Transfer learning

Pavel Severilov

Moscow Institute of Physics and Technology
(National Research University)

April 8th, 2020.

Definitions

Domain

A **domain** $\mathcal{D}$ is composed of two parts, i.e., a feature space $\mathcal{X}$ and a marginal distribution $P(X)$. In other words, $\mathcal{D} = \{\mathcal{X}, P(X)\}$. And the symbol X denotes an instance set, which is defined as $X = \{\mathbf{x} | \mathbf{x}_i \in \mathcal{X}, i = 1, \cdots, n\}$

Task

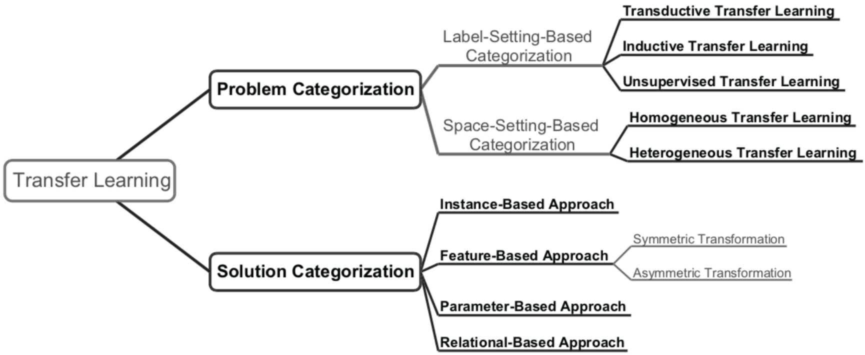
A **task** $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a decision function f, i.e., $\mathcal{T} = \{\mathcal{Y}, f\}$. The decision function f is an implicit one, which is expected to be learned from the sample data.

Definitions

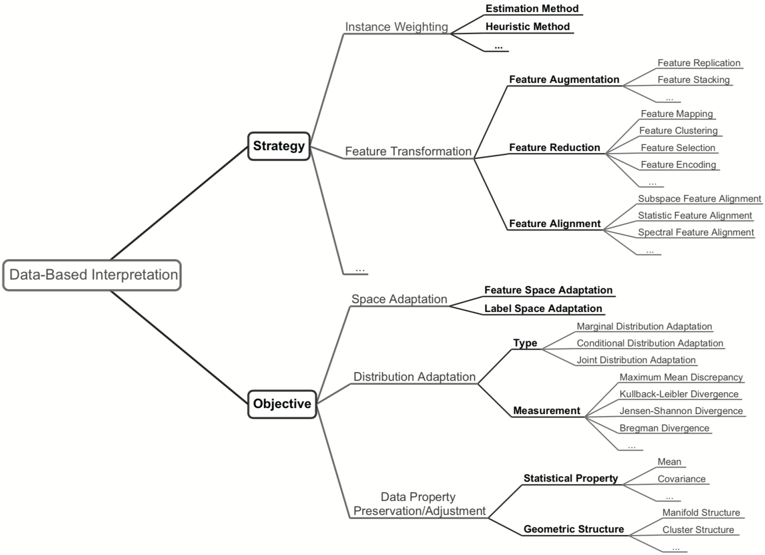
Transfer Learning

Given some observations corresponding to $m^S \in \mathbb{N}^+$ source domains and tasks (i.e., $\{(\mathcal{D}_{S_i}, \mathcal{T}_{S_i}) \,|\, i = 1, \cdots, m^S\}$), and some observations about $m^T \in \mathbb{N}^+$ target domains and tasks (i.e., $\left\{\left(\mathcal{D}_{T_j}, \mathcal{T}_{T_j}\right) |j = 1, \cdots, m^{\tilde{T}}\right\}$), **transfer learning** utilizes the knowledge implied in the source domains to improve the performance of the learned decision functions $f^{T_j}(j = 1, \cdots, m^T)$ on the target domains

Categorization



Data-based interpretation



Instance Weighting Strategy

- Scenario: a large number of labeled source-domain and a limited number of target-domain instances are available and $P^S(X) \neq P^T(X)$, $P^S(Y|X) = P^T(Y|X) \rightarrow$ adapting the marginal distributions.

$$\begin{aligned}\mathbb{E}_{(\mathbf{x},y)\sim P^T}[\mathcal{L}(\mathbf{x},y;f)] &= \mathbb{E}_{(\mathbf{x},y)\sim P^S}\left[\frac{P^T(\mathbf{x},y)}{P^S(\mathbf{x},y)}\mathcal{L}(\mathbf{x},y;f)\right] \\ &= \mathbb{E}_{(\mathbf{x},y)\sim P^S}\left[\frac{P^T(\mathbf{x})}{P^S(\mathbf{x})}\mathcal{L}(\mathbf{x},y;f)\right]\end{aligned}$$

- $\Rightarrow$ learning task:

$$\min_f \frac{1}{n^S}\sum_{i=1}^{n^S}\beta_i\mathcal{L}\left(f\left(\mathbf{x}_i^S\right), y_i^S\right) + \Omega(f),$$

where Ω – regularizer, $\beta_i\ \left(i = 1, \cdots, n^S\right)$ is the weighting parameter $\left(P^T\left(\mathbf{x}_i\right)/P^S\left(\mathbf{x}_i\right)\right)$.

Instance Weighting Strategy

- Kernel Mean Matching (KMM) resolves the estimation problem of β_i by

$$\arg\min_{\beta_i \in [0,B]} \left\| \frac{1}{n^S} \sum_{i=1}^{n^S} \beta_i \Phi\left(\mathbf{x}_i^S\right) - \frac{1}{n^T} \sum_{j=1}^{n^T} \Phi\left(\mathbf{x}_j^T\right) \right\|_{\mathcal{H}}^2$$

$$\text{s.t.} \left| \frac{1}{n^S} \sum_{i=1}^{n^S} \beta_i - 1 \right| \leq \delta,$$

  where $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ a map into a feature space, $\mathcal{H}$ – Reproducing Kernel Hilbert Space (RKHS), δ is a small parameter, and B is a parameter for constraint.

- There are some other studies attempting to estimate the weights.

Feature Transformation Strategy

Feature-based approaches transform each original feature into a new feature representation

Objective is to reduce the distribution difference of the source and the target domain instances: Maximum Mean Discrepancy (MMD) is widely used:

$$\mathrm{MMD}\left(X^S, X^T\right) = \left\| \frac{1}{n^S} \sum_{i=1}^{n^S} \Phi\left(\mathbf{x}_i^S\right) - \frac{1}{n^T} \sum_{j=1}^{n^T} \Phi\left(\mathbf{x}_j^T\right) \right\|_{\mathcal{H}}^2$$

- Feature Augmentation
- Feature Mapping, Clustering, Selection, Encoding
- Feature Alignment
- ...

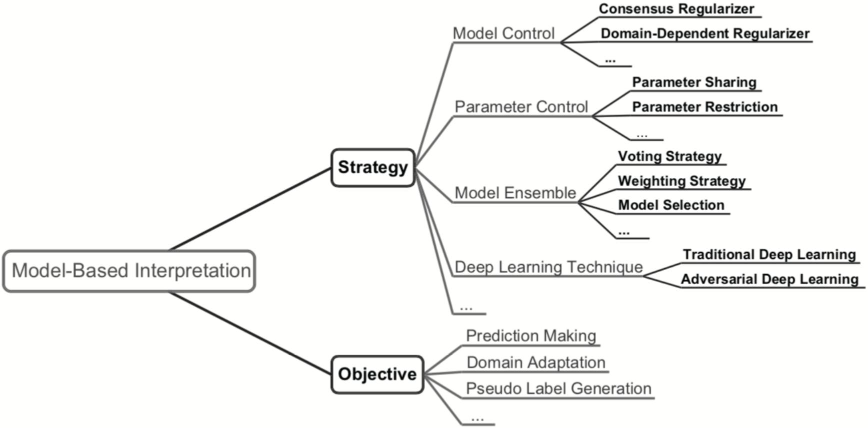
Example: Feature Mapping

Feature extraction:

$$\min_{\Phi} \left(\mathrm{DIST}\left(X^S, X^T; \Phi\right) + \lambda\Omega(\Phi)\right) / \left(\mathrm{VAR}\left(X^S \cup X^T; \Phi\right)\right)$$

This objective function aims to find a mapping function Φ that minimizes the marginal distribution difference between domains and meanwhile makes the variance of the instances as large as possible.

Model-based interpretation



Model Control Strategy

Idea: add the model-level regularizers to the learner's objective function
Example: Domain Adaptation Machine (DAM). The objective function is given by:

$$\min_{f^T} \mathcal{L}^{T,L}\left(f^T\right) + \lambda_1 \Omega^{\mathrm{D}}\left(f^T\right) + \lambda_2 \Omega\left(f^T\right)$$

where the first term represents the loss function, the second term denotes different data-dependent regularizer, and the third term is regularizer to control the complexity of the target classifier f^T

Other works varies by regularizers: Consensus Regularization Framework (CRF), Fast-DAM (specific algorithm of DAM), Univer-DAM (extension of the Fast-DAM)

Parameter Control and Model Ensemble Strategies

Parameter Control

Idea: share the parameters of the source learner to the target learner

Example: if we have a neural network for the source task, we can freeze most of its layers and only finetune the last few layers to produce a target network.

Model Ensemble

Idea: combine a number of weak classifiers to make the final predictions

Example: TaskTrAdaBoost: a group of candidate classifiers are constructed by performing AdaBoost on each source domain. Then a revised version of AdaBoost is performed on the target-domain instances to construct the final classifier.

Non-adversarial Deep Learning Technique

Example: Transfer Learning with Deep Autoencoders (TLDA)

$$X^S \xrightarrow{(W_1,b_1)} Q^S \xrightarrow[\text{Softmax Regression}]{(W_2,b_2)} R^S \xrightarrow{(\hat{W}_2,\hat{b}_2)} \tilde{Q}^S \xrightarrow{(\hat{W}_1,\hat{b}_1)} \tilde{X}^S,$$

$\Uparrow$

KL Divergence

$\Downarrow$

$$X^T \xrightarrow{(W_1,b_1)} Q^T \xrightarrow[\text{Softmax Regression}]{(W_2,b_2)} R^T \xrightarrow{(\hat{W}_2,\hat{b}_2)} \tilde{Q}^T \xrightarrow{(\hat{W}_1,\hat{b}_1)} \tilde{X}^T.$$

Autoencoders share the same parameters

TLDA

1. Reconstruction Error $\mathcal{L}_{\mathrm{REC}}$ Minimization: The output of the decoder should be extremely close to the input of encoder.
2. Distribution Adaptation: The distribution difference between Q^S and Q^T should be minimized.
3. Regression Error $\mathcal{L}_{\mathrm{REG}}$ Minimization: The output of the encoder on the labeled source-domain instances (R^S), should be consistent with the corresponding label information Y^S.

$\Rightarrow$ the objective function of TLDA:

$$\min_{\Theta} \mathcal{L}_{\mathrm{REC}}(X, \tilde{X}) + \lambda_1 \mathrm{KL}\left(Q^S \| Q^T\right) + \lambda_2 \Omega(W, b, \hat{W}, \hat{b}) + \lambda_3 \mathcal{L}_{\mathrm{REG}}\left(R^S, Y^S\right)$$

Adversarial Deep Learning Technique

- **Idea**: inspired by GANs find transferable representations that is applicable to both the source domain and the target domain
- **Example**: Domain-Adversarial Neural Network (DANN)

$$
\begin{array}{ccccc}
 & & \xrightarrow[\text{Predictor}]{\text{Label}} & \begin{array}{c}\hat{Y}^{S,L}\\ \hat{Y}^{T,U}\end{array} & \\
 & & \uparrow & & \\
\begin{array}{c}X^{S,L}\\ X^{T,U}\end{array} & \xrightarrow[\text{Extractor}]{\text{Feature}} & \left.\begin{array}{c}Q^{S,L}\\ Q^{T,U}\end{array}\right\} & \xrightarrow[\text{Classifier}]{\text{Domain}} & \begin{array}{c}\hat{S}\\ \hat{T}\end{array} \ (\text{Domain Label})
\end{array}
$$

The feature extractor acts like the generator, which aims to produce the domain-independent feature representation for confusing the domain classifier (discriminator). Output $\hat{Y}^{T,U}$ is the predicted labels of the unlabeled target-domain instances.