Towards more Reliable Transfer Learning

Summary and Reproduction of Wang et al.'s [1] work

Hadrien Mariaccia hadrien.mariaccia@dauphine.eu
Kenza Hammou hammou.kenza.cs@gmail.com
Olivier Randavel olivier.randavel@gmail.com
Amine El Iraki amineliraki@hotmail.com

***Abstract*—Wang et al [1] address the challenge of multi-source transfer learning with two independent approaches. One focusing on an improving method of multi-source transfer learning. The other focusing on a multi-source active learning methods which increases transfer learning performances. Firstly, this paper summarizes Wang et al. work and secondly, shows a reproduction of some of their experimentations.**

I. Introduction

Used to single-source transfer learning where knowledge of one source problem is transferred to solve the target problem. The more realistic task of multi-source transfer learning consists in assuming the existence of multiple source domains related to a target domain all suffering from labelled data scarcity. Firstly, the authors address this challenge with a new method called Peer-weighted Multi-source Transfer Learning (PW-MSTL). This method uses sources reliabilities and source-target similarities to perform multi-source transfer learning on a target problem. Secondly, the authors address the challenge of labelled data scarcity in the source domains by designing an Adaptive Multi-source Active Learning (AMSAT) method, this method is based on source-target proximity and assumes the existence of a budget limited oracle. AMSAT aims to label relevant source elements in order to increase multi-source transfer learning accuracy on the target.

II. Theoretical framework

We consider a multi-source transfer learning setting. We denote K auxiliary sources and S domains with labeled and unlabeled data : we define $S_k = S_k^L \cup S_k^U$ with $S_k^L = (x_i^{S_k^L}, y_i^{S_k^L})_{i=1}^{n_k^L}$ the k^{th} labeled set and $S_k^U = (x_i^{S_k^L})_{i=1}^{n_k^U}$ the k^{th} unlabeled set. Finally we have the unlabeled target data $T = (x_i^T)_{i=1}^{n_T}$.

A. Re-weighted MMD

Before training each classifier on each source, the weights of the k^{th} source aggregate data ($\alpha_i^k \geq 0$) have to be determined. An adaptation of the Maximum Mean Discrepancy (MMD) statistic that estimate the distribution discrepancy is used : re-weighted MMD (1).

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

ϕ is a feature map onto a Reproducing Kernel Hilbert Space (RKHS), avoiding the complex density estimation. The optimization can be solved by applying the kernel trick.

B. Relationship matrix

This matrix is used by both algorithms. It captures inter-source similarities by computing the classification error made by an estimator ($\hat{h}_j$), trained on a source S_j, on the source S_i compare to all other sources $j' \neq i$ on the same source S_i. In other words, if the error done by the classifier of the source S_j for the data of S_i is big comparing to other classifiers $j' \neq i$ then R_{ij} will be big.

$$R_{i,j} = \left\{ \begin{array}{cc} \frac{exp(\beta \hat{\varepsilon}_{S_i}(\hat{h}_j))}{\sum_{j' \in [K], j' \neq i} exp(\beta \hat{\varepsilon}_{S_i}(\hat{h'}_j))}, & i \neq j \\ 0, & otherwise \end{array} \right\} \quad (2)$$

C. Source importance weights

The source importance weights is parametrized considering both source proximity and reliability, as follows:

$$w = \delta[\mu I_K + (1-\mu)R] \quad (3)$$

where $I_K \in R^{K*K}$ is the identity matrix, $\delta \in R^K$, K is a vector measuring pairwise source-target proximities that may be manually specified according to domain knowledge or estimated from data, the concentration factor $\mu \in [0, 1]$ is a scalar that introduces an additional degree of freedom in quantifying the trade-off between proximity and reliability and R is the source relationship matrix. They took $\mu = \frac{1}{K}$ in their experiments.

III. Algorithms

A. PW - MSTL

This algorithm is used to predict the class of a target data by performing transfer learning over all sources. In this configuration, the authors chose a binary classification. Firstly, the algorithm trains all classifiers by solving the re-weighted MMD formula for all sources. Then, for each target data and for each classifier, if the confidence of the k^{th} classifier on the instance $x^{(t)}$ is too low, we compute the sum of all other source classifiers predictions on the instance $x^{(t)}$ weighted by the matrix R, ie, the inter-sources proximities matrix. If the confidence is high enough, we keep the result prediction of the current source classifier. Finally the label of the target data is deduced by calculating the sign of the sum of all source

classifiers predictions, weighting by the source importance weights coefficient.

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**
    - $\hat{p}_k^{(t)} = \left|\hat{h}_k(x^{(t)})\right|$
- $\hat{y}^(t) = sign(\sum_{k \in [K]} w_k \hat{p}_k^{(t)})$

B. 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 $S_{k^{(t)}}^L \leftarrow S_{k^{(t)}}^L \cup \{x^{(t)}\}$
- $Update S_{k^{(t)}}^L \leftarrow S_{k^{(t)}}^L \setminus \{x^{(t)}\}$;
- Update classifier $\hat{h}_k$.

AMSAT is a two stage active learning framework assuming the budget limited availability of an oracle in the source domains. Firstly, AMSAT selects the source domain to query based on 2 criteria. On one side, if the distribution of labelled data over source domains is to far from a uniform distribution, the probability of choosing one source is then inversely proportional to its relative labelled data scarcity. On another side, if the distribution of labelled data over source domains is close to a uniform distribution, the probability of choosing one source is defined by (3). Secondly, AMSAT queries the most informative instance by solving Eq.(4).

1) Selection criterion:

$$x = argmax_{x_i \in S_{k^{(t)}}^U} E[(\hat{y}_i - y_i)^2 | x_i] \alpha_i^{k^{(t)}} \quad (4)$$

This formula consists in choosing an instance on which the classifier is weak (by taking uncertainty into consideration) but also weighted by (1) in order to select the most source-target representative instance.
This process goes along until all the budget is spent.

IV. Experiments

The experiments have 2 aims : "(i) evaluate PW-MSTL with sources with diverse reliabilities, (ii) evaluate the effectiveness of AMSAT in constructing more reliable classifiers via active learning".

On one hand, concerning PW-MSTL, they consider a binary classification problem, on 3 different datasets : a synthetic one generated through gaussian random variables (600 examples, 100 per domain), and two real-world datasets, one for a spam detection purpose (6000 mails, 400 per user), and the other for sentiment analysis on Amazon reviews (20000 reviews, 2000 per domain). One domain is set as the target, and is divided randomly into a test set (40%) and a training set (60%) composed of unlabeled data. Other domains are set as sources.
Note that for each dataset, all domains (target and sources) have instances that are in the same features space. PW-MSTL will be compared to five other methods: one single-source method, one aggregate method, and three ensemble methods. The classifier is trained using SVM and fixing b_1 = 1. The same Gaussian kernel is used for fair comparisons between the models.

$$k(x_i, x_j) = exp(-\|x_i - x_j\|^2 / \gamma) \quad (5)$$

On the other hand, the AMSAT method uses four different experiments. First samples are generated from a random distribution. Then they use the uncertainty method which picks the examples that are most uncertain or close to the decision boundary. Besides, the representative méthod choose exemple most representative regarding equation (1). Finally the proximity method selects the source examples that are most similar to the target domain. The setup implemented by the authors is running as followed : sources are initialised with various fraction of labeled data. AMSAT is performed 30 times and the average accuracy on the test target data is reported. Also it uses a query budget to 10% of the total number of source examples in the dataset.

V. Experiments reproduction

This part aims to confirms results mentioned in the paper and try to show forgotten bias on the interpretation of the results. The authors provide a git repository [2]. To start we will go through the git and describe the results obtained. Then we will follow setup indications from the paper to make our experiments.

The git has been created in 2018 and hasn't been modified since then. The code is written in python 2.6. The authors created two simulators that created samples from random distributions. Parameters are used to create k sources which contains each d features and n samples. This example performs a binary classification through AMSAT and PW-MSTL algorithms.

First, we modify the code from python 2 to python 3. We make our experiments using Colab CPU and 12go of RAM. Our first approach was to run python codes named "MultiSource_Transfer_Learning_Simulator.py" and "Multi-Source_Active_Learning_Simulator.py". After 30 loops, we obtained the results presented on Fig 1. It seems that both algorithms does not improve accuracy. Indeed, in the case of binary classification, a random classification performs around 50%.

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

Fig. 1. Uniformly generated Data

Then, we decided to generate samples from a Gaussian distribution as mentioned in the synthetic methods. Code can be found under our repository and is named "Data_generator.py". Some instructions were not precised for example, we decided to generate $\Delta\mu$ from a uniform distribution and took p in $\{0.00001, 0.0001, 0.002, 0.0005, 0.001\}$ for each sources following the advise ($p \leq 0.001$). Finally, parameters for labelling function were not obvious, then we set σ to 1, w_0^T a vector of ones, $\delta = 1$, generates Δw and ϵ from a zero-mean normal distribution

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

Results are shown below. These results are better than the uniform generated sample and shown that transfer learning with a low proximity result in a good accuracy. However, the Active Learning gives no improvements.

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

Fig. 2. Gaussian generated data

Besides, we want to work on results stated in the below table.

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**

These results were obtained using the synthetic method described previously and two others data-sets. Unfortunately, we are not able to deal with these two data-sets as the data is not prepossessed (figure 4) and extracted features from those datasets are not described either. The vocabulary concerning sentiment data-set is not given. Therefore creating a use-able data-set would require a lot of work and no certainty that the data are processed the same way as the authors.

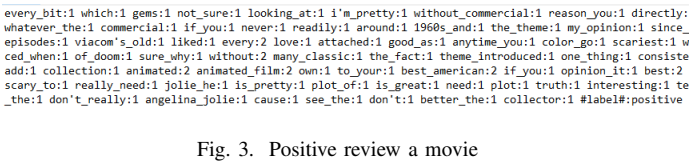

Fig. 3. Positive review a movie

However, it is possible to reproduce results from the Synthetic method. But, the authors seem to use a different set up than the one describe in the experiment section. Indeed, in the table 1, they use a different fraction of labeled data for four source sets, but they describe previously the use of 5 labelled sources. Also, this implementation has not been done on their GitHub repository and required some development.

VI. Review

This paper provides a clear explanation of the two algorithms implemented. They are available through the GitHub. Experiments help to prove the contribution of their solution and show improvement of the accuracy.

However, several details are missing. The setup detailed in the PW-MSTL experiments is not the one used to show results in the table 1. The implementation available in the GitHub only simulate random uniform samples. Besides, when launching the python file, results are not significant and around 50%. This is disappointed as it does not confirmed results obtained in the paper. Therefore, Gaussian random experiments require time to implement and it could have been

provided in the git. Moreover methods stated in the results table such as DAM is not available in the GitHub.

Other experiments on sentiment and Spam data also required a lot of work to reproduce. In fact, the prepossessing of the data is not detailed in this paper. The lack of information introduces a bias which could make impossible to reproduce the experiments.

Data-sets chosen are not representative of the reality. Indeed, all sources from the sentiment Amazon data-set have the same features and the task is the same for every source. When using transfer learning, sources are generally different and the authors do not tell a word on it.

Finally the experiments have not been applied in real life cases and the GitHub has not been updated since its creation two years ago. This proves that lack of explanations doesn't give the opportunity to test and to reproduce experiments by others.

Following previous remarks, we made researches in order to test experiments on Amazon sentiment data-set. We ended on this paper [3] written by one common researcher Mr. Wang, two months later. It uses a logistic regression to evaluate weights of each sources in the case of transfer learning. The paper describes over a whole section the prepossessing of the Amazon data. However, it seems that no implementation is made available to the reader. Besides some other critics still remain the same, experiments are done in the same way : task of each source is the same and each source have the same features.

REFERENCES

[1] Zirui Wang and Jaime G. Carbonell, "Towards more Reliable Transfer Learning" CoRR. abs/1807.02235, 2018.

[2] iedwardwangi, "ReliableMSTL" https://github.com/iedwardwangi/ReliableMSTL, 2018

[3] Yang Wang, Quanquan Gu, and Donald Brown, "Differentially Private Hypothesis Transfer Learning, 2018