Transfer Learning

Adapted from a Survey of **Sinno Jialin Pan**
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

Transfer Learning: Definition

Transfer Learning (TL):
The ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks (in new domains)

It is motivated by human learning. People can often transfer knowledge learnt previously to novel situations

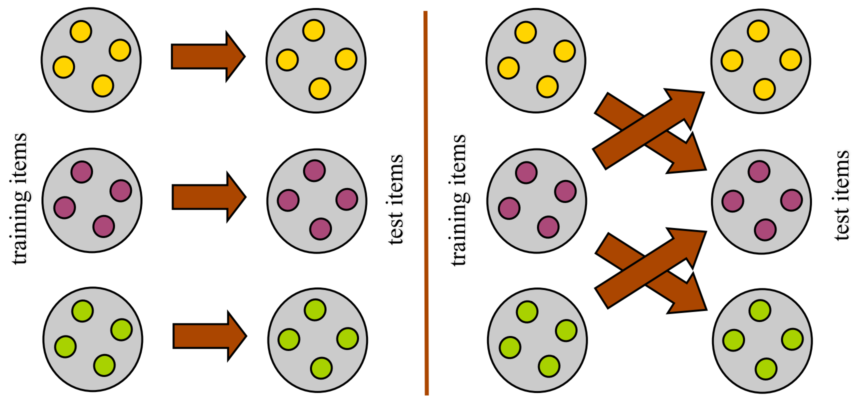
- ✓ Chess → Checkers
- ✓ Mathematics → Computer Science
- ✓ Table Tennis → Tennis

Traditional ML vs. TL

(P. Langley 06)

Traditional ML in multiple domains

Transfer of learning across domains



Humans can learn in many domains.

Humans can also transfer from one domain to other domains.

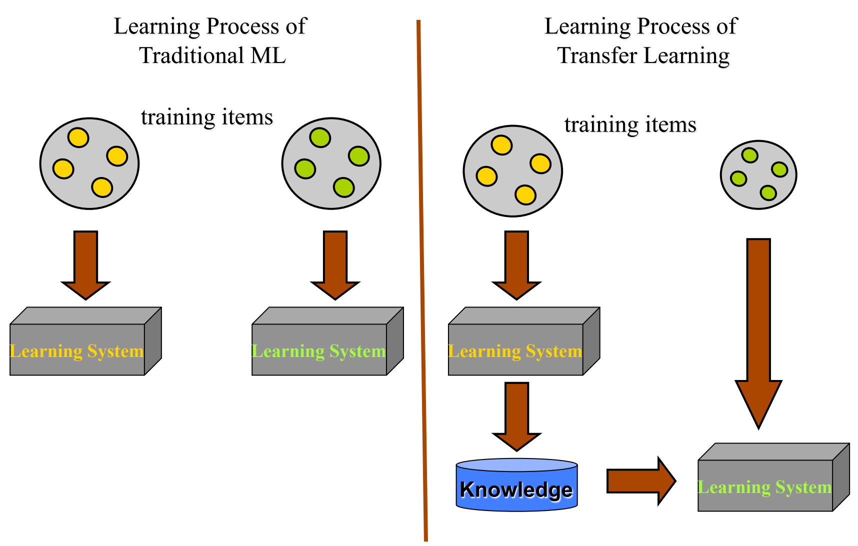
Motivation

- The need for transfer learning may arise when the data can be easily outdated
- It is expensive or impossible to re-collect training data and rebuild the models
- Knowledge transfer or transfer learning between task domains is desirable

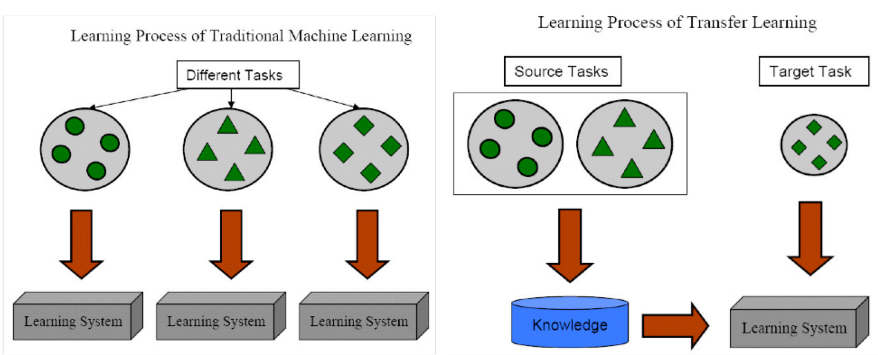
Examples

- Web document classification:
  - Labeled examples: University Web pages associated with category information via manual labeling
  - Task: classification of a newly created Web site where data features and data distributions might be different
- Sentiment classification:
  - Labeled examples: reviews of products (e.g., brand of a camera) with annotation (positive or negative review)
  - Task: classification of reviews of new products into positive or negative reviews

Traditional ML vs. TL



Traditional ML vs. TL



(a) Traditional Machine Learning (b) Transfer Learning

Notation

Domain:

It consists of two components: A feature space $\mathcal{X}$ **, a marginal distribution** $\mathcal{P}(X)$, where $X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$

In general, if two domains are different, then they may have different feature spaces or different marginal distributions.

Task:

Given a specific domain and label space $\mathcal{Y}$ **, for each** x_i **in the domain, to predict its corresponding label** y_i, where $y_i \in \mathcal{Y}$

In general, if two tasks are different, then they may have different label spaces or different conditional distributions

$\mathcal{P}(Y|X)$, where $Y = \{y_1, ..., y_n\}$ and $y_i \in \mathcal{Y}$

Notation

In many cases, two domains and two tasks are considered:

Source domain:

$$\mathcal{P}(X_S), \text{ where } X_S = \{x_{S_1}, x_{S_2}, ..., x_{S_{n_S}}\} \in \mathcal{X}_S$$

Task in the source domain:

$$\mathcal{P}(Y_S|X_S), \text{ where } Y_S = \{y_{S_1}, y_{S_2}, ..., y_{S_{n_S}}\} \text{ and } y_{S_i} \in \mathcal{Y}_S$$

Target domain:

$$\mathcal{P}(X_T), \text{ where } X_T = \{x_{T_1}, x_{T_2}, ..., x_{T_{n_T}}\} \in \mathcal{X}_T$$

Task in the target domain

$$\mathcal{P}(Y_T|X_T), \text{ where } Y_T = \{y_{T_1}, y_{T_2}, ..., y_{T_{n_T}}\} \text{ and } y_{T_i} \in \mathcal{Y}_T$$

Transfer Learning

- Given a source domain, a learning task in the source domain, a target domain, and a learning task in the target domain, transfer learning aims to help improve the learning of the task in the target domain using the knowledge achieved in the source domain

Assumption:

source domain <> target domain

or

source learning task <> target learning task

Why Transfer Learning?

- In some domains, labeled data are in short supply.
- In some domains, the calibration effort is very expensive.
- In some domains, the learning process is time consuming.

- *How to extract knowledge learnt from related domains to help learning in a target domain with a few labeled data?*
- *How to extract knowledge learnt from related domains to speed up learning in a target domain?*

✌ **Transfer learning techniques may help!**

What/How/When

Three main research issues in transfer learning:

- What to transfer
- How to transfer
- When to transfer (avoid negative transfer)

Inductive Transfer Learning

Instance-transfer Approaches

- Assumption: the source domain and target domain data use exactly the same features and labels.
- Motivation: Although the source domain data can not be reused directly, there are some parts of the data that can still be reused by re-weighting.
- Main Idea: Discriminatively adjust weighs of data in the source domain for use in the target domain.

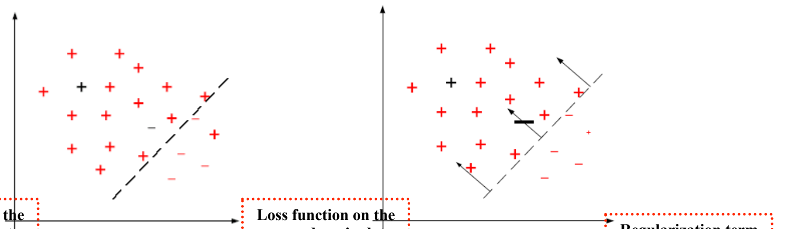
Inductive Transfer Learning

--- Instance-transfer Approaches

Non-standard SVMs

[Wu and Dietterich ICML-04]

Uniform weights | **Correct the decision boundary by re-weighting**



Loss function on the target domain data | **Loss function on the source domain data** | **Regularization term**

- Differentiate the cost for misclassification of the target and source data

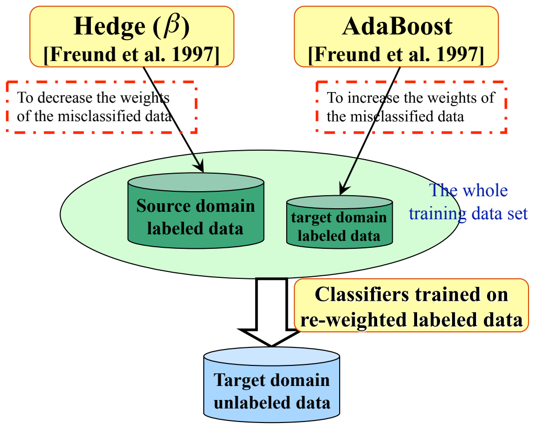
$$J(h) = \sum_{i}^{n_T} L(h(x_{T_i}), y_{T_i}) + \lambda \sum_{j}^{n_S} L(h(x_{S_j}), y_{T_j}) + R(h)$$

Inductive Transfer Learning

--- Instance-transfer Approaches

TrAdaBoost

[Dai et al. ICML-07]



Boosting for Transfer Learning

- Given: few labeled data in target domain; many labeled data in source domain.
- The data in target domain are not sufficient to train a model.
- The method allows users to use a small amount of newly labeled data to leverage the old data and construct a high-quality classification model for the new data.

Boosting for Transfer Learning

- Which old data are useful for the target domain?
- Although training data in source domain are out-dated, parts of the data can still be reused in training a classifier for the new data.
- Idea: leverage the training data in target domain to vote on the usefulness of each of the training data in source domain.

Boosting for Transfer Learning

- **Same-distribution training data**: training data in target domain. They have the same distribution as test data.
- **Diff-distribution training data**: training data in source domain. They may have a different distribution from test data.
- Task: use boosting to filter out the diff-distribution training data that are very different from the same-distribution data by automatically adjusting the weights of training instances. Use remaining diff-distribution data as additional training instances.

Boosting (review)

- Boosting creates an ensemble of classifiers by re-sampling the data, which are then combined by majority voting
- In boosting, the re-sampling strategy is geared to provide the **most informative training data for each consecutive classifier**

Boosting (Adaboost.M1) Freund and Schapire, 1996

- Generates a set of classifiers, and combines them through weighted majority voting of the classes predicted by the individual classifiers
- Classifiers are trained using instances drawn from an iteratively updated distribution of the training data
- The distribution ensures that instances misclassified by the previous classifier are more likely to be included in the training data of the next classifier
- Thus, consecutive classifiers' training data are more geared towards increasingly hard-to-classify instances

Algorithm AdaBoost.M1

Input:

- Sequence of N examples $S = [(\mathbf{x}_i, y_i)], i = 1, \cdots, N$ with labels $y_i \in \Omega, \Omega = \{\omega_1, \ldots, \omega_C\}$;
- Weak learning algorithm **WeakLearn**;
- Integer T specifying number of iterations.

Initialize $D_1(i) = \frac{1}{N}, i = 1, \cdots, N$ (11)

Do for $t = 1, 2, \ldots, T$:

1. Select a training data subset S_t, drawn from the distribution D_t.
2. Train **WeakLearn** with S_t, receive hypothesis h_t.
3. Calculate the error of h_t: $\varepsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i)$. (12)

   If $\varepsilon_t > 1/2$, **abort**.
4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$. (13)
5. Update distribution

$$D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(\mathbf{x}_i) = y_i \\ 1, & \text{otherwise} \end{cases} \quad (14)$$

   where $Z_t = \sum_i D_t(i)$ is a normalization constant chosen so that D_{t+1} becomes a proper distribution function.

Test – Weighted Majority Voting: Given an unlabeled instance $\mathbf{x}$,

1. Obtain total vote received by each class

$$V_j = \sum_{t:h_t(\mathbf{x}) = \omega_j} \log \frac{1}{\beta_t}, \ j = 1, \ldots, C. \quad (15)$$

2. Choose the class that receives the highest total vote as the final classification.

Boosting (property)

- Freund and Schapire proved that, provided that is always $\epsilon_t < 0.5$, the error rate of boosting on a given training data set, under the original uniform distribution, approaches zero exponentially fast as T increases.

Boosting (property)

- Thus, a succession of weak classifiers can be boosted to a strong classifier that is at least as accurate as, and usually more accurate than, the best weak classifier on the training data.

Boosting for Transfer Learning

Notation:

- T_d : diff-distribution training data (size n)
- T_s : same-distribution training data (size m)
- $T = \{(x_i, y_i)\}$: combined training set (size n+m)

$$x_i = \begin{cases} x_i^d, & i = 1,\cdots,n \\ x_i^s, & i = n+1,\cdots,n+m \end{cases}$$

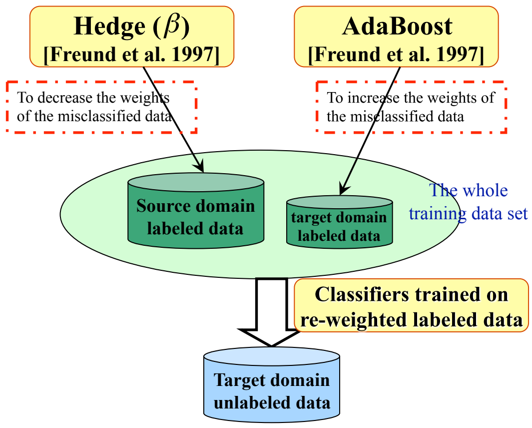
- Train a classifier $\hat{c} : X \to Y$ that minimizes prediction error on the unlabeled data set S. We assume $Y \in \{0,1\}$

Inductive Transfer Learning

--- Instance-transfer Approaches

TrAdaBoost

[Dai et al. ICML-07]



Algorithm 1 `TrAdaBoost`

Input the two labeled data sets T_d and T_s, the unlabeled data set S, a base learning algorithm **Learner**, and the maximum number of iterations N.

Initialize the initial weight vector, that $\mathbf{w}^1 = (w_1^1, \ldots, w_{n+m}^1)$. We allow the users to specify the initial values for $\mathbf{w}^1$.

For $t = 1, \ldots, N$

1. Set $\mathbf{p}^t = \mathbf{w}^t / (\sum_{i=1}^{n+m} w_i^t)$.
2. Call **Learner**, providing it the combined training set T with the distribution $\mathbf{p}^t$ over T and the unlabeled data set S. Then, get back a hypothesis $h_t : X \to Y$ (or $[0, 1]$ by confidence).
3. Calculate the error of h_t on T_s:
$$\epsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t \cdot |h_t(x_i) - c(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t}.$$
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$ and $\beta = 1/(1 + \sqrt{2 \ln n / N})$. Note that, ϵ_t is required to be less than $1/2$.
5. Update the new weight vector:
$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i) - c(x_i)|}, & 1 \le i \le n \\ w_i^t \beta_t^{-|h_t(x_i) - c(x_i)|}, & n+1 \le i \le n+m \end{cases}$$

Output the hypothesis

$$h_f(x) = \begin{cases} 1, & \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)} \ge \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-\frac{1}{2}} \\ 0, & \text{otherwise} \end{cases}$$

Experimental Evaluation

- Datasets: 20 Newsgroups, SRAA, Reuters-21578

Data Set	KL-divergence	Size	
		$\|T_d\|$	$\|T_s \cup S\|$
rec vs talk	1.102	3,669	3,561
rec vs sci	1.021	3,961	3,965
sci vs talk	0.854	3,374	3,828
auto vs aviation	1.126	8,000	8,000
real vs simulated	1.048	8,000	8,000
orgs vs people	0.303	1,016	1,046
orgs vs places	0.329	1,079	1,080
people vs places	0.307	1,239	1,210
edible vs poisonous	1.315	4,608	3,516

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Experimental Evaluation

- Baseline methods

Baseline	Training Data		Test Data	Basic Learner
	labeled	unlabeled		
SVM	T_s	$\emptyset$	S	SVM
SVMt	$T_s \cup T_d$	$\emptyset$	S	SVM
TSVM	T_s	S	S	TSVM
TSVMt	$T_s \cup T_d$	S	S	TSVM

Experimental Results

- Error rates for supervised learning
- (same-distr/diff-distr=0.01):

Data Set	SVM	SVMt	AUX	TrAdaBoost(SVM)
rec vs talk	0.222	0.127	0.127	**0.080**
rec vs sci	0.240	0.164	0.153	**0.097**
sci vs talk	0.234	0.177	0.173	**0.125**
auto vs aviation	0.131	0.192	0.188	**0.096**
real vs simulated	0.140	0.219	0.210	**0.119**
orgs vs people	0.494	0.285	0.287	**0.280**
orgs vs places	0.423	0.440	0.433	**0.315**
people vs places	0.412	0.255	0.257	**0.216**
edible vs poisonous	0.127	0.135	0.082	**0.071**

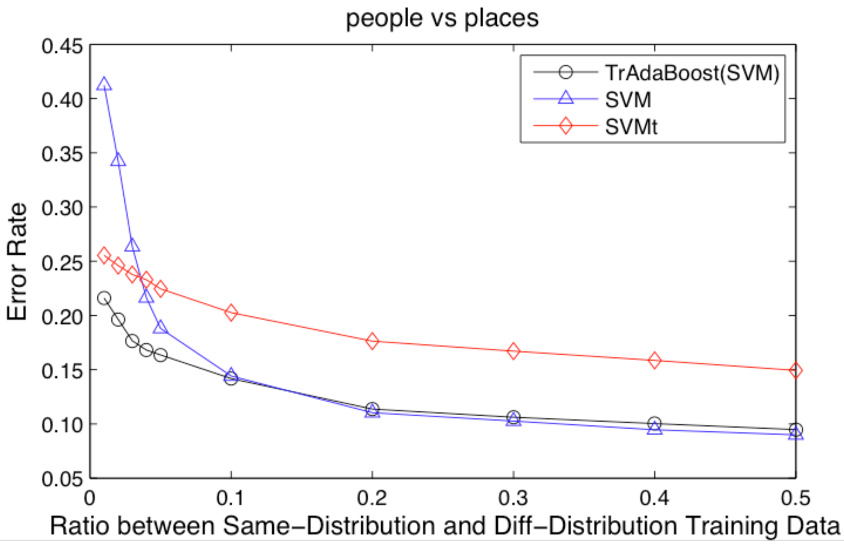
Experimental Results

- Error rates for semi-supervised learning
- (only 1 positive and 1 negative example in same-distr):

Data Set	TSVM	TSVMt	TrAdaBoost(TSVM)
rec vs talk	0.059	0.040	**0.021**
rec vs sci	0.067	0.062	**0.013**
sci vs talk	0.173	0.106	**0.075**
auto vs aviation	0.043	0.103	**0.038**
real vs simulated	0.144	0.131	**0.102**
orgs vs people	0.358	0.292	**0.248**
orgs vs places	0.424	0.436	**0.304**
people vs places	0.307	0.225	**0.179**
edible vs poisonous	0.439	0.179	**0.160**

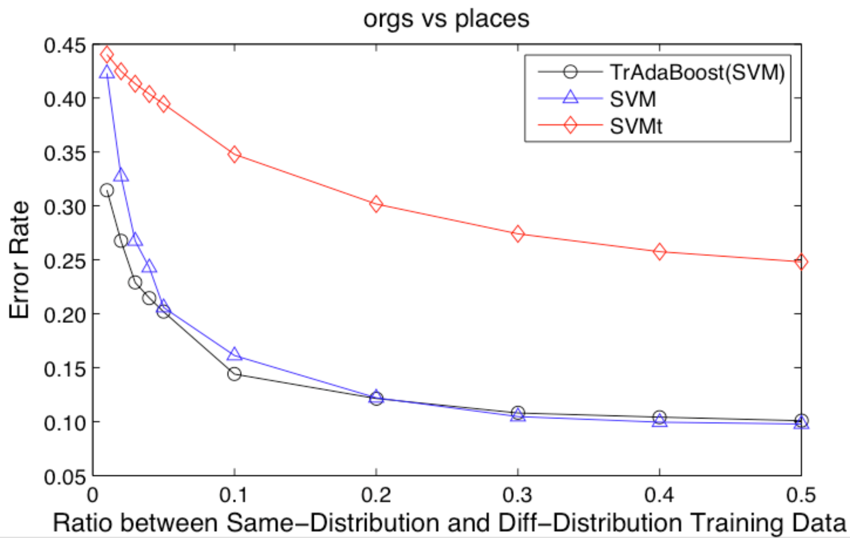
Experimental Results

- Does transfer learning always help?



Experimental Results

- Does transfer learning always help?

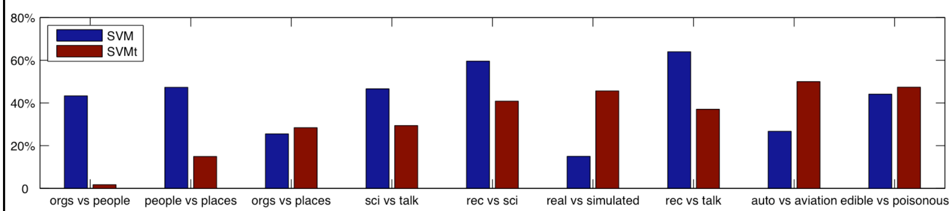


Experimental Results

- Transfer learning does not always reduce the generalization error, and sometimes even lower the performance on the test set (Caruana, 1997)
- This phenomenon is called *negative transfer* (similar to cross-talk phenomenon in neural networks)
- TrAdaBoost does not guarantee to improve the basic learner either

Experimental Results

- Error reduction vs. KL-divergence



Issues

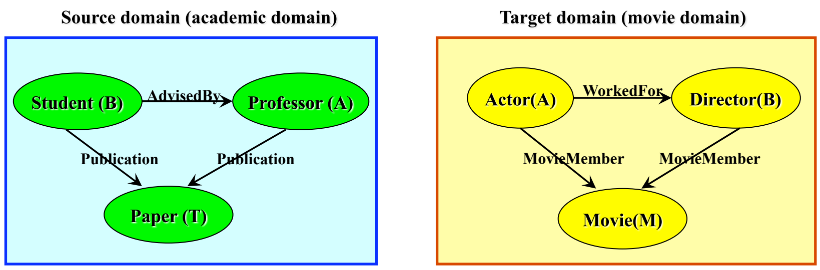
- The authors (Dai et al., ICML 2007) have shown convergence of the prediction error on the same distribution data, but the improvement is sensitive to the quality (or KL-divergence) of diff-distribution examples.
- Can we analyze auxiliary data and determine in a principled way whether transfer learning will help? (when? question)
- TrAdaBoost can transfer knowledge from only one distribution. How can we deal with multiple distributions simultaneously?

Inductive Transfer Learning

Relational-knowledge-transfer Approaches

TAMAR

[Mihalkova et al. AAAI-07]



Inductive Transfer Learning

Relational-knowledge-transfer Approaches

TAMAR

[Mihalkova et al. AAAI-07]

Assumption: If the target domain and source domain are related, then there may be some relationship between domains being similar, which can be used for transfer learning

Input:

1. Relational data in the source domain and a statistical relational model, Markov Logic Network (MLN), which has been learnt in the source domain.
2. Relational data in the target domain.

Output: A new statistical relational model, MLN, in the target domain.

Goal: To learn a MLN in the target domain more efficiently and effectively.