Feature Spaces-based Transfer Learning

Hua Zuo, Guangquan Zhang, Vahid Behbood, Jie Lu

Decision System & e-Service Intelligence Lab, Centre for Quantum Computation & Intelligent Systems,
Faculty of Engineering and Information Technology,
University of Technology Sydney, P.O. Box 123, Broadway NSW, Australia
Hua.Zuo@student.uts.edu.au, Guangquan.Zhang@uts.edu.au, Vahid.Behbood@uts.edu.au, Jie.Lu@uts.edu.au

Abstract

Transfer learning provides an approach to solve target tasks more quickly and effectively by using previously-acquired knowledge learned from source tasks. Most of transfer learning approaches extract knowledge of source domain in the given feature space. The issue is that single perspective can't mine the relationship of source domain and target domain fully. To deal with this issue, this paper develops a method using Stacked Denoising Autoencoder (SDA) to extract new feature spaces for source domain and target domain, and define two fuzzy sets to analyse the variation of prediction accuracy of target task in new feature spaces.

Keywords: transfer learning, deep learning, feature extraction, fuzzy sets

1. Introduction

Although machine learning technologies have made great achievements in many research areas, most of these technologies work under the same assumption that source domain and target domain have the same feature space and distribution. It means that if feature space or distribution of target data changes, the prediction model trained using source data can't be used for target tasks, so new model should be built using adequate labeled target data, which is time-consuming and sometimes unavailable. In real world situations, very few labeled target data can be obtained, and collecting new labeled data and constructing a new prediction model for target tasks is impossible. If knowledge exploited from similar but not identical source domain with plenty of labeled data can be utilized to target tasks, building a well prediction model for target task becomes possible.

Transfer learning has emerged as a way of exploiting knowledge from source domain to improve the performance of target tasks. Unlike traditional machine learning, transfer learning considers source domain and target domain are different. Many techniques and methods are proposed to transfer knowledge from source domain to target domain. For instance, a fuzzy bridge refinement-based domain adaptation method based on fuzzy system and similarity concepts is developed to modify the target instances' labels which are predicted by the prediction model trained using source data [1]. On the basis of the above method, dissimilarity is also introduced as another criterion to modify the labels of target instances [2].

As the rise of deep learning, it is applied as a new way to improve the performance of transfer learning. Deep learning is an emerging research area and is considered to be an intelligent feature extraction module that offers great flexibility in extracting multilevel features. Most deep learning algorithms use neural network as framework because multiple hidden layers of neural network have greater expressive power to capture the intricate non-linear representations of data. Many deep learning algorithms are proposed, such as Convolutional Neural Network, Stacked Denoising Autoencoder (SDA), Restricted Boltzman Machines, Deep Belief Networks (CNN), Hybrid Monte-Carlo Sampling, etc. These technologies all work well in machine learning area, but only some of them, for instance CNN and SDA, are used in transfer learning. CNN, which consists of alternating layer of convolutional and max pooling, is constructed on the basis of multi-stage Hubel-Wiesel architecture [3]. Based on CNN, many methods are presented to improve the performance of transfer learning. CNN and multiple tasks learning are combined together to transfer knowledge [4]. In this model, target task and related tasks are trained together to build a neural network with shared input and hidden layers, and separately output neurons. In this case, each task only has one corresponding output neuron. The model is then extended to the situation in which each task has multiple output neurons [5] . Likewise, based on the multi-stage Hubel-Wiesel architectures, whether the layers in the neural network model trained using source data can be reused for target tasks is detected. For the target task model, the input and hidden layers of the model for source task can be reused, but the last output layer needs to be retrained. However, all layers in the model of target task can be fine-tuned. In this case, the parameters in input and hidden layers obtained from source tasks can be treated as initialization parameters of the model for target task, and this strategy is especially promising for a model in which good initialization is very important [6]. SDA is another deep learning structure, which has the greater expressive power to capture a useful "hierarchical grouping" or "part-whole decomposition" of the input. SDA is used to extract meaningful representation for the reviews in sentiment classification problem [7]. In order to reduce the high computational cost and deal with the issue of lacking scalability to high-dimensional features in SDA, Minmin Chen et al. proposed marginalized SDA method, in which no optimization algorithms are needed to learn parameters [8]. According to various degrees of complexity in transfer leaning problems, SDA provides a more flexible way to construct the specific model. The number of layers transferred to the new model de-

pends on the high-level or low-level feature representations that are needed. This means if low-level features are needed, only the first layer parameters are transferred to the target task [9, 10]. Features extracted from SDA are used to transfer knowledge, but how many layers should be transferred, and whether high-level or low-level feature representation is needed is difficult to identify in specific problem. In this paper, knowledge is transferred in multiple feature spaces that are extracted from SDA. And two fuzzy sets are built to indicate the variation of prediction accuracy of target task in these feature spaces. The main contributions are: (1) exploit and transfer knowledge in multiple feature spaces extracted from SDA; (2) analyse the variation of prediction accuracy of target task in multiple feature spaces, and get the best feature space for transfer learning.

The paper is organized in the following way. We start with an introduction to transfer learning and SDA in Section 2. Then a method on the basis of multiple feature spaces is introduced, and two fuzzy sets are defined to analyse the variation of prediction accuracy of target task in multiple feature spaces in Section 3. Finally, conclusion and future work are given in Section 4.

2. Transfer learning and deep learning

This section reviews related work in two areas: transfer learning and Stacked Denoising Autoencoder, one of the deep learning techniques.

2.1. Basic concepts of transfer learning

To understand and analyse the process of transfer learning more clearly, this section first give the notations and definitions about transfer learning that will be used throughout the whole paper.

Definition 2.1 (Domain) [11] A domain, which is denoted by $D=\{\chi,P(X)\}$ consists of two components:

(1) Feature space χ ; and

(2) Marginal probability distribution $P(X)$ where $X=\{x_1,\cdots,x_n\}\in\chi$.

Definition 2.2 (Task) [11] A task, which is denoted by $T=\{Y,f(\cdot)\}$, consists of two components:

(1) A label space $Y=\{y_1,\cdots,y_m\}$; and

(2) An objective predictive function $f(\cdot)$ which is not observed and is to be learned by pairs $\{x_i,y_i\}$.

Definition 2.3 (Transfer learning) [11] Given a source domain D_s and learning task T_s , target domain D_t and learning task T_t, transfer learning aims to improve the learning of the target predictive function $f_t(\cdot)$ in D_t using the knowledge in D_s and T_s where $D_s\neq D_t$ or $T_s\neq T_t$.

In the above definition, the condition $D_s\neq D_t$ implies that either $\chi_s\neq\chi_t$ or $P_s(X)\neq P_t(X)$. Similarly, the condition $T_s\neq T_t$ implies that either $Y_s\neq Y_t$ or $f_s(\cdot)\neq f_t(\cdot)$. In addition, there are some explicit or implicit relationships between the feature spaces of two domains such that we imply that the source domain and target domain are related. It should be mentioned that when the target and source domains are the same ($D_s=D_t$) and their learning tasks are also the same ($T_s=T_t$), the learning problem becomes a traditional machine learning problem.

In this work, we focus on domain adaptation, which belongs to tansductive transfer learning setting. In this situation, no labeled data in the target domain is available while a lot of labeled data in the source domain are available.

Suppose that the source domain is $D_s=\{\chi,P_s(X)\}$, and the target domain is $D_t=\{\chi,P_t(X)\}$. In domain adaptation situation, feature space in source domain and target domain is the same, so we use the identical symbol χ , but the marginal probability distribution is different, i.e. $P_s(X)\neq P_t(X)$.

2.2. Stacked Denoising Autoencoder

Deep learning is a new area in machine learning, which has been introduced with the objective of moving machine learning closer to one of its original goal: Artificial Intelligence. Deep neural network is considered to be an intelligent feature extraction module that offers great flexibility in extracting high-level features in machine learning. The prominent characteristic of deep neural network is its multiple hidden layers, which can capture the intricate non-linear representations of data.

SDA is one of the deep learning methods that can learn multiple feature spaces. The core idea of SDA is that unsupervised learning is used to pre-train each layer [12]. Next, the construction of SDA is elaborated with more details.

First, let's recall neural network, which is a supervised learning. Suppose we have access to labeled training examples (x,y), and neural networks give the way of defining a complex, non-linear form of hypotheses $h_{W,b}(x)$, with parameters W,b that fit to the data. A neural network is putting together by hooking together many simple neurons, so that the output of a neuron can be the input of another. For example, here is a small neural network:

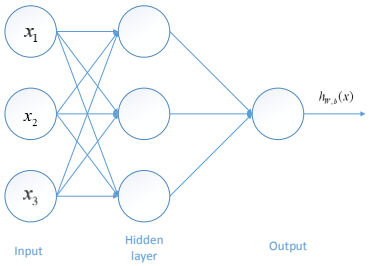

Figure 1: Neural network

In Figure 1, there are three layers in this neural network. The leftmost layer of the network is called input layer, and there are three neurons in the input layer, so the input data can be expressed as $x=(x_1,x_2,x_3)$ .The rightmost layer is the output layer. The middle layer of

nodes is called the hidden layer, because their values are not observed in the training set. In this neural network, the parameters are $(W,b)=(W^{(1)},b^{(1)},W^{(2)},b^{(2)})$, where $W_{ij}^{(l)}$ is the parameter associated with the connection between unit j in layer l, and unit i in layer $l+1$. Also, $b_i^{(l)}$ is the bias associated with unit i in layer $l+1$. The training of neural network is to find the optimal parameters (W,b) to minimize the distance between y and $h_{W,b}(x)$.

An antoencoder neural network is an unsupervised learning algorithm. Now suppose that we have a set of unlabeled training examples $\{x^{(1)},x^{(2)},x^{(3)},\cdots x^{(m)}\}$, where $x^{(i)}\in R^n$, where n is the number of neurons in the input layer. In the antoencoder neural network, the numbers of input and output neurons are equal in order to make the target values be equal to the inputs, i.e. $h_{W,b}(x^{(i)})=x^{(i)}$. An autoencoder is shown in figure 2:

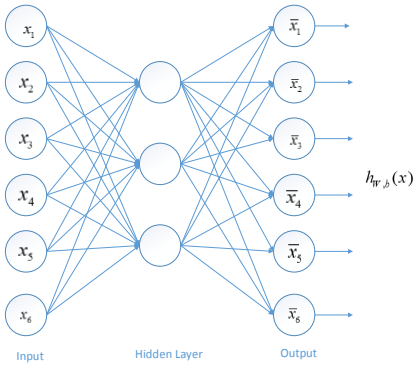

Figure 2: Autoencoder

The structure of autoencoder is designed to reconstruct the input data. The reconstruction accuracy is obtained by minimizing the average reconstruction error between the original data and the reconstructed instances. The neurons in the hidden layer have the ability of reconstructing the input data, so they can be treated as a new feature space for the original data.

The training process of antoencoder is to minimize the below formula:

$$\frac{1}{m}\sum_{i=1}^{m} g(x^{(i)},h_{W,b}(x^{(i)})) \qquad (1)$$

where $g(x^{(i)},h_{W,b}(x^{(i)}))$ is the distance between the input data and the reconstructed data.

But this reconstruction criterion alone may lead to the obvious solution, which simply copies the input. In view of this situation, Pascal [13] gave the definition of a good representation and followed it as new criterion to reconstruct. They defined "a good representation is the one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input". So in order to extract new features that are stable and robust under corruptions of the input, denoising is advocated as a training criterion in antoencoder to extract features capture useful structure in the input distribution. First all the data will be added with some noise, so the original data $x^{(i)}$ becomes $\bar{x}^{(i)}$, and the function needed to be optimized becomes:

$$\frac{1}{m}\sum_{i=1}^{m} g(x^{(i)},h_{W,b}(\bar{x}^{(i)})) \qquad (2)$$

SDA is stacking many denoising autoencoder together. SDA consists of layers of denoising autoencoders in which the outputs of each layer are wired to the inputs of the successive layer. Figure 3 gives an example of SDA.

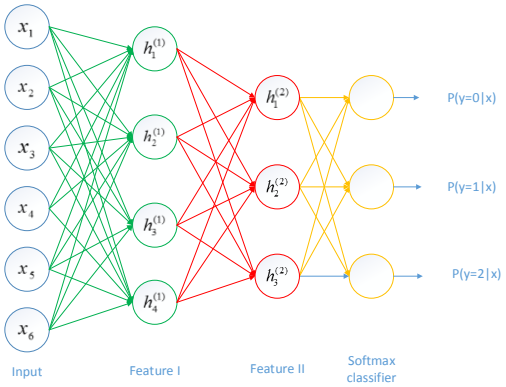

Figure 3: Stacked Denoising Autoencoder

In this SDA, there are one input layer and two hidden layers, followed by a softmax classifier layer. The dimension of the input data is 6, and the number of labels is 3. A good way to obtain the parameters of SDA is to use greedy layer-wise training. Next, the whole training process of SDA is given.

In order to get the parameters between the input layer and the first hidden layer, an antoencoder is trained on the raw input x to learn primary features $h^{(1)}$ on the raw input. The process is shown in figure 4.

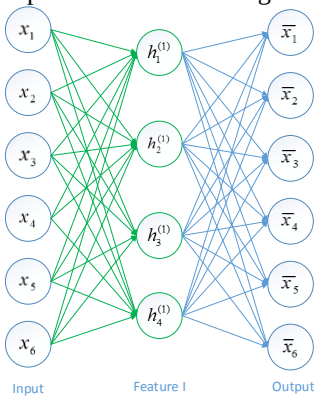

Figure 4: Autoencoder for training Feature I

Next, the raw input can be feed into this trained antoencoder in Figure 4, obtaining the primary feature activations $h^{(1)}$ for the input x. Then, use these primary features as the "raw input" to another antoencoder to learn secondary features $h^{(2)}$ on the primary features. The process is shown in figure 5.

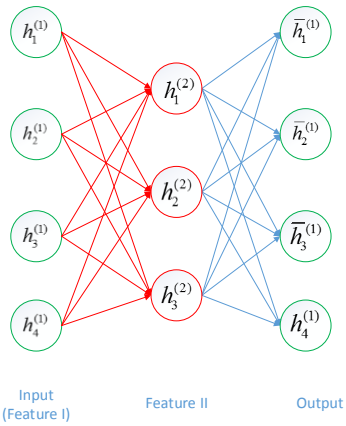

Figure 5: Autoencoder for training Feature II

Following this, the primary features are feed into the autoencoder in Figure 5 to obtained the secondary feature activation $h^{(2)}$ for each of the primary features $h^{(1)}$, which correspond to the primary features of the corresponding inputs x. Then treat these secondary features as "raw input" to a softmax classifier, training it to map secondary features to labels. The process is shown in figure 6.

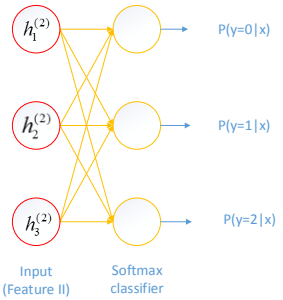

Figure 6: The softmax classifier

Finally, combine all three layers together to form a SDA with two hidden layer and a final softmax classifier capable of classifying the original data as desired.

In this paper, SDA is used to extract feature spaces, so we construct SDA only including input layer and hidden layers (feature space layers). So the structure of SDA is like the one in Figure 7, if we want to extract two feature spaces.

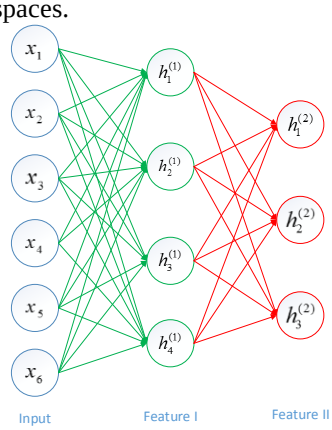

Figure 7: SDA without output layer

When training the SDA with structure in Figure 7, unlabeled data are needed as the input to extract feature spaces.

3. Knowledge transfer in different feature spaces extracted from SDA

SDA provides a way to find new feature spaces for source domain and target domain. The new feature spaces extracted from SDA are obtained by minimizing the distance between marginal probability distribution of source data and target data.

We use SDA to extract new feature spaces for source domain and target domain separately. Two SDA are constructed for source domain and target domain, denoted as SDA(s) and SDA(t), and they have the same structure illustrated in figure 8, in which three feature spaces can be formed. The number of feature space layers can be changed, and it depends on how many new feature spaces you want to form.

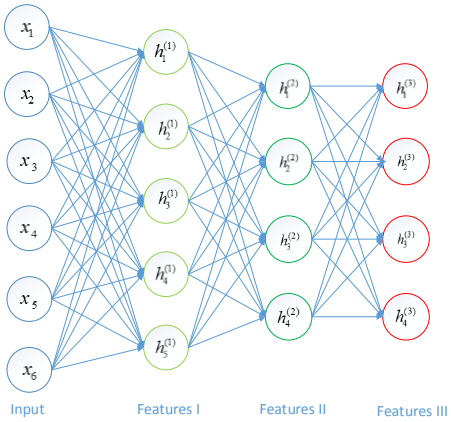

Figure 8: Using SDA to extract feature spaces

The input of SDA(s) and SDA(t) are unlabeled source data and unlabeled target data separately. When train SDA(s) and SDA(t), three denoising autoencoders are built for source domain and target domain separately.

In domain adaptation, source domain and target domain have the same feature space χ, but the marginal probability distribution are different $P_s(X) \neq P_t(X)$. The purpose of SDA is to form new feature spaces for the input data, denoted as χ^j, $j=1,2,3$. At the same time, the corresponding marginal probability distributions of source domain and target domain change into $P_s(X^j)$ and $P_t(X^j)$ separately, $j=1,2,3$.

So from SDA(s) and SDA(t), we get three new feature spaces χ^j, $j=1,2,3$, and the marginal probability distributions of source domain and target domain also change. We get the following changes:

Source domain: $\chi \to \chi^j$, $P_s(X) \to P_s(X^j)$, $j=1,2,3$

Target domain: $\chi \to \chi^j$, $P_t(X) \to P_t(X^j)$, $j=1,2,3$

To reduce the gap between source domain and target domain, the distance between marginal probability distribution of source domain and target domain is minimized in every new feature space. So when training the denoising antoencoders in SDA(s) and SDA(t), the following function is needed to be optimized:

$$\min \quad d(P_s(X^j) - P_t(X^j)) \qquad j=1,2,3 \qquad (3)$$

where $d(P_s(X^j)-P_t(X^j))$ is the distance of $P_s(X^j)$ and $P_t(X^j)$ in feature space χ^j, and $P_s(X^j)$ is obtained by optimizing formula (2) in SDA(s), and $P_t(X^j)$ is obtained by optimizing formula (2) in SDA(t).

Actually, the essence of the above procedures is to find functions f_{sj} and f_{tj}, $j=1,2,3$, such that there are $f_{sj}:\chi\to\chi^j$ in source domain, and $f_{tj}:\chi\to\chi^j$ in target domain. Functions f_{sj} satisfy the condition that χ^j has the ability to reconstruct χ in source domain, and f_{tj} satisfy the condition that χ^j has the ability to reconstruct χ in target domain.

Using the mapping functions f_{sj} and f_{tj}, $j=1,2,3$ learned from SDA(s) and SDA(t), source data and target data can be projected to new feature space χ^j, $j=1,2,3$. So the source domain and target domain become D_s^j and D_t^j, where $D_s^j=\{\chi^j,P_s(X^j)\}$, and $D_t^j=\{\chi^j,P_t(X^j)\}$.

In D_s^j and D_t^j, $P_s(X^j)$ and $P_t(X^j)$ have the smallest distance. So the prediction model trained by labeled source data can be used to unlabeled target data, and we can get the prediction accuracy of the target task , denoted as a_j, $j=1,2,3$.

We extract three feature spaces, and the prediction accuracy of the target task is different in every feature space. We can choose the best feature space with the highest prediction accuracy for target task. But what we want to get is not only the optimal feature space for transfer learning, but also the variation of prediction accuracy of target task in these feature spaces. Next, two fuzzy sets are constructed to indicate the relationship between the distance of marginal probability distributions in source domain and target domain and prediction accuracy of target task.

Suppose that $D=\{d_j \mid j=1,2,3\}$, where d^j is the distance of $P_s(X^j)$ and $P_t(X^j)$ in feature space χ^j, and the optimization result of formula (3). Next two fuzzy sets are defined on D.

Definition 3.1 A is a fuzzy set, named "Positive Transfer (PT)" and defined on domain of discourse D. The membership degree of each element in D belonging to PT is defined as follows:

$$A(d_j)=h(a_j) \qquad (4)$$

where $d^j\in D$, a_j is the prediction accuracy of target task when transfer learning is implemented on feature space χ^j. $h(\cdot)$ is an increasing function, the choice of $h(\cdot)$ depends on specific problem.

Definition 3.2 B is a fuzzy set, named "Negative Transfer (NT)" and defined on domain of discourse D. The membership degree of each element in D belonging to NT is defined as follows:

$$B(d_j)=1-h(a_j) \qquad (5)$$

Based on fuzzy sets A and B, every element d^j in D can be expressed as a pair $(A(d^j),B(d^j))$. $A(d^j)$ represents the membership degree of belonging to Positive Transfer, and $B(d^j)$ represents the membership degree of belonging to Negative Transfer. If variation of d^j belonging to fuzzy set PT is needed, $A(d^1)$, $A(d^2)$, and $A(d^3)$ can be compared.

4. Conclusion and further study

The present method utilizes SDA to extract multiple feature spaces, and a prediction model for target task can be built in every feature space. The feature spaces are obtained by minimizing the distance between marginal probability distribution of source domain and target domain. The prediction accuracy of target task in every feature space is different, so two fuzzy sets are constructed to indicate the variation of prediction accuracy of target task in these feature spaces. In the process of optimizing the feature spaces, the choice of distance function of marginal probability distributions plays an important role. And the optimization of feature spaces should be related, because the formations of feature spaces are conjoint. These issues will be considered in further study.

References

[1] V. Behbood, J. Lu and G. Zhang, Fuzzy refinement domain adaptation for long term prediction in banking ecosystem. IEEE Trans. Industrial Informatics 10 (2), pages 1637-1646, 2014.

[2] V. Behbood, J. Lu and G. Zhang, Fuzzy bridge refinement domain adaptation: Long-term bank failure prediction. International Journal of Computational Intelligence and Applications 12(01), 2013.

[3] D. H. Hubel, and T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology, 160(1), pages 106-154, 1962.

[4] A. Ahmed, K. Yu, W. Xu, Y. Gong and E. Xing, Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks, in Computer Vision–ECCV 2008, Springer, pages 69-82, 2008.

[5] J. Huang, J. Li, D. Yu, L. Deng and Y. Gong, Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7304-7308, 2013.

[6] D. C. Ciresan, U. Meier, and J. Schmidhuber. Transfer learning for Latin and Chinese characters with deep neural networks. International Joint Conference on Neural Networks (IJCNN), 2012.

[7] X. Glorot, A. Bordes and Y. Bengio, Domain adaptation for large-scale sentiment classification: a

deep learning approach. Proceedings of the 28 th International Conference on Machine Learning, 2011.

[8] M. Chen, Z. Xu and K.Q. Weinberger, Marginalized denoising autoencoders for domain adaptation. Proceedings of the 29 th International Conference on Machine Learning, 2012.

[9] K. Chetak, L. Silva and L. Alexandre, Report: Improving CNN by reusing features trained with transductive transfer setting, 2014.

[10] C. Kandaswamy, et al, Improving deep neural network performance by reusing features trained with transductive transference. Proceedings of the 24th International Conference on Artificial Neural Networks, pages 265-272, 2014.

[11] S.J. Pan, and Q. Yang, A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), pages 1345-1359, 2010.

[12] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, Greedy layer-wise training of deep networks, in advances in neural information processing systems 19 (NIPS‘06), pages 153-160, MIT Press 2007.

[13] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P. Manzagol, Stacked Denoising Autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research 11, pages 3371-3408,2010.