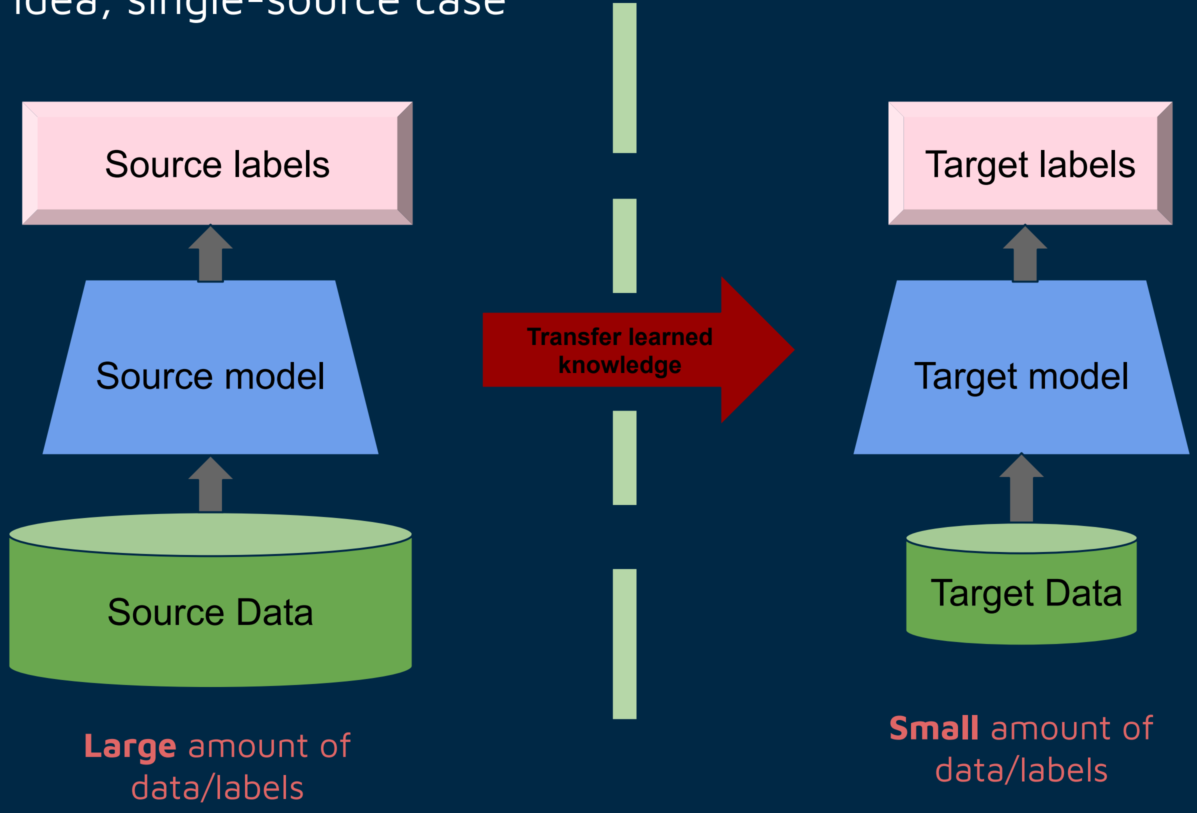
Towards more Reliable
Transfer Learning

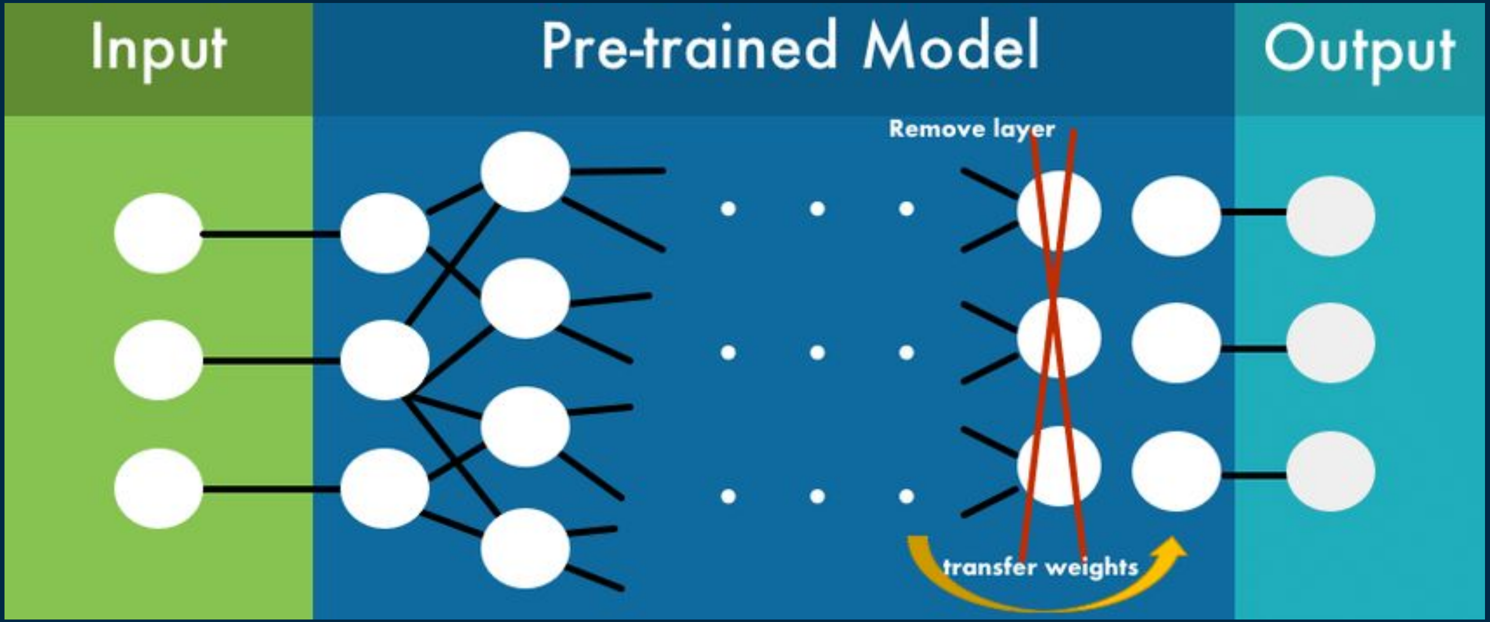
Table of contents

Transfer Learning

- Basic idea, single-source case

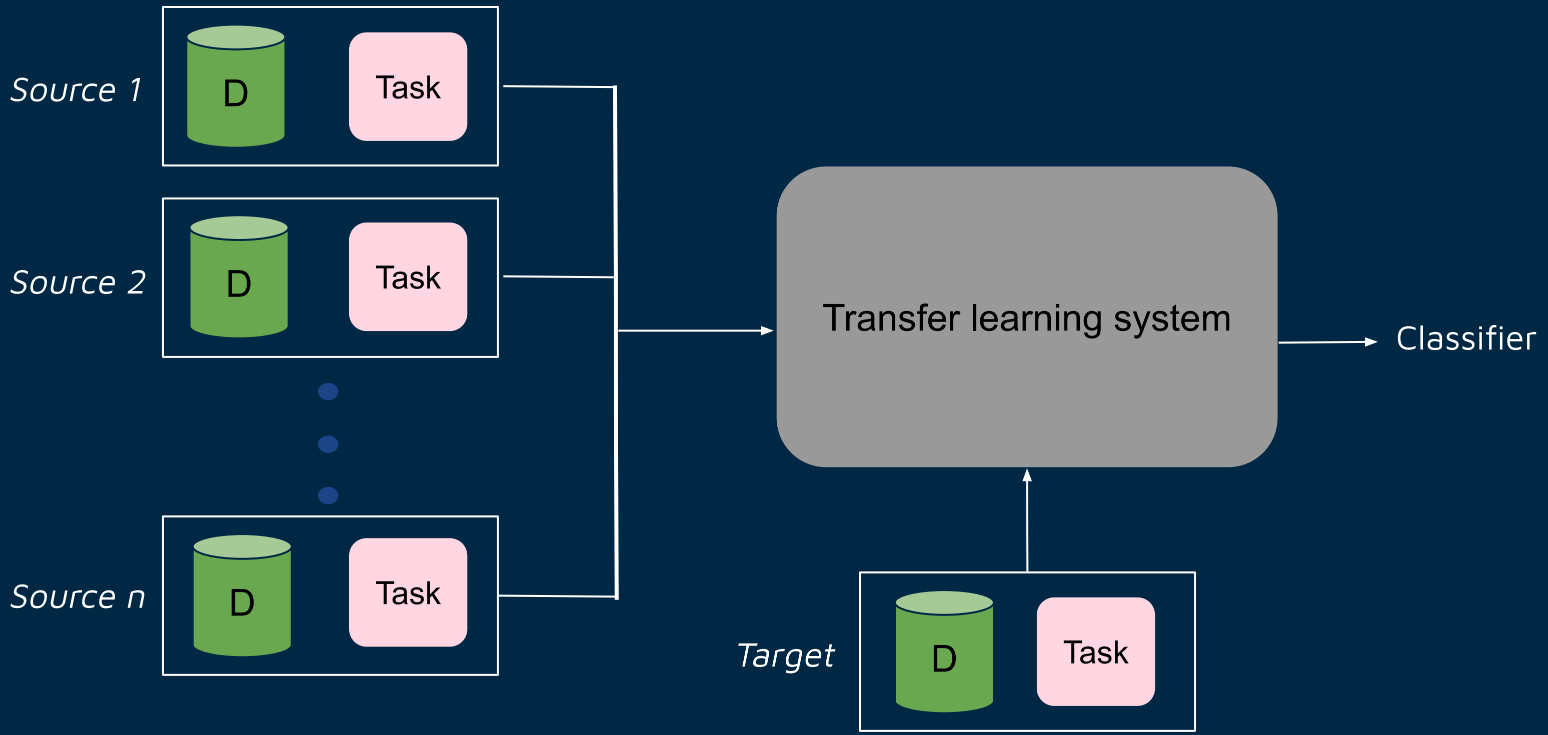


Transfer Learning



Transfer Learning

- Multi-source case



Transfer Learning

- Ideally : sources are relevant and reliable

- In a multi-source scenario, there is an assumption :

*All sources are **equally reliable**, i.e. have labeled data of the same or comparable **quantity** and **quality***

Motivations

- What if sources have diverse reliabilities ?
  have different relations to the target task ?

⇒ sources with different quality and quantity of labeled data and different proximity to a target task

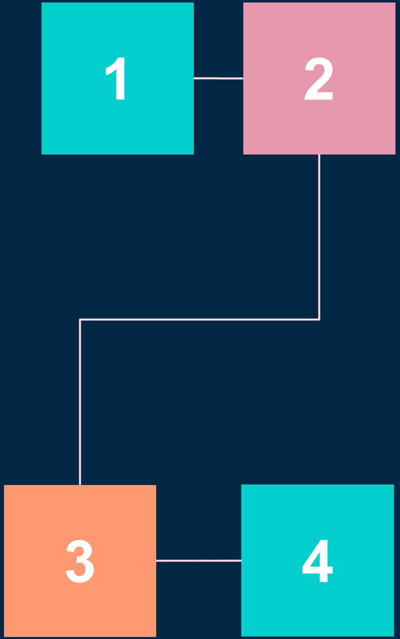
Challenges

1) Create a transfer learning method combining *domain similarity* and *sources reliabilities* (**PW-MSTL**)

2) Create an **active** transfer learning incorporating *distribution matching* and *uncertainty sampling* (**AMSAT**)

Context

- K auxiliary sources
- $S_k = S_k^L \cup S_k^U$
- Unlabeled target data



Re-weighted MMD

Adaptation of the Maximum Mean Discrepancy for determining the weights of the k source aggregate data

Relationship matrix

This matrix captures inter-source similarities by computing the classification error made by an estimator, trained on a source S_j, on the source S_i compared to all other estimator on the same source S_i

$$R_{i,j} = \begin{cases} \dfrac{exp(\beta \hat{\varepsilon}_{S_i}(\hat{h}_j))}{\sum_{j' \in [K], j' \neq i} exp(\beta \hat{\varepsilon}_{S_i}(\hat{h'}_j))}, & i \neq j \\ 0, & otherwise \end{cases}$$

Source importance weights

It is parametrized considering both source proximity and reliability

K : vector measuring pairwise source-target proximities

Mu : concentration factor

R : the source relationship matrix

ALGORITHMS

01

Peer-weighted multi-source transfer learning (PW-MSTL)

Algorithm 1: PW-MSTL

input: $S = S^L \cup S^U$: source data; T: target data; μ : concentration factor; b : confidence tolerance

for $k = 1, ..., K$ **do**

  Compute α^k by solving (1)

  Train a classifier $\hat{h}_k$ on the α^k weighted S_k^L

Compute δ and R by computing (2)

Compute w by computing (3)

for $t = 1, ..., T$ **do**

  for $k = 1, ..., K$ **do**

    if $\left|\hat{h}_k(x^{(t)})\right| < b$ **then**

      $\hat{p}_k^{(t)} = \sum_{m \in [K], m \neq k} R_{km} \left|\hat{h}_m(x^{(t)})\right|$

    else

      $\hat{p}_k^{(t)} = \left|\hat{h}_k(x^{(t)})\right|$

  $\hat{y}(t) = sign(\sum_{k \in [K]} w_k \hat{p}_k^{(t)})$

Goal : predict the class of a target data by performing transfer learning over all sources.

Peer-weighted multi-source transfer learning (PW-MSTL)

Algorithm 1: PW-MSTL

input: $S = S^L \cup S^U$: source data; T: target data; μ : concentration factor; b : confidence tolerance

for $k = 1, ..., K$ **do**
  Compute α^k by solving (1)
  Train a classifier $\hat{h}_k$ on the α^k weighted S_k^L
Compute δ and R by computing (2)
Compute w by computing (3)
for $t = 1, ..., T$ **do**
  for $k = 1, ..., K$ **do**
    if $\left|\hat{h}_k(x^{(t)})\right| < b$ **then**
      $\hat{p}_k^{(t)} = \sum_{m\in[K],m\neq k} R_{km} \left|\hat{h}_m(x^{(t)})\right|$
    else
      $\hat{p}_k^{(t)} = \left|\hat{h}_k(x^{(t)})\right|$
  $\hat{y}^{(t)} = sign(\sum_{k\in[K]} w_k \hat{p}_k^{(t)})$

$$min_{\alpha^k} \left\| \frac{1}{n_k^L + n_k^U} \sum_{i=1}^{n_k^L+n_k^U} \alpha_i^k \phi(x_i^{S_k}) - \frac{1}{n_T} \sum_{i=1}^{n_T} \phi(x_i^T) \right\|_H^2$$

+

$$R_{i,j} = \begin{cases} \frac{exp(\beta \hat{\varepsilon}_{S_i}(\hat{h}_j))}{\sum_{j'\in[K],j'\neq i} exp(\beta \hat{\varepsilon}_{S_i}(\hat{h'}_j))}, & i \neq j \\ 0, & otherwise \end{cases}$$

$$\omega = \delta * [\mu \mathbf{I}_K + (1-\mu)\mathbf{R}]$$

Peer-weighted multi-source transfer learning (PW-MSTL)

Algorithm 1: PW-MSTL

input: $S = S^L \cup S^U$: source data; T: target data; μ : concentration factor; b : confidence tolerance

for $k = 1, ..., K$ **do**

  Compute α^k by solving (1)

  Train a classifier $\hat{h}_k$ on the α^k weighted S_k^L

Compute δ and R by computing (2)

Compute w by computing (3)

for $t = 1, ..., T$ **do**

  for $k = 1, ..., K$ **do**

    if $\left|\hat{h}_k(x^{(t)})\right| < b$ **then**

      $\hat{p}_k^{(t)} = \sum_{m \in [K], m \neq k} R_{km} \left|\hat{h}_m(x^{(t)})\right|$

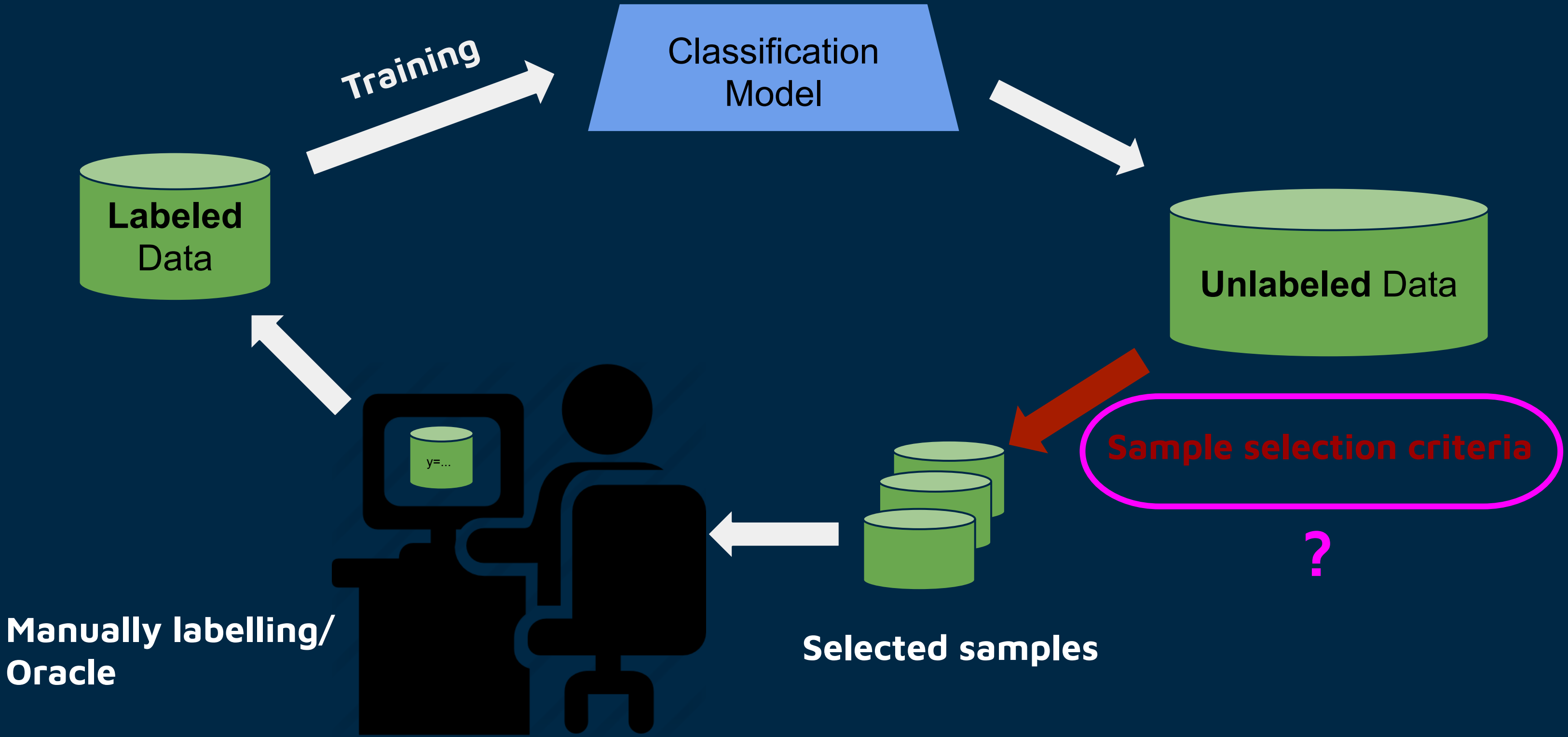
    else

      $\hat{p}_k^{(t)} = \left|\hat{h}_k(x^{(t)})\right|$

  $\hat{y}(t) = sign(\sum_{k \in [K]} w_k \hat{p}_k^{(t)})$

Classify testing instances by weighted vote by the source weights coefficient

Adaptive multi-source active transfer (AMSAT)



Adaptive multi-source active transfer (**AMSAT**)

How to select unlabeled instances that are the **most representative** and avoid **information redundancy** ?

Adaptive multi-source active transfer (**AMSAT**)

Algorithm 2: AMSAT

input: $S = S^L \cup S^U$: source data; T: target data; μ : concentration factor; B : Budget

for $k = 1, ..., K$ **do**
- Compute α^k by solving (1)
- Train a classifier $\hat{h}_k$ on the α^k weighted S_k^L.

for $t = 1, ..., B$ **do**
- Compute $\beta_i^{(t)} = \frac{n_i^L}{\sum_i n_i^L}$
- Draw a Bernoulli random variable $P^(t)$ with probability $D_{KL}(\beta^{(t)}||uniform)$.
- **if** $P^(t) = 1$ **then**
  - Set $Q^{(t)} = \frac{1}{\beta^{(t)}}$
- **else**
  - Compute $w^{(t)}$ as (3) and set $Q^{(t)} = w^{(t)}$
- Draw $k^{(t)}$ from $[K]$ with distribution $Q^{(t)}$.
- Select $x^{(t)}$ according to (4) and query the label for it.
- Update $S_{k^{(t)}}^L \leftarrow S_{k^{(t)}}^L \cup \{x^{(t)}\}$
- $Update$S$_{k^{(t)}}^L \leftarrow S_{k^{(t)}}^L$ \; $\{x^{(t)}\}$
- Update classifier $\hat{h}_k$.

AMSAT is a 2 stage active learning framework assuming the **budget** limited availability of an oracle in the source domains.

Adaptive multi-source active transfer (**AMSAT**)

Algorithm 2: AMSAT

input: $S = S^L \cup S^U$: source data; T: target data; μ : concentration factor; B : Budget

for $k = 1, ..., K$ **do**
  Compute α^k by solving (1)
  Train a classifier $\hat{h}_k$ on the α^k weighted S^L_k.

for $t = 1, ..., B$ **do**
  Compute $\beta_i^{(t)} = \frac{n_i^L}{\sum_i n_i^L}$
  Draw a Bernoulli random variable $P^(t)$ with probability $D_{KL}(\beta^{(t)}||uniform)$.
  if $P^(t) = 1$ **then**
    Set $Q^{(t)} = \frac{1}{\beta^{(t)}}$
  else
    Compute $w^{(t)}$ as (3) and set $Q^{(t)} = w^{(t)}$
  Draw $k^{(t)}$ from $[K]$ with distribution $Q^{(t)}$.
  Select $x^{(t)}$ according to (4) and query the label for it.
  Update $S^L_{k^{(t)}} \leftarrow S^L_{k^{(t)}} \cup \{x^{(t)}\}$
  $Update$S$^L_{k^{(t)}} \leftarrow S^L_{k^{(t)}}$ \;
  $\{x^{(t)}\}$
  Update classifier $\hat{h}_k$.

AMSAT selects source domain to query based on 2 criteria :

- If sources are too unbalanced, more likely to explore less labeled sources.
- If sources are balanced, more likely to exploit more useful source.

Adaptive multi-source active transfer (**AMSAT**)

Algorithm 2: AMSAT

input: $S = S^L \cup S^U$: source data; T: target data; μ : concentration factor; B : Budget

for $k = 1, ..., K$ **do**

  Compute α^k by solving (1)

  Train a classifier $\hat{h}_k$ on the α^k weighted S^L_k.

for $t = 1, ..., B$ **do**

  Compute $\beta_i^{(t)} = \frac{n_i^L}{\sum_i n_i^L}$

  Draw a Bernoulli random variable $P^{(}t)$ with probability $D_{KL}(\beta^{(t)} || uniform)$.

  if $P^{(}t) = 1$ **then**

    Set $Q^{(t)} = \frac{1}{\beta^{(t)}}$

  else

    Compute $w^{(t)}$ as (3) and set $Q^{(t)} = w^{(t)}$

  Draw $k^{(t)}$ from $[K]$ with distribution $Q^{(t)}$.

  Select $x^{(t)}$ according to (4) and query the label for it.

  Update $S^L_{k(t)} \leftarrow S^L_{k(t)} \cup \{x^{(t)}\}$

  $Update S^L_{k(t)} \leftarrow S^L_{k(t)} \setminus \{x^{(t)}\}$;

  Update classifier $\hat{h}_k$.

AMSAT selects a source among the K ones according to Q distribution...

...and queries the most informative instance by solving :

$$x = argmax_{x_i \in S^U_{k(t)}} E[(\hat{y}_i - y_i)^2 | x_i] \alpha_i^{k(t)}$$

RESULTS

02

Datasets

- Synthetic dataset (randomly generated data)

- Spam Detection
  - Discovery challenge : Several separate inboxes but with only little training data and little unlabeled data in the inboxes available. To be successful in this setting we assume that a learning algorithm needs to generalize over the different users in a way that user specific properties are taken into account but the data from the other users is utilized in a way that enhances the classification performance.

- Sentiment Analysis
  - Multi-Domain Sentiment Dataset : The Multi-Domain Sentiment Dataset contains product reviews taken from Amazon.com from many product types (domains). Some domains (books and dvds) have hundreds of thousands of reviews. Others (musical instruments) have only a few hundred.

Results (PW-MSTL)

Table 1. Classification accuracy (%) on the target domain, given that source domains contain diverse {1%,5%,15%,30%} labeled data.

Method	Synthetic		Spam			Sentiment									
	case1	case2	user7	user8	user3	electronics	toys	music	apparel	dvd	kitchen	video	sports	book	health
KMM	82.7	88.8	92.0	91.8	89.7	77.6	77.4	71.0	78.3	72.4	78.4	72.1	79.1	71.2	77.4
KMM-A	87.3	91.4	92.0	92.0	91.8	74.6	76.3	70.3	75.8	72.4	75.2	70.5	76.7	69.7	74.9
A-SVM	70.8	89.4	84.5	87.8	86.8	70.8	73.7	67.7	73.6	62.6	72.8	62.5	73.7	66.9	71.4
DAM	75.8	91.0	83.8	85.4	86.8	71.3	73.7	68.0	75.1	62.5	72.1	62.0	73.0	68.0	72.5
PW-MSTL$_b$	85.5	90.8	91.5	92.6	90.3	78.0	78.7	70.7	79,5	73.2	78.3	72.5	79.5	71.5	77.7
PW-MSTL	**88.4**	**92.6**	**93.8**	**95.6**	**92.8**	**79.3**	**81.9**	**74.6**	**82.7**	**76.7**	**80.7**	**76.2**	**82.7**	**74.8**	**80.9**

$\%_L$	Method	Synthetic	Spam			Sentiment				
			user7	user8	user3	electronics	toys	music	apparel	dvd
10%	KMM	87.0	89.1	91.2	90.3	75.0	74.6	68.3	75.6	70.2
	KMM-A	91.1	91.3	90.7	91.0	74.8	76.5	70.2	76.8	71.3
	A-SVM	89.4	88.4	91.9	89.2	77.1	78.1	69.9	78.2	68.9
	DAM	89.7	89.6	90.4	91.3	77.5	79.0	69.9	79.8	69.0
	PW-MSTL$_b$	90.2	89.7	92.4	92.1	77.7	78.7	69.7	78.9	73.5
	PW-MSTL	91.2	**92.5**	**94.9**	**93.1**	**79.8**	**81.5**	**73.3**	**81.3**	**76.4**
50%	KMM	95.6	92.6	94.0	91.8	81.6	81.7	75.0	82.2	76.9
	KMM-A	97.2	91.4	93.8	**94.7**	80.4	82.4	74.5	82.7	77.1
	A-SVM	96.4	91.5	95.2	93.4	81.7	83.4	74.7	84.3	76.0
	DAM	96.6	92.7	93.1	93.2	83.5	84.5	73.4	84.4	77.3
	PW-MSTL$_b$	96.6	92.9	95.2	93.5	83.6	84.7	74.4	85.0	80.4
	PW-MSTL	97.2	**94.5**	**95.7**	93.7	**84.8**	**86.4**	**76.9**	**87.2**	**82.0**

Results (AMSAT)

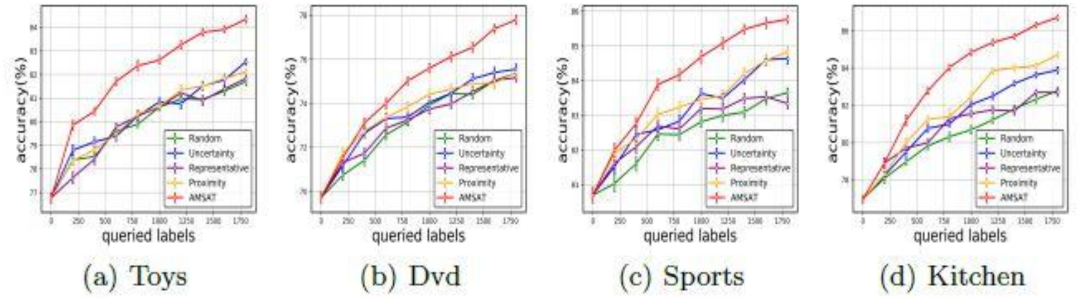
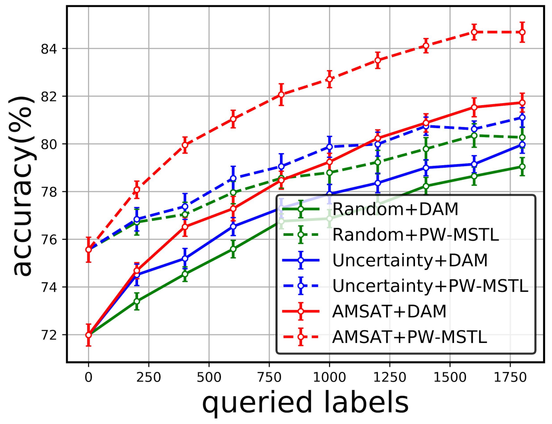


Fig. 2. Performance comparison of active learning methods on *Sentiment*: initial labeled fractions randomly selected from $\{1\%, 5\%, 15\%, 30\%\}$.



EXPERIMENTS

03

Experiments

The authors provide a git repository.

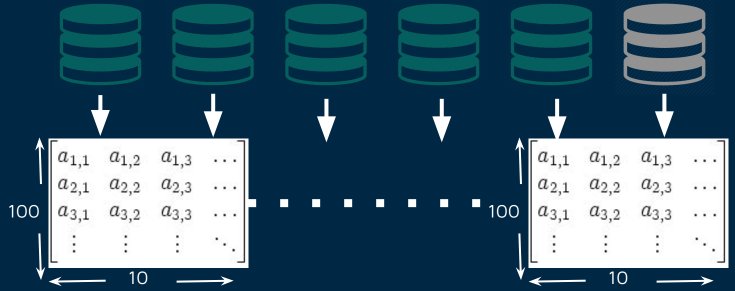
Unfortunately, these python files only generate uniform sample to simulate data from various sources.

The data used for experiments is not available and the preprocessed is not describe

Algo	k, d, n, budget	Sample Generation	Base_model	Accuracy on Test set
PW-MSTL	10, 10, 100, na	Uniform distribution	LinearSVC	0.501
	5, 10, 100, na			0.496
AMSAT	10, 10, 100, 100			Before AL: 0.513 After AL: 0.507
	5, 10, 100, na			Before AL: 0.510 After AL: 0.509

Synthetic experiments : [PW-MSTL]

Synthetic dataset. We generate a synthetic data for 5 source domains and 1 target domain. The samples $x \in \mathbb{R}^{10}$ are drawn from Gaussian distributions $\mathcal{N}(\mu_T + \mathbf{p}\Delta\mu, \sigma)$, where μ_T is the mean of the target domain, $\Delta\mu$ is a random fluctuation vector and $\mathbf{p}$ is the variable controlling the proximity between each source and the target (higher $\mathbf{p}$ indicates lower proximity). We then consider a labeling function $f(x) = sign((w_0^T + \delta\Delta w)x + \epsilon)$, where w_0 is a fixed base vector, Δw is a random fluctuation vector and ϵ is a zero-mean Gaussian noise term. We set δ to small values as we assume labeling functions are similar. Using different $\mathbf{p}$ values, We generate 50 positive points and 50 negative points as training data for each source domain and additional 100 balanced testing samples for the target domain.



$$\mathcal{N}(\mu_T + \mathbf{p}\Delta\mu,\ \sigma^2)$$

p : proximity parameters [0.00001, 0.0001, 0.002, 0.0005, 0.001]

$\Delta\mu$: uniform random distribution

μ_T : mean of target domain

$f(x) = sign((w_0^T + \delta\Delta w)x + \epsilon)$ → labellisation

ϵ & Δw ~ zero-mean gaussian distribution , $w^T_0 = 1$, $\delta = 1$, σ = 1

Algo	k, d, n, budget	proximity	Sample Generation	Base_model	Accuracy on Test set
PW-MSTL	5, 10, 100, na	0.00001, 0.0001, 0.002, 0.0005, 0.001	Gaussian distribution	LinearSVC	0.88
AMSAT	5, 10, 100, na				Before AL: 0.876 After AL: 0.876

Synthetic experiments : [PW-MSTL]

Table 1. Classification accuracy (%) on the target domain, given that source domains contain diverse {1%,5%,15%,30%} labeled data.

Method	Synthetic		Spam			Sentiment									
	case1	case2	user7	user8	user3	electronics	toys	music	apparel	dvd	kitchen	video	sports	book	health
KMM	82.7	88.8	92.0	91.8	89.7	77.6	77.4	71.0	78.3	72.4	78.4	72.1	79.1	71.2	77.4
KMM-A	87.3	91.4	92.0	92.0	91.8	74.6	76.3	70.3	75.8	72.4	75.2	70.5	76.7	69.7	74.9
A-SVM	70.8	89.4	84.5	87.8	86.8	70.8	73.7	67.7	73.6	62.6	72.8	62.5	73.7	66.9	71.4
DAM	75.8	91.0	83.8	85.4	86.8	71.3	73.7	68.0	75.1	62.5	72.1	62.0	73.0	68.0	72.5
PW-MSTL$_b$	85.5	90.8	91.5	92.6	90.3	78.0	78.7	70.7	79,5	73.2	78.3	72.5	79.5	71.5	77.7
PW-MSTL	**88.4**	**92.6**	**93.8**	**95.6**	**92.8**	**79.3**	**81.9**	**74.6**	**82.7**	**76.7**	**80.7**	**76.2**	**82.7**	**74.8**	**80.9**

The synthetic experiment is not the same as the one described previously.

Different methods are not implemented in the git, except the SVM.

The treatment of partial labeled dataset is not implemented

Also the two others data set are not useable. The data available is unprocessed and required lot of work and it will create approximations

REVIEW

04

Review

- The method presented is very generical And can be applied to any classification problem and even more.

- Both methods, although independant, can be combined and provide interesting results

- The synthetic experiment described is not tested. Authors didn't share results.

- This experiment has been made on our hand with approximations because of lack of details

- Experiments are not accurately reproductible

- Different sources are always taken from a same set of features and have the same shape. This doesn't represent the reality.

Openings

Differentially Private Hypothesis Transfer Learning

Yang Wang(✉)[1], Quanquan Gu[2], and Donald Brown[1]

[1] Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA, USA
{yw3xs, deb}@virginia.edu
[2] Department of Computer Science, University of California, Los Angeles, CA, USA
qgu@cs.ucla.edu

Abstract. In recent years, the focus of machine learning has been shifting to the paradigm of transfer learning where the data distribution in the target domain differs from that in the source domain. This is a prevalent setting in real-world classification problems and numerous well-established theoretical results in the classical supervised learning paradigm will break down under this setting. In addition, the increasing privacy protection awareness restricts access to source domain samples and poses new challenges for the development of privacy-preserving transfer learning algorithms. In this paper, we propose a novel differentially private multiple-source hypothesis transfer learning method for logistic regression. The target learner operates on differentially private hypotheses and importance weighting information from the sources to construct informative Gaussian priors for its logistic regression model. By leveraging a publicly available auxiliary data set, the importance weighting information can be used to determine the relationship between the source domain and the target domain without leaking source data privacy. Our approach provides a robust performance boost even when high quality labeled samples are extremely scarce in the target data set. The extensive experiments on two real-world data sets confirm the performance improvement of our approach over several baselines.

Keywords: Differential privacy · Transfer learning.

It uses a logistic regression to evaluate weights of each source in the case of transfer learning

The Amazon sentiment dataset is used and a whole section is dedicated to preprocessing.

But no implementation is made available to the reader

Thanks

Do you have any questions ?