Transfer learning!

Meichen Lu (ml574@cam.ac.uk)

June 22, 2018

Contents

1 Introduction

1.1 Motivation

Difference between training and test stages “many machine learning methods work well only under a common assumption: the training and test data are drawn from the same feature space and the same distribution. When the distribution changes, most statistical models need to be rebuilt from scratch using newly collected training data.”

Human intelligence The study of Transfer learning is motivated by the fact that people can intelligently apply knowledge learned previously to solve new problems faster or with better solutions.

NIPS-95 Learning to learn lifelong machine-learning methods that retain and reuse previously learned knowledge.

1.2 Definitions

TL defined by DARPA the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks. In this definition, transfer learning aims to extract the knowledge from one or more *source tasks* and applies the knowledge to a *target task*.

inductive learning : “the learned classifier can be applied to any unseen instance”

transductive learning : "we only care about predictions on a pre-specified set of unlabeled instances."

1.3 Related machine learning methods

1.3.1 Multi-task learning

- TL: train only on source task; Multi-task learning train on both source and target tasks.

2 TL for data mining

2.1 Notations

A *domain* $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = x_1, ..., x_n \in \mathcal{X}$.

Given a specific domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a *task* consists of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$.

- Task is denoted by $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$
- $f(\cdot)$ is not observed but can be learned from the training data, which consist of pairs x_i, y_i, where $x_i \in X$ and $y_i \in \mathcal{Y}$.
- $f(\cdot)$ can be used to predict the corresponding label, $f(x)$, of a new instance x.
- $f(x)$ can be written as the conditional probability function $P(y|x)$

Consider only one source domain $\mathcal{D}_S$, and denote *source domain data* as $D_S = \{(x_{S_1}, y_{S_1}), ..., (x_{S_n}, y_{S_n})\}$. Consider only one target domain $\mathcal{D}_T$, and denote *target domain data* as $D_T = \{(x_{T_1}, y_{T_1}), ..., (x_{T_n}, y_{T_n})\}$.

Definition 1 (*Transfer Learning*) Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$

Different source and target domain:

1. Different feature space $\mathcal{X}_S \neq \mathcal{X}_T$ **OR**
2. Same feature space $\mathcal{X}_S = \mathcal{X}_T$ but different marginal probability distribution $P(X_S) \neq P(X_T)$.

Different source and target tasks:

1. Different label space $\mathcal{Y}_S \neq \mathcal{Y}_T$ **OR**
2. Same label space $\mathcal{X}_S = \mathcal{X}_T$ but different conditional probability distribution $P(Y_S|X_S) \neq P(Y_T|X_T)$.

Difference between $f(x)$ and $P(Y|X)$? Which is more stringent? Does $P(Y|X)$ imply a knowledge of the underlying distribution?

2.2 Categorization of TL techniques

Three research issues

1. **What to transfer** asks which part of knowledge can be transferred across domains or tasks.
2. **When to transfer** asks in which situations, transferring skills should be done.
3. **How to transfer** asks which learning algorithm need to be developed to transfer the knowledge

Categorization based on different situations between the source and target domains and tasks

1. *Inductive TL*: the target task is different from the source task, no matter whether the source and target domains are the same or not.
   Some **labeled data in the target domain** are required to *induce* an objective predictive model. Regarding the source domain:
   (a) A lot of labeled data in the source domain are available.
       In this case, the inductive transfer learning setting is similar to the multi-task learning setting. However, the inductive transfer learning setting only aims at achieving high performance in the target task by transferring knowledge from the source task while multi-task learning tries to learn the target and source task simultaneously.
   (b) No labeled data in the source domain are available.
       Related to self-taught learning.
2. *Transductive TL* the source and target tasks are the same, while the source and target domains are different.
   No labeled data in the target domain are available while a lot of labeled data in the source domain are available.
   (a) $\mathcal{Y}_S \neq \mathcal{Y}_T$
   (b) $\mathcal{Y}_S = \mathcal{Y}_T$ but $P(X_S) \neq P(X_T)$.
       i. Different domains but single task: domain adaptation
       ii. Single domain and single task: sample selection bias/co-variate shift
3. *Unsupervised TL*: no label in either source or target domain. *Inductive learning* on solving unsupervised learning tasks in the target domain, such as clustering, dimensionality reduction and density estimation.

categorization based on “what to transfer”

1. **Instance-transfer** assumes that certain parts of the data in the source domain can be reused for learning in the target domain by re-weighting. Instance re-weighting and importance sampling are two major techniques in this context.
2. **Feature-representation-transfer**: the knowledge used to transfer across domains is encoded into the learned feature representation.
3. **Parameter-transfer**: the source tasks and the target tasks share some parameters or prior distributions of the hyper-parameters of the models.

4. **Relational-knowledge transfer** assumes that some relationship among the data in the source and target domains are similar.

2.3 Algorithms

2.3.1 Homogeneous transfer learning

Homogeneous transfer learning is when $\mathcal{X}_S = \mathcal{X}_T$.

Instance-based transfer learning

- Conditional probability based multi-source domain adaptation (CP-MDA) approach. **Use pseudo labels (semi-supervised)**
- two stage weighting framework for multi-source domain adaptation (2SW-MDA). **Weight the source using marginal distribution**

Asymmetric feature-based transfer learning

- Daume FEDA
- Domain transfer multiple kernel learning (DTMKL)
- Joint domain adaptation (JDA)
- Adaptation Regularization based transfer learning (ARTL)

Parameter-based TL

- Multi-model knowledge transfer (MMKT) method: transfer the SVM hyperplane information
- Multi-source TrAdaBoost algorithm (called MsTrAdaBoost)
- TaskTrAdaBoost: transfers **internal learner parameter** information from the source to the target.

Symmetric feature-based transfer learning

- Transfer component analysis (TCA)
- Spectral feature alignment (SFA)
- Stacked denoising autoencoder (SDA)
- Geodesic flow kernel (GFK) is proposed that finds a **low-dimensional feature space**, which **reduces the marginal distribution differences** between the labeled source and unlabeled target domains.
- Discriminative clustering process (DCP) to equalize the marginal distribution of the labeled source and unlabeled target domains.

2.3.2 Heterogeneous transfer learning

Heterogeneous transfer learning is the scenario where the source and target domains are represented in different feature spaces.

Symmetric feature-based transfer learning

- Structural correspondence learning (SCL)
- Heterogeneous Spectral Mapping (HeMap): first step is to find a common **latent input space** between the source and target domains using a spectral mapping technique.
- Domain adaptation manifold alignment (DAMA) algorithm, proposes using a manifold alignment process to perform a symmetric transformation of the domain input spaces. Each domain is modeled as a manifold.
-

3 Instance weighting framework

3.1 Probabilistic Framework

Two types of differences

1. Instance difference: $P_t(Y|X) = P_s(Y|X)$ but $P_t(X) \neq P_s(X)$. Same problem as covariance shift or sample selection bias.
2. Labeling difference: $P_t(Y|X) \neq P_s(Y|X)$.

3.1.1 Using source domain

Probabilistic framework using empirical risk minimisation. Components:

- $\mathcal{H}$ Hypothesis space: all possible prediction functions
- $L(x, y, f)$ Loss function

Estimating f_t^* for target domain using samples from source domain:

$$f_t^* = \arg\min_{f\in\mathcal{H}} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \frac{P_t(x,y)}{P_s(x,y)} P_s(x,y) L(x,y,f) \tag{1}$$

$$\approx \arg\min_{f\in\mathcal{H}} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \frac{P_t(x,y)}{P_s(x,y)} \tilde{P}_s(x,y) L(x,y,f) \tag{2}$$

$$= \arg\min_{f\in\mathcal{H}} \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{P_t(x_i^s, y_i^s)}{P_s(x_i^s, y_i^s)} L(x_i^s, y_i^s, f) \tag{3}$$

$$= \arg\min_{f\in\mathcal{H}} \frac{1}{N_s} \sum_{i=1}^{N_s} \underbrace{\frac{P_t(x_i^s)}{P_s(x_i^s)}}_{\alpha_i} \underbrace{\frac{P_t(y_i^s|x_i^s)}{P_s(y_i^s|x_i^s)}}_{\beta_i} L(x_i^s, y_i^s, f) \tag{4}$$

3.1.2 Using labeled target domain

$$f_t^* = \arg\min_{f\in\mathcal{H}} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} P_t(x,y) L(x,y,f) \tag{5}$$

$$= \arg\min_{f\in\mathcal{H}} \frac{1}{N_{t,l}} \sum_{i=1}^{N_{t,l}} L(x_i^{t,l}, y_i^{t,l}, f) \tag{6}$$

3.1.3 Using unlabeled target domain

$$f_t^* = \arg\min_{f\in\mathcal{H}} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} P_t(x) P_t(y|x) L(x,y,f) \tag{7}$$

$$\approx \arg\min_{f\in\mathcal{H}} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \tilde{P}(x) P_t(y|x) L(x,y,f) \tag{8}$$

$$= \arg\min_{f\in\mathcal{H}} \frac{1}{N_{t,u}} \sum_{i=1}^{N_{t,u}} \sum_{y\in\mathcal{Y}} \underbrace{P_t(y|x_i^{t,u})}_{\gamma_i(y)} L(x_i^{t,y}, y, f) \tag{9}$$

3.1.4 Combining all domains

The loss from all data are added together

$$\hat{f}_t = \arg\min_{f\in\mathcal{H}} \left[\lambda_s \sum_{i=1}^{N_s} \alpha_i \beta_i L(x_i^s, y_i^s, f) + \lambda_{t,l} \sum_{i=1}^{N_{t,l}} L(x_i^{t,l}, y_i^{t,l}, f) + \lambda_{t,u} \sum_{y\in\mathcal{Y}} \gamma_i(y) L(x_i^{t,y}, y, f) + \lambda R(f) \right] \tag{10}$$

where

$$\alpha_i \geq 0, \beta_i \geq 0, \sum_{i=1}^{N_s} \alpha_i \beta_i = N_s$$
$$0 \leq \gamma_i(y) \leq 1$$

3.2 Estimating marginal distribution correction α_i

3.2.1 Logistic regression

Build a probabilistic model to compute $P_t(x)$ and $P_s(x)$.

$$\frac{P_t(x)}{P_s(x)} = \frac{P(x|d=t)}{P(x|d=s)} = \frac{P(d=t|x)P(x)}{P(d=t)} \cdot \frac{P(d=s)}{P(d=s|x)P(x)} = \frac{P(d=s)}{P(d=t)} \cdot \frac{P(d=t|x)}{P(d=s|x)} \propto \frac{P(d=t|x)}{P(d=s|x)} \tag{11}$$

Mathod proposed by Zadrozny (2004) treat the estimation of $P(d=t|x)$ and $P(d=s|x)$ as a logistic regression problem:

$$P(d=t|x) = \frac{1}{1+\exp(-\theta x)} \tag{12}$$

$$P(d=s|x) = \frac{1}{1+\exp(\theta x)} \tag{13}$$

3.2.2 Mean matching

Find a set of weights $\{\alpha_i\}$ such that the difference between the weighted mean of the two domains is minimised.

$$\{\alpha_i\}_{i=1}^{N_s} = \arg\min_{\alpha_i} \left\| \frac{1}{N_s}\sum_{i=1}^{N_s} \alpha_i x_i^s - \frac{1}{N_{t,l}+N_{t,u}} \left(\sum_{i=1}^{N_{t,l}} x_i^{t,l} + \sum_{i=1}^{N_{t,u}} x_i^{t,u} \right) \right\|^2 \tag{14}$$

3.3 Setting conditional distribution correction β

To set $\frac{P_t(y_i^s|x_i^s)}{P_s(y_i^s|x_i^s)}$, Train the same model for labeled source instances and labeled target instances.

3.3.1 Direct estimation

$$\beta_i = \frac{P(y_i^s|x_i^s,\hat{\theta}_t)}{P(y_i^s|x_i^s,\hat{\theta}_t)} \tag{15}$$

β_i are then normalised with sum 1.

3.3.2 Instance pruning

Remove (x^s, y^s) when the estimated $P(y_i^s|x_i^s,\hat{\theta}_t)$ is too small

3.4 Setting instance weighting γ

4 FEDA

4.1 Previous techniques

Straightforward models

- Source only
- Target only
- All
- Weighted
- Pred: use prediction made by source only model as additional features and train a second model on target data, augmented with this new feature
- Linear interpolation of the two predictions

Prior model by Chelba and Acero (2004) match the weights of the target model with the source model. Regularise using $\lambda\|\boldsymbol{w} - \boldsymbol{w}^s\|_2^2$ instead of $\lambda\|\boldsymbol{w}\|_2^2$

4.2 Algorithm

Given $\mathcal{X} = \mathbb{R}^F$ for $F > 0$, we define the augmented input space by $\breve{\mathcal{X}} = \mathbb{R}^{3F}$ and the mappings $\boldsymbol{\Phi}^s, \boldsymbol{\Phi}^t : \mathcal{X} \to \breve{\mathcal{X}}$ as

$$\boldsymbol{\Phi}^s = \langle \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{0} \rangle, \quad \boldsymbol{\Phi}^t = \langle \boldsymbol{x}, \boldsymbol{0}, \boldsymbol{x} \rangle \tag{16}$$

Kernelised version For a kernel $K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}}$:

$$\boldsymbol{\Phi}^s = \langle \Phi(x), \Phi(x), \boldsymbol{0} \rangle, \quad \boldsymbol{\Phi}^t = \langle \Phi(x), \boldsymbol{0}, \Phi(x) \rangle \tag{17}$$

Thus

$$\breve{K}(x, x') = \begin{cases} 2K(x, x') & \text{same domain} \\ K(x, x') & \text{diff. domain} \end{cases} \tag{18}$$

5 Structural correspondence learning (SCL)

5.1 Overview

Find the correspondence of features from the source domain to the target domain.

5.2 Algorithm

1. Define pivot features
2. Train pivot predictors to find the correspondence of features with pivot features
3. Use SVD to find low-dimensional feature mapping
4. Use low-dimensional feature mapping to enhance the original feature.

Input

- Labeled source data $(\boldsymbol{x}_t, y_t)_{t=1}^T, \boldsymbol{x} \in \mathbb{R}^n$
- Unlabeled data from both domains $\boldsymbol{x_j}, \boldsymbol{x} \in \mathbb{R}^n$

5.2.1 Pivot features

Pivot features are features which occur frequently in the two domains and behave similarly in both.

- Choose the most frequent features in both domains
- Choose the features with maximum mutual information

5.2.2 Pivot predictors

Given m pivot features, train m linear predictors on all the data. why not on the non-pivot features?

$$f_\ell = \text{sgn}(\hat{\boldsymbol{w}} \cdot \boldsymbol{x}), \quad \ell = 1...m \tag{19}$$

The weight vectors $\hat{\boldsymbol{w}}$ encode the covariance of the non-pivot features with the pivot features.

5.2.3 Low-dimensional feature mapping

SVD for matrix W

$$W = \begin{pmatrix} | & | & & | \\ \hat{\boldsymbol{w}}_1 & \hat{\boldsymbol{w}}_2 & \cdots & \hat{\boldsymbol{w}}_\ell \\ | & | & & | \end{pmatrix} = U\Sigma V^T$$

From the SVD, construct a low-dimensional feature representation θ using the first h left eigenvector, i.e.

$$\theta = U[1:h, :]^T \in \mathbb{R}^{h \times n} \tag{20}$$

5.2.4 Supervised training and inference

For each training instance t, augment the feature vector with the shared features $\theta \boldsymbol{x}_t$

6 Spectral feature alignment (SFA)

6.1 Algorithm

1. Find
2. Train pivot predictors to find the correspondence of features with pivot features
3. Use SVD to find low-dimensional feature mapping
4. Use low-dimensional feature mapping to enhance the original feature.

Input

- Labeled source data $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^{N_s}, \boldsymbol{x} \in \mathbb{R}^m$
- Unlabeled data from both domains $\{\boldsymbol{x_j}\}^{N_t}, \boldsymbol{x} \in \mathbb{R}^m$
- The number of cluster K
- The number of domain-independent features m

6.1.1 Find domain-independent features

Select domain-independent features by minimising the mutual information of the features to the domain. The re are l domain-independent features W_{DI} and $m-l$ domain-specific features W_{DS}.

6.1.2 Construct bipartite feature graph

The score m_{ij} that measures the relationship between the feature $i \in W_{DS}$ and feature $j \in W_{DI}$, later used to build a co-occurence matrix $\mathbf{M} \in \mathbb{R}^{(m-l)\times l}$. The score of m_{ij} is the total number of co-occurrence of $w_i \in W_{DS}$and $w_j \in W_{DI}$ in $\mathcal{D}_s$ and $\mathcal{D}_t$.

Given two binary matrices

$$\Phi_{DI} = \begin{bmatrix} \phi_{DI}(x_1^s) \\ \phi_{DI}(x_2^s) \\ \vdots \\ \phi_{DI}(x_{N_t}^t) \end{bmatrix} \in \mathbb{R}^{N\times l}, \quad \Phi_{DS} = \begin{bmatrix} \phi_{DS}(x_1^s) \\ \phi_{DS}(x_2^s) \\ \vdots \\ \phi_{DS}(x_{N_t}^t) \end{bmatrix} \in \mathbb{R}^{N\times(m-l)}$$

The weight matrix is $\mathbf{M} = \Phi_{DS}^T \Phi_{DI} \in \mathbb{R}^{(m-l)\times l}$

6.1.3 Spectral feature clustering

1. Form a weight matrix $\mathbf{M}$
2. Form an affinity matrix $\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{0} \end{bmatrix}$
3. Form a diagonal matrix $\mathbf{D}$ where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ and construct the matrix $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$

4. Find the K largest eigenvectors of $\mathbf{L}$ and form the matrix $\mathbf{U}$
5. Define feature mapping function $\varphi(x) = x\mathbf{U}_{[1:m-l,:]}$

6.1.4 Feature augmentation

Enhance each data with the feature mapping function, whose input are the domain specific features.

$$\tilde{x}_i = [x_i, \gamma\varphi(\phi_{DS}(x_i))]$$

7 Transfer component analysis (TCA)

7.1 Previous feature-based domain adaptation methods

7.1.1 Stationary subspace analysis (SSA)

Aim: match distributions in latent spaces von Buhau (2009) "SSA is a general purpose algorithm for the explorative analysis of non-stationary data, i.e. data whose statistical properties change over time. SSA can help to detect, characterize and visualize temporal changes in complex high-dimensional data sets."

SSA factorizes a multivariate time-series into its stationary and non-stationary components.

7.1.2 maximum mean discrepancy embedding (MMDE)

Aim: learn a shared latent space underlying the domains where distance between distributions can be reduced while the data variance can be preserved

7.2 Algorithm

Problem: Labeled source data, unlabeled target data

Aim Learn a set of common *transfer components* underlying **both** domains that

- Minimise the difference in data distributions of the different domains
- Preserve the data properties

7.2.1 Unsupervised domain adaptation: TCA

Quantify the first goal: minimising the distance : applying Maximum mean discrepancy (MMD) for **dimensionality reduction**

Measuring the distance between two marginal distributions in the latent space defined by the transformation ϕ, i.e. minimise the distance between $P(\phi(X_S))$ and $P(\phi(X_T))$

$$\text{Dist}(X'_S, X'_T) = \| \frac{1}{n_1} \sum_{i=1}^{n_1} \phi(x_{S_i}) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(x_{T_i}) \|^2_{\mathcal{H}} \tag{21}$$

1. Kernel trick (i.e. MMDE): Learn the corresponding kernel matrix K that correspond to the low-dimensional latent space using a nonlinear mapping ϕ. K is called the empirical kernel map features.
   Con
   - Cannot generalise to unseen patterns ?
   - Expensive semi-definite programming solver
2. Improved algorithm
   (a) Use $\tilde{W} \in \mathbb{R}^{(n_1+n_2)\times m}$ to transform the empirical kernel map features to an m-dimensional space ($m \ll n_1 + n_2$)
   (b) $W = K^{-1/2}\tilde{W}$
   (c) (last step):
   $$\text{Dist}(X'_S, X'_T) = \text{tr}((KWW^TK)L) = \text{tr}(W^TKLKW) \tag{22}$$

Quantify the second goal: preserving data properties Maximise the data variance.

Final optimisation problem

$$\min_W \text{tr}(W^TKLKW) + \mu\text{tr}(W^TW) \text{ s.t. } W^TKHKW = I \tag{23}$$

- $\text{tr}(W^TW)$ maximise the variance ?
- μ is a tradeoff parameter that balances the two goals
- H is the centering matrix (subtract mean from the data)
- $W^TKHKW = I$, $I \in \mathcal{R}^{m\times m}$ is the constraint that for learning normal vectors ?
- Optimise using Lagrangian multipliers.

7.2.2 Semi-supervised TCA: SSTCA

Three objectives:

Distribution matching Same as unsupervised TCA

Label dependence We propose to maximally align the embedding

$$\tilde{K}_{yy} = \gamma K_l + (1-\gamma) K_v \tag{24}$$

- Label dependence: $[K_l]_{i,j} = k_y y(y_i, y_j)$ for source domain data
- Data variance: $K_v = I$ serves to maximise the variance on both the source and target domain data.
- γ is a tradeoff parameter that balances label dependence and data variance.

Locality preserving using manifold information to improve classification performance via Lapacian smoothing.

Final optimisation problem

$$\min_W \operatorname{tr}(W^T KLKW) + \mu \operatorname{tr}(W^T W) + \frac{\lambda}{n^2} \operatorname{tr}(W^T K \mathcal{L} KW) \text{ s.t. } W^T KH\tilde{K}_{yy}HKW = I \tag{25}$$

- λ is another tradeoff parameter.
- $n^2 = (n_1 + n_2)^2$ is a normalisation term
- Optimise using Lagrangian multipliers.