Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

Deep Nonlinear Feature Coding for Unsupervised Domain Adaptation

Pengfei Wei[1], **Yiping Ke**[1], **Chi Keong Goh**[2]
Nanyang Technological University, Singapore[1],
Rolls-Royce Advanced Technology Centre, Singapore[2]
{pwei001,ypke}@ntu.edu.sg, ChiKeong.Goh@Rolls-Royce.com

Abstract

Deep feature learning has recently emerged with demonstrated effectiveness in domain adaptation. In this paper, we propose a Deep Nonlinear Feature Coding framework (DNFC) for unsupervised domain adaptation. DNFC builds on the marginalized stacked denoising autoencoder (mSDA) to extract rich deep features. We introduce two new elements to mSDA: domain divergence minimization by Maximum Mean Discrepancy (MMD), and nonlinear coding by kernelization. These two elements are essential for domain adaptation as they ensure the extracted deep features to have a small distribution discrepancy and encode data nonlinearity. The effectiveness of DNFC is verified by extensive experiments on benchmark datasets. Specifically, DNFC attains much higher prediction accuracy than state-of-the-art domain adaptation methods. Compared to its basis mSDA, DNFC is able to achieve remarkable prediction improvement and meanwhile converges much faster with a small number of stacked layers.

1 Introduction

Conventional machine learning needs sufficient labeled data to achieve satisfactory prediction performance. Nonetheless, the acquiring of labeled data is an expensive and time-consuming process. Domain adaptation [Ben-David *et al.*, 2007; Pan and Yang, 2010; Margolis, 2011] provides an effective way to manage the label scarcity of data. The objective of domain adaptation is to learn a model that works well in a target domain where none or scarce labeled data is available, by leveraging upon the knowledge from a different but related source domain with plenty of labeled data. The main challenge of domain adaptation lies in the distribution discrepancy between source and target domains [Ben-David *et al.*, 2007]. Therefore, a direct application of the model learnt from the source domain to the target domain often results in poor performance, and thus effective adaptation is in demand.

Depending on the availability of labeled data in the target domain, domain adaptation can be categorized into unsupervised one (with no labeled data) and semi-supervised one (with scarce labeled data). In this paper, we focus on unsupervised domain adaptation, which is a more difficult task since no labeled data in the target domain can be used to guide the model learning.

In order to address the distribution discrepancy in unsupervised domain adaptation, various feature-based methods [Pan *et al.*, 2011; Gong *et al.*, 2012; Fernando *et al.*, 2013; Long *et al.*, 2014] are developed. They share the same underlying intuition which tries to find new feature representations well aligning the two domains. Among them, deep feature learning [Glorot *et al.*, 2011; Chen *et al.*, 2012; Zhou *et al.*, 2014; Ding *et al.*, 2015] has attracted much attention with demonstrated effectiveness. It learns deep features jointly from source and target data in an unsupervised manner. By doing so, generic concepts that exist in both domains can be extracted and captured in the new feature space. Such concepts can be further sharpened when more layers of deep features are constructed. Therefore, when operating on the deep feature spaces, the model learnt from one domain is able to adapt better to the other.

One typical unsupervised deep feature learning method is the marginalized stacked denoising autoencoder (mSDA) [Chen *et al.*, 2012]. It extracts multi-layer deep features by reconstructing the input data from a number of randomly corrupted ones. By averaging over an infinite number of corruptions, robust features can be extracted efficiently without materializing any data corruption. The feature representation generated by mSDA has yielded very impressive performance for cross-domain sentiment analysis. It has also been employed as a building block in other domain adaptation methods [Zhou *et al.*, 2014; Ding *et al.*, 2015].

Despite its success, mSDA suffers from two limitations. First, it does not pay attention to the domain divergence that might arise in the new feature space. As a result, distribution discrepancy on the extracted features may still be large. Second, it injects the nonlinearity after the feature learning. Consequently, the extracted features may not capture much nonlinear relationship in the data, which compromises one of the biggest strengths of deep learning.

In this paper, we propose a Deep Nonlinear Feature Coding framework (DNFC) to address these two limitations of mSDA for unsupervised domain adaptation. In DNFC, the minimization of domain divergence and the exploitation of data nonlinearity are incorporated seamlessly into the deep

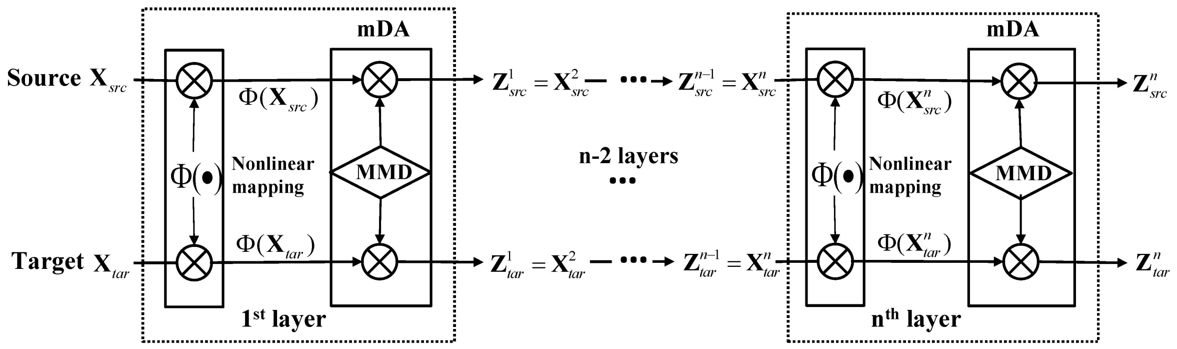

mDA
Source $\mathbf{X}_{src}$
Target $\mathbf{X}_{tar}$
$\Phi(\mathbf{X}_{src})$
$\Phi(\mathbf{X}_{tar})$
$\Phi(\bullet)$
Nonlinear mapping
MMD
$\mathbf{Z}^1_{src} = \mathbf{X}^2_{src}$
$\mathbf{Z}^{n-1}_{src} = \mathbf{X}^n_{src}$
$\mathbf{Z}^1_{tar} = \mathbf{X}^2_{tar}$
$\mathbf{Z}^{n-1}_{tar} = \mathbf{X}^n_{tar}$
n-2 layers
1st layer
nth layer
$\Phi(\mathbf{X}^n_{src})$
$\Phi(\mathbf{X}^n_{tar})$
$\mathbf{Z}^n_{src}$
$\mathbf{Z}^n_{tar}$

Figure 1: Deep nonlinear feature coding framework

feature learning. More specifically, we introduce two new elements, Maximum Mean Discrepancy (MMD) and kernelization, into 1-layer mSDA and stack the single-layer nonlinear feature coding to create higher-level deep features. As illustrated in Figure 1, at each layer, we measure the distribution discrepancy between the source and target domains in the new deep feature space by MMD and minimize it in the feature learning process. This enforces the distribution discrepancy in the new feature representation, on which domain adaptation is performed, to be minimized. Moreover, we also develop a kernelization that fuses the nonlinearity exploitation with the feature learning. Both elements play important roles in learning good features for domain adaptation and are enclosed in a unified framework.

2 Related Work

A number of feature-based methods have been developed for unsupervised domain adaptation. These methods aim to learn a common feature representation shared by source and target domains, on which the model built from the source domain adapts well to the target domain. One typical approach is to construct *subspaces* as new feature representations. Recent studies include GFK [Gong *et al.*, 2012], DASA [Fernando *et al.*, 2013], and TJM [Long *et al.*, 2014]. GFK integrates an infinite number of subspaces that lie between source and target along the geodesic flow on a Grassmann manifold. DASA directly aligns the subspaces of the two domains, without considering intermediate subspaces. TJM extracts a subspace by jointly matching features and reweighting instances under the kernel PCA.

As another type of feature-based method, deep feature learning has recently emerged and demonstrated its effectiveness for unsupervised domain adaptation. Marginalized denoising autoencoder (mDA) [Chen *et al.*, 2012] learns feature embedding by reconstructing the original data from the randomly corrupted ones. To extract multiple layers of deep features, mDA are stacked to mSDA in a layer-wise manner. Thanks to its effectiveness in capturing deep structures embedded in data, mSDA is also used as a building block for other domain adaptation methods [Zhou *et al.*, 2014; Ding *et al.*, 2015]. In addition, autoencoders are also applied to transfer model parameters across different domains [Deng *et al.*, 2013; Kandaswamy *et al.*, 2014; Zhuang *et al.*, 2015]. In this paper, we incorporate two new elements into mSDA, which makes it more powerful for domain adaptation.

Our work adopts Maximum Mean Discrepancy (MMD) [Gretton *et al.*, 2006] to quantify the domain divergence. MMD is widely used to measure the distance between two distributions in domain adaptation [Pan *et al.*, 2011; Long *et al.*, 2013; Wang *et al.*, 2015]. It avoids density estimations and uses the mean embeddings of the two distributions in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ for distance calculation. Given two sets of samples $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_{n_1}\}$ and $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_{n_2}\}$, drawn from the distributions P and Q respectively, the empirical MMD is calculated as:

$$||\frac{1}{n_1}\sum_{i=1}^{n_1}\Phi(\mathbf{x}_i) - \frac{1}{n_2}\sum_{i=1}^{n_2}\Phi(\mathbf{y}_i)||_{\mathcal{H}},$$

where $\Phi : \mathbb{R} \to \mathcal{H}$ with $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^{\mathrm{T}}\Phi(\mathbf{x}_j)$ as the kernel of $\mathcal{H}$.

3 A Deep Nonlinear Coding Approach

In this section, we introduce the proposed Deep Nonlinear Feature Coding (DNFC) approach in detail.

3.1 Problem Statement

Denote $\mathbf{X}_{src}$ as a set of n_s labeled samples from source domain: $\mathbf{X}_{src} = \{\mathbf{x}^s_1, ..., \mathbf{x}^s_{n_s}\}$ and $\mathbf{Y}_{src} = \{y^s_1, ..., y^s_{n_s}\}$, where $\mathbf{x}^s_i \in \mathbb{R}^d$ and y^s_i is the class label. Denote $\mathbf{X}_{tar}$ as a set of n_t unlabeled samples from target domain: $\mathbf{X}_{tar} = \{\mathbf{x}^t_1, ..., \mathbf{x}^t_{n_t}\}$, where $\mathbf{x}^t_i \in \mathbb{R}^d$. Denote $\mathbf{X} = \mathbf{X}_{src} \cup \mathbf{X}_{tar} = [\mathbf{x}_1, ..., \mathbf{x}_{n_s}, ..., \mathbf{x}_n] = [\mathbf{f}_1; \mathbf{f}_2; ...; \mathbf{f}_d] \in \mathbb{R}^{d \times n}$, where $n = n_s + n_t$ and $\mathbf{f}_i \in \mathbb{R}^n$ is the *i-th* dimensional feature vector. The task is to predict the labels $\mathbf{Y}_{tar}$ in the target domain, by leveraging the labelled data from the source domain.

3.2 Main Idea

The proposed DNFC aims to extract deep structures from source and target data, and use them as features in domain adaptation. DNFC is based on the marginalized stacked denoising autoencoder (mSDA) [Chen *et al.*, 2012], which extracts multiple layers of deep features by stacking the marginalized denoising autoencoder (mDA). Though mSDA has demonstrated encouraging results for domain adaptation

[Chen *et al.*, 2012; Zhou *et al.*, 2014; Ding *et al.*, 2015], it suffers from two limitations. First, the deep features are extracted by minimizing the reconstruction error solely. The divergence between source and target in the deep feature space is not taken care of. Second, mSDA learns a linear mapping, after which the nonlinearity is inserted by applying a nonlinear squashing function. As a result, the nonlinear relationship embedded in the data may not be well exploited and captured.

To address these two limitations, we propose to bring in two new elements: (1) Maximum Mean Discrepancy (MMD) for domain divergence minimization; and (2) kernelization for nonlinear coding. In the subsequent subsections, we first introduce the original mDA, followed by these two new elements. We then describe how we incorporate them seamlessly with mSDA in a unified framework for domain adaptation.

3.3 The Original mDA

mDA is a type of autoencoder that corrupts the inputs before mapping them into the deep representation [Chen *et al.*, 2012]. It aims to learn a linear mapping $\mathbf{W}$ to reconstruct the original data from the corrupted ones. Its objective function is formulated as:

$$L(\mathbf{W})_{mDA} = \frac{1}{2mn} tr[(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}})^{\mathrm{T}}(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}})], \quad (1)$$

where $\overline{\mathbf{X}} = [\mathbf{X}, \mathbf{X}, ..., \mathbf{X}]$, is the *m*-times repeated version of the input $\mathbf{X}$; $\widetilde{\mathbf{X}} = [\widetilde{\mathbf{X}^1}, \widetilde{\mathbf{X}^2}, ..., \widetilde{\mathbf{X}^m}]$ is the corrupted version of $\overline{\mathbf{X}}$ with *m* different corruptions of $\mathbf{X}$ at a feature corruption probability p; and $tr(\cdot)$ is the operator to calculate the trace of a matrix. A robust mapping $\mathbf{W}$ in Eq. (1) can be obtained by averaging over infinitely many *m* corruptions, without actually constructing any corruption. This process is named marginalization and enables efficient implementation.

3.4 Domain Divergence Minimization by MMD

We propose to use MMD to quantify the domain divergence in the learnt deep feature space and incorporate it in the learning of $\mathbf{W}$. This is particularly essential for unsupervised learning of deep features since no labeled data in the target domain is available to guide the feature learning.

Denote $\widetilde{\mathbf{x}_i^r}$ as the *i-th* sample in the *r-th* corrupted version of $\mathbf{X}$. Following the definition of MMD, the empirical distance between source and target domains in the *r-th* reconstructed space is:

$$\begin{aligned} &\|\frac{1}{n_s}\sum_{i=1}^{n_s}\mathbf{W}\widetilde{\mathbf{x}_i^r} - \frac{1}{n_t}\sum_{i=n_s+1}^{n}\mathbf{W}\widetilde{\mathbf{x}_i^r}\|^2 \\ =&tr(\mathbf{W}\widetilde{\mathbf{X}^r}\mathbf{M}\widetilde{\mathbf{X}^r}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}) \end{aligned} \quad (2)$$

where $\mathbf{M} = [M_{i,j}]_{n \times n}$ with $M_{i,j} = \frac{1}{n_s^2}$ if $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_{src}$; $M_{i,j} = \frac{1}{n_t^2}$ if $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_{tar}$; otherwise $M_{i,j} = -\frac{1}{n_s n_t}$.

Combining Eq. (1) and Eq. (2), and considering m corrupted versions, we formulate the objective function as:

$$\begin{aligned} L(\mathbf{W}) = &tr[(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}})^{\mathrm{T}}(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}})] \\ &+ \theta tr(\mathbf{W}\widetilde{\mathbf{X}}\widetilde{\mathbf{M}}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}), \end{aligned} \quad (3)$$

where $\widetilde{\mathbf{M}}$ is a block diagonal matrix with *m* copies of $\mathbf{M}$ as the diagonal elements, and $\theta > 0$ is the balancing parameter. Eq. (3) has a closed-form solution:

$$\begin{aligned} &\mathbf{W} = \mathbf{P}(\mathbf{Q}_1 + \theta\mathbf{Q}_2)^{-1} \text{ with} \\ &\mathbf{P} = \overline{\mathbf{X}}\widetilde{\mathbf{X}}^{\mathrm{T}},\ \mathbf{Q}_1 = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^{\mathrm{T}},\ \mathbf{Q}_2 = \widetilde{\mathbf{X}}\widetilde{\mathbf{M}}\widetilde{\mathbf{X}}^{\mathrm{T}}. \end{aligned} \quad (4)$$

The computation of $\mathbf{W}$ relies on the construction of m corruptions of $\mathbf{X}$. The more corruptions are constructed, the more robust $\mathbf{W}$ is. Since the corruptions are obtained independently with the same corruption probability p for each feature, we can apply the weak law of large numbers and derive the expectations of $\mathbf{P}$, $\mathbf{Q}_1$ and $\mathbf{Q}_2$ as $m \to \infty$.

Define $\mathbf{q} = [1-p, 1-p, ..., 1-p] \in \mathbb{R}^d$, $\mathbf{R} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$, $\mathbf{S} = \mathbf{X}\mathbf{M}\mathbf{X}^{\mathrm{T}}$, and $\mathbf{T} = \mathbf{X}\mathbf{N}\mathbf{X}^{\mathrm{T}}$ ($\mathbf{N}$ is a diagonal matrix whose diagonal elements are the same as those in $\mathbf{M}$).

For $\mathbf{Q}_1$, we have:

$$\begin{aligned} &E[\mathbf{Q}_1] = E[\widetilde{\mathbf{X}^r}\widetilde{\mathbf{X}^r}^{\mathrm{T}}] \\ \Rightarrow &E[\mathbf{Q}_1]_{\alpha,\beta} = E[\widetilde{\mathbf{f}_\alpha^r}\widetilde{\mathbf{f}_\beta^r}^{\mathrm{T}}] = E[\sum_{i=1}^{n}\widetilde{f_{\alpha,i}^r}\widetilde{f_{\beta,i}^r}] \end{aligned}$$

When $\alpha \neq \beta$,

$$E[\mathbf{Q}_1]_{\alpha,\beta} = \mathbf{q}_\alpha\mathbf{q}_\beta\sum_{i=1}^{n}f_{\alpha,i}f_{\beta,i} = \mathbf{q}_\alpha\mathbf{q}_\beta\mathbf{R}_{\alpha,\beta}$$

When $\alpha = \beta$,

$$E[\mathbf{Q}_1]_{\alpha,\alpha} = \mathbf{q}_\alpha\sum_{i=1}^{n}f_{\alpha,i}f_{\alpha,i} = \mathbf{q}_\alpha\mathbf{R}_{\alpha,\alpha}$$

For $\mathbf{Q}_2$, we have:

$$E[\mathbf{Q}_2]_{\alpha,\beta} = E[\widetilde{\mathbf{f}_\alpha^r}\mathbf{M}\widetilde{\mathbf{f}_\beta^r}^{\mathrm{T}}] = E[\sum_{i=1}^{n}\sum_{j=1}^{n}\widetilde{f_{\alpha,i}^r}M_{i,j}\widetilde{f_{\beta,j}^r}]$$

When $\alpha \neq \beta$,

$$E[\mathbf{Q}_2]_{\alpha,\beta} = \mathbf{q}_\alpha\mathbf{q}_\beta\sum_{i=1}^{n}\sum_{j=1}^{n}f_{\alpha,i}M_{i,j}f_{\beta,j} = \mathbf{q}_\alpha\mathbf{q}_\beta\mathbf{S}_{\alpha,\beta}.$$

When $\alpha = \beta$,

$$\begin{aligned} E[\mathbf{Q}_2]_{\alpha,\alpha} &= \mathbf{q}_\alpha^2\sum_{i=1}^{n}\sum_{\substack{j=1 \\ i\neq j}}^{n}f_{\alpha,i}M_{i,j}f_{\alpha,j} + \mathbf{q}_\alpha\sum_{i=1}^{n}f_{\alpha,i}^2M_{i,i} \\ &= \mathbf{q}_\alpha^2\mathbf{S}_{\alpha,\alpha} + \mathbf{q}_\alpha(\mathbf{1} - \mathbf{q}_\alpha)\mathbf{T}_{\alpha,\alpha}. \end{aligned}$$

Similarly, we can obtain the expectation of $\mathbf{P}$ as $E[\mathbf{P}]_{\alpha,\beta} = \mathbf{q}_\beta\mathbf{R}_{\alpha,\beta}$.

After obtaining these expectation matrices, the mapping $\mathbf{W}$ can be computed as:

$$\mathbf{W} = E[\mathbf{P}](E[\mathbf{Q}_1] + \theta E[\mathbf{Q}_2])^{-1}. \quad (5)$$

By incorporating MMD into mDA, we not only take advantage of mDA in obtaining discriminative deep features but also ensure that the new feature space brings the two domains as close as possible. As evidenced by our experimental results, this leads to much better adaptation performance. We also remark that the incorporation of MMD is seamless in the sense that the marginalization property of mDA is preserved and thus the efficiency of the proposed approach is ensured.

3.5 Nonlinear Coding by Kernelization

With the learning of $\mathbf{W}$, only the linear relationship in the input data is captured. In the original mDA, the nonlinearity is inserted by applying a nonlinear squashing function on the output of mDA. Consequently, the nonlinearity in the data may not be well exploited because it is not taken care of in the learning of $\mathbf{W}$. Note that with the incorporation of MMD, such a neglect also harms the minimization of domain divergence in the deep feature space.

In order to address this issue, we propose a kernelized solution. Specifically, we use a nonlinear mapping function Φ to map the original data to a RKHS. We then reconstruct the original data by randomly corrupting the mapped data in the RKHS. By doing so, we factor in the nonlinearity in the learning of $\mathbf{W}$ and thus in the extracted deep features.

Suppose that $\mathbf{X}$ is mapped to a RKHS $\mathcal{H}$ by a nonlinear mapping function $\Phi : \mathbb{R}^d \to \mathcal{H}$. Denote $\Phi(\mathbf{X}) = [\Phi(\mathbf{x}_1), ..., \Phi(\mathbf{x}_{n_s}), ..., \Phi(\mathbf{x}_n)]$ as the mapped data matrix and $\mathbf{K} = \Phi(\mathbf{X})^{\mathrm{T}}\Phi(\mathbf{X})$ as the corresponding kernel matrix. The mDA term to reconstruct the original data from the corrupted mapped data now becomes:

$$L(\mathbf{W})_{mDA} = \frac{1}{2mn} tr[(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})^{\mathrm{T}}(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})]$$

where $\widetilde{\Phi(\mathbf{X})} = [\widetilde{\Phi(\mathbf{X})}^1, \widetilde{\Phi(\mathbf{X})}^2, ..., \widetilde{\Phi(\mathbf{X})}^m]$ represents the corrupted mapped data with *m* different corruptions of $\Phi(\mathbf{X})$.

With the incorporation of nonlinearity, the objective function that combines mDA and MMD terms becomes:

$$\begin{aligned} L(\mathbf{W}) =& tr[(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})^{\mathrm{T}}(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})] \\ &+ \theta tr(\mathbf{W}\widetilde{\Phi(\mathbf{X})}\widetilde{\mathbf{M}}\widetilde{\Phi(\mathbf{X})}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}) \end{aligned}$$

According to [Niyogi, 2004], the linear mapping $\mathbf{W}$ in $\mathcal{H}$ can be represented as a linear combination of the data points in $\mathcal{H}$, that is, $\mathbf{W} = \mathbf{W}_k\Phi(\mathbf{X})^{\mathrm{T}}$. The objective function thus becomes:

$$\begin{aligned} L(\mathbf{W}_k) =& tr[(\overline{\mathbf{X}} - \mathbf{W}_k\widetilde{\mathbf{K}})^{\mathrm{T}}(\overline{\mathbf{X}} - \mathbf{W}_k\widetilde{\mathbf{K}})] \\ &+ \theta tr(\mathbf{W}_k\widetilde{\mathbf{K}}\widetilde{\mathbf{M}}\widetilde{\mathbf{K}}^{\mathrm{T}}\mathbf{W}_{\mathbf{k}}^{\mathrm{T}}), \end{aligned} \tag{6}$$

where $\widetilde{\mathbf{K}} = [\widetilde{\mathbf{K}^1}, \widetilde{\mathbf{K}^2}, ..., \widetilde{\mathbf{K}^m}]$ is the *m* corrupted kernel matrix with $\widetilde{\mathbf{K}^r} = [\widetilde{\mathbf{k}_1^r}; \widetilde{\mathbf{k}_2^r}; ...; \widetilde{\mathbf{k}_n^r}]$, $\widetilde{\mathbf{k}_i^r} = [\widetilde{k_{i,1}^r}, \widetilde{k_{i,2}^r}, ..., \widetilde{k_{i,n}^r}]$ and $\widetilde{k_{i,j}^r} = \Phi(\mathbf{x}_i)^{\mathrm{T}}\widetilde{\Phi(\mathbf{x}_j)}^r$. The closed-form solution of Eq. (6) is:

$$\begin{aligned} &\mathbf{W}_k = \mathbf{P}_k(\mathbf{Q}_{k1} + \theta\mathbf{Q}_{k2})^{-1} \text{ with} \\ &\mathbf{P}_k = \overline{\mathbf{X}}\widetilde{\mathbf{K}}^{\mathrm{T}},\ \ \mathbf{Q}_{k1} = \widetilde{\mathbf{K}}\widetilde{\mathbf{K}}^{\mathrm{T}},\ \ \mathbf{Q}_{k2} = \widetilde{\mathbf{K}}\widetilde{\mathbf{M}}\widetilde{\mathbf{K}}^{\mathrm{T}}. \end{aligned} \tag{7}$$

Since corrupting $\Phi(\mathbf{X})$ corresponds to corrupting the kernel matrix $\mathbf{K}$, we assume that each element $\widetilde{k_{i,j}^r}$ is corrupted with probability p. We now define the kernelized counterparts: $\mathbf{q}' = [1-p, 1-p, ..., 1-p] \in \mathbb{R}^n$, $\mathbf{R}' = \mathbf{K}\mathbf{K}^{\mathrm{T}}$, $\mathbf{S}' = \mathbf{K}\mathbf{M}\mathbf{K}^{\mathrm{T}}$, and $\mathbf{T}' = \mathbf{K}\mathbf{N}\mathbf{K}^{\mathrm{T}}$. The expected values of $\mathbf{Q}_{k1}$ and $\mathbf{Q}_{k2}$ can be obtained in a similar way to those of $\mathbf{Q}_1$ and $\mathbf{Q}_2$:

$$E[\mathbf{Q}_{k1}]_{\alpha,\beta} = E[\sum_{i=1}^{n} \widetilde{k_{\alpha,i}^r}\widetilde{k_{\beta,i}^r}] = \mathbf{q}'_\alpha\mathbf{q}'_\beta\mathbf{R}'_{\alpha,\beta},$$

$$E[\mathbf{Q}_{k1}]_{\alpha,\alpha} = E[\sum_{i=1}^{n} \widetilde{k_{\alpha,i}^r}\widetilde{k_{\alpha,i}^r}] = \mathbf{q}'_\alpha\mathbf{R}'_{\alpha,\alpha},$$

$$E[\mathbf{Q}_{k2}]_{\alpha,\beta} \ = E[\sum_{i=1}^{n}\sum_{j=1}^{n} \widetilde{k_{\alpha,i}^r}\, M_{i,j}\widetilde{k_{\beta,j}^r}] = \mathbf{q}'_\alpha\mathbf{q}'_\beta\mathbf{S}'_{\alpha,\beta},$$

$$\begin{aligned} E[\mathbf{Q}_{k2}]_{\alpha,\alpha} &= E[\sum_{i=1}^{n}\sum_{\substack{j=1 \\ i\neq j}}^{n} \widetilde{k_{\alpha,i}^r}M_{i,j}\widetilde{k_{\alpha,j}^r} + \sum_{i=1}^{n} \widetilde{k_{\alpha,i}^r}M_{i,i}\widetilde{k_{\alpha,i}^r}] \\ &= \mathbf{q}'_\alpha\mathbf{q}'_\alpha\mathbf{S}'_{\alpha,\alpha} + \mathbf{q}'_\alpha(\mathbf{1} - \mathbf{q}'_\alpha)\mathbf{T}'_{\alpha,\alpha}. \end{aligned}$$

Similarly, we define $\mathbf{U} = \mathbf{X}\mathbf{K}^{\mathrm{T}}$ and obtain the expectation of the matrix $\mathbf{P}_k$: $E[\mathbf{P}_k]_{\alpha,\beta} = \mathbf{q}'_\beta\mathbf{U}_{\alpha,\beta}$. Then, we obtain the mapping matrix:

$$\mathbf{W}_k = E[\mathbf{P}_k](E[\mathbf{Q}_{k1}] + \theta E[\mathbf{Q}_{k2}])^{-1}. \tag{8}$$

Our nonlinear feature coding at a single layer is then defined as:

$$\mathbf{Z} = \mathbf{W}_k\mathbf{K}. \tag{9}$$

It encloses both the data nonlinearity and domain similarity in the deep feature space.

3.6 Deep Nonlinear Feature Coding (DNFC)

Similar to the stacking of mDA in mSDA, we stack our single-layer feature coding in Section 3.5 so as to create richer deep feature representations. The framework is illustrated in Figure 1. The stacking is performed by feeding the output $\mathbf{Z} = [\mathbf{Z}_{src}, \mathbf{Z}_{tar}]$ of each layer into the next layer as the input. In this way, higher-level deep structures can be captured. To avoid overfitting the data nonlinearity during stacking, we conduct cross validation at each layer on the source data to select a suitable kernel function. Our proposed DNFC is summarized in **Algorithm** 1.

For domain adaptation, we concatenate the outputted feature coding at all L layers as the final feature representation. A classifier is then trained on the labeled source data and applied to the target domain to perform the prediction.

Algorithm 1 Deep Nonlinear Feature Coding

```
Input: Source data matrix $\mathbf{X}_{src}$, target data matrix $\mathbf{X}_{tar}$,
and the number of layers $L$.
for $k = 1$ to $L$ do
    1. Select kernel function using cross validation on
    source;
    2. Learn coding $\mathbf{Z}_{src}^k$ and $\mathbf{Z}_{tar}^k$ by Eq. (9);
    3. Set $\mathbf{X}_{src}^{k+1} = \mathbf{Z}_{src}^k$ and $\mathbf{X}_{tar}^{k+1} = \mathbf{Z}_{tar}^k$.
end for
Output: Feature coding $\{\mathbf{Z}_{src}^k, \mathbf{Z}_{tar}^k\}, (k = 1, ..., L)$.
```

4 Experimental Results

In this section, we evaluate the performance of DNFC by comparing with state-of-the-art domain adaptation methods. We also study the properties of DNFC and analyze the effect of the two new elements in DNFC.

4.1 Datasets & Experimental Setting

We use two benchmark datasets that are widely used in domain adaptation.

Amazon product review dataset [Blitzer *et al.*, 2007] contains sentiment reviews from four product categories: books (B), DVD (D), electronics (E) and kitchen appliance (K). Each review is characterized by unigram and bigram tf-idf features and labeled as positive or negative. Each domain has about 5,000 samples. When a domain is selected as source (target), all the samples in this domain are used as training (test) data. By pairing up the domains for adaptation task, we have 12 domain pairs, denoted as 'source→target'. For example, K→E means that category K is the source domain and E is the target domain.

20-Newsgroups [Dai *et al.*, 2007] consists of about 20,000 documents coming from four top categories: computer (C), recording (R), science (S), and talk (T). Each top category has four subcategories. Top categories are treated as labels, while subcategories are treated as related domains. Therefore, six binary prediction tasks are formed: C-R, C-S, C-T, R-S, R-T, and S-T. We take the task C-R for instance. Top category C is the positive class and R is the negative class. Two subcategories under each class are selected to constitute the source domain, while another two subcategories are selected to form the target domain. By exchanging the roles of the two domains, we have two domain pairs for the prediction task C-R, denoted as C-R1 and C-R2.

We compare DNFC with five baselines: NN, which predicts target labels using the 1-nearest neighbour classifier trained on the source domain with original features, DASA [Fernando *et al.*, 2013], GFK [Gong *et al.*, 2012], TJM [Long *et al.*, 2014], and mSDA [Chen *et al.*, 2012]. Following [Gong *et al.*, 2012; Long *et al.*, 2014], we use 1-NN as the base classifier since it avoids model parameter tuning. For subspace-based methods DASA, GFK, and TJM, we use the default subspace dimension d if it is specified by the authors, otherwise we set d to be the smallest one that contributes 85% of the energy. For TJM, we use the default kernel function specified by the authors. In our DNFC, we conduct the cross validation on source to automatically select between 'rbf' and linear kernels at each layer. For mSDA and DNFC, we set the default number of layers as three and do the cross validation on source to select the best corruption probability p between 0.1 and 0.9 with step size 0.1.

4.2 Comparison with Baselines

The classification accuracies of DNFC and five baselines on the two benchmarks are reported in Table 1. The best result on each domain pair is highlighted in bold, and the runner-up is highlighted in bold-italic.

As can be seen in Table 1, our DNFC achieves very promising performance and is a clear winner. Out of 24 domain pairs, DNFC performs the best in 21 and the second best in 3. For mSDA, on which DNFC is based, it is the best for 3 domain pairs and the runner-up for 15 domain pairs. The remaining 6 runner-ups mostly go to TJM, which is the most recent subspace-based method with nonlinear feature mapping. The performance improvement of DNFC over the second best method mSDA is up to 14.12% and is 5.32% on average. This demonstrates that the two new elements in DNFC are very effective in achieving better domain adaptation.

Table 1: Accuracy (%) on two benchmark datasets

Dataset	NN	DASA	GFK	TJM	mSDA	DNFC
B → D	51.04	59.42	55.49	61.92	***64.79***	**66.66**
B → E	53.26	57.36	55.84	59.92	***65.79***	**66.43**
B → K	56.37	60.68	58.86	61.65	***64.01***	**69.35**
D → B	52.51	61.29	56.95	63.34	***67.09***	**70.84**
D → E	55.23	59.23	57.28	63.16	***67.07***	**67.31**
D → K	56.53	61.36	59.27	65.52	**69.06**	***68.35***
E → B	51.45	56.82	54.29	61.16	***62.13***	**62.22**
E → D	51.37	57.21	54.24	62.94	***63.19***	**65.16**
E → K	54.84	65.62	61.66	70.84	**75.49**	***72.73***
K → B	50.93	51.93	51.44	***61.74***	52.95	**62.65**
K → D	51.86	57.31	54.36	***63.35***	60.79	**63.83**
K → E	53.55	59.38	54.53	69.37	***69.67***	**74.11**
C-R1	56.21	57.75	60.33	***60.75***	53.90	**68.98**
C-R2	55.22	60.73	60.31	63.36	**79.30**	***78.03***
C-S1	60.31	61.08	55.86	63.13	***63.47***	**77.59**
C-S2	54.51	***61.07***	56.73	60.39	58.86	**72.40**
C-T1	59.62	72.15	68.61	77.13	***85.93***	**87.18**
C-T2	58.78	68.56	67.82	76.45	***87.38***	**91.27**
R-S1	59.29	62.91	49.54	64.34	***68.63***	**77.38**
R-S2	62.31	50.67	65.60	62.90	***71.84***	**74.70**
R-T1	51.89	60.89	53.50	***67.99***	59.75	**76.61**
R-T2	53.49	57.80	58.32	***64.07***	60.68	**69.82**
S-T1	54.78	62.63	53.26	65.18	***69.54***	**82.02**
S-T2	57.39	64.74	59.57	62.77	***75.28***	**83.14**
Average	55.11	60.36	57.65	64.72	67.36	**72.68**

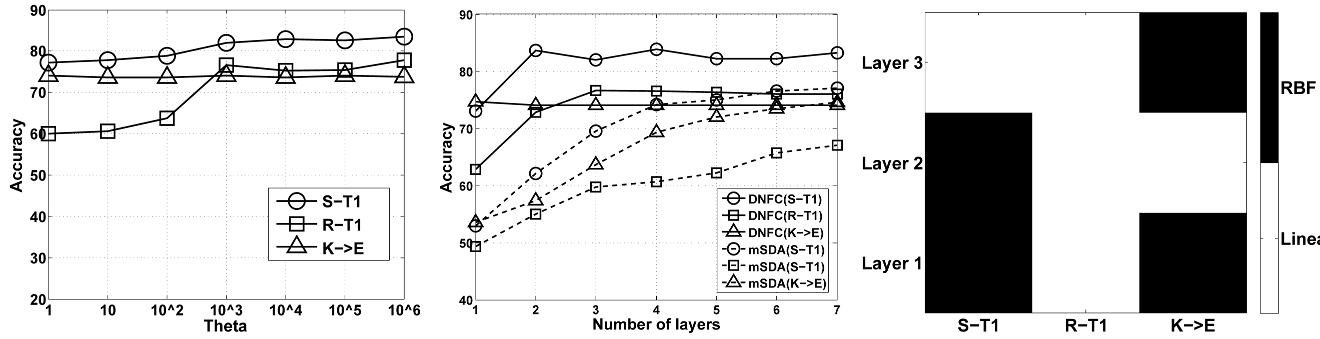
The results also show that the deep-feature-based methods, DNFC and mSDA, significantly outperform the subspace-based methods. This demonstrates the superiority of exploiting deep structures. The success of DNFC and TJM (with respect to other subspace-based methods) also reveals the necessity of capturing the data nonlinearity. Our DNFC takes advantage of both and thus is able to significantly boost the domain adaptation performance.

4.3 Property Study on DNFC

In this subsection, we study our proposed DNFC in four aspects: 1) sensitivity analysis on the balancing parameter θ; 2) the influence of the number of layers; 3) the kernel functions selected in each layer; and 4) the effect of feature concatenation. Due to space limit, we only report the results of three domain pairs: S-T1, R-T1, and K→E.

As shown in Figure 2(a), with the increase of θ, the classification accuracy generally increases until it gets stable. The larger θ is, the more sensitive the MMD term is to the objective function. When the empirical MMD gets stable, larger values of θ cannot further improve the performance. For all three domain pairs, this happens when θ reaches 10^3, which is also the case for all other domain pairs. In our experiments, we set $\theta = 10^3$ as default.

Figure 2(b) shows how the number of layers affects the performance of DNFC. The results of mSDA are also shown as a reference. In general, both DNFC and mSDA obtain better results when more layers are stacked. However, they have different trends. With the increase in the number of layers,



(a) Sensitivity analysis on the balancing parameter θ

(b) Influence of the number of layers

(c) Kernel functions selected at each layer

Figure 2: Property study on DNFC

the performance of DNFC converges very fast, while that of mSDA improves gradually. Even after 7 layers of stacking, mSDA is still not as good as DNFC. In particular, the accuracy of mSDA is about 5% worse than that of DNFC for S-T1 and R-T1 with 7 layers of stacking. The different trends may be due to the different ways of exploiting the data nonlinearity. mSDA injects the nonlinearity to the output of each layer, while DNFC incorporates the consideration of nonlinearity into the feature learning process. As a result, the data nonlinearity can be captured by DNFC more quickly with a small number of stacked layers. Considering that more stacked layers incur heavier computational overheads, the quicker convergence of DNFC also becomes a big strength, which makes it more practically useful in real applications.

Figure 2(c) shows the kernels selected by the cross validation on the source domain at different layers of DNFC. It demonstrates the flexibility of selecting different suitable kernels for different layers even for the same domain pair.

We also investigate the impact of concatenating all layers of features. We test DNFC using the final layer features only and find that the performance is just slightly compromised. On 20-Newsgroups, its average accuracy is only 0.53% lower than the one with concatenated features. This suggests that one can simply use the final layer features in DNFC if the slightly lower accuracy is not a big concern.

4.4 Effect of New Elements in DNFC

In this subsection, we assess the individual effect of MMD and kernelization in DNFC, as well as their joint effects, with respect to the performance of domain adaptation.

We develop two variants of mSDA: one with MMD but not kernelization, the other with the kernelization but not MMD. The two variants are referred to as mSDA_MMD and mSDA_Kernel, respectively. We compare DNFC with mSDA and the two variants on the 20-Newsgroups dataset.

The results are reported in Table 2. Overall, either MMD or kernelization alone improves the performance. This demonstrates the effectiveness of each individual element in DNFC. Moreover, as evidenced by the performance of DNFC, integrating MMD and kernelization together is able to further boost the improvement. We notice that the impacts of MMD and kernelization could vary even within the same domains.

Table 2: Accuracy (%) on 20-Newsgroups for assessing the effect of new elements in DNFC

Dataset	mSDA	mSDA_MMD	mSDA_Kernel	DNFC
C-R1	53.90	71.21	58.27	68.98
C-R2	79.30	77.95	75.40	78.03
C-S1	63.47	70.92	73.48	77.59
C-S2	58.86	63.63	69.59	72.40
C-T1	85.93	89.00	87.46	87.18
C-T2	87.38	87.49	90.75	91.27
R-S1	68.63	75.71	65.85	77.38
R-S2	71.84	71.33	65.26	74.70
R-T1	59.75	72.54	59.94	76.61
R-T2	60.68	67.97	52.87	69.82
S-T1	69.54	78.05	77.29	82.02
S-T2	75.28	76.73	83.35	83.14
Average	69.55	75.21	71.63	**78.26**

This is because their effects are affected by the choices of p and the kernel, which are selected using cross validation on source data. We also observe occasional inferior performance of DNFC and the variants to mSDA. This may be again due to the parameter selection on source data, which might result in overfitting when generalizing to target domain.

5 Conclusion

In this paper, we propose a deep nonlinear feature coding framework for unsupervised domain adaptation. It incorporates MMD and kernelization into mSDA to extract nonlinear deep features with minimum domain divergence. Extensive experimental studies show that DNFC achieves very promising results. It consistently and significantly outperforms all baselines in terms of prediction accuracy. Compared to the best baseline mSDA, DNFC attains up to 14% better accuracy and converges much faster in terms of the number of stacked layers. The effects of MMD and kernelization on the performance of DNFC are also investigated. It is shown that DNFC benefits from both elements, as well as their joint effects. All the results demonstrate that DNFC is an effective and promising solution to unsupervised domain adaptation.

Acknowledgments

This work was done within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

References

[Ben-David *et al.*, 2007] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[Blitzer *et al.*, 2007] John Blitzer, Mark Dredze, Fernando Pereira, et al. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.

[Chen *et al.*, 2012] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *29th International Conference on Machine Learning (ICML)*, 2012.

[Dai *et al.*, 2007] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219. ACM, 2007.

[Deng *et al.*, 2013] Jun Deng, Zixing Zhang, Erik Marchi, and Bjorn Schuller. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 511–516. IEEE, 2013.

[Ding *et al.*, 2015] Zehngming Ding, Ming Shao, and Yun Fu. Deep low-rank coding for transfer learning. In *IJCAI - International Joint Conference on Artificial Intelligence*, 2015.

[Fernando *et al.*, 2013] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2960–2967. IEEE, 2013.

[Glorot *et al.*, 2011] Xavier Glorot, Antoine Bordes, and Yoshua 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.

[Gong *et al.*, 2012] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073. IEEE, 2012.

[Gretton *et al.*, 2006] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2006.

[Kandaswamy *et al.*, 2014] Chetak Kandaswamy, Lynette M Silva, Luís A Alexandre, Ricardo Sousa, Jorge M Santos, and Joaquim Marques de Sá. Improving transfer learning accuracy by reusing stacked denoising autoencoders. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 1380–1387. IEEE, 2014.

[Long *et al.*, 2013] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2200–2207. IEEE, 2013.

[Long *et al.*, 2014] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer joint matching for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1410–1417. IEEE, 2014.

[Margolis, 2011] Anna Margolis. A literature review of domain adaptation with unlabeled data. *Rapport Technique, University of Washington*, page 35, 2011.

[Niyogi, 2004] X Niyogi. Locality preserving projections. In *Neural information processing systems*, volume 16, page 153. MIT, 2004.

[Pan and Yang, 2010] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[Pan *et al.*, 2011] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, 22(2):199–210, 2011.

[Wang *et al.*, 2015] Wei Wang, Hao Wang, Chen Zhang, and Fanjiang Xu. Transfer feature representation via multiple kernel learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Zhou *et al.*, 2014] Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W Tsang, and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[Zhuang *et al.*, 2015] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: transfer learning with deep autoencoders. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 4119–4125. AAAI Press, 2015.