

Distributed Heterogeneous Transfer Learning

Paolo Mignone^{1,2}, Gianvito Pio^{1,2} and Michelangelo Ceci^{1,2}

¹Department of Computer Science, Via Orabona, 4, 70125, University of Bari Aldo Moro, Bari, Italy

²Big Data Lab, National Interuniversity Consortium for Informatics (CINI), Via Ariosto, 25, 00185, Rome, Italy

Abstract

Transfer learning has proved to be effective for building predictive models for a target domain, by exploiting the knowledge coming from a related source domain. However, most existing transfer learning methods assume that source and target domains have common feature spaces. Heterogeneous transfer learning methods aim to overcome this limitation, but they often make strong assumptions, e.g., on the number of features, or cannot distribute the workload when working in a big data environment. In this manuscript, we present a novel transfer learning method which: *i*) can work with heterogeneous feature spaces without imposing strong assumptions; *ii*) is fully implemented in Apache Spark following the MapReduce paradigm, enabling the distribution of the workload over multiple computational nodes; *iii*) is able to work also in the very challenging Positive-Unlabeled (PU) learning setting.

We conducted our experiments in two relevant application domains for transfer learning: the prediction of the energy consumption in power grids and the reconstruction of gene regulatory networks. The results show that the proposed approach fruitfully exploits the knowledge coming from the source domain and outperforms 3 state-of-the-art heterogeneous transfer learning methods.

Keywords

Heterogeneous transfer learning, Positive-Unlabeled setting, Distributed computation, Apache Spark

1. Introduction

Machine learning algorithms aim to train a model function that describes and generalizes a set of observed examples, called training data. The learned model function can be applied to unseen data with the same feature space and data distribution of the training data. However, in several real scenarios, it is difficult or expensive to obtain training data described through the same feature space and following the same data distribution of the examples where the prediction functions will be applied. A possible solution to the challenges raised by these scenarios come from the design and application of *transfer learning* methods [1], that aim to learn a predictive function for a *target* domain, by exploiting also an external but related *source* domain.

A relevant example of these scenarios can be observed in the energy field, where predicting the customer energy consumption is a typical task [2]. When new customers are connected to the energy network in a new area/district, they could not be well represented by the available training data to make accurate predictions for them. In this case, transfer learning would enable the exploitation of data related to other customers, also residing in different areas, leveraging common characteristics in terms of type of customers or in terms of behavior.

SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy


✉ paolo.mignone@uniba.it (P. Mignone); gianvito.pio@uniba.it (G. Pio); gianvito.pio@uniba.it (M. Ceci)

🌐 <http://www.di.uniba.it/~mignone/> (P. Mignone); <http://www.di.uniba.it/~gianvitopio/> (G. Pio);

<http://www.di.uniba.it/~ceci/> (M. Ceci)

🆔 0000-0002-8641-7880 (P. Mignone); 0000-0003-2520-3616 (G. Pio); 0000-0002-6690-7583 (M. Ceci)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

In several scenarios, data for the target and the source domains come from multiple data sources, exhibiting different data representations that make the direct adoption of classical transfer learning approaches unfeasible. A possible solution may consist in performing time-consuming manual feature engineering steps, aiming to find commonalities among the data sources and to somehow make the feature spaces homogeneous. However, such manual operations can be subjective, error-prone, and even unfeasible when no detailed information are available about the features at hand, or when the features in the target and source domains are totally different, i.e., heterogeneous. To overcome this issue, several approaches have been proposed in the literature to automatically identify the best match between features of different domains, or to identify a novel shared feature space. Such methods cover different real-domain applications, such as biomedical analysis [3], object recognition from images [4], or multilingual sentiment classification [5]. However, to the best of our knowledge, existing methods exhibit one or more of the following limitations: *i*) they make strict assumptions on the number of features, i.e., even if they are able to work with heterogeneous feature spaces, they are required to have the same dimensionality [6, 7, 8]; *ii*) they are not able to distribute the workload over multiple computational nodes; *iii*) in the case of classification tasks, they require a fully labeled training set, or at least a proper representation of each class, i.e., they are not able to work in the Positive-Unlabeled (PU) setting, where the training set consists of only positive labeled instances and unlabelled instances [9], that is very common, for example, in the biological domain [10, 11]; *iv*) they are able to work by exploiting the background knowledge of specific application domains [12, 13].

To overcome these limitations, in this paper we proposed a novel heterogeneous transfer learning method, called STEAL, that simultaneously exhibit the following characteristics:

- It is able to work also in the very challenging Positive-Unlabeled (PU) learning setting;
- It is implemented using the Apache Spark framework following the MapReduce paradigm, enabling the distribution of the workload over multiple computational nodes;
- It exploits the Kullback-Leibler (KL) divergence to align the descriptive variables of the source and target domains and, therefore, to work with heterogeneous domains described through feature spaces having different dimensionalities;
- It is not tailored for specific application domains, but can be considered a general purpose approach applicable to multiple real-world scenarios.

2. The proposed method STEAL

In this section, we describe the proposed method STEAL, emphasizing its peculiarities. The distribution of the workload over multiple computational nodes is achieved by designing the algorithms using the MapReduce paradigm. The pseudo-code description of the algorithms exploits the Resilient Distributed Dataset (RDD) data structure available in Apache Spark.

The workflow followed by STEAL consists of 3 stages. The first two stages are dedicated to align the feature spaces and to identify a proper matching between target and source instances. Finally, the third stage consists of training a distributed variant of Random Forests (RF) available in Apache Spark, from the obtained hybrid training set. In the following, we report some further details about the first two stages.

Stage 1 - Feature Alignment. In the first stage, in the case of PU learning setting, we first estimate the labels of the unlabeled instances by resorting to a clustering-based approach.

Specifically, we apply a prototype-based clustering algorithm to identify k representatives of positive instances from the positive examples of the target domain, and exploit them to estimate a score for each unlabeled instance. Specifically, we adopt a distributed variant of the k -means algorithm, since it is well established in the literature and has been effectively exploited in previous works in the PU learning setting [10, 11, 12, 13]. The score we compute for each unlabeled instance is in the interval $[0, 1]$, where a value close to 0.0 (resp., 1.0) means that the unlabeled example is likely to be a negative (resp., positive) example.

Methodologically, the score is computed according to the similarity between the feature vector associated with the unlabeled instance and the feature vectors of the cluster prototypes.

As similarity function, we consider $sim(a, b) = 1 - \frac{\sqrt{\sum_{i=1}^r (a_i - b_i)^2}}{r}$, that computes the similarity between the instance vectors a and b , in a r -dimensional space, based on the Euclidean distance, after applying a min-max normalization (in the range $[0, 1]$) to all the features. After this step, we obtain a fully labeled dataset composed by true and estimated labels, where the computed score represents the confidence on the fact that a given unlabelled instance is a positive instance.

Subsequently, if the dataset is highly dimensional¹, STEAL applies a distributed variant of the PCA, in order to identify a reduced set of (non-redundant) new features. We extract $\sqrt{\min(p_s, p_t)}$ new features, where p_s and p_t are the initial dimensionalities of the source and the target feature spaces, respectively.

Finally, we focus on the main objective of this stage, namely, the identification of an *alignment* between the descriptive variables of the source and target domains. This step is necessary in order to make the instances of the source and target domains directly comparable (through a distance/similarity measure), to carry our the following Stage. Specifically, the goal is to find a correspondent feature in the source domain, for each feature of the target domain. To this aim, we compute the asymmetric KL divergence for each target-source pair of descriptive variables, that quantifies the loss of information caused by a given feature of the source domain when used to approximate a given feature of the target domain. Formally, given the i -th feature of the target domain F_{t_i} , and the j -th feature of the source domain F_{s_j} , we compute the discrete variant of the KL divergence between F_{t_i} and F_{s_j} as follows:

$$KL(F_{t_i}, F_{s_j}, X_t, X_s) = \sum_{x \in (un(F_{t_i}, X_t) \cap un(F_{s_j}, X_s))} p(x, F_{t_i}, X_t) \ln \frac{p(x, F_{t_i}, X_t)}{p(x, F_{s_j}, X_s)}, \quad (1)$$

where X_t and X_s are the training instances of the target and source domains, respectively; $un(y, w)$ represents the set of distinct values of the feature y in the dataset w ; $p(x, y, w)$ represents the relative number of occurrences of the value x on the feature y in the dataset w , after a proper discretization of the feature y ².

In Figure 1 we graphically show this alignment step, while the pseudo-code is reported in Algorithm 1. STEAL starts by considering the values assumed by the attributes as the key of a paired RDD. The result is then used to find common values assumed by target and source attributes, through a join operation. The frequency of each distinct value is then computed for each target and source attribute, which are then exploited to estimate the probabilities used in

¹This can be considered as an optional step, that mainly depends on the availability of computational resources.

²We discretize continuous features using the equal-width strategy (bin size of 0.01), after min-max normalization.

Algorithm 1: alignSourceDomain(X_t, X_s)

Data: X_t, X_s : RDD. Target and source domain instances (in the $\langle instance_id, feature, value \rangle$ form)

Result: $X_{s_aligned}$: RDD. Source domain instances aligned in the p_t -dimensional feature space.

```
1 begin
2   tValKey ←  $X_t$ .map{case(instance_id, feature, value) → ⟨value, feature⟩};
3   sValKey ←  $X_s$ .map{case(instance_id, feature, value) → ⟨value, feature⟩};
4   // Join the target and source structures according to the feature values
   commVals ← tValKey.join(sValKey)
5   // Count the frequencies of values for the target features
   tValFreq ← commVals.map{case(value, tFeat, sFeat) → ⟨⟨value, tFeat⟩, 1⟩}.reduceByKey((a, b) → a + b)
6   // Compute the number (cardinality) of value matches for each target feature
   tAttrCard ← tValFreq.map{case(⟨⟨value, feat⟩, freq⟩) → ⟨feat, freq⟩}.reduceByKey((a, b) → a + b)
7   // Compute the value probabilities for the target domain
   tValueProbs ← tValFreq.map{case(⟨value, feat⟩, freq) → ⟨feat, ⟨value, freq⟩⟩}
8   .join(tAttrCard).map{case(feat, ⟨value, freq, card⟩) → ⟨value, feat, freq/card⟩}
9   // Compute the value probabilities for the source domain in the same way
   sValProbs ← ...
10  // Compute the KL divergences
   divergences ← tValProbs.cartesian(sValProbs)
11  .map{case(⟨tVal, tFeat, tProb⟩, ⟨sVal, sFeat, sProb⟩) → ⟨⟨tFeat, sFeat⟩, tProb · log(tProb/sProb)⟩}
12  .reduceByKey((aKLterm, bKLterm) → aKLterm + bKLterm)
13  // Find the best target-source features matching by minimizing the KL divergences
   minDivergences ← divergences
14  .map{case(⟨tFeat, sFeat⟩, KLdiv) → ⟨tFeat, ⟨sFeat, KLdiv⟩⟩}
15  .reduceByKey{case(⟨sFeat1, KLdiv1⟩, ⟨sFeat2, KLdiv2⟩) →
16    if(KLdiv1 < KLdiv2) then ⟨sFeat1, KLdiv1⟩ else ⟨sFeat2, KLdiv2⟩}
17  // Align source features to target features, according to the minimum divergences
   return align( $X_s$ , minDivergences)
```

the computation of the KL terms. Such probabilities are computed by dividing the frequencies by the number of matches (in the join operation) that involved a given attribute. Finally, we compute the KL divergence for each target-source pair of features, which are then exploited to find the best (i.e., with the minimum divergence) source feature for each target feature.

Note that the proposed approach implicitly performs a feature selection on the source domain, since some of the features could not be matched to any target feature. Analogously, the same feature of the source domain could be selected multiple times for different features of the target domain, if it appears strongly aligned to all of them from a statistical viewpoint.

Stage 2 - Instance Matching. After the first stage, target and source instances are represented in an aligned feature space, and are also directly comparable through similarity/distance measures. A straightforward approach to exploit the instances of the source domain would be that of appending them to the training set of the target domain. However, this approach would lead to the possible introduction of noisy instances, i.e., not properly representing the data distribution of the target domain, increasing the chance of negative transfer phenomena.

On the contrary, we aim to finely exploit only a subset of source instances, by *attaching* them to specific target instances, through feature concatenation, according to their similarity. In this way, STEAL augments the descriptive features of target instances with those coming from

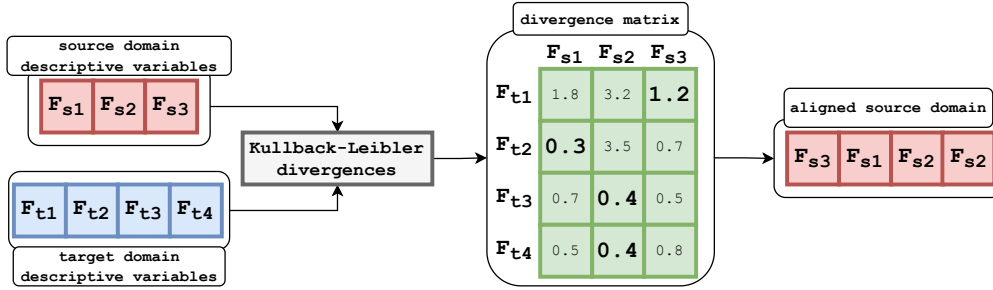


Figure 1: A graphical focus on the feature alignment between the source and the target domains.

aligned, best-matching, source instances.

STEAL computes the Cartesian product between $num.Samples$ subsets (randomly selected, with replacement) of target and source instances, and *reduce* the resultset by taking the source instances with the maximum similarity. The adoption of multiple random subsets, instead of working on the whole set of instances alleviates the negative effects due to the possible match of a target instance to an improper source instance, since the same target instance may be sampled multiple times and associated with different source instances. At the end of the instance matching step, we obtain a single target-source hybrid training set by merging the target-source concatenated instances constructed from each sample. For space constraints, we omit the pseudo-code description of this stage.

3. Experiments

We conducted our experimental evaluation in a relevant application domain where transfer learning can be strongly beneficial, namely in biology, and specifically in the reconstruction of the human gene regulatory network (target domain), using also the knowledge coming from the mouse gene regulatory network (source domain). The reconstruction of a gene regulatory network consists in the identification of currently unknown gene regulatory activities. A gene regulatory network consists of a set of genes (represented as nodes) and a set of known regulations (represented as edges). The reconstruction of such networks can be performed by resorting to link prediction methods [14], working in the Positive-Unlabelled learning setting.

We compared the results achieved by STEAL with some state-of-the-art heterogeneous transfer learning competitors, namely **TJM** [6], **JGSA** [7] and **BDA** [8]. Note that even if they are able to work with heterogeneous feature spaces, the latter are required to have the same dimensionality. Moreover, we also evaluated the results obtained by some baseline approaches, that are: *i*) **T (no transfer)**, that is a predictive model learned only from the dataset of the target domain; *ii*) **S (optimal feature alignment)**, that is a predictive model trained only from the source domain dataset, assuming that the optimal feature alignment is known a priori; *iii*) **T + S (optimal feature alignment)**, that is a predictive model trained from both the target and the source instances, assuming that the optimal feature alignment is known a priori.

In our experiments, we used the dataset adopted in [11], which consists of two gene regulatory networks, one for the human organism and one for the mouse organism. We considered both the available versions of the dataset, i.e., the heterogeneous version, where human and mouse genes

	Heterogeneous		Homogeneous		Homogeneous Reduced	
	Human	Mouse	Human	Mouse	Human	Mouse
Positive interactions	235,706	14,613	235,706	14,613	4,714	4,714
Unlabeled interactions	235,706	235,706	235,706	235,706	4,714	4,714
Gene features	174	161	6	6	6	6
Gene-pair features	348	322	12	12	12	12

Table 1
Quantitative information of the considered datasets.

are described by 174 and 161 expression levels, respectively, and the homogeneous version obtained by averaging the expression levels per organ. Moreover, since competitor systems ran out of memory on our server equipped with 64GB RAM, we also ran the experiments on a reduced version of the homogeneous dataset consisting of 2% randomly selected instances. Detailed quantitative information of the considered datasets are shown in Table 1.

The task at hand naturally falls in the PU learning setting, since unobserved links cannot be considered negative examples. To perform the estimation of the label confidence (Stage 1), we identified 3 clusters and 2 clusters for the mouse and the human positive instances, respectively. Such values were identified through the silhouette cluster analysis. For the heterogeneous version of the dataset, we ran STEAL with the distributed PCA, that led to identify 18 new features. As regards the instance matching, 10 samples were considered (i.e., $numSamples=10$), each involving 10% random training examples from the source and from the target domains.

All the experiments were performed in the 10 fold cross-validation setting, where each fold consists of 9/10 positive examples for training and 1/10 positive examples for testing, while all the unlabeled examples are considered for both training and testing.

As evaluation measure, since the dataset does not contain any negative example, we considered the recall@ k defined as $\frac{TP_k}{TP+FN}$, where TP_k is the number of returned true positive interactions within the first top- k interactions, and $(TP + FN)$ corresponds to the number of positive examples in the testing fold, and computed the area under the recall@ k curve (AUR@K). Note that the computation of other well-known measures, such as the Area Under ROC curve (AUROC) and the Area Under Precision-Recall curve (AUPR), is not possible at all in the positive-unlabeled setting, without introducing wrong biases in the ground truth.

From the results shown in Table 2, we can observe that STEAL is able to outperform all the considered baselines, in both the heterogeneous and homogeneous settings. In particular, the results show that STEAL outperforms the counterpart that does not exploit the source domain, i.e., **T (no transfer)**, by 27% in the homogeneous setting and by 9.7% in the heterogeneous setting, proving the benefits of exploiting the additional knowledge coming from the mouse organism. On the other hand, using only the source domain (i.e., **S (optimal feature alignment)**), leads to a reduction of the AUR@K with respect to using the target data only. This means that, although source data can be beneficial for the target task, their exploitation should be done in a smart manner. Note that, as expected, appending source instances to the target dataset, even if the optimal feature alignment is known a priori (see **T+S (optimal feature alignment)**), does not provide as much as benefits as the method adopted by STEAL, that outperforms this approach by 23.4% and 10% in the homogeneous and heterogeneous settings, respectively.

Method	Heterogeneous	Homogeneous
T (no transfer)	0.585	0.533
S (optimal feature alignment)	0.569	0.544
T+S (optimal feature alignment)	0.583	0.551
T+S (STEAL)	0.642	0.680

Method	Homogeneous Reduced
JGSA	0.500
TJM	0.554
BDA	0.558
STEAL	0.589

Table 2

On the left, the AUR@K results obtained by STEAL and baseline approaches on the full datasets. On the right, the AUR@K results obtained by STEAL and competitors on the reduced dataset.

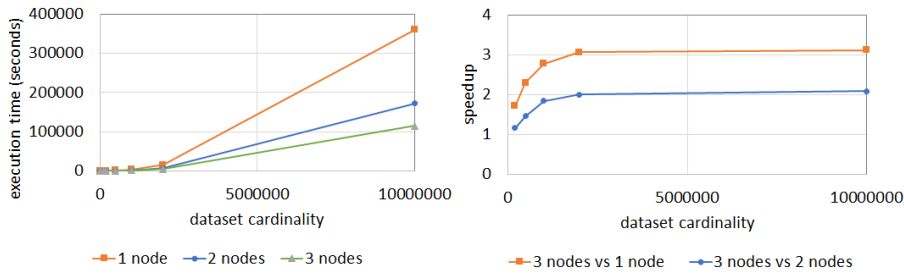


Figure 2: Running times and speedup factors measured on a synthetic dataset of 10 millions examples

The comparison with competitors on the reduced dataset shows that STEAL is able to outperform all of them. Specifically, STEAL achieves an improvement of 17.8% with respect to **JGSA**, 6.3% with respect to **TJM**, and 5.5% with respect to **BDA**. Note that the superiority of STEAL over these competitors is not limited to the observed AUR@K on the reduced dataset, since **JGSA**, **TJM** and **BDA** were not able to complete the experiments on the full dataset.

To assess the advantages of STEAL also from a computational viewpoint, we performed a scalability analysis. We measured the running times and computed the speedup factor to evaluate the ability of STEAL to exploit additional computational nodes. This analysis was performed on a cluster with 1, 2, or 3 computational nodes, by resorting to a synthetic dataset with 10 millions instances. The results of this analysis, depicted in Figure 2, show that, although the communication overhead in principle increases, STEAL is able to take advantage of possible additional computational nodes. Specifically, the measured speedup factors are close to ideal results, i.e., they quickly converge to 2 and 3, respectively, with 2 and 3 computational nodes.

4. Conclusions

In this discussion paper, we proposed a novel heterogeneous transfer learning method, called STEAL, that can work in the PU learning setting also with heterogeneous feature spaces, thanks to a feature alignment step based on the KL divergence. The obtained results demonstrate the effectiveness of the proposed method with respect to baseline and state-of-the-art competitors. Moreover, it exhibits very interesting scalability results, emphasizing its possible applicability on large datasets. We evaluated the performance of our method in the reconstruction of gene regulatory networks, that obtained significant benefits from the application of STEAL. Currently, we are evaluating the effectiveness of STEAL in other application domains, including the prediction of the energy consumption in power grids, and we are working on extending it with the possibility of exploiting multiple source domains.

Acknowledgments

Dr. Paolo Mignone acknowledges the support of Apulia Region through the REFIN project “Metodi per l’ottimizzazione delle reti di distribuzione di energia e per la pianificazione di interventi manutentivi ed evolutivi” (CUP H94I20000410008, Grant n. 7EDD092A).

References

- [1] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, *Proc. IEEE* 109 (2021) 43–76.
- [2] K. Amasyali, N. M. El-Gohary, A review of data-driven building energy consumption prediction studies, *Renewable and Sustainable Energy Reviews* 81 (2018) 1192–1205.
- [3] Y. Wang, Z. Xia, J. Deng, X. Xie, M. Gong, X. Ma, TLGP: a flexible transfer learning algorithm for gene prioritization based on heterogeneous source domain, *BMC Bioinform.* 22-S (2021) 274.
- [4] W. Li, L. Duan, D. Xu, I. W. Tsang, Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation, *IEEE Trans. Pattern Anal. Mach. Intell.* 36 (2014) 1134–1148.
- [5] J. T. Zhou, S. J. Pan, I. W. Tsang, Y. Yan, Hybrid heterogeneous transfer learning through deep learning, in: *AAAI 2014*, 2014, pp. 2213–2220.
- [6] M. Long, J. Wang, G. Ding, J. Sun, P. S. Yu, Transfer joint matching for unsupervised domain adaptation, in: *CVPR 2014*, 2014, pp. 1410–1417.
- [7] J. Zhang, W. Li, P. Ogunbona, Joint geometrical and statistical alignment for visual domain adaptation, *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 2017-January* (2017) 5150–5158.
- [8] J. Wang, Y. Chen, S. Hao, W. Feng, Z. Shen, Balanced distribution adaptation for transfer learning, in: *ICDM 2017*, 2017, pp. 1129–1134.
- [9] B. Zhang, W. Zuo, Learning from positive and unlabeled examples: A survey, in: *ISIP 2008 / WMWA 2008*, Moscow, Russia, 23-25 May 2008, 2008, pp. 650–654.
- [10] P. Mignone, G. Pio, Positive unlabeled link prediction via transfer learning for gene network reconstruction, in: *ISMIS 2018*, 2018, pp. 13–23.
- [11] P. Mignone, G. Pio, D. D’Elia, M. Ceci, Exploiting transfer learning for the reconstruction of the human gene regulatory network, *Bioinformatics* 36 (2020) 1553–1561.
- [12] P. Mignone, G. Pio, S. Džeroski, M. Ceci, Multi-task learning for the simultaneous reconstruction of the human and mouse gene regulatory networks, *Scientific Reports* 10 (2020) 22295.
- [13] G. Pio, P. Mignone, G. Magazzù, G. Zampieri, M. Ceci, C. Angione, Integrating genome-scale metabolic modelling and transfer learning for human gene regulatory network reconstruction, *Bioinformatics* 38 (2021) 487–493.
- [14] L. Lu, T. Zhou, Link prediction in complex networks: A survey, *Physica A: Statistical Mechanics and its Applications* 390 (2011).