation including context-based, graph-based and PCA-based approaches.

5.3.1 Unsupervised context-based alignment

As mentioned in Section 5.1, the basic requirement of conceptual domain adaptation that makes it different from other domain adaptation methods using deep learning is the independence of the adaptation on the lower-level representational properties of the data. In Section 5.2.3, we proposed using a basic supervised score in evaluating the fitness of the mapping matrices. In this section, we introduce a new approach that (i) does not rely on lower-level representational properties of the data, solely depends on conceptual properties, and (ii) it does not require the target data to be labeled.

According to Section 5.1(b), the role of the mapping matrix is to indicate the correspondent concepts between source and target domains. Furthermore, correspondent concepts are defined as abstract entities that carry same semantic meaning. In order to provide a solution that characterizes the correspondence between concepts, we need to provide a clear definition of semantic meaning.

As mentioned in Section 5.1, concepts are represented in the highest (top) layer of a deep network. While we consider local representation of the concepts in this document, we introduce superimposition of concepts as follows:

Definition 4 (superimposition)**.** *According to [26], superimposition happens when two or more concepts are represented using the same distributed pattern. The representation r^s is a superimposition of concepts defined as follows:*

$$r^s = \{c_j^s : c_j^s \in D^s\} \tag{5.6}$$

We consider cases where multiple concepts are represented through a single high-level representation, each represented locally by a single unit.

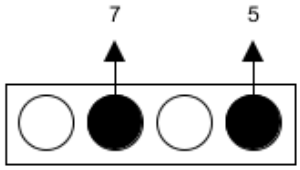


Figure 5.3: Superimposition of concepts 7 and 5 in their local representations

In this document, we assume that the meaning of a concept c^s is dependent on the context it appears in and the context is defined based on the other concepts that are in superimposition with the c^s. The context of c^s is defined as follows:

$$X(c^s) = \{c_i^s \in D^s \exists r^s : c^s \in r^s, c_i^s \in r^s\} \tag{5.7}$$

Based on the definition above, as the context of a concept reflects its semantic meaning, we redefine correspondence between concepts as similarity of their contexts as follows:

$$c^s \leftrightarrow c^t \iff X(c^s) = X(c^t) \tag{5.8}$$

and the similarity between two contexts is defined as follows:

$$X(c^s) = X(c^t) \iff \forall c_i^s \in X(c^s) \exists c_j^t \in X(c^t) : c_i^s \leftrightarrow c_j^t \tag{5.9}$$

In order to provide a good measurement for the fitness of the mapping matrix, we need to evaluate the distance between two contexts as the number of concepts that do not co-occur with both correspondent concepts and it is defined as follows:

$$\begin{aligned} \forall c^s \in D^s, c^t \in D^t, U = \{c : c \in D^s \cup D^t\}, V = \{c \in U : c\Delta c^s, c\Delta c^t\} : \\ ||X(c^s) - X(c^t)|| = n(U) - n(V) \end{aligned} \tag{5.10}$$

where $c\Delta c^s$ is the co-occurrence of concepts c and c^s defined as follows:

$$c_i^s \Delta c_j^s \iff \exists z_k^s : c_i^s \in z_k^s, c_j^s \in z_k^s \tag{5.11}$$

where z_k^s is a superimposed pattern over the concepts in domain D^s.

By defining the measure of distance between the two contexts, we can evaluate the fitness of mapping matrix M as follows:

$$\forall i, j, c_i^t \in D^t, c_j^s \in D^s : score(M) = \sqrt{\sum_{i,j} |X(c_i^t) - X(c_j^s)|} \tag{5.12}$$

5.3.2 Unsupervised graph-based alignment

In another approach to unsupervised conceptual domain adaptation, we propose performing graph matching to align the source and target representations.

By considering each hidden unit as a vertex of a graph and using the co-occurrences between them (can be implemented as mutual information, both activation, etc) as edges, we build the adjacency matrix corresponding to the high level representation of the data. Using the two adjacency matrices for source and target, we would be able to match the two graphs by minimizing the following equation:

$$SC_{graph} = ||D^s - PA^tP^T||_F^2 \tag{5.13}$$

where D^s and D^t are the adjacency matrices of data in source and target domains.

For our experiments, we used GraphM tool [1] with PATH matching algorithm [36].

Using the score, we would be able to find the best mapping between source and target domains. Figure 5.4 illustrates the process of domain adaptation using the unsupervised graph-based method.

5.3.3 Unsupervised PCA-based Alignment

In previous section, we concluded that due to representational discrepancy between domains, after concept extraction we require alignment between concepts such that

[1] http://projects.cbio.mines-paristech.fr/graphm/

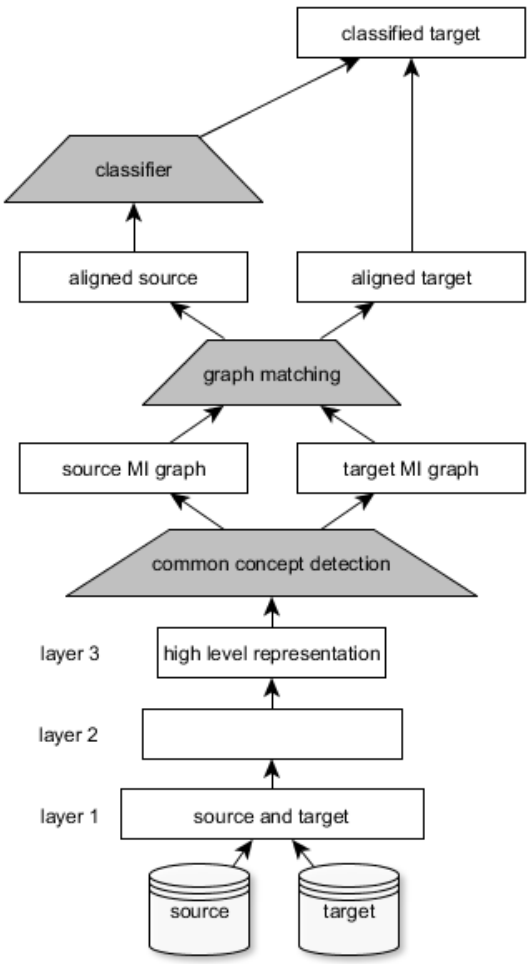


Figure 5.4: graph-based alignment of high level representation of data for conceptual domain adaptation

source concepts are aligned with their correspondent concepts in target domain. Using the aligned target concepts, we would be able to perform AI task on a model trained on source domain.

Accordingly, in this section we propose using principal component analysis (PCA) to align the two high-level representations of the data. The architecture proposed is

illustrated in Figure 5.5. The main idea is that principal components can be helpful in aligning the two domains. For instance, by projecting source and target data on their principle components, one could assume that the first eigenvector of the source which captures the highest variation, corresponds to the similar principal component of the target and so on. Similarly, we can assume that eigen values of two domains should be nearly equal as they are suppose to show correspondence between the domains.

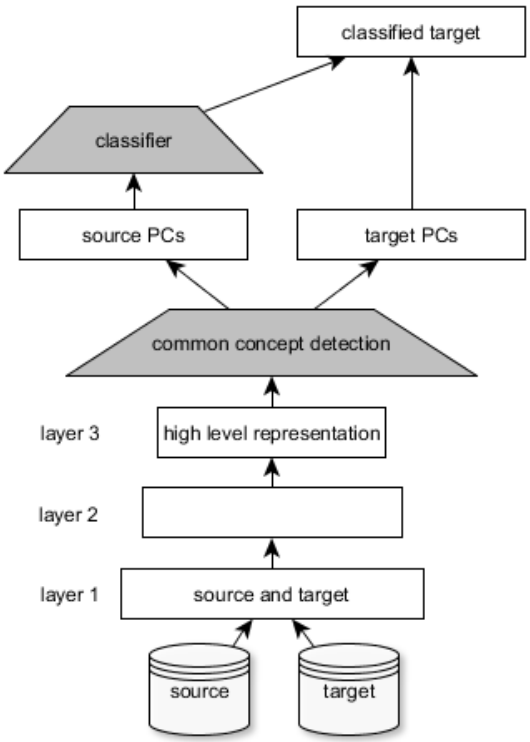


Figure 5.5: PCA alignment of high-level representation of data for conceptual domain adaptation

In order to evaluate the goodness of the alignment in PCA-based alignment approach, we propose three methods including i) minimizing discrepancy between domains, ii) minimizing the angle between correspondent eigen vectors between two domains and iii) minimizing the difference between eigen values between two domains.

In the first experiments mentioned above, we use discrepancy measure for evaluating the goodness of the alignment. As a result, we train a classifier that discriminates source from target data. The goodness score would be:

$$sc_{disc} = |2acc - 1| \tag{5.14}$$

where acc is the accuracy of the classifier.

In the second trial, we use the change of variation between the correspondent principle components as a measure of goodness of the alignment. Accordingly, the score is defined as follows:

$$sc_{var} = \sqrt{\sum_{i} (\lambda_i^c - \lambda_i^t)^2} \tag{5.15}$$

where $\lambda_i^\bullet$ is the eigen vector corresponding to the ith eigen value in $\bullet$ domain.

The intuition behind this measure is that by aligning the principal components, the source data, and target data, should have similar variances along the correspondent principle components.

In the third attempt, instead of variations, we use the angle between two principle components.

$$\cos\theta = \frac{v^s \cdot v^t}{||v^s||||v^t||} \tag{5.16}$$

Using this method, the less the angle between correspondent principle components, the better the alignment is applied and that would increase the domain adaptation performance.

Chapter 6

Experiments and Analysis

6.1 Experiment settings

With appropriate initialization of the network and also using ReLU nonlinearities, there would be no necessity for pre-training the network; directly full-training could achieve comparable performance on the data. However, in order to observe the results of our experiments on each layer separately and provide analysis of the effect of depth, we used stacked denoising autoencoder [34] architecture for training the model and a modified version of Matlab implementation of the network provided in [28][1]. The model is comprised of nine layers of denoising autoencoders, each using batch gradient descent for training. The stopping criteria for training were set to no improvement in the last 20 iterations or the number of iterations exceeding 500. For learning-rate scheduling, we used the method mentioned in [4] as follows:

[1] https://github.com/rasmusbergpalm/DeepLearnToolbox

$$\epsilon_t = \frac{\epsilon_0 \tau}{max(t, \tau)} \tag{6.1}$$

where τ is the minimum iterations that need to be passed in order to reduce the learning rate and it was set to 20 iterations.

As mentioned in [4], we performed layer-wise search for finding hyper-parameters. Accordingly, we performed a grid search on sets of hyper-parameter values and chose the best setting. The size of the hidden layer was chosen from $\{1L, 2L/3, 1L/2, 1L/5\}$ where L is the size of the previous layer. Similarly, the range of the learning rate and corruption level hyper-parameters where chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and $\{0.0, 0.3, 0.5\}$ correspondingly. The final setting was chosen manually by visualizing the reconstruction error and number of iterations over hyper-parameters. The chosen hyper-parameters were set to $2L/3$, 1, and 0.3, respectively.

For the genetic algorithm, at each iteration we kept 20% of the population as elite instances. Also, 80% of the rest of the population would be generated using crossover operation and 20% using mutation. The size of the population was set to 100 instances. The mutation operator would mutate one gene at each time. The algorithm would stop if the best score of the population would not improve for at least 200 iterations. For the classification in the fitness function, we used nearest neighbor classifier with $k = 1$ and L_1 distance. Using this approach, each instance would be labeled similar to the instance in the training set with maximum number of similar concepts activated in both.

We performed domain adaptation on digit recognition task and used MNIST [24][2], USPS [13][3] and rotated USPS. The datasets where processed to standard format of 16×16 grayscale images. Using MNIST versus rotated USPS, provides us a domain adaptation scenario with inherent lower-level representational discrepancy along with similar semantically correspondences between the domains.

6.2 CDA in presence of representational discrepancy

In the first set of experiments, we evaluate the performance of conceptual domain adaptation in the presence of inherently low-level representational discrepancy. Accordingly, we compared the domain adaptation accuracy on two datasets MNIST and rotated USPS tested using solely joint-training approach and conceptual domain adaptation. We hypothesize that the solely joint training approach would suffer from the representational discrepancy in lower levels as introduced in Section 4.1.

We investigated three main aspects of the conceptual domain adaptation including the role of adjustment, role of depth and role of joint training.

[2]http://yann.lecun.com/exdb/mnist/
[3]http://statweb.stanford.edu/ tibs/ElemStatLearn/data.html

6.2.1 The Role of adjustment

In our initial experiments, we performed the adjustment approach on two domain adaptation scenarios including MNIST to USPS and MNIST to rotated USPS. Based on accuracy performance, we chose a 5 layer architecture with each layer 2/3 of the size of the previous layer. Figure 6.1 shows the improvement of domain adaptation performance in each scenario using adjustment approach.

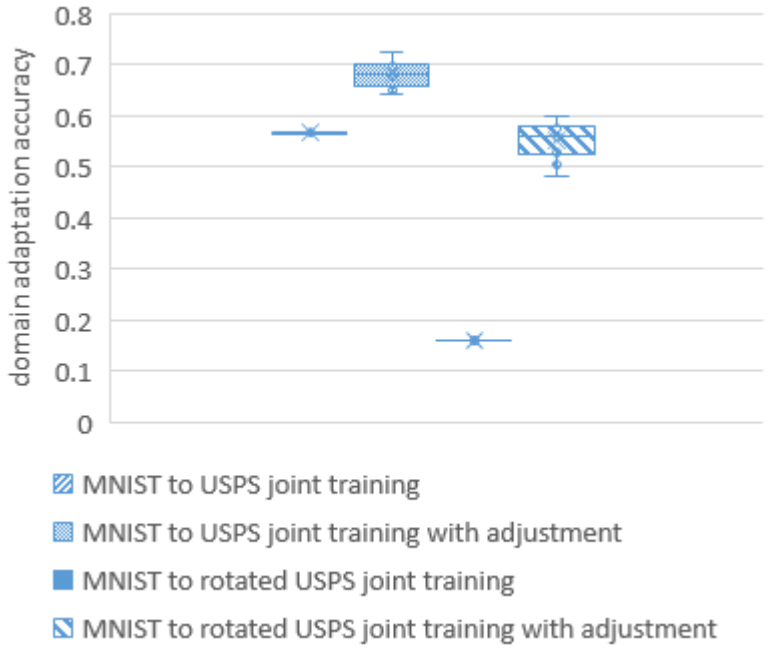
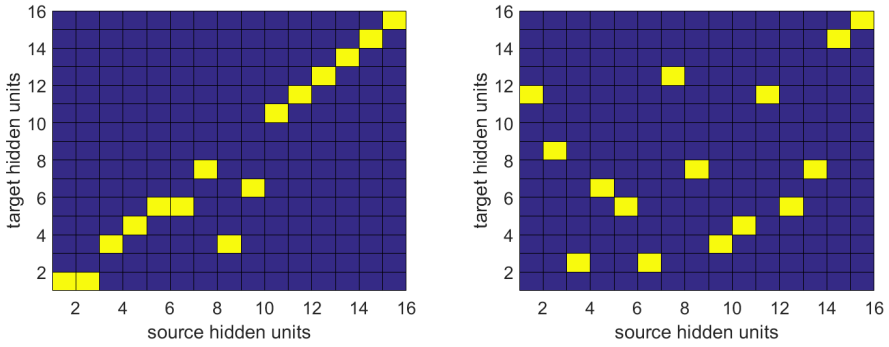


Figure 6.1: Improvement in domain adaptation accuracy using search based adjustment

In the MNIST to USPS scenario, correspondent concepts between the two domains have stronger representational similarities compared to the other scenario (MNIST to rotated USPS). Therefore, the proposed approach shows only a small



(a) MNIST to USPS (b) MNIST to rotated USPS

Figure 6.2: The mapping matrices for two domain adaptation scenarios

increase in performance. On the MNIST to rotated USPS scenario, the adjustment approach displays a significant performance improvement due to the high degree of low-level representational discrepancy. Overall, the proposed search-based framework shows improved performance, regardless of the presence –or lack of– low-level representational discrepancy.

Furthermore, as illustrated in Figure 6.2, we realize that unlike MNIST to rotated USPS, in MNIST to USPS scenario the mapping matrix will be nearly identity matrix. This observation confirms that MNIST to USPS contains representational similarities between correspondent concept so there would be less requirement for adjustment in the next step.

6.2.2 The Role of depth

In order to assess the effectiveness of the proposed mapping with respect to the number of hidden layers in the deep autoencoder, we capture the deviation of M^*

from the identity matrix I_n (where we assume M^* and I_n are square matrices). In the extreme case where $M^* = I_n$, then the mapping is direct: concepts are represented by the same hidden unit. Figure 6.3 shows the adjustment degree for each no. of layers in a deep autoencoder, on the two domain-adaptation scenarios described above. The adjustment degree is the percentage of the hidden units in source-adjusted representation that are not mapped from the same hidden unit in target representation through adjustment.

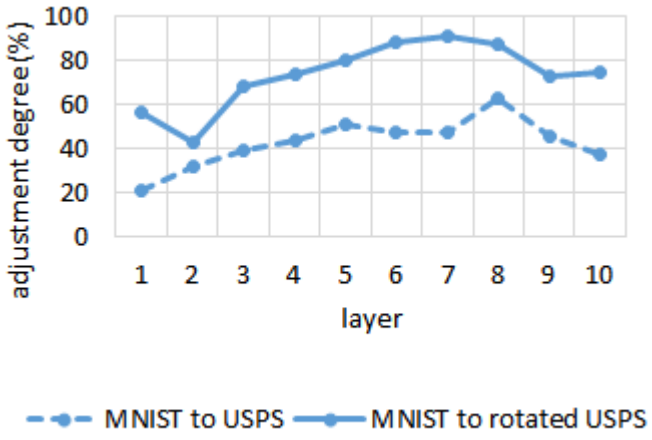


Figure 6.3: The adjustment degree for each layer of the network

Figure 6.3 shows that in lower layers we observe more identity mappings. In other words, low-level less abstract features contain more representational rather than conceptual relationships; therefore, in order to achieve better domain adaptation performance in lower layers, representational alignments applied by joint training are sufficient and no further conceptual alignments using adjustments are required.

Figure 6.3 shows that adapting MNIST to rotated USPS requires more adjustment compared to USPS. This observation agrees with the fact that MNIST and USPS contain more representational similarities that can be equipped in alignment by joint training. But, rotated USPS contains correspondent concepts with less representational similarities and more conceptual similarities with MNIST concepts.

If we consider the performance of layers 1 to 5 in Figure 6.4 we realize that using adjustment, adding layers would generally improve the performance of domain adaptation comparing to the raw data in layer 1. However, the performance by the solely joint training approach is not affected by the depth factor. The same behavior has been observed in MNIST to USPS scenario. This observation leads to the conclusion that conceptual domain adaptation compared to joint training truly benefits from the depth in deep networks by extracting more abstract concepts.

Although conceptual domain adaptation benefits from depth in layers 1 to 5, after layer 5, as seen in Figure 6.4, the performance decreases by adding layers. This observation suggests an increase in adjustment degree (Figure 6.3) in higher layers: Due to representational discrepancy in higher layers, there would be more adjustments required to align the correspondent units. Therefore, more hidden units are required to maintain the information contained in the correspondent hidden units. However, due to the 2/3 rule for designing the architecture, in each layer we are reducing the size of the network, so more information would be lost and the performance will degrade by adding layers.

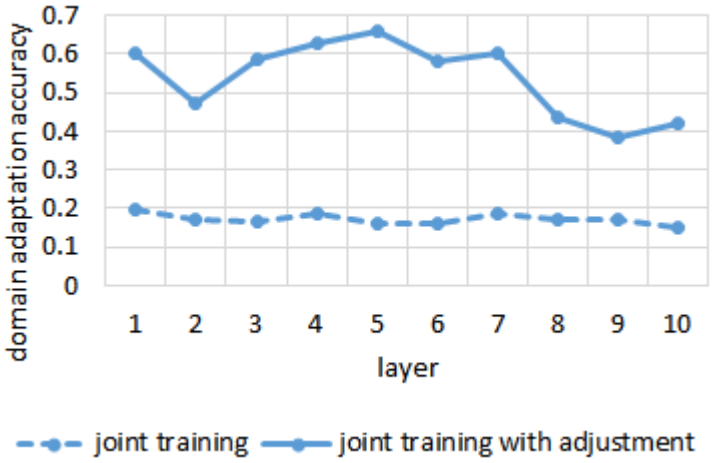
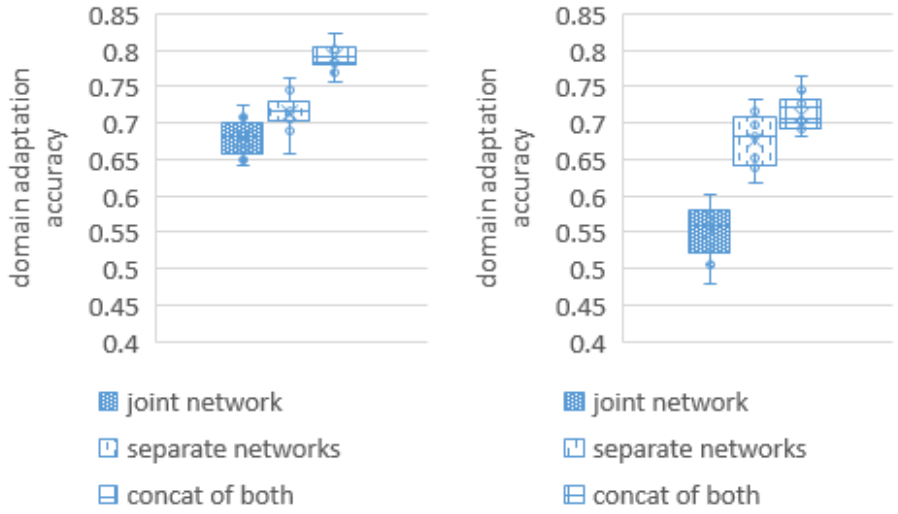


Figure 6.4: The effect of depth on domain adaptation performance for MNIST to rotated USPS scenario

6.2.3 The Role of jointly learning new concepts

The first step in conceptual domain adaptation is learning high-level concepts (Section 5.1). We compared two approaches in learning the concepts including (i) jointly learning the source and target concepts in the same network and (ii) separately learning the source and target concepts in different networks. Also, we considered a third case (iii) where the representation of each data point is constructed by concatenating the representation from the previous two approaches. For each case, the adjustment was performed by constructing the mapping matrix. The size of the mapping matrix is dependent on the output size of the network used for training the source and target data in each case as follows:

In joint adjustment, similar to the previous experiments, we searched for the best mapping matrix with h rows and columns where h is the number of hidden units in



(a) MNIST to USPS (b) MNIST to rotated USPS

Figure 6.5: The effect of joint training versus separate training

the high-level representation. In separate adjustment, the size of the matrix is $h_t \times h_s$, where h_t and h_s correspond to the size of high-level representations trained by target and source separate networks, respectively. Finally, in concatenated adjustment, the mapping matrix is a $(h + h_t) \times (h + h_s)$ matrix initiated as four sub-matrices as follows:

$$M = \begin{bmatrix} J & 0 \\ 0 & S \end{bmatrix} \tag{6.2}$$

where J is the a sub-matrix corresponding to jointly learned concepts and S corresponding to the separately learned concepts. The J and S matrix are initiated as diagonal matrix and random mapping, respectively. Figure 6.5 compares the domain adaptation performance for the three approaches.

Figure 6.5 shows separate networks in MNIST to rotation of USPS significantly

improves the domain adaptation accuracy compared to using a single joint network. The reason can be traced to the inability of the joint network in capturing correspondent concepts for each domain separately when there is low representational similarity between the two concepts. In other words, we can conclude that in single joint network architecture, when the concepts have low representational relationships, the information coming from one concept affects the ability of capturing other corresponding concepts from other domains, therefore performance decreases. Using separate networks for each domain, will force the networks to capture each correspondent concept separately without interfering in the learning process of each other.

Despite the fact that separate networks will assist in learning correspondent concepts, there would be still some concepts between two domains with representational similarities that can be captured with joint network. Therefore, in practical applications, it might be a reasonable approach to use the concatenation of both representations to achieve improved performance.

6.3 Unsupervised CDA

After applying the supervised version of our algorithm on a scenario having lower-level representational discrepancy, we observe increase in domain adaptation performance.

Despite the improvement in performance, however, in supervised domain adaptation, we require the target dataset be labeled. Therefore, the need for an unsupervised version of conceptual domain adaptation is essential.

In this section, considering the proposed approaches in 5.1, we will try to achieve similar domain adaptation performance compared to the supervised one.

6.3.1 Graph-based approach

In order to experiment with the graph-based approach in conceptual domain adaptation, we used an exhaustive search over the 9th layer of the stacked denoising autoencoder trained on MNIST to rotated USPS domain adaptation scenario. Performing the exhaustive search increases the computational cost, but it will help us diagnose the process and verify if the results are related to the search mechanism or the goodness evaluation approach.

We used GraphM package[4] for finding a matching matrix between two adjacency matrices of source and target. Using the matching matrix, we will be able to adjust the two domains by simply finding correspondent common concepts in target domain and assign it to the corresponding source concepts.

While each layer of the network consists of certain number of hidden units, a certain portion of them should be chosen in our experiment as *common concepts*. Common concepts can be considered as concepts that are not domain-specific. Setting the size of common concepts to two, we find all combinations of pairs of concepts from source and pairs of concepts from target followed by a matching between corresponding adjacency graphs.

[4]http://projects.cbio.mines-paristech.fr/graphm/

Using the matching score from all combinations, we did not observed any improvement in domain adaptation error. Furthermore, as shown in Figure 6.6, the graph matching score does not show any correlation to domain adaptation error. Therefore, graph matching score is not be a good evaluation measure for goodness of unsupervised conceptual domain adaptation.

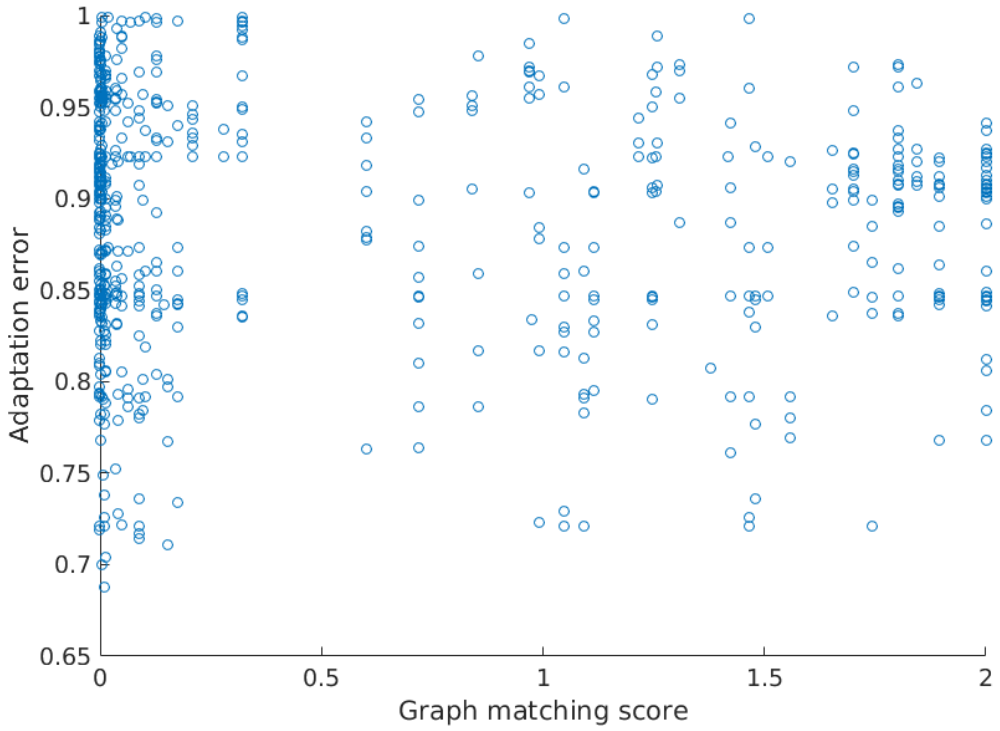


Figure 6.6: Relation between graph-matching score and domain adaptation error.

We assumed that minimizing the graph-matching score we would be able to minimize domain adaptation error, having plotted the error for minimum score (Figure 6.7). But we realized that the graph-matching score doesn't provide sufficient

information to find the best combination of common concepts that satisfy the minimum domain adaptation error.

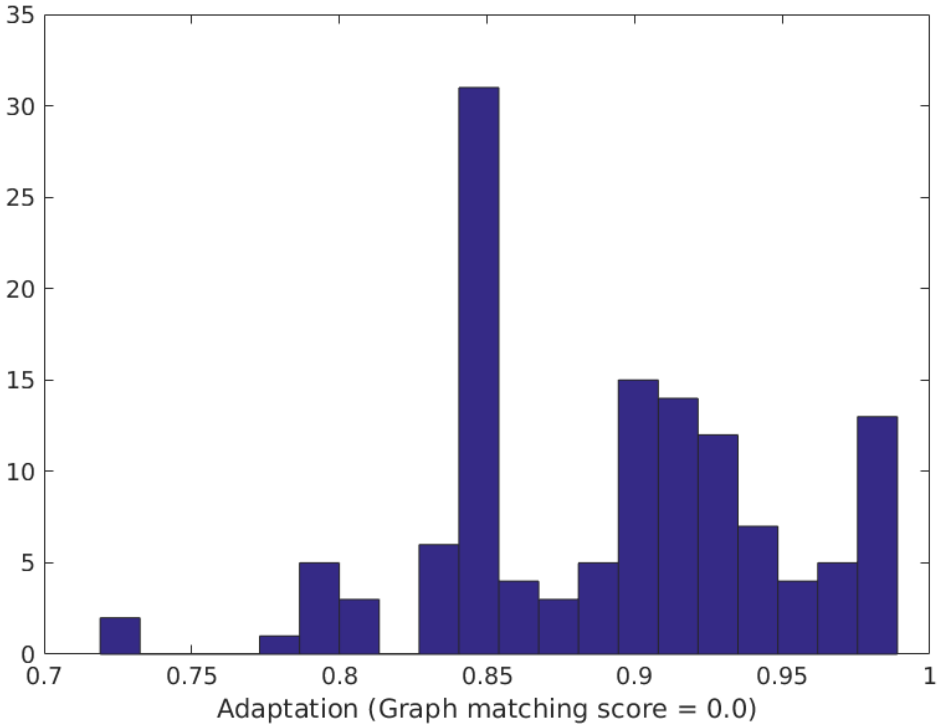


Figure 6.7: Distribution of domain adaptation error for pairs of common concepts when graph matching is in it's minimum value (0.0).

6.3.2 PCA-based approach

The graph-based conceptual domain adaptation results in Figure 6.6 suggest graph-matching is not an appropriate method for aligning correspondent concepts. We now experiment with a second method: PCA-based conceptual domain adaptation.

The idea of PCA-based conceptual domain adaptation is that PCA would provide us the orthogonal principle components of the data in order of eigenvalues, and that we can use this in order to align the correspondent concepts.

The concepts in the source and target in PCA-based approach are separately projected onto their principle components unlike the graph-based approach. Figure 5.5 shows the different steps for this method.

The first PC in source corresponds to the first PC in target. We train the classifier on source and use it to classify the target using this idea for aligning the PCs,

Using PCA as the alignment method, similar to graph-based approach, we require defining specific measures that evaluate the goodness of the alignment. We have tested two measures defined as follows:

- **Principle components** In the first attempt, we used the mean of angles between correspondent PCs of two domains. If the data in source and target are aligned, the score proposed would return a low value and vice versa.
- **Discrepancy** In the second approach we use the discrepancy measure defined in equation below:

$$S = |2 \times acc - 1| \tag{6.3}$$

where acc is the classification accuracy for a classifier that discriminates source from target data.

As we didn't observe good results applying the graph-based method, we switched to the next option (PCA-based). In PCA-based approach, although the semantics of concepts in higher-layers are changed, we use the idea of correspondence between units in two domains as the direction to align the two domains.

After running an exhaustive search over combinations of two hidden units with discrepancy measure as the score, we realize that similar to the graph-based approach, the score from PCA-based approach does not correlate with domain adaptation either. Figure 6.8 shows that minimizing the PCA-based score (discrepancy in this experiment), does not help minimizing domain adaptation error.

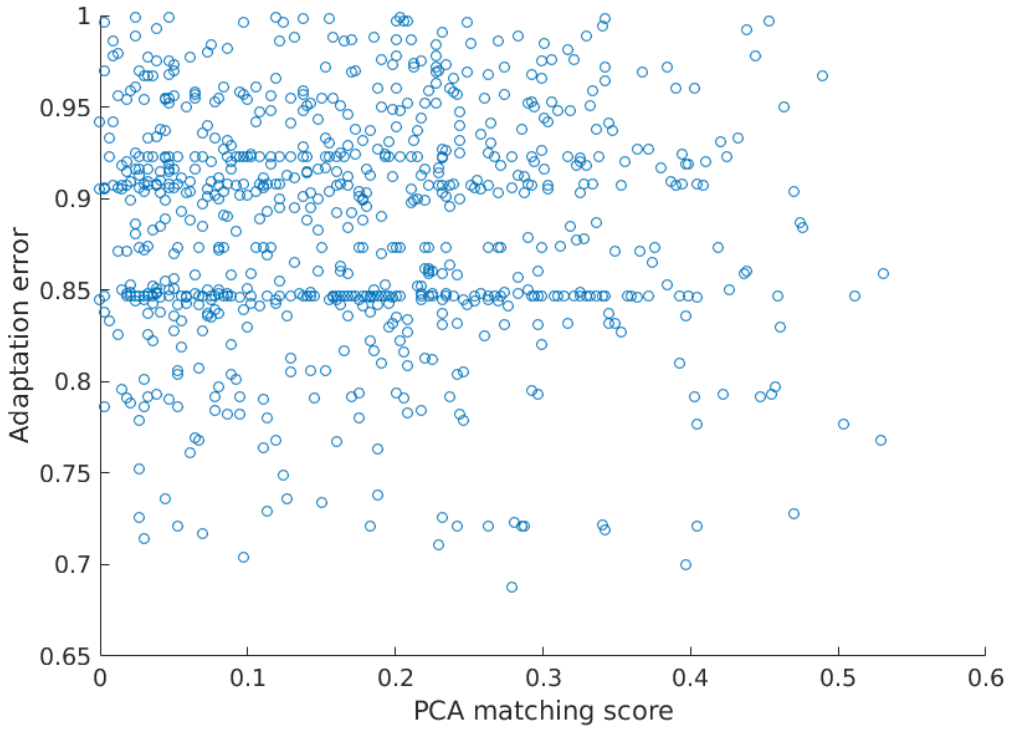


Figure 6.8: Relation between PCA-matching score and domain adaptation error.

Chapter 7

Contributions

By implementing multiple layers of non-linearities, deep networks have been able to extract high-level complex concepts from data. Despite the value behind extracting high-level concepts from data, deep networks have been naively used as new representations for classification. In other words, the use of extracted concepts have been limited to simply providing new representations that would facilitate the task of building improved models due to the capacity to disentangle factors of variation.

After introducing conceptual domain adaptation, we not only used deep networks to extract high-level concepts, but also we equipped the technique specifically for an alignment step required to perform domain adaptation. In this regard, apart from providing disentangled representations, the high-level concepts provide underlying semantic meaning that provides the basis for correspondences in conceptual alignment. In other words, the information provided by high-level neurons have been previously only used for generating new models, but in the new proposed approach,

correspondence between concepts (assuming local representation) provides information for alignment as well.

In general, two domains are suitable for domain adaptation when they are different but related. In conceptual domain adaptation, we consider the relatedness of two domains as having correspondent concepts with same semantic meaning. Furthermore, we identified a discrepancy between two domains called inherently lower-level representational discrepancy.

By relaxing the dependency of the alignment on representational similarities between concepts and solely relying on conceptual similarities, conceptual domain adaptation is beneficial for novel domain adaptation problems with inherently lower-level representational discrepancy between domains.

7.0.1 Future work

Apart from the supervised version of conceptual domain adaptation, we have attempted three other approaches to solve the problem of conceptual domain adaptation in the unsupervised version.

The main problem with unsupervised approaches is that their score does not align with the equivalent supervised score. In other words, when the unsupervised score (as in context-based, graph-based or PCA-based methods) is minimized, it doesn't guarantee that the domain adaptation error is also minimized. Therefore, further investigation is required in this regard.

Specifically, in graph-based and PCA-based approaches we restricted our experiment to the 9th layer of the network with a total of 8 hidden units and we chose size 2 as the number of common concepts. This limitation on the size of the experiment prevents us from getting a broad view of the problem.

As introduced in Section 5.2.3, initially we proposed a supervised version of conceptual domain adaptation. Although the proposed approach requires the target data fully labeled, it might also be practical to apply this method in a semi-supervised scenario where the target data is partially labeled.

The basic assumption regarding conceptual domain adaptation proposed in this document is the (strictly) local representation of high-level concepts in two domains. Accordingly, in future works, one might consider finding correspondences between concepts in their distributed –rather than local– representations.

Performing conceptual domain adaptation between two domains not only provides us the ability to transfer the model from one domain to the other, but also it facilitates the transfer of knowledge between the domains in forms of correspondences. Accordingly, the alignment model itself could be used as a translation model between two domains that indicates the correspondences between the two domains.

Although by definition of domain adaptation, transferring a model can be performed where there is only discrepancy between the marginal probability distribution of the two domains, a more general case such as transductive transfer learning as introduced in [29], shows that the difference between the two domains might lie in their feature spaces. In conceptual domain adaptation, one might be able to tackle

such transductive transfer learning problems as well simply by training two different networks, one for each domain.

Following up with the idea above, one should be able to tackle domain adaptation problems where not only the labeled samples are not available from the target domain, but also the target data itself is not available at the training phase of the model. In such cases, the source network can be trained independently of the target domain; when the target data is available, a separate network can then be trained and concepts can be aligned to perform domain adaptation.

Finally, conceptual domain adaptation is the first attempt that combines deep networks to extract high-level concepts with the semantic information hidden in the same concepts. In the future, we expect more research taking place showing new perspectives regarding the type of conceptual information amenable for extraction using deep networks.

Chapter 8

Conclusion

Assuming an ideal extraction of underlying high-level concepts using deep networks, in Section 5.2 we proposed performing simple target representation adjustment in order to align two domains. Our adjustment approach for aligning representations of correspondent concepts between source and target domain achieves reasonable performance results, supporting the fundamental hypothesis proposed in conceptual domain adaptation, namely that there exists correspondent concepts that can be used to measure the relatedness between two domains. The problem of domain adaptation is thus assumed to be solvable conditioned on prior knowledge about the proposed relatedness between domains.

Having assumed a lower-level representational discrepancy to the domain adaptation problem by rotating the target dataset in hand-written digit recognition problem, we observed a significant drop in domain adaptation accuracy. For this case specifically, conceptual domain adaptation using basic supervised fitness evaluation

of mapping matrix proved to provide a novel alternative solution to existing joint training approaches. Furthermore, by observing patterns across mapping matrices, we realized that the failure of the joint training approach is due to misalignment of correspondent concepts in different hidden units.

One of the main challenges in conceptual domain adaptation is finding correspondences between concepts in two domains which requires a measurable definition of semantic meaning of concepts. Assuming the context of a concept as an indicator of its semantic meaning, and furthermore, assuming that superimposition of concepts provide contextual information about concepts, we proposed context-based fitness evaluation of correspondences. We hypothesize that relying on the proposed evaluation method would relax the dependency of alignment on lower-level representational properties of data, while maintaining the domain adaptation performance comparable to the basic supervised evaluation approach.

We also explored an unsupervised conceptual domain adaptation that does not rely on labeled target data. We proposed three approaches including context-based, graph-based and PCA-based conceptual domain adaptation. Here we proposed three search techniques: (i) genetic algorithm, (ii) exhaustive search, (iii) heuristic search. To be able to evaluate the proposed approach, we used exhaustive search for graph-based and PCA-based methods. Using exhaustive search on a layer with a fixed number of common concepts helps us evaluate the score regardless of the goodness of the search mechanism. We concluded that more research is needed in the unsupervised approach to efficiently identify correspondent concepts.

Bibliography

[1] A. Arnold, R. Nallapati, and W. W. Cohen. A comparative study of methods for transductive transfer learning. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 77–82. IEEE, 2007.

[2] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, and others. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[3] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[4] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[5] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[6] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[7] L. Bruzzone and M. Marconcini. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):770–787, 2010.

[8] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

[9] S. Chopra, S. Balakrishnan, and R. Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML Workshop on Challenges in Representation Learning*.

[10] B. C. Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[11] J. Deng, Z. Zhang, F. Eyben, and B. Schuller. Autoencoder-based unsupervised domain adaptation for speech emotion recognition.

[12] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*, volume 53. Springer, 2003.

[13] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[14] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[15] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2551–2559.

[16] M. Ghifary, W. B. Kleijn, and M. Zhang. Deep hybrid networks with good out-of-sample object recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5437–5441. IEEE, 2014.

[17] M. Ghifary, W. B. Kleijn, and M. Zhang. Domain adaptive neural networks for object recognition. In *Pacific Rim International Conference on Artificial Intelligence*, pages 898–904. Springer, 2014.

[18] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[19] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520, 2011.

[20] I. J. Goodfellow, Q. V. Le, A. M. Saxe, H. Lee, and A. Y. Ng. Measuring Invariances in Deep Networks. In *NIPS*, volume 9, pages 646–654, 2009.

[21] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[22] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. A field guide to dynamical recurrent neural networks. IEEE Press, 2001.

[23] M. Kan, S. Shan, and X. Chen. Bi-shifting auto-encoder for unsupervised domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3846–3854, 2015.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105, 2015.

[26] B. B. Murdock Jr. Convolution and correlation in perception and memory. *Perspectives on Learning and Memory*, pages 105–119, 2014.

[27] H. V. Nguyen, H. T. Ho, V. M. Patel, and R. Chellappa. Dash-n: Joint hierarchical domain adaptation and feature learning. *IEEE Transactions on Image Processing*, 24(12):5479–5491, 2015.

[28] R. B. Palm. Prediction as a candidate for learning deep hierarchical models of data. *Technical University of Denmark*, 5, 2012.

[29] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[30] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[31] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[32] S. Thorpe. Localized versus distributed representations. In *The Handbook of Brain Theory and Neural Networks*, pages 549–552. MIT Press, 1998.

[33] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[35] R. A. Wilson and F. C. Keil. *The MIT Encyclopedia of the Cognitive Sciences*. MIT press, 2001.

[36] M. Zaslavskiy, F. Bach, and J.-P. Vert. A path following algorithm for graph matching. *Image and Signal Processing*, pages 329–337, 2008.