Indian Statistical Institute, Kolkata

M. Tech. (Computer Science) Dissertation

Transductive Transfer Learning Using Autoencoders

A dissertation submitted in partial fulfillment of the requirements
for the award of Master of Technology
in
Computer Science

Author:
Sayontan Ghosh
Roll No: CS-1408

Supervisor:
Prof. Nikhil R. Pal
ECSU Unit, ISI

M.Tech(CS) DISSERTATION THESIS COMPLETION CERTIFICATE

Student: Sayontan Ghosh (CS1408)

Topic: Transductive Transfer Learning Using Autoencoder

Supervisor: Prof. Nikhil R. Pal

This is to certify that the thesis titled "***Transductive Transfer Learning Using Autoencoder***" submitted by **Sayontan Ghosh** in partial fulfillment for the award of the degree of Master of Technology is a bonafide record of work carried out by him under my supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission. The results contained in this thesis have not been submitted to any other university for the award of any degree or diploma.

Date: Prof. Nikhil R. Pal

Dedication

To my parents and my well wishers, without your help and encouragement it would not have been possible.

Acknowledgements

I would like to thank my dissertation supervisor Prof. Nikhil R. Pal for agreeing to guide me and for helping me to undertake work in the topic.

Abstract

A major assumption, that traditional machine learning algorithms make, is that training and test data come from the same domain. In other words, these data are represented in the same feature space and follow the same data distribution. However, in a real world scenario, this assumption may be violated due various reasons. These reasons include different marginal distributions, different feature spaces, different predictive distribution and different label spaces of the source and target domain datasets. In these kind of scenarios, a special learning startegy, called transfer learning is useful. Transfer leanring gains knowledge while performing one task, and then applies that knowledge to improve the performance of a different but related task.

In this thesis, we will specifically deal with transductive trasnfer learning. In this setting, a labelled source domain dataset and an unlabelled target domain dataset is available. Moreover, both the domains have the same feature space but follow different marginal distributions. Our aim is to maximize the classification accuracy on the target domain. To accompolish this task, we propose two methods using autoencoders. The first method is a supervised one. In this strategy, we try to extract features which not only encodes infor-

mation common to both the domains but also have discriminating power for the source domain. In the second method, in an unsupervised fashion we try to get good representation for target domain that is close to source domain. To achieve this, at first we train an autoencoder on the source datset. After that, we train another autoencoder on the target dataset that is similar to the previously trained autoencoder in terms of both weights and biases. We have tested our methods on three dataset of different type to show their generic nature. We also analyze our methods by discussing the pros and cons associated with them. We at last provide some ideas to improve their performance further.

Contents

Introduction

Data mining and machine learning algorithms work under the assumption that the test and the training data, both belong to the same space and have the same conditional density distribution; but this assumption need not always hold true. More so in todays world where there is an abundance of data and sometimes the data come from similar but different sources, contradiction of this assumption has become more prominent with time. Often data get generated at multiple related sources without much additional cost, but annotaion/labelling of such data requires manual intervention and usually it is both time consuming and expensive. So a method for ***Transfer learning or domain adpatation*** between related domain tasks can be quite helpful.

There are a number of real life scenario where transfer learning can be helpful. One example [18] is the problem of image classification, where we have to classify images taken from different devices into one of the many classes. Suppose we have trained a classifier to classify images taken by webcams. Can we use the same classifier to classify images taken by DSLR cameras. The answer is expected to be no, as images from DSLR and webcam have different processes that generate the images, so a classifier, which is trained on certain features that are specific to webcam domain, may (usually

will) give poor result when tested on data having features which are specific to DSLR cameras.

Another example where transfer learning is useful is the case of brain computer interface [9]. If a system for predicting wether a person will get a migrain attack in the next one hour is trained using the EEG signal of one person is applied to another person, it may (should usually) give poor results as different individuals usually have some characteristics of brain wave which are different. Altough training one calssifier for each individual is a soulution, it is an expensive process both for the designer of the classifier as well as the individual on which the classifier is being trained. Moreover, sometimes this may not even be feasible.

A third example [17] can be seen in the case of labelling of pixels in remotely sensed images. It is quite expensive to label a remotely sensed image at the pixels level as there may be millions pixel in it. Also when the sattelite files over the same area again the weather might not be the same and thus prediction may degrade over time. So, transfer learning could be usedful when a classifier trained on the images taken during one season is to be tested on images taken during different season .

In this thesis we propose a few methods to tackle the problem of transfer learning in a transductive setting using autoencoder. Our objective is to find a representation for source and target data in a new latent space such that distance between the distributions of source and target data in the new space is less then that in the original space. This enables us to effectively use the labelled source domain data for training the classifier and then test it on target domain data.

Chapter 1

Preliminary Knowledge

In this chapter we will review some topics which are required to understand the proposed methods.

1.1 Transductive transfer learning

In this section, we present the notations and formal definations related to transductive transfer learning. The problem of transductive transfer learning can be formally defined [15] as,
given

- a set of labelled source domain data points $\mathbf{X_S} = \{(\mathbf{x_s}^1, \mathbf{y_s}^1), (\mathbf{x_s}^2, \mathbf{y_s}^2), \ldots, (\mathbf{x_s}^m, \mathbf{y_s}^m)\}$ which are drawn i.i.d from a distribution $\mathcal{P}_s(\boldsymbol{x})$; $\mathbf{x_s}^i \in \mathcal{X}_s \ \forall \ i \in \{1, 2, \ldots, m\}$, $\mathbf{x_s}^i \in \mathbb{R}^p$.

- a set target domain data points $\mathbf{X_T} = \{(\mathbf{x_t}^1), (\mathbf{x_t}^2), \ldots, (\mathbf{x_t}^n)\}$ which are drawn i.i.d from a distribution $\mathcal{P}_t(\boldsymbol{x})$ such that the target domain

class labels are not known at the time of training ; $\mathbf{x_t}^i \in \mathcal{X}_t \ \forall \ i \in \{1, 2, \ldots, n\}$, $\mathbf{x_t}^i \in \mathbb{R}^q$.

If either of the following conditions is satisfied

- $\mathcal{X}_s \neq \mathcal{X}_t$
- $\mathcal{X}_s = \mathcal{X}_t$ but $\mathcal{P}(X_s) \neq \mathcal{P}(X_t)$

then the aim of transductive trasnfer learning is to train a classifier to maximize the classification accuracy on the target domain. Here we assume $p = q$ and also both source and target domains are repersented by the same set of features.

Some of the approaches used for solving this problem are [15]

- **Instance transfer** [15] In this method some of the labelled data in the source domain are reweighted by methods such as instance sampling and importance sampling for use in target domain .
- **Feature representation transfer** [15] Here the main aim is to learn a good representation for the target domain. Knowledge to be transfered across the domains are encoded in the learned feature representation which is expected to improve the target classification task.

There are three major issues which are to be considered in transfer learning [15]

- **What to transfer** which takes into consideration which part of the knowledge is to be transfered across domains as knowledge can be either specific to a particular domain or can be common to both the domains.

- **How to transfer** which deals with devising a learning algorithm which transfers the knowledge across the domain after it decides what knowledge is to be transfered.

- **When to transfer** asks in which scenario knowledge should be transferd and when it should not be transfered as in some situations source and the target domains may be totally unrelated and forcibily trying to transfer knowledge might lead to negetive transfer, which actually degrades the performance on target domain.

1.2 Autoencoder

An autoencoder is a simple multilayer perceptron having an input layer, an output layer and one or more hidden layers which tries to learn the identity function of the input. It consists of two parts, the encoder and the decoder which can be defined as follows:
Given an input data set $\mathbf{X}$ belonging to source domain space $\mathcal{X}$ the encoder and decoder are mapping functions ϕ and ψ such that

$\phi \ : \ \mathcal{X} \rightarrow \mathcal{F}$

$\psi \ : \ \mathcal{F} \rightarrow \mathcal{X}$

$\underset{\psi,\phi}{\operatorname{argmin}} \left\| X - \hat{X} \right\|^2$ where $\hat{X} = \psi\left(\phi\left(X\right)\right)$.
Another choice of error function which can be minimized to train an autoencoder is the cross entropy between the input and the reconstructed input.

1.2.1 Notations and conventions

Here we will follow the setup as given by Nielsen in [14].

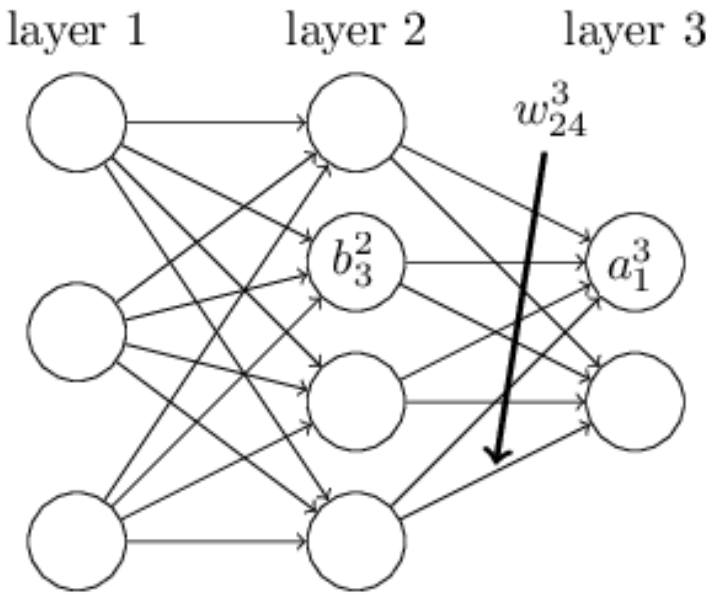

Figure 1.1: Neural network along with the notations used to refer the weights, biases and other network parameters

For the network given in Figure 1.1, notations followed in this thesis are given in Table 1.1

Table 1.1: Notations used for neural networks

w_{jk}^l	weight of the connection from the k^{th} neuron in the $(l-1)^{th}$ layer to j^{th} neuron in the l^{th} layer
b_t^l	bias of t^{th} neuron in the l^{th} layer
z_{ij}^l	weighted sum of inputs to the j^{th} neuron in the l^{th} layer for the i^{th} example, i.e., $\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$
a_{ij}^l	activation of j^{th} neuron in the l^{th} layer for the i^{th} example, i.e., $a_{ij}^l = \sigma\left(\sum_k w_{jk}^l a_{ik}^{l-1} + b_j^{l-1}\right)$, where $\sigma\left(z_j^l\right)$ is the activation function

When implementing an neural network in matrix form, the notations followed are given in table 1.2

Table 1.2: Notations used for neural network is implemented in matrix form

$\mathbf{W}^l$	weight matrix for layer l whose j^{th} row and k^{th} column is w_{jk}^l
$\mathbf{B}^l$	bias vector of the l^{th} layer
$\mathbf{Z}^l$	pre activation vector of the l^{th} layer
$\mathbf{A}^l$	activation vector of the l^{th} layer, $\mathbf{A}^l = \sigma\left(\mathbf{W}^l\mathbf{A}^{l-1} + \mathbf{B}^l\right)$, here σ is applied to each element of the vector

1.2.2 Training

The cost function we will use here to train an autoencoder is cross-entropy as it addresses the problem of learning slowdown to some extent [21].

Corss-entropy between two discrete distributions p and q over a set X is defined as

$$H(p,q) = -\sum_{x \in X} p(x) \log q(x) \tag{1.1}$$

So if we have data set of size n with input as $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ such that $\boldsymbol{x}_i \in \mathbb{R}^{d_1}$, $\boldsymbol{y}_i \in \mathbb{R}^{d_2}$ $\forall\ i \in \{1, 2, \ldots, n\}$ i.e $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id_1}]^T$, $\boldsymbol{y}_i = [y_{i1}, y_{i2}, \ldots, y_{id_2}]^T$ and we want to train a neural network, then the cost function to be minimized is

$$C = -\frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{d_2} \left[y_{ij} \ln a_{ij}^L + (1 - y_{ij}) \ln(1 - a_{ij}^L) \right]. \tag{1.2}$$

Here, a_{ij}^L is the output of the j^{th} unit of the final layer for the i^{th} input $\boldsymbol{x}_i$. without giving the complete derivation, the expression for partial derivative

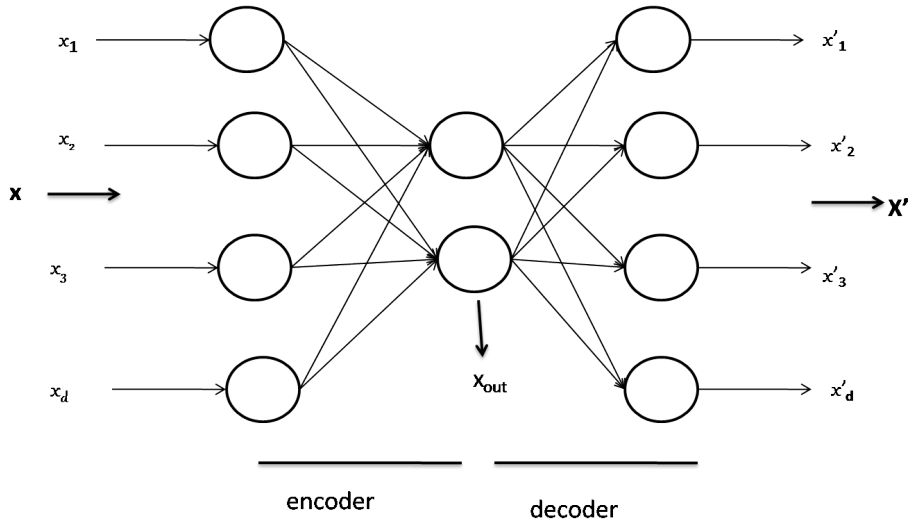

Figure 1.2: Autoencoder with $\boldsymbol{x}$ as input data, $\acute{\boldsymbol{x}}$ as the reconstructed input and $\boldsymbol{x}_{out}$ as the trasnformed data

of cost with respect to weights and biases of a network with L layer is given by

$$\frac{\partial C}{\partial w_{jk}^L} = \frac{1}{n}\sum_{i=1}^{n} a_{ik}^{L-1}(a_{ij}^L - y_{ij}), \tag{1.3}$$

$$\frac{\partial C}{\partial b_j^L} = \frac{1}{n}\sum_{i=1}^{n}(a_{ij}^L - y_{ij}). \tag{1.4}$$

Now, if for i^{th} training example with cost C_i

$$\delta_{ij}^l = \frac{\partial C_i}{\partial z_{ij}^l} \tag{1.5}$$

then

$$\frac{\partial C}{\partial w_{jk}^l} = \sum_{i=1}^{n} a_{ik}^{l-1}\delta_{ij}^l, \tag{1.6}$$

$$\frac{\partial C}{\partial b_j^l} = \sum_{i=1}^{n} \delta_{ij}^l. \tag{1.7}$$

So the updation rule for the weights and the biases for the neural network are

$$w_{jk}^l(new) = w_{jk}^l(old) - \eta\frac{\partial C}{\partial w_{jk}^l(old)}, \tag{1.8}$$

$$b_j^l(new) = b_j^l(old) - \eta\frac{\partial C}{\partial b_j^l(old)}. \tag{1.9}$$

The partial derivatives of cost with respect to weights and biases of non penultimate layers are obtained by backpropagating the error from the l^{th} layer to the $l-1^{th}$ layer as done in a multilayer perceptron, so it's derivation is skipped here. Note that for an autoencoder y_{ij} is nothing but the normalized input x_{ij} and a_{ij}^L is the output computed by the j^{th} output node, which we denote by $\hat{x}_{ij}$ for an autoencoder. The training process of an autoencoder is summarized in Algorithm 1.

Algorithm 1: Autoencoder training algorithm

Data: Input data $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$

Result: Autoencoder trained for input data

```
Initialize the MLP;
while Termination condition is not met do
    foreach x_i ∈ X do
        Do a feed-forward pass to compute activations at all hidden
         layers;
        At the output layer compute the output for x_i;
        Measure the instantenous cost function;
        Backpropagate the error and update the weights and biases
         using the equations mentioned;
    end
end
```

A major problem that we face while training an autoencoder is overfitting where instead of learning a compressed representation of the input, the network tends to memorize a mapping of input to itself. Few of the methods which are used to prevent overfitting in an autoencoder are :

- **Using tied weights** [13], constraining the weights of the decoder to be the transpose of the weights of the encoder reduces the degree of freedom and prevents memorization.
- **Using L1 or L2 regularization** [14] on the weights penalizes weights of the network to be small. Additionally L1 regularization tends to induce sparsity in the network as reduces some of the weights to zero and shrinks rest of the weights also by same amount.
- **Using Drouput** [19], where we randomly select some of the neurons which are not in the input and the output layers and carry on the training as if these neurons do not exist in the network and update the weights. Then we restore the deleted neurons and again repeat the same preocess.
- **Using Denoising autoencoder** [13], where the network is trained to reconstruct the input from a corrupted version of it. Corruption can be a additive stochastic corruption like, randomly masking entries of the input by making them zero.

Chapter 2

Related work

2.1 Related work

The initial motivation for study of transfer learning in the field of machine learning was possibly first discussed in a 1995 NIPS workshop on "Learning to Learn" which discussed the concept of never ending learning where knowledge gained from past experience could be utilised in the future. Feature based methods are widely exploited in the field of transfer learning. The idea for getting a shared representation for related tasks can be traced back to the work of Ando el al. [2]; Argyriou et al. [3] where shared feature representation is exploited where as the additional tasks are used as an inductive bias while learning. In multiple domain learning, Blitzer et al. [4] described a heuristic to construct new feature representations of the data for transfer learning.

In self-taught learning where unlabelled source data are used to improve the supervised classification performace on labelled target data set in [16] first a high-level set of bases is learned from an unlabeled data set which may have

different labels from the labeled data, and then labelled data is projected on the bases to get new feature representations for the target data's classification problem.

Dimensionality reduction approaches have been widely studied for transfer learning problem, but they may not be very effective for transfer learning as there is no gaurentee that the distribution of both source and target domain data in the reduced latent space will be similar. More recent dimensionality reduction technique modified to be used for transfer learning used the maximum variance unfolding (MVU) (Weinberger et al. [22]), which is motivated by designing kernels for kernel principal component analysis (KPCA) from the data itself. Low-dimensional feature representation of the data is extracted by MVU to maximize the variance of the embedding while preserving the local distances between neighboring data points. MVU can be formulated as a semidefinite programming (Lanckriet et al. [12]) optimization problem and solved by general optimization packages. After getting an estimate of the kernel matrix K, PCA is applied to the kernel matrix K to choose the top few eigen vectors on which the original data are projected to get a low dimensional feature representation.

One important problem in trasnfer learning is to get a representation minimizing the distance between two distributions, i.e., minimize the ***Kullback Leibler divergence***. But the problem requires intermediate density eastimation. So to tackle this problem which is expensive, a nonparametric eastimate of the distance between two distributions is required. ***Maximum Mean Discrepancy (MMD)*** is one such distance measure for comparing distributions based on Reproducing Kernel Hilbert space (Borgwardt et al. [8])

(*RKHS*). Say $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_1}]^T$ and $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{n_2}]^T$ are independently identicall distributed random variable, such that $X \sim \mathcal{P}$ and $Y \sim \mathcal{Q}$ then the eastimate of distance between X and Y using MMD can be defined as [8]

$$\mathcal{D}(X, Y) = \sup_{||f||_{\mathcal{H}} \leq 1} \left(\frac{1}{n_1} \sum_{i=1}^{n_1} f(x_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} f(y_i) \right)$$

where $\mathcal{H}$ is a universal RKHS ([20]) with the following constrains

(i) $\mathcal{D}(X, Y) \geq 0$;

(ii) $\mathcal{D}(X, Y) = 0$ iff $\mathcal{P} = \mathcal{Q}$, given $n_1, n_2 \to \infty$

Based upon the fact that in a RKHS $f(\boldsymbol{x})$ can be written as $f(\boldsymbol{x}) = \langle \phi(\boldsymbol{x}), f(\boldsymbol{x}) \rangle$ where $\phi : \mathcal{X} \to \mathcal{H}$ the expression for MMD between X and Y can be rewritten as

$$\mathcal{D}(X, Y) = ||\frac{1}{n_1} \sum_{i=1}^{n_1} \phi(\boldsymbol{x}_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(\boldsymbol{y}_i)||_{\mathcal{H}}$$

So basically MMD theory ([8]) states that distance between distributions of two samples is the distance between the mean of the two samples in RKHS.

Chapter 3

Transfer learning through simulteneous autoencoding and classification

Our proposed method for dealing with the problem of transfer learning is mainly based on an autoencoder trained on both source and target data which also integrates some discriminatory information from the source data. Thus it simultenously integrates a supervised method with an unsupervised feature extraction scheme. We shall discuss the pros and cons related to each method and also suggest few modifications in the methods which might lead to improvement.

To state the problem specific to our work, we refer back to Subsection 2.1 for notation. Let $\mathcal{X}_s$ and $\mathcal{X}_t$ be the source and target input space such that $\mathcal{X}_s \subset \mathbb{R}^{d_1}$, $\mathcal{X}_t \subset \mathbb{R}^{d_1}$ but $\mathcal{P}_s(\boldsymbol{x}) \neq \mathcal{P}_t(\boldsymbol{x})$. Our aim is to design a classifier with good performance on the target data X_t.

3.1 Feature extraction

Most works on transfer learning state that while transferring knowledge across domains, domain specific information should be removed while information common to both domain should be used to train the classifier. However we think that transfer of domain specific information to certain extent may be benificial for transfer learning task. Further all common attributes may not contribute equally in discriminating between the classes as represented by the souce data. Consequently those common attibutes that have better discriminating power for the source data should get heigher importancewhile transferring knowledge. So for this method we perform feature extraction such that the extracted feature tries to primarily encode the domain invariant information but it also exploits some class specific source domain information.

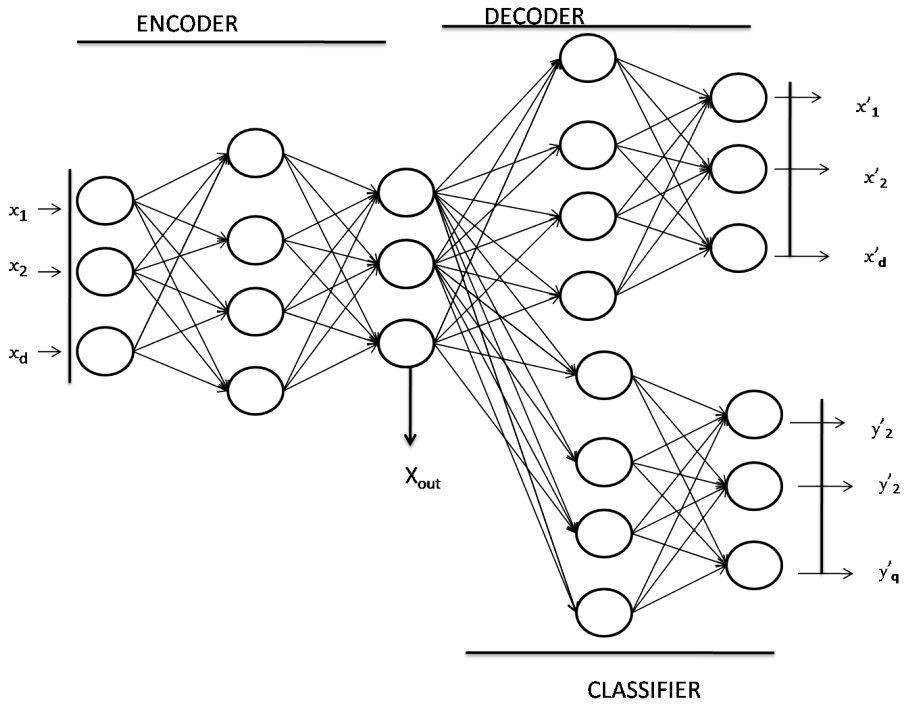

Figure 3.1: Neural network architecture for simulteneous autoencoding and source classification. For $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T$ as the input $\hat{\boldsymbol{x}} = [\acute{x}_1, \acute{x}_2, \ldots, \acute{x}_d]^T$ is the reconstructed output of the autoencoder, $\boldsymbol{y} = [y_1, y_2, \ldots, y_q]^T$ is the class label of $\boldsymbol{x}$ and $\hat{\boldsymbol{y}} = [\acute{y}_1, \acute{y}_2, \ldots, \acute{y}_q]^T$ is computed output by the classifier model. $\boldsymbol{x}_{out}$ is the output from the encoder layer (here the third layer).

To fullfill these requirements we use a neural network architecture as shown in Figure 3.1. In this network, the autoencoder is augmented with a classifier module which receives input from the encoder layer. Our objective is to find a suitable latent space (represented) by the autoencoder layer, which does a good job of encodingof both source and target data and at the same time has some discriminating power for the source data. To achieve this, the proposed network in 3.1 attempts to minimize the following cost function :

$$C(W;(X_S,Y_S),X_T) = -\frac{\alpha}{2(m)}\sum_{i=1}^{m}\sum_{j=1}^{d_2}\left[y_{ij}\ln a_{ij}^{L} + (1-y_{ij})\ln(1-a_{ij}^{L})\right] +$$

$$\frac{\lambda}{2(n+m)}\sum_{i,j,l} {w_{ij}^{l}}^{2} - \frac{\beta}{2(n+m)}\sum_{i=1}^{(m+n)}\sum_{j=1}^{d_1}[\hat{x}_{ij}\ln x_{ij} + (1-\hat{x}_{ij})\ln(1-x_{ij})] \tag{3.1}$$

In Figure 3.1, W is a vector of all learnable weights and biases of the network.

Feature extraction using the aforementioned transfer learning method is summarized as an algorithm next.

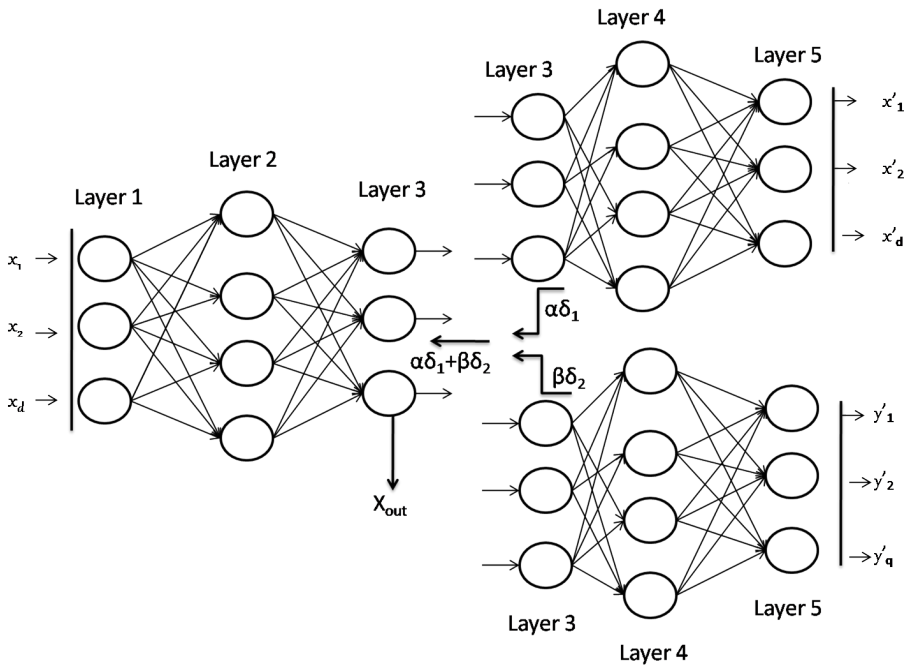

Figure 3.2: Flow of error in the network architecture as shown in Figure 3.1 for minimizing C in 3.1. Here δ_1 is the error derivative backpropagated due to the autoencoding error and δ_2 is the error derivative backpropagated due to classification error

Algorithm 2: Autoencoder training algorithm

Data: Labelled source domain data $X_S = \{\boldsymbol{x}_{S1}, \boldsymbol{x}_{S2}, \ldots, \boldsymbol{x}_{Sm}\}$ and unlabelled target domain data $X_T = \{\boldsymbol{x}_{T1}, \boldsymbol{x}_{T2}, \ldots, \boldsymbol{x}_{Tn}\}$ such that $\boldsymbol{x}_{Si} \in \mathbb{R}^{d_1}$ and $\boldsymbol{x}_{Tj} \in \mathbb{R}^{d_1}$, initial value of α, β and λ as α_o, β_o and λ_o respectively

Result: A unified encoded representation for both source and target data as $\hat{X}_S$ and $\hat{X}_T$ in a latent space, such that encoded representation has enough information to reconstruct both source and target data as well as it has adequate discriminatory power

Initialize the MLP;

Mix the source and target domain data such that $X_N = X_S \bigcup X_T$

while *Termination condition is not met* **do**

  foreach $\boldsymbol{x}_i \in X_N$ **do**

$$\frac{\partial C}{\partial w_{jk}^L} = -\frac{\alpha}{n} \sum_{\boldsymbol{x}_i \in X_S} a_{ik}^{L-1}(\acute{y}_{ij} - y_{ij}) + \frac{\lambda}{(m+n)} w_{jk}^L - \frac{\beta}{(n+m)} \sum_{\boldsymbol{x}_i \in X_N} a_{ik}^{L-1}(\acute{x}_{ij} - x_{ij}) \tag{3.2}$$

$$\frac{\partial C}{\partial b_j^L} = -\frac{\alpha}{n} \sum_{\boldsymbol{x}_i \in X_S} (\acute{y}_{ij} - y_{ij}) - \frac{\beta}{(n+m)} \sum_{\boldsymbol{x}_i \in X_N} (\acute{x}_{ij} - x_{ij}) \tag{3.3}$$

$$\delta_{ij}^L = \frac{\partial C_i}{\partial z_{ij}^l} \tag{3.4}$$

$$\frac{\partial C_i}{\partial w_{jk}^l} = \sum_{i=1}^{n} a_{ik}^{l-1} \delta_{ij}^l, \tag{3.5}$$

$$\frac{\partial C}{\partial b_j^l} = \sum_{i=1}^{n} \delta_{ij}^l. \tag{3.6}$$

$$w_{jk}^l(new) = w_{jk}^l(old) - \eta \frac{\partial C}{\partial w_{jk}^l(old)}, \tag{3.7}$$

$$b_j^l(new) = b_j^l(old) - \eta \frac{\partial C}{\partial b_j^l(old)}. \tag{3.8}$$

  end

3.2 Designing the classifier

Let S_i be the i^{th} architecture selected from a set of architechtures $\mathbf{S}$ and α_j is the j^{th} value of α selected from set of values $\boldsymbol{\alpha}$. We denote a pair of architecture and α by (S_i, α_j). For each (S_i, α_j) we can get a unique set of features. As for selecting the optimal architecture, some labelled validation data are required which is not present in our problem setting. Therefore, selecting a single architecture optimal for our task is not possible. Infact it was experimentally seen that if we use the feature corresponding to each (S_i, α_j) pair and design a classifier using them, then there are many (S_i, α_j) pairs which, if used, would have given better result than our proposed methods but there is no systematic way to know which particular (S_i, α_j) pair would yield the best result).

Method 1 In this method we consider, only set of features generated by (S_i, α_j) pair for which $\alpha_j = 0$. This method is primarily used to evaluate the effectiveness of the proposed method where, we use $\alpha \neq 0$ in order to inject the class specific source data information. In the porposed method, we assume that after the transformation, source and the target domain are closer then they originally were. So it can be expected that preformance of transformed target data would be similar to that of the source data. So we choose the (S_i, α_j) pair for which the source domain accuracy is maximum. Then using the source feature set generated by that (S_i, α_j) a classifier is trained. Then using the target feature set corresponding to that particular (S_i, α_j) pair as the test set, the classifier previously trained is tested .

Method2 This method is similar to method one. The only change made here is that, the constrain that α_j has to be zero is removed, i.e., we use $\alpha \neq 0$. This allows us to check the effectiveness of adding the supervised learning part in cost function.

3.3 Experimental result

For the autoencoder used for feature extraction, we use autoencoder with three hidden layer and sigmoidal activation function. Weights of the MLP are initialized as Gaussian random variables with mean 0 and standard deviation $1/\sqrt{n_{in}}$ where n_{in} is the number of input weights [14]. Biases are initialized as a Gaussian with mean 0 and standard deviation 1 inorder to prevent the saturation of neurons. The input data $(X_S \bigcup X_T)$ for the autoencoder was normalized between 0 and 1 by using a common transformation, where the minimum and the maximum values used were obtained from $X_S \bigcup X_T$.
For classifier we will be using of Support Vector Machines with linear kernel. The grid used for grid search for Linear kernel was same for all the tasks. The parameter **C** for the SVM was obtained after performing 5 fold cross validation on the transformed source domain data. Then using the parameter obtained from the crossvalidation we train a classifier on transformed source domain data. This classifier is then tested using the transformed target domain data to get the accuracy. The result reported here are the mean of accuracy obtained after repeating the experiment five times

3.3.1 Reuters dataset

The original Reuters 21578 dataset was collected for the purpose of text categoriztion, i.e., deceiding weather a document belongs to any of the set of prespecified categories. The documents used in the Reuters 21578 dataset were collected from the Reuters newswire of 1987. Here we are using the already preprocessed data in (refer the source). The dataset used here are catergorized into top categories which are further subdivided into subcategories. For the creation of source and target domain, data from the same top category but different subcategories are treated to be different domains. Here the three top categories are ***orgs, people and places***.

Here we took MLP with three hidden layers and fixed the values of $\alpha = 1$, $\lambda = 0.02$ and used the values of β as [0.0,.1, .2, .4, .8, 1.2, 1.6, 2.0]. Due to computational constrain, encoder architecture which were used for feature extraction are [[d_1, 90, 30], [d_1, 90, 32], [d_1, 90, 34], [d_1, 90, 36], [d_1, 90, 38], [d_1, 90, 40], [d_1, 90, 50]], where d_1 is the dimension of the input data. The result obtained using this method along with other methods is given in table 3.1

3.3.2 Office dataset

Office dataset is an image dataset used for multiclass transfer learning task. It consists of images belonging to one of the 31 different categories such as helmet, bike etc generated by three differnet sources DSLR, Webcam and images collected from amazon website. Here images generated by each source can be considered as a distinct domain and the task is given labelled images

Dataset	SVM	TSVM [11]	C^3E [1]	RTriTL [24]	Method 1	Method 2
Orgs-People	72.86	76.94	81.81	**81.88**	74.37 ± 0.32	76.28 ± 0.48
Orgs-Places	65.15	70.08	68.92	**78.95**	65.78 ± 0.21	68.78 ± 0.98
People-People	54.69	59.72	68.61	**69.68**	58.46±.32	68.46 ± .98

Table 3.1: Reuter datset accuracy along with other methods

from one of the domains, predict the categoy of images from a different domain.

As done by other investigators, SURF Bag of words histogram features of images, vector quantized to 800 dimension as given in [18] are used. Like the reuter data set we normalize data to values between 0 and 1.

Here we took MLP with three hidden layers, fixed the value of $\alpha = 1$, $\lambda = 0.02$ and used the values of β as [0.0,.1, .2, .4, .8, 1.2, 1.6, 2.0]. Due to computational constrains, encoder architecture which were used for feature extraction are [[d_1, 100, 40], [d_1, 100, 50], [d_1, 100, 70], [d_1, 100, 80], [d_1, 100, 90]], where d_1 is the dimension of input data. The result obtained using this method along with other methods are given in Table 3.2

Dataset	Gopalan et al.[7]	Gong et al. [6]	Jhu et al. [10]	Method 1	Method 2
Amazon-Webcam	39±2.0	15±0.4	50.71±0.8	17.39 ± .29	19.26±1.04
DSLR-Webcam	26±0.8	44.6±0.3	36.85±1.9	20.64± .39	23.47±.85
Webcam-DSLR	19±1.2	49.7±0.5	32.89± .69	31.37± .46	16.96 ± 1.24

Table 3.2: Office datset accuracy along with other methods

3.4 Some remarks

The result obtained using this method shows that it is not always necessary that, injection of domain specific knowledge would deteriorate the transfer learning perfromance. Infact careful transfer of domain specific knowledge might sometimes be beneficial for transfer learning task. But how much to transfer and when to transfer are two crucial questions that need to be answered to make an effective use of this method. Furthermore when ratio of number of data points in source to target domain is very high or vey low then there is a high chance that the MLP would be more biased towards learning the distribution of a particular domain.

Chapter 4

Transfer learning by mirroring autoencoder

Here we present another method for improving transductive transfer learning using unsupervised feature extraction . We will divide the task into two sub tasks

1. Extaction of features
2. Designing the classifier

4.1 Extaction of features

This method for feature extraction for transfer learning is based on Feature representation principle where we will try to get a good representation for target domain which is close to source domain. So we want our features to be extracted in such a way that they encode information related to the target

domain as well as resemble the source domain which we try to accomplish using autoencoder.

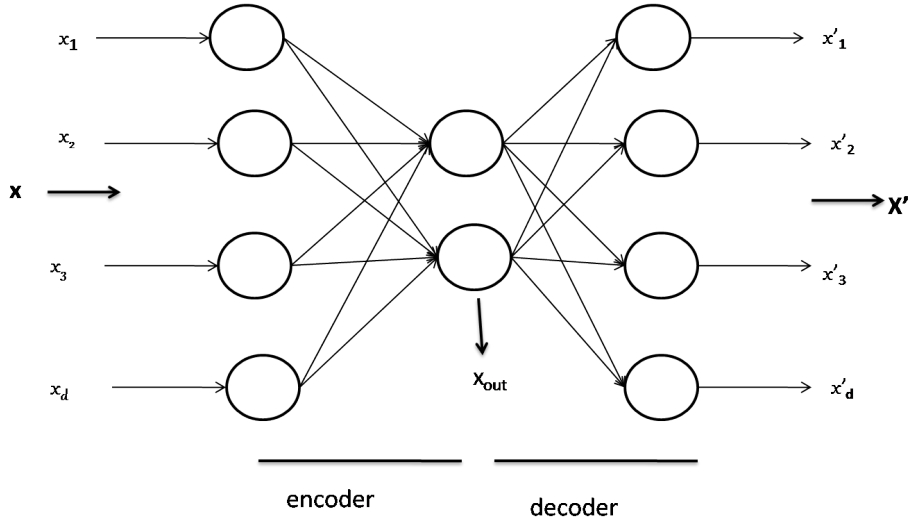

Figure 4.1: Autoencoder with $\boldsymbol{x}$ as input data, $\acute{\boldsymbol{x}}$ as the reconstructed input and $\boldsymbol{x}_{out}$ as the trasnformed data

As autoencoder trained on a data following a distribution encodes the charecteristics of the distribution of data in the weights and the biases of the network, so inspired by the work of Hal Daumé III in [5] for supervised transfer learning, we use a regularizer in the unsupervised feature extraction phase. Here first we train an autoencoder on the source data and then train a seperate autoencoder on the targer data such that it is similar to the autoencoder trained on source domain in terms of weights and biases. Feature, extracted by the target domain autoencoder will have, as a result of this, both the information about the target as well as the source distribution encoded in it.

To state the problem specific to our paper, referring back to Subsec-

tion 1.1: $\mathcal{X}_s$ and $\mathcal{X}_t$ are the source and target input space such that $\mathcal{X}_s \subset \mathbb{R}^{d_1}$, $\mathcal{X}_t \subset \mathbb{R}^{d_1}$ but $\mathcal{P}_s(\boldsymbol{x}) \neq \mathcal{P}_t(\boldsymbol{x})$, our aim is to improve the classification accuracy on X_t when no labelled data are available for X_t. To accomplish this we add a regularizer $\lambda \left[\sum_{i,j,l} \left(w_{ij}^{l(t)} - \acute{w}_{ij}^{l}\right)^2\right] + \left[\sum_{i,l} \left(b_i^{l(t)} - \acute{b}_i^{l}\right)^2\right]$ along with the normal cross entropy cost function where $(w_{ij}^{l(t)}, b_i^{l(t)})$ and $(\acute{w}_{ij}^{l}, \acute{b}_i^{l})$ are the weights and biases of the autoencoder trained on target domain and terminal weights of autoencoder trained on source domain repspectively. So the algorithm for our proposed method is stated as Algorithm 3 next.

Algorithm 3: Feature extraction algorithm

Data: Labelled source domain data $\mathbf{X_S}$ and an unlabelled target domain data $\mathbf{X_T}$

Result: Transformed source and target domain data $\acute{\mathbf{X}}_\mathbf{S}$ and $\acute{\mathbf{X}}_\mathbf{T}$ with a hope that $D(\mathcal{P}_s(\boldsymbol{x}), \mathcal{P}_t(\boldsymbol{x}) < D(\acute{\mathcal{P}}_s(\boldsymbol{x}), \acute{\mathcal{P}}_t(\boldsymbol{x}))$ where $D(P_1, P_2)$ is the distance between two distributions P_1 & P_2.

1 Train an autoencoder A_S with weights and biases ($w^{(s)}$, $b^{(s)}$) on the source domain data to minimize the cost function

$$\begin{aligned} C_S = & -\frac{1}{2m}\sum_{i=1}^{n}\sum_{j=1}^{d_1}\left[x_{sj}^i \ln a_{ij}^L + (1-x_{sj}^i)\ln(1-a_{ij}^L)\right] \\ & + \frac{\lambda}{2m}\sum_{i,j,l} {w_{ij}^{l(s)}}^2 \end{aligned} \tag{4.1}$$

Let the weights and biases that minimize C_S be $\acute{w} = \left\{\acute{w}_{ij}^l\right\}$ and $\acute{b} = \left\{\acute{b}_i^l\right\}$;

2 Initialize an autoencoder A_T with weight and bias set $(\acute{w}, \acute{b})$ such that the initial weights and bias of A_T are equal to the final weights and bias of A_S;

3 Train A_T on the target domain data to minimize the cost function

$$\begin{aligned} C_T = & -\frac{1}{2n}\sum_{i=1}^{m}\sum_{j=1}^{d_1}\left[x_{tj}^i \ln a_{ij}^L + (1-x_{tj}^i)\ln(1-a_{ij}^L)\right] \\ & + \frac{\lambda_1}{2n}\sum_{i,j,l} {w_{ij}^{l(t)}}^2 + \frac{\lambda_2}{2n}\sum_{i,j,l}\left(\acute{w}_{ij}^l - w_{ij}^{l(t)}\right)^2 \\ & + \frac{\lambda_2}{2n}\sum_{i,l}\left(\acute{b}_i^l - b_i^{l(t)}\right)^2 \end{aligned} \tag{4.2}$$

;

4 Feedforward the source and target data X_S and X_T as input to the autoencoder A_T and extract the output $\acute{X}_S$ and $\acute{X}_T$ from the encoder part of A_T to get the transformed data in the new latent feature space.;

4.2 Designing the classifier

After getting the source and traget data in the new feature space, as per our hypothesis, the distribution of $\acute{X}_S$ is expected to be closer to $\acute{X}_T$ than it originally was. So now to predict the class label of $\acute{X}_T$ we can train a classifier on $\acute{X}_S$ and test it on $\acute{X}_T$.

Let S_i be the i^{th} architecture selected from a set of architechtures $\mathbf{S}$ and λ_{2j} is the j^{th} value of λ_2 selected from a set of values $\boldsymbol{\lambda_2}$. We denote a pair of architecture and λ_2 by (S_i, λ_{2j}). Now for each (S_i, λ_{2j}) we can get a unique set of features. As for selecting the optimal architecture, some labelled validation data are required which is not present in our problem setting. Therefore, selecting a single architecture optimal for our task is not possible. Infact it was experimentally seen if we use the feature corresponding to each (S_i, λ_{2j}) pair and design a classifier using them, then there are many (S_i, λ_{2j}) pairs which, if used, would have given better result than our proposed methods (discussed below) but there is no systematic way to know which particular (S_i, λ_{2j}) pair would yield the best result.

(i) Method 2: Ensemble 1

Since the set of featues representation generated by different (S_i, λ_{2j}) pairs could be quite diverse in themselves, so in order to exploit this diversity ensemble method comes as a natural choice as it helps in reducing the variance in performance This is what we do, we create an ensemble of M*N classifier, where M is the number fo architectures and N is the number of λ_2 values tried. The algorithm for designing the classifier is stated in Algorithm 4.

Method 2: Ensemble 2

As per our method it is assumed that after the transformation, source and

Algorithm 4: Ensemble-1

Data: Labelled source domain data $\acute{\mathbf{X}}_{\mathbf{S}}$ and an unlabelled target domain data $\acute{\mathbf{X}}_{\mathbf{T}}$ in new latent space for each (S_i, λ_{2j}) pair

Result: Given an $\boldsymbol{x} \in \mathbf{X_T}$, predict the class label of $\boldsymbol{x}$

1 For each set of source features generated by a (S_i, λ_{2j}) pair, train a classifier C_{ij}; $i = 1, \ldots, M; j = 1, \ldots N$;

2 For each target domain data point, pass it's corresponding representation generated by (S_i, λ_{2j}) pair through the classifier C_{ij} and let the label predicted by it be L_{ij};

3 For a K class problem, let L_p be the number of classifiers that have predicted the class label of a target data point under consideration as $p \,\forall\, p \in \{1, 2, \ldots K\}$, $\sum_{p=1}^{K} L_p = MN$;

4 Class label of $\boldsymbol{x}^i$ will be assigned as l such that $l = \underset{p \in \{1,2,\ldots,K\}}{\operatorname{argmax}} L_p.$;

5 Ties can be broken randomly;

target domain data are more similar in terms of their distributions. As per our cost function, greater the value of λ_2, more source information is encoded in the extracted features, so if a classifier performs good on the source domain for a particular architecture with $\lambda_2 = 0$, then it will perform better for a higher value of λ_2. So instead of creating an ensemble for all the architectures, we create an ensemble of set of features corresponding to a particular architecture and all the values of λ_2 associated with that architecture. The architecture is selected gives highest accuracy on source for $\lambda_2 = 0$.

Method 3: Optimal

In this method we assume, we have an algorithm to choose the optimal (S_i, λ_{2j}) pair. Then we use the source domain dataset corresponding to that (S_i, λ_{2j}) pair to train the classifier, followed by testing on the target domain data generated by that (S_i, λ_{2j}) pair.

4.3 Experimental result

For the autoencoder A_S which is trained on the source dataset, we use one hidden layer autoencoder with sigmoidal activation function. Weights of the MLP are initialized as Gaussian random variables with mean 0 and standard deviation $1/\sqrt{n_{in}}$ where n_{in} is the number of input weight. Biases are initialized as a Gaussian with mean 0 and standard deviation 1 inorder to prevent the saturation of neurons. For the autoencoder A_T trained on the target domain dataset, the weights and the biases are initialized using the terminal weights and biases of A_S.

For designing the classifier we will be using an ensemble of Support Vector

Machines with linear kernel. We are using linear kernel. The grid used for grid search to find the best value of **C** was same for all the tasks. The parameter **C** for the SVM was obtained after performing 5 fold cross validation on the transformed source domain data and then using the optimal parameter obtained from the crossvalidation is used to train the classifier on transformed source domain using the entire source data. This classifier is then used to test the transformed target domain data. Note that for method 1 we get one classifier for each (S_i, λ_{2j}) pair. For method 2 The results reported here are the mean and the standard deviation of the accuracy obtained after repeating the experiment five times.

4.3.1 Reuters dataset

The information about the Reuters dataset is provided in Subsection 3.2.1. For creating an ensmeble of classifiers, set of architectures for autoencoder used here is $\mathbf{S} = \{[D_{in}, i, D_{in}]\ \forall i \in \{30, 32, 34, 36, 38, 40, 42, 44, 48, 50\}\}$, where $[D_1, D_2, \ldots, D_n]$ represents an autoencoder with D_1 neurons in it's input layer, D_2 in the first hidden layer and so on. In our experiment we used a three layer autoencoder and the set of parameter λ_2 used is $\boldsymbol{\lambda_2}$= { 0.0, 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0}. The result of the proposed method along with the results of other methods is tabulated in Table 4.1

Table 4.1: Reuter dataset accuracy along with results from other methods.

Dataset	TSVM [11]	C^3E [1]	RTriTL [24]	Ensemble 1	Ensemble 2	Optimal
Orgs-People	76.94	81.81	**81.88**	80.73±.65	79.86 ± .49	82.67 ± .36
Orgs-Places	70.08	68.92	**78.95**	71.68±.79	71.43 ± .59	73.85 ± .53
People-People	59.72	68.61	**69.68**	64.48±.69	65.23 ± 1.03	67.78 ± .41

We find that for Reuter datasets, both Ensemble 1 and Ensemble 2 perform consistently better than TSVM. Compared to C^3E, Ensemble 1 and Ensemble 2 both provide better result for Orgs-places. For the other two datasetC^3E is slightly better. However for the other two dataset RTriTL is much better than our method. On this context it is worth mentioning that RTriTL is a method specifically developed for text data.

4.3.2 Office dataset

The information about the Office dataset is provided in Subsection 3.2.2. For creating an ensmeble of classifiers, set of architectures for autoencoder used here is $\mathbf{S} = \{[D_{in}, i, D_{in}]\ \forall i \in \{40, 50, 60, 70, 80, 90, 100, 120, 150, 200\}\}$. As earlier $[D_1, D_2, \ldots, D_n]$ represents an autoencoder with D_1 neurons in it's input layer, D_2 in the first hidden layer and so on and the set of parameter λ_2 used is $\boldsymbol{\lambda_2}$= { 0.0, 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0}. The result of the proposed method is reported in table 4.2

Table 4.2: Reuter dataset accuracy along with results from other methods.

Dataset	Gopalan et al.[7]	Gong et al. [6]	Jhu et al. [10]	Ensmeble 1	Ensemble 2	Optimal
Amazon-Webcam	39 ± 2.0	15±0.4	$\mathbf{50.71 \pm 0.8}$	24.49±.65	21.89 ± .78	26.27± .36
DSLR-Webcam	26±0.8	$\mathbf{44.6 \pm 0.3}$	36.85±1.9	31.47±.43	31.65± .67	38.47± .52
Webcam-DSLR	19±1.2	$\mathbf{49.7 \pm 0.5}$	32.89±	45.09 ± .39	48.09 ± .53	52.85 ± .47

For the office dataset no single method is consistently better. Given, there is a mechanism for selecting the optimal (S_i, λ_{2j}) pair, there is a significant improvement in the performance. Infact our method outperform most of the methods on Webcam-DSLR data by large margin.

4.3.3 Landmine dataset

The remote sensing problem is based on the data collected from real landmines. In this problem there are total of 29 sets of data, collected from fields of different types. Each data is represented as a 9-dimensional feature vector extracted from radar images, and the class label is either true or false. Since each of the 29 datasets are from differnt terrains, each of the dataset corresponding to a terrain can assumed to follow a distinct distribution. As per the work reported in [23], dataset 1 to 10 are collected from foliated rocks while 20 to 24 are collected from bare earth region. Thus we combine dataset 1 to 5 to create the source domain dataset. Datasets 20 to 24 are combined to create the target domain dataset.

Since landmine datasets are imbalance in nature, i.e., number of positive cases is much less than number of negetive cases. So before testing this dataset using our proposed method for feature extraction, a prepocessing step is required. So firstly, in the source domain, we over sample the minority class data using the SMOTE algorithm. By using SMOTE we create D seperate minority oversampled datasets. Each of these D data sets are then fed to the network corresponding to (S_i, λ_{2j}) pair, let the tuple corresponding to the d^{th} oversampled dataset and (S_i, λ_{2j}) pair be denoted by (S_i, λ_{2j}, D_d).

Method: Bi level ensemble Now to design the classifier, we use a two level ensemble startegy. In the first level, an ensemble for each (S_i, λ_{2j}) is created, by training D classifier using source data generated by (S_i, λ_{2j}, D_k) $\forall k \in \{1, 2, \ldots D\}$ followed by majority voting. Then in the second level, ensemble of classifiers of all (S_i, λ_{2j}) pair is created followed by majority voting. Here we report the F1_score, recall score, precision score and the

G_score(geometric mean of precision and recall) for the target data using our method as well using SVM in Table 4.3. We can observe that our method

Table 4.3: Landmine dataset performance scores along with results from SVM

Method	F1_score	G_score	recall score	precision score
SVM	.0926	.1568	.2219	.1039
Bi level ensemble	.1973	.2285	.3928	.1971

has led to significant improvement in all the performance measure for this dataset.

4.4 Some remarks

Here we can see that the proposed method, lead to significant increase in the transfer learning performance for some data sets, but not for all. Experimental results also seems to solidify the hypothesis that the source domain information can be encoded in the feature extracted for target domain data by making the autoencoder trained on target domain similar to the autoencoder trained on source domain data in terms of weights and biases. This method also helps in increasing the diversity of classifiers used for creating ensemble of classifiers. As the autoencoders are trained seperately on source and target domain dataset, this method can easily handle the case when source and target data are highly disproportionate in terms of number of instances. Furthermore, this method is very general in nature, i.e., it does not uses any domain specific features and thus can be used for any kind of

dataset.

Chapter 5

Conclusion and future work

In this thesis we propose two types of schemes for transfer learning using autoencoder. The first type of scheme is more biased towards proving the hypothesis that encoding some class information in the extracted feature might lead to improved transfer learning performance. While the result tends to agree with the hypothesis for most of the data set there are certain data sets for which this methods leads to performance degradation. The main obstacle in effificiently using this method is knowing when injecting class information might lead to performance improvement and how much to inject. Other issues with this method are disproportionate source and target data size could lead to network being biased towards a particular domain and the scenerio where the source and target domain are extremely dissimilar. The second problem can be easily tackled by choosing the weights of the regularizing term.

In the second family of schemes, we try find domain invariant features by performing unsupervised feature extraction using autoencoders which are known

to be good at extracting latent hierarchial features. The approach is based on the hypothesis that similarity of the distribution between two datasets is "proportional" to the similarity of the autoencoders in term of the weights and the biases. Our results indeed show that this approach significantly can improve the transfer learning performance.

For further improving the performance, a function of distance between the source and target domains can be included as a regularizer in the cost function. Instead of using gradient descent based optimization schemes, 2nd order optimization methods which gives better mimima and takes lesser number of epochs to converge can be used. Due to the promising results shown by deep architectures, autoencoders having multiple hidden layers learned by sequential stacking of autoencoders using greedy method can be used. As autoencoders inherently learn hierarchical features, combination of intermediate representations can be used as features.

Bibliography

[1] Ayan Acharya, Eduardo R Hruschka, Joydeep Ghosh, and Sreangsu Acharyya. Transfer learning with cluster ensembles. *ICML Unsupervised and Transfer Learning*, 27:123–132, 2012.

[2] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[4] John Blitzer, Ryan McDonald, and Fernando 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.

[5] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[6] 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.

[7] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 international conference on computer vision*, pages 999–1006. IEEE, 2011.

[8] 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.

[9] Vinay Jayaram, Morteza Alamgir, Yasemin Altun, Bernhard Scholkopf, and Moritz Grosse-Wentrup. Transfer learning in brain-computer interfaces. *IEEE Computational Intelligence Magazine*, 11(1):20–31, 2016.

[10] I-Hong Jhuo, Dong Liu, DT Lee, and Shih-Fu Chang. Robust visual domain adaptation with low-rank reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2168–2175. IEEE, 2012.

[11] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[12] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004.

[13] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72:1–19, 2011.

[14] Micheal A. Nielsen. *Neural Networks and Deep Learning.*

[15] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[16] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[17] Suju Rajan, Joydeep Ghosh, and Melba M Crawford. Exploiting class hierarchies for knowledge transfer in hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 44(11):3408–3417, 2006.

[18] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

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

[20] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of machine learning research*, 2(Nov):67–93, 2001.

[21] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning

useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[22] Kilian Q Weinberger, Fei Sha, and Lawrence K Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 106. ACM, 2004.

[23] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan):35–63, 2007.

[24] Fuzhen Zhuang, Ping Luo, Changying Du, Qing He, Zhongzhi Shi, and Hui Xiong. Triplex transfer learning: exploiting both shared and distinct concepts for text classification. *IEEE transactions on cybernetics*, 44(7):1191–1203, 2014.