Distant Domain Transfer Learning\textsuperscript{a}

Jindong Wang
wangjindong@ict.ac.cn

May 2017

\textsuperscript{a}This slide is based on AAAI-17 paper: Ben Tan, Yu Zhao, Sinno Jialin Pan and Qiang Yang: Distant domain transfer learning.
Background of Transfer learning

Introduction
  Author
  DDTL: Distant Domain Transfer Learning

Selective Learning Algorithm for DDTL
  Instance Selection via Reconstruction Error
  Incorporation of Side Information
  Learning Algorithm

Experiments and Analysis

Related Work

Conclusion
Background
What is transfer learning?

Problems
- Building every model from scratch is time-consuming and expensive.
- But there are many existing knowledge. Can we reuse them?

(a) Traditional Machine Learning
(b) Transfer Learning
### Common Definition
- Wikipedia: research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a **different** but **related** problem [wik].

### TL vs Traditional ML

**Traditional ML:**
- Training and testing samples must be in the **same** feature distributions.
- Training samples must be **enough**.

**Transfer learning:**
- Source and target domains do **not** need to be in the same distributions.
- **Less** training samples, even **none**.
Example: sentiment classification

DVD → Electronics: Only got sentiment on DVD, how to transfer it to electronics?

Proceedings

- Data mining: ACM SIGKDD, IEEE ICDM, PKDD
- ML & AI: ICML, NIPS, AAAI, IJCAI, ECML
- Applications: ACM TIST, ACM SIGIR, WWW, ACL, IEEE TKDE

Many apps include image classification, natural language processing, activity recognition, and Wifi localization.
Introduction
Author Information

There are 4 authors of this paper:

**Tan Ben**
- Ph.D candidate at HKUST

**Yu Zhang**
- Research associate at HKUST

**Sinno Jialin Pan**
- Assistant professor at NTU
- Google scholar citations: 4,000+
- [http://www.ntu.edu.sg/home/sinnopan/](http://www.ntu.edu.sg/home/sinnopan/)

**Qiang Yang**
- Head of CSE HKUST
- Fellow of AAAI/IEEE/AAAS/IAPR
- Google scholar citations: 30,000+
Introduction
Distant domain transfer learning

Traditional TL: the source and target domain are close \[PY10\]
DDTL: the source and target domain can be totally different!

- Task 1: Cat → Tiger, good performance for traditional TL.
- Task 2: Face → Airplane, bad performance for traditional TL.
Introduction
Distant domain transfer learning

Traditional TL: the source and target domain are close [PY10]
DDTL: the source and target domain can be totally different!

- Task 1: Cat → Tiger, good performance for traditional TL.
- Task 2: Face → Airplane, bad performance for traditional TL.

How to conduct transfer learning in such scenario when source and target domain are totally different?
**Introduction**

Problem definition

**Distant domain transfer learning (DDTL):** exploit the unlabeled data in the **intermediate** domains to build a bridge between source and target domain.

Input:
- Labeled source domain $\mathcal{S} = \{(x^1_S, y^1_S), \ldots, (x^n_S, y^n_S)\}$
- Unlabeled target domain $\mathcal{T} = \{(x^1_T, y^1_T), \ldots, (x^n_T, y^n_T)\}$
- Mixture of unlabeled intermediate domains: $\mathcal{I} = \{(x^1_I), \ldots, (x^n_I)\}$

Output:
- labels of target domain

Constraints:
- $p_T(x) \neq p_S(x), p_T(x) \neq p_I(x)$ and $p_T(y|x) \neq p_S(y|x)$
- similarity between $\mathcal{S}$ and $\mathcal{T}$ is very small
Selective Learning Algorithm (SLA) is proposed to solve the DDTL problem, which is based on autoencoder.

**Autoencoder**

An unsupervised feed-forward neural network with an input layer, hidden layer and output layer.

- **Encoding**: \( h = f_e(x) \)
- **Decoding**: \( \hat{x} = f_d(h) \)
- **Objective**: \( \min \sum_{i=1}^{n} ||\hat{x}_i - x_i||_2^2 \)

To capture spatial information, a convolutional autoencoder is desired.
Motivation: if data from source / intermediate domain is similar and useful to the target domain, then one should be able to find a pair of encoding and decoding functions that have small reconstruction error.

Objective: learn a pair of encoding and decoding functions by minimizing reconstruction errors on source, intermediate and target domain simultaneously.

\[ J_1(f_e, f_d, v_S, v_T) = \frac{1}{n_S} \sum_{i=1}^{n_S} ||\hat{x}_i^S - x_i^S||^2_2 + \frac{1}{n_I} \sum_{i=1}^{n_I} ||\hat{x}_i^I - x_i^I||^2_2 + \frac{1}{n_T} ||\hat{x}_i^T - x_i^T||^2_2 + R(v_S, v_T) \]
Motivation: if data from source / intermediate domain is similar and useful to the target domain, then one should be able to find a pair of encoding and decoding functions that have small reconstruction error.

Objective: learn a pair of encoding and decoding functions by minimizing reconstruction errors on source, intermediate and target domain simultaneously.

\[
\mathcal{J}_1(f_e, f_d, v_S, v_T) = \frac{1}{n_S} v^i_S \| \hat{x}^i_S - x^i_S \|^2_2 + \frac{1}{n_I} v^i_I \| \hat{x}^i_I - x^i_I \|^2_2 + \frac{1}{n_T} \| \hat{x}^i_T - x^i_T \|^2_2 + R(v_S, v_T) \\
\]

\( v_S \) and \( v_T \) are selection indicators. \( R(\cdot, \cdot) \) is the regularization term.

\[
R(v_S, v_T) = -\frac{\lambda_S}{n_S} \sum_{i=1}^{n_S} v^i_S - \frac{\lambda_T}{n_T} \sum_{i=1}^{n_T} v^i_T
\]
Learning in Eq. (1) is in unsupervised manner, consider to add some side information:

\[
J_2(f_c, f_e, f_d) = \frac{1}{n_S} \sum_{i=1}^{n_S} v^i_S \mathcal{L}(y^i_S, f_c(h^i_S)) + \frac{1}{n_T} \sum_{i=1}^{n_T} v^i_T \mathcal{L}(y^i_T, f_c(h^i_T)) \\
+ \frac{1}{n_I} \sum_{i=1}^{n_I} v^i_I g(f_c(h^i_I))
\]

(3)

\(f_c(\cdot)\) is a classification function, \(g(\cdot)\) is the entropy function:

\[g(z) = -z \ln z - (1 - z) \ln(1 - z)\] for \(0 \leq z \leq 1\).

Overall objective function:

\[
\min_{\Theta, v} J = J_1 + J_2
\]

(4)

\(s.t.\ v^i_S, v^i_T \in \{0, 1\}\)

Where \(\Theta\) denotes all parameters \((f_c(\cdot), f_d(\cdot), f_e(\cdot))\) and \(v = \{v_S, v_T\}\).
Learning Algorithm

Technique: **Block Coordinate Decent (BCD)**, where in each iteration, variables in each block are optimized sequentially while keeping other variables fixed.

- fix $v$, update $\Theta$ using back propagation;
- fix $\Theta$, obtain $v$ as follows:

$$v^i_S = \begin{cases} 
1 & \text{if } \mathcal{L}(y^i_s, f_c(f_e(x^i_S))) + \|\hat{x}^i_S - x^i_S\|^2 < \lambda_S \\
0 & \text{otherwise}
\end{cases} \quad (5)$$

$$v^i_T = \begin{cases} 
1 & \text{if } \|\hat{x}^i_I - x^i_I\|^2 + g(f_c(f_e(x^i_I))) < \lambda_I \\
0 & \text{otherwise}
\end{cases} \quad (6)$$

Based on the above equations, only samples with low reconstruction error and high prediction confidence will be selected and used.
The learning algorithm is as follows:

**Algorithm 1** The Selective Learning Algorithm (SLA)

1: **Input:** Data in $S$, $T$ and $I$, and parameters $\lambda_S$, $\lambda_I$, and $T$;
2: Initialize $\Theta$, $\nu_S = 1$, $\nu_I = 0$; // All source data are used
3: **while** $t < T$ **do**
4: Update $\Theta$ via the BP algorithm; // Update the network
5: Update $\nu$ by Eqs. (4) and (5); // Select “useful” instances
6: $t = t + 1$
7: **end while**
8: **Output:** $\Theta$ and $\nu$. 
Add convolution layers to the network, it can be viewed as a generalized autoencoder or convolutional autoencoder with side information.

**SAN**: supervised autoencoder, only autoencoder

**SCAN**: supervised convolutional autoencoder, using convolution
Experiment Overview

2 datasets
- **Caltech-256**: 30,607 images from 256 classes
- **Animals with Attributes (AwA)**: 30,475 images with 50 classes

3 categories of baseline methods
- **Supervised learning**: SVM and CNN
- **Transfer learning**: ASVM, GFK, LAN, DTL and TTL
- **Self-taught learning method**

3 experiments
- Source and target domain are distant
- Visualize some intermediate domain data
- Evaluate the learning order of SLA
Average accuracies of different algorithms on Caltech-256 and AwA:

<table>
<thead>
<tr>
<th></th>
<th>SVM</th>
<th>DTL</th>
<th>GFK</th>
<th>LAN</th>
<th>ASVM</th>
<th>TTL</th>
<th>STL</th>
<th>SLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘horse-to-face’</td>
<td>84 ± 2</td>
<td>88 ± 2</td>
<td>77 ± 3</td>
<td>79 ± 2</td>
<td>76 ± 4</td>
<td>78 ± 2</td>
<td>86 ± 3</td>
<td>92 ± 2</td>
</tr>
<tr>
<td>‘airplane-to-gorilla’</td>
<td>75 ± 1</td>
<td>62 ± 3</td>
<td>67 ± 5</td>
<td>66 ± 4</td>
<td>51 ± 2</td>
<td>65 ± 2</td>
<td>76 ± 3</td>
<td>84 ± 2</td>
</tr>
<tr>
<td>‘face-to-watch’</td>
<td>75 ± 7</td>
<td>68 ± 3</td>
<td>61 ± 4</td>
<td>63 ± 4</td>
<td>60 ± 5</td>
<td>67 ± 4</td>
<td>75 ± 5</td>
<td>88 ± 4</td>
</tr>
<tr>
<td>‘zebra-to-collie’</td>
<td>71 ± 3</td>
<td>69 ± 2</td>
<td>56 ± 2</td>
<td>57 ± 3</td>
<td>59 ± 2</td>
<td>70 ± 3</td>
<td>72 ± 3</td>
<td>76 ± 2</td>
</tr>
</tbody>
</table>

**Conclusion**: For distant domains, SLA algorithm **outperforms** other methods.
Visualization of the selected intermediate data over iterations of face-to-airplane and face-to-watch:

Conclusion:

- At the beginning, the intermediate domains are similar to source domain. In the end, it’s more similar to the target domain.
- The number of positive examples in source domain decreases, and the value of objective function also decreases.
Comparison results with different learning orders on the Caltech-256 and AwA datasets (Orders of intermediate domains changed):

**Conclusion**: Three types of different orders obtain worse results than SLA, and Category is close to SLA because this strategy is close to SLA.
DDTL is a novel difficult problem with many state-of-the-art methods not applying to it.

- **Typical transfer learning** approaches like instance reweighting [DYXY07] and feature mapping [PTKY11] do not apply to this problem, as they assume the domains are close.

- **Transitive transfer learning** [TSZY15]: manually select one intermediate domain as the bridge; ours automatically select many domains.

- **TLMS** [MMR09]: all the source domains in TLMS are labeled and closely related to the target domain.

- **Self-taught learning** [RBL07]: use all domain data to learn; ours use intermediate domains.

- **Semi-supervised autoencoder** [WRMC12]: uses labeled and unlabeled data for learning; ours use intermediate domains and we use convolutional layers.
Conclusion

Contributions of this paper

▶ **First** work to study DDTL problem using mixture intermediate domains
▶ Propose **SLA** algorithm for DDTL problem
▶ Extensive experiments on real-world datasets show the **effectiveness** of SLA

What we should learn from this paper

▶ Good layout, consider it as a template for **algorithm-paper**
▶ Introduction, tables and figures are good
▶ Experiment: **more datasets, more analysis**
References I


Thank you for your listening