Big Data Research 37 (2024) 100456

Contents lists available at ScienceDirect

Big Data Research

journal homepage: www.elsevier.com/locate/bdr

Distributed Heterogeneous Transfer Learning

Paolo Mignone [a,b], Gianvito Pio [a,b,*], Michelangelo Ceci [a,b,c]

[a] Department of Computer Science, University of Bari Aldo Moro, Bari, Italy
[b] Big Data Laboratory, National Interuniversity Consortium for Informatics (CINI), Rome, Italy
[c] Department of Knowledge Technologies, Jozef Stefan Institute, Ljubljana, Slovenia

ARTICLE INFO

Dataset link: https://figshare.com/articles/software/STEAL/19482533

Keywords:
Heterogeneous transfer learning
Distributed algorithms
Apache spark
Cerebral stroke prediction
Energy consumption prediction
Gene regulatory network reconstruction

ABSTRACT

Transfer learning has proved to be effective for building predictive models even in complex conditions with a low amount of available labeled data, by constructing a predictive model for a target domain also using the knowledge coming from a separate domain, called source domain. However, several existing transfer learning methods assume identical feature spaces between the source and the target domains. This assumption limits the possible real-world applications of such methods, since two separate, although related, domains could be described by totally different feature spaces. Heterogeneous transfer learning methods aim to overcome this limitation, but they usually *i)* make other assumptions on the features, such as requiring the same number of features, *ii)* are not generally able to distribute the workload over multiple computational nodes, *iii)* cannot work in the Positive-Unlabeled (PU) learning setting, which we also considered in this study, or *iv)* their applicability is limited to specific application domains, i.e., they are not general-purpose methods.

In this manuscript, we present a novel distributed heterogeneous transfer learning method, implemented in Apache Spark, that overcomes all the above-mentioned limitations. Specifically, it is able to work also in the PU learning setting by resorting to a clustering-based approach, and can align totally heterogeneous feature spaces, without exploiting peculiarities of specific application domains. Moreover, our distributed approach allows us to process large source and target datasets.

Our experimental evaluation was performed in three different application domains that can benefit from transfer learning approaches, namely the reconstruction of the human gene regulatory network, the prediction of cerebral stroke in hospital patients, and the prediction of customer energy consumption in power grids. The results show that the proposed approach is able to outperform 4 state-of-the-art heterogeneous transfer learning approaches and 3 baselines, and exhibits ideal performances in terms of scalability.

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 comes from the design and application of *transfer learning* methods. In general, the goal of transfer learning is to learn a predictive function for a *target* domain by exploiting also data from a separate, but related domain, called *source* domain [33,44,11,32,52].

A relevant example of these scenarios can be observed in the energy field, where predicting the customer energy consumption is a typical task [1]. When new customers are connected to the energy network in a new area/district, they could not be well represented by the available (scarce) training data, producing totally inaccurate models. In this case, the adoption of a transfer learning method would enable the exploitation of data related to other customers, also residing in different areas, leveraging common characteristics, e.g., in terms of customer type (residential, enterprise, etc.) or in terms of behavior (high consumption during the weekends, similar consumption trends during the holiday, etc.).

However, in most cases, 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

* Corresponding author at: Department of Computer Science, University of Bari Aldo Moro, Bari, Italy.
E-mail addresses: paolo.mignone@uniba.it (P. Mignone), gianvito.pio@uniba.it (G. Pio), michelangelo.ceci@uniba.it (M. Ceci).

https://doi.org/10.1016/j.bdr.2024.100456
Received 18 October 2023; Received in revised form 27 March 2024; Accepted 22 April 2024
Available online 14 May 2024

solutions 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 is available about the features at hand, or when the features in the target and source domains are totally different. 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 shared feature space [43,42,20,50,24,49,40]. Such methods cover different real applications, such as sensor data analysis [43], genetic analysis with respect to different types of cancers [42], object recognition from images [20], or multilingual sentiment classification [50]. 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, the latter are required to have the same dimensionality [24,49,40]; *ii)* they are not able to distribute the workload over multiple computational nodes, that makes them inapplicable to large datasets; *iii)* in the case of classification tasks, they require that all the classes are properly represented by labeled training instances, i.e., they are not able to work in the Positive-Unlabeled (PU) learning setting [47], that is very common in some applications (see, for example, the biological domain [27,30]); *iv)* they are able to work only in presence of a background knowledge that enables the match between the target and the source domains in a specific application context [29,34].

In this paper, we propose a novel distributed heterogeneous transfer learning method, called STEAL (Source-Target hEtetogeneous ALignment), which solves these limitations.

In particular, STEAL does not exhibit the limitation *i)*, since it employs a novel approach that allows it to work with heterogeneous feature spaces also having a different number of features. Such an approach is based on the adoption of the Kullback-Leibler (KL) divergence to identify which feature of the source domain can be properly adopted to represent a given feature of the target domain. In this way, STEAL can work when the number of features of the source domain is higher than that of the target domain, leading to an implicit feature selection: when a given feature of the source domain is not matched to any feature of the target domain, it is considered irrelevant and discarded. Moreover, since the same feature can be selected multiple times, STEAL can also work when the dimensionality of the source domain is smaller than that of the target domain. Note that this behavior is not strictly dependent on the dimensionality of the feature spaces since, anyway, a given feature of the source domain might be never selected (if does not appear to be appropriate to replace any feature of the target space) or selected multiple times (if it appears strongly correlated to multiple features of the target domain).

The limitation *ii)*, namely, the distribution of the workload, is fully overcome by STEAL, since it is implemented using the MapReduce paradigm within the Apache Spark framework. Note that to achieve the full distribution of the workload, it is necessary to design all the algorithms in a different way, since not all the data structures (and their operators) commonly available in any programming language can be adopted. On the contrary, all the procedures need to be designed on top of specific distributed data structures made available by the framework, using the (quite small) set of available operators, that also require the definition of proper functions that need to satisfy some strict properties (see Appendix A for details).

Also the limitation *iii)* exhibited by existing approaches, i.e., the impossibility to work in the Positive-Unlabeled (PU) learning setting, is fully overcome by STEAL. Indeed, it exploits a clustering-based approach to estimate the labels and the confidence thereof for (possibly huge amount of) unlabeled instances, and to exploit them during the training phase, together with the (usually small amount of) labeled instances.

Finally, STEAL does not exhibit the limitation *iv)*, since the strategies adopted to handle the heterogeneity of feature spaces, the distribution of the workload, and the PU learning setting, are not dependent on specific application domains. Therefore, STEAL can be considered a general purpose approach applicable to multiple real-world scenarios.

The remaining of the paper is organized as follows. Section 2 introduces some useful background definitions and briefly reviews existing methods that are related to this paper. In Section 3, we explain the proposed distributed heterogeneous transfer learning method. In Section 4, we report an analysis of the computational complexity, while in Section 5, we describe the conducted experimental evaluation and discuss the obtained results. Finally, Section 6 concludes the manuscript and outlines some future work.

2. Background

As introduced in the previous section, transfer learning aims to enhance a task in the target domain by exploiting the knowledge coming from a source domain. In the following, we provide a formal definition of *domain* and *task*.

- A domain is defined as $D = \{F, P(X)\}$, where:
  - F is a feature space;
  - X is a set of observations;
  - $P(X)$ is the marginal probability distribution over X.
- A task is defined as $T = \{Y, f\}$, where:
  - Y is the output space of the prediction task;
  - f is a predictive function learned from a set of training examples in the form $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in Y$.

Therefore, given:

- $D_s = \{F_s, P(X_s)\}$ the source domain,
- $D_t = \{F_t, P(X_t)\}$ the target domain,
- $T_s = \{Y_s, f_s\}$ the source task,
- $T_t = \{Y_t, f_t\}$ the target task,

the goal is to improve the performance of the target predictive function f_t, by exploiting also the knowledge coming from D_s and T_s, where $D_s \neq D_t$ and/or $T_s \neq T_t$. In this context, the term *knowledge* may refer to entire predictive models, portions of predictive models (e.g., weights of a neural network, or a set of rules extracted from a decision tree), or hyper-parameters thereof.

According to the above definitions, two different transfer learning settings can be defined: *homogeneous transfer learning*, where $F_s = F_t$ and $P(X_s) \neq P(X_t)$, and *heterogeneous transfer learning*, where $F_s \neq F_t$ and, naturally, $P(X_s) \neq P(X_t)$. The latter is more challenging, more frequent in real-world applications, and is the focus of the present work. Therefore, in the following subsection, we briefly discuss existing works falling in such a category.

2.1. Related work on heterogeneous transfer learning

Most of existing heterogeneous transfer learning methods focus on the alignment of the input spaces of the source and target domains [11]. Such approaches have been exploited in several applications, including image recognition [12,19,51,36,20], text classification [35,50,39], drug efficacy classification [38], human activity classification [16], software defect classification [31], gene network reconstruction [27,30,29], and energy consumption prediction [15].

Long et al. [24] and Wu et al. [46] exploited the Maximum Mean Discrepancy (MMD) as a distance measure between feature distributions. Specifically, Wu et al. [46] proposed a method called Knowledge Preserving and Distribution Alignment (KPDA) that performs data augmentation on the target space with the aim to maximize the domain distribution alignment with respect to the source domain. KPDA is

based on matrix operations, and exploits Laplacian graph terms to preserve the closeness (resp., distance) among instances of the same class (resp., of different classes) in the latent common space. Long et al. [24] proposed a method called TJM that exploits MMD in a *reproducing kernel Hilbert space* to perform visual domain adaptation, that is based on the learning of a classifier for a new domain using labeled images from another domain.

Courty et al. [10] proposed a method called JDOT (Joint Distribution Optimal Transportation) which exploits few labeled instances in the source domain, and the data distribution of unlabeled instances in both source and target domains. JDOT performs a nonlinear transformation between the input space distributions of both domains and the output space of the source domain. To this aim, JDOT uses a map between the two domains called *optimal transport* to handle changes in both marginal and conditional distributions through a transformation function. Such a function combines both the distances between the samples and a loss function measuring the discrepancy in terms of labels.

Recently, Liang et al. [21] proposed a two-step method that aims to capture possible commonalities among heterogeneous related tasks (in this case, $T_s \neq T_t$). In the first step, the method learns one model for each task independently. Then it constructs a so-called *model matrix*, representing the information conveyed by the set of learned models. Subsequently, it learns low-rank and group-sparse patterns from such a model matrix. In the second step, a centralized transfer process shares the knowledge among the tasks in a privacy-preserving fashion based on the Wishart noise [45].

Wang and Breckon [41] proposed a method called Cross-Domain Structure Preserving Projection (CDSPP), that aims to learn domain-specific projections to map a sample of the features from source and target domains into a common subspace where data distributions are sufficiently aligned. CDSPP is also able to work in a semi-supervised setting. Similarly, Zhang et al. [49] proposed a method called JGSA that aims to match the source and the target feature spaces, in a statistical fashion. In addition, the method also exploits a geometry-based alignment, by projecting both source and target data into a low-dimensional subspace that simultaneously preserves the source domain class information and the target domain variance. This method has been specifically applied in the context of cross-domain object recognition, hand-written digit recognition, and action recognition.

Wang et al. [40] proposed a method called Balanced Distribution Adaptation (BDA) that adapts the distributions between the source and target domains and is also able to work with imbalanced data. BDA also analyzes the importance of the marginal (i.e., in the input space) and conditional (i.e., in the output space) distribution discrepancies between the source and the target domains. The authors applied the proposed method on five image datasets, evaluating multiple transfer *directions* from a dataset to another.

A different approach is considered in a recent study [23], where the authors exploited graphs to describe complex connections between instances. This approach is particularly important for the task of person re-identification, that aims to identify the same person from different surveillance videos. Specifically, the authors proposed a Heterogeneous Graph-driven Optimization scheme (H-GO) that captures different structures from unlabeled data. H-GO consists of four stages: *i)* supervised learning in the source domain; *ii)* construction of a heterogeneous graph in the target domain by exploiting unlabeled images captured by multiple cameras; *iii)* a heterogeneous affinity propagation procedure, applied on the heterogeneous graph, to reveal the structural affinity between images; *iv)* a heterogeneous affinity learning algorithm, applied on the identified structural affinity, to incrementally optimize the visual model.

Focusing on existing methods that, similar to ours, exploit the KL divergence, we can find two relevant works. In the first (i.e., the work by [18]), the authors exploit the Non-negative Matrix Factorization (NMF) for the specific application of document clustering. In the second [6], the authors adopt the KL divergence to align learned embedding spaces from the source and the target domains, in the specific application of sentiment classification of textual data. Contrary to such approaches, STEAL is also able to work in the PU learning setting and to solve general learning tasks without being constrained to specific applications. Moreover, our method (eventually) works in latent spaces, by exploiting the PCA that, contrary to embedding methods that aim at accurately reconstructing the original space, leads to orthogonal (i.e., independent) new features.

Transfer learning approaches have also been deeply adopted in the biological context [27,30,29], specifically for the identification of human gene regulatory activities (target domain), by exploiting the information conveyed by the mouse gene regulatory network (source domain). In particular, Mignone et al. [29] reconstructed both the mouse and the human gene regulatory networks, by taking into account a biological background knowledge (i.e., orthologous genes, between the mouse and the human organisms). Such a background knowledge strongly enhances the predictive power of the method, but, in most cases, a known mapping between source and target features and/or instances is not available. On the contrary, STEAL does not rely on the availability of such a background knowledge, and is able to automatically identify a proper matching for both feature and instance viewpoints.

Transfer learning was also recently exploited for cerebral stroke prediction [48,4,22,7]. Specifically, Zhang et al. [48] proposed a deep learning architecture working on cerebral images for Time Since Stroke (TSS) onset classification. The method trains a model on an easier clinical task (i.e., stroke detection) and then refines the model with different thresholds for TSS (e.g., TSS $< 4.5h$).

Ban et al. [4] adopted deep learning models for stroke detection from facial images. Specifically, they exploited pre-trained models on images in order to predict facial paralysis caused by acute stroke in time-series data from video clips. Lin et al. [22] trained deep learning models for the prognosis of stroke recovery by analyzing stroke survivor patients' data during stretching activities. However, the works by [48], [4] and [22] are suitable only for image datasets, and the proposed architectures are strictly dependent on the specific application domain. Therefore, they cannot be considered general methods applicable for transferring knowledge among different domains.

Chen et al. [7] proposed a deep learning method for transfer learning aiming at exploiting multiple source domains, such as hypertension and diabetes prediction, in order to improve the performance of imbalanced classification in brain stroke prediction. This method is more general and independent of the data type with respect to the previously mentioned approaches, since it is able to work with tabular data, i.e., instances represented as feature vectors. The authors aim to improve the performance of predictive models for low-ranked hospital data, usually small and imbalanced, by exploiting also external data from top-ranked hospital data. However, this method works only with homogeneous feature spaces by exploiting common attributes manually selected among the datasets about hypertension, diabetes, and brain stroke.

Transfer learning approaches have also been applied in the energy field. Gao et al. [15] solved the task of energy consumption prediction in Chinese buildings, by fine-tuning pre-trained models. However, this approach is not applicable if there is no available pre-trained model for the domain at hand. Hooshmand & Sharma [17] considered the task of developing predictive models with limited data for the forecasting of the daily electricity demand. They first developed a predictive model based on Convolutional Neural Networks (CNN) from the source domain. Then, they fine-tuned the last layer of the pre-trained CNN in order to address the challenge of limited training data in the target domain.

Overall, transfer learning attracted much attention in the research community, and several methods have been proposed to solve different tasks in different application domains. However, as already emphasized in the previous section, none of the existing approaches simultaneously

exhibits all the peculiarities of the method STEAL proposed in this paper.

3. The proposed method STEAL

In this section, we describe the proposed method STEAL. Its ability to work also in the Positive-Unlabeled (PU) learning setting is based on a clustering approach [30] that estimates the labels and the confidence thereof for unlabeled instances, according to their similarity with positive instances. We also recall that, among our objectives, there is the distribution of the workload over multiple computational nodes, in order to be able to work with large datasets (large number of instances). The distribution of the workload is achieved by properly designing the algorithms we propose (that compose the method STEAL) according to the MapReduce paradigm, under the umbrella of the Apache Spark framework. Therefore, all the pseudo-code descriptions of the proposed algorithms reported in this Section will follow the MapReduce paradigm, applied on the Resilient Distributed Dataset (RDD) data structure available in Apache Spark. In order to make them more easily interpretable, in Appendix A, we report a brief explanation of how the distributed approach in Apache Spark is implemented and the semantics of the main operators that can be applied to RDDs.

The possible high dimensionality (large number of features) of the data is faced through a distributed variant of the Principal Component Analysis (PCA).[1] Finally, in order to work with heterogeneous domains, STEAL exploits the Kullback-Leibler (KL) divergence to *align* the descriptive variables of the source and target domains. Such an alignment is then exploited to find a proper matching between source and target instances, in order to augment the training set from a descriptive viewpoint.

The clustering-based estimation of the labels, the adoption of the distributed PCA and the feature alignment procedure will be detailed in Section 3.1. In Section 3.2, we will describe the approach adopted to match source and target training instances. Finally, in Section 3.3 we provide some details about the considered predictive models learned in a distributed fashion. The general workflow followed by STEAL is formalized in Algorithm 1.

Algorithm 1: STEAL.

Data:

$\cdot X_t$: RDD. Target instances, represented as p_t-dimensional distributed feature vectors
$\cdot X_s$: RDD. Source instances, represented as p_s-dimensional distributed feature vectors

Result:

$\cdot f: \mathbb{R}^{p_t} \rightarrow Y_t$. Learned predictive function for the target domain

```
1  begin
       // ------- STAGE 1 - INSTANCE LABELING, FEATURE
          REDUCTION and ALIGNMENT ------
       // If the learning setting is PU, estimate the label for
          unlabeled instances
2      if positive_unlabeled(X_t, X_s) then
3          X_t ← instanceLabeling(X_t) // Algorithm 2
4          X_s ← instanceLabeling(X_s) // Algorithm 2
5      Y_t ← X_t.select(label)
6      X_t ← X_t.drop(label)
       // If the user chooses to perform dimensionality
          reduction
7      if highly_dimensional(X_t, X_s) then
8          p ← sqrt(min(p_t, p_s))
9          X_t ← distributed_PCA(X_t, p)
10         X_s ← distributed_PCA(X_s, p)
       // Align source features according to their best
          matching with target features.
11     X_{s_aligned} ← alignSourceDomain(X_t, X_s) // Algorithm 3
       // -------------------------------------------------------
       // ------- STAGE 2 - INSTANCE
          MATCHING-----------------------------
12     trainingset ← instanceMatching(X_t, X_{s_aligned}) // Algorithm 4
       // ------- STAGE 3 - DISTRIBUTED MODEL
          TRAINING----------------------
13     f ← train_predictive_model(trainingset, Y_t)
14     return f
```

3.1. Stage 1 - instance labeling, feature reduction and alignment

In the first stage, in the case of a PU learning setting, we aim at estimating the labels of the unlabeled instances. For this purpose, we apply a prototype-based clustering algorithm in order 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 (see Algorithm 2). In STEAL, we exploit the silhouette cluster analysis [37] to automatically estimate the optimal number of clusters k (see Algorithm 2, line 3).

The considered clustering algorithm, known as *k-means*|| [3],[2] is a distributed version of the *k-means*++ algorithm [2].

The initialization step of *k-means*++ operates iteratively: it randomly selects only the first centroid, whereas, for the subsequent centroids, it computes, for each remaining point, the probability of being chosen as a centroid based on the Euclidean distances from the previously selected centroids. This strategy aims to ensure that subsequent centroids are maximally distant from the existing ones, mitigating the risk of converging to local minima prematurely. The adopted implementation, *k-means*||, independently selects more than one centroid at each iteration. As this process could generate more than k centroids, the algorithm prunes them by assigning weights based on the number of data points closest to each centroid from the original dataset.

Once the initial centroids are determined, the algorithm proceeds with the classical Lloyd's update algorithm. For further details, refer to the work by [3].

The motivation behind the adoption of a solution based on the *k-means* algorithm is that it has been proven to be effective in the PU learning setting [27,30,29,34,28]. The score we compute for each unlabeled instance is in the interval $[0, 1]$, where a value close to 0 means that the unlabeled example is likely to be a negative example, whereas a value close to 1 means that the unlabeled example is likely to be a 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 (see Algorithm 2, line 9). As similarity function, we consider the function sim: $\mathbb{R}^r \times \mathbb{R}^r \rightarrow [0, 1]$ defined as:

$$sim(a, b) = 1 - \frac{\sqrt{\sum_{i=1}^{r} \left(a_i - b_i\right)^2}}{r} \tag{1}$$

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 (pseudolabels), where the computed score represents the confidence on the fact that a given unlabeled instance is a positive instance. Note that in this phase we only assign pseoudolabels to both source and target datasets. These pseoudolabels are only used

[1] spark.apache.org/docs/latest/api/scala/org/apache/spark/mllib/feature/PCA.html.

[2] spark.apache.org/docs/latest/api/scala/org/apache/spark/ml/clustering/KMeans.html.

Algorithm 2: instanceLabeling(X).

Data:

·X: RDD. Set of labeled and unlabeled instances

Result:

·X_l: RDD. Set of fully labeled instances

```
1  begin
2      P ← getPositiveInstances(X)
3      k ← sil(P) // Estimates the optimal number of clusters via
           silhouette cluster analysis
4      C ← distributed_kmeans(P, k) // Identify the positive
           instance prototypes
5      X_l ← X.map(x→{
6          if x ∈ P then
7              (x, 1.0) // Positive instance → score = 1.0
8          else
9              (x, max_(c∈C)(sim(x, c))) // Equation (1)
10         end
11     })
12     return X_l
13 end
```

for giving STEAL the opportunity to manage and exploit PU data during the training.

Subsequently, as mentioned in the previous section, if the dataset is highly dimensional,[3] STEAL applies a distributed variant of the PCA, in order to identify a reduced set of (non-redundant) new features. We extract $\sqrt{min(k_s, k_t)}$ new features, according to the generally accepted solution adopted when defining the number of clusters k of clustering algorithms [26].

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 out the subsequent step (see Section 3.2).

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. Indeed, by selecting the feature of the source domain exhibiting the minimum loss for each feature of the target domain, we identify a proper alignment such that instances in the source domain are represented as similar as possible to instances of the target domain. More 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 KL divergence between F_{t_i} and F_{s_j} as:

$$KL(F_{t_i}, F_{s_j}, X_t, X_s) = \sum_{x \in (un(F_{t_i}, X_t) \cap un(F_{s_j}, X_s))} p(x, F_{t_i}, X_t) \ln \frac{p(x, F_{t_i}, X_t)}{p(x, F_{s_j}, X_s)}, \tag{2}$$

where $un(F, X)$ represents the set of distinct values of the feature F in the dataset X, and $p(x, F, X)$ represents the relative number of occurrences of the value x on the feature F in the dataset X, after a proper discretization of the feature F.[4]

[3] Note that this can be considered as an optional step and the need to apply a feature reduction step mainly depends on the availability of computational resources.

[4] In STEAL, we discretize continuous features using equal-width discretization strategy with a bin size of 0.01, after a min-max normalization.

Algorithm 3: alignSourceDomain(X_t, X_s).

Data:

·X_t: RDD. Target domain instances (in the $\langle instance\_id, feature, value \rangle$ form);
·X_s: RDD. 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
       // Consider each possible value as key, independently
           from the feature where it appears
2      tValKey ← X_t.map{case(instance_id, feature, value) → ⟨value, feature⟩}
3      sValKey ← X_s.map{case(instance_id, feature, value) → ⟨value, feature⟩}
       // Join the target and source structures according to
           the feature values
4      commonVals ← tValKey.join(sValKey)
       // Count the frequencies of values for the target
           features
5      tValueFreq ← commonVals
6          .map{case(value, tFeat, sFeat) → ⟨⟨value, tFeat⟩, 1⟩}
7          .reduceByKey((aCount, bCount) → aCount + bCount)
       // Count the frequencies of values for the source
           features
8      sValueFreq ← commonVals
9          .map{case(value, tFeat, sFeat) → ⟨⟨value, sFeat⟩, 1⟩}
10         .reduceByKey((aCount, bCount) → aCount + bCount)
       // Compute the number (cardinality) of value matches for
           each target feature
11     tAttrCard ← tValueFreq
12         .map{case(⟨⟨value, feat⟩, freq⟩) → ⟨feat, freq⟩}
13         .reduceByKey((aFreq, bFreq) → aFreq + bFreq)
       // Compute the number (cardinality) of value matches for
           each source feature
14     sAttrCard ← sValueFreq
15         .map{case(⟨⟨value, feat⟩, freq⟩) → ⟨feat, freq⟩}
16         .reduceByKey((aFreq, bFreq) → aFreq + bFreq)
       // Compute the value probabilities for the target domain
17     tValueProbs ← tValueFreq
18         .map{case(⟨value, feat⟩, freq) → ⟨feat, ⟨value, freq⟩⟩}
19         .join(tAttrCard)
20         .map{case(feat, ⟨value, freq, card⟩) → ⟨value, feat, freq/card⟩}
21     sValueProbs ← sValueFreq
22         .map{case(⟨value, feat⟩, freq) → ⟨feat, ⟨value, freq⟩⟩}
23         .join(sAttrCard)
24         .map{case(feat, ⟨value, freq, card⟩) → ⟨value, feat, freq/card⟩}
       // Compute the KL divergences
25     divergences ← tValueProbs
26         .cartesian(sValueProbs)
27         .map{case(⟨tValue, tFeat, tProb⟩, ⟨sValue, sFeat, sProb⟩) →
               ⟨⟨tFeat, sFeat⟩, tProb · log(tProb/sProb)⟩}
28         .reduceByKey((aKLterm, bKLterm) → aKLterm + bKLterm)
       // Find the best target-source features matching by
           minimizing the KL divergences
29     minDivergences ← divergences
30         .map{case(⟨tFeat, sFeat⟩, KLdiv) → ⟨tFeat, ⟨sFeat, KLdiv⟩⟩}
31         .reduceByKey{case(⟨sFeat1, KLdiv1⟩, ⟨sFeat2, KLdiv2⟩) →
32             if(KLdiv1 < KLdiv2)
33                 ⟨sFeat1, KLdiv1⟩
34             else
35                 ⟨sFeat2, KLdiv2⟩}
       // Align source features to target features, according
           to the minimum divergences
36     X_{s_aligned} ← align(X_s, minDivergences)
37     return X_{s_aligned}
```

In Figs. 1 and 2 we graphically show this alignment step. In Algorithm 3, we report its pseudocode description using the MapReduce paradigm. The algorithm starts by considering the values assumed by the features as the key of a paired RDD (Algorithm 3, lines 2-3). The result is then used to find common values assumed by target and source features, through a join operation (Algorithm 3, line 4). The frequency

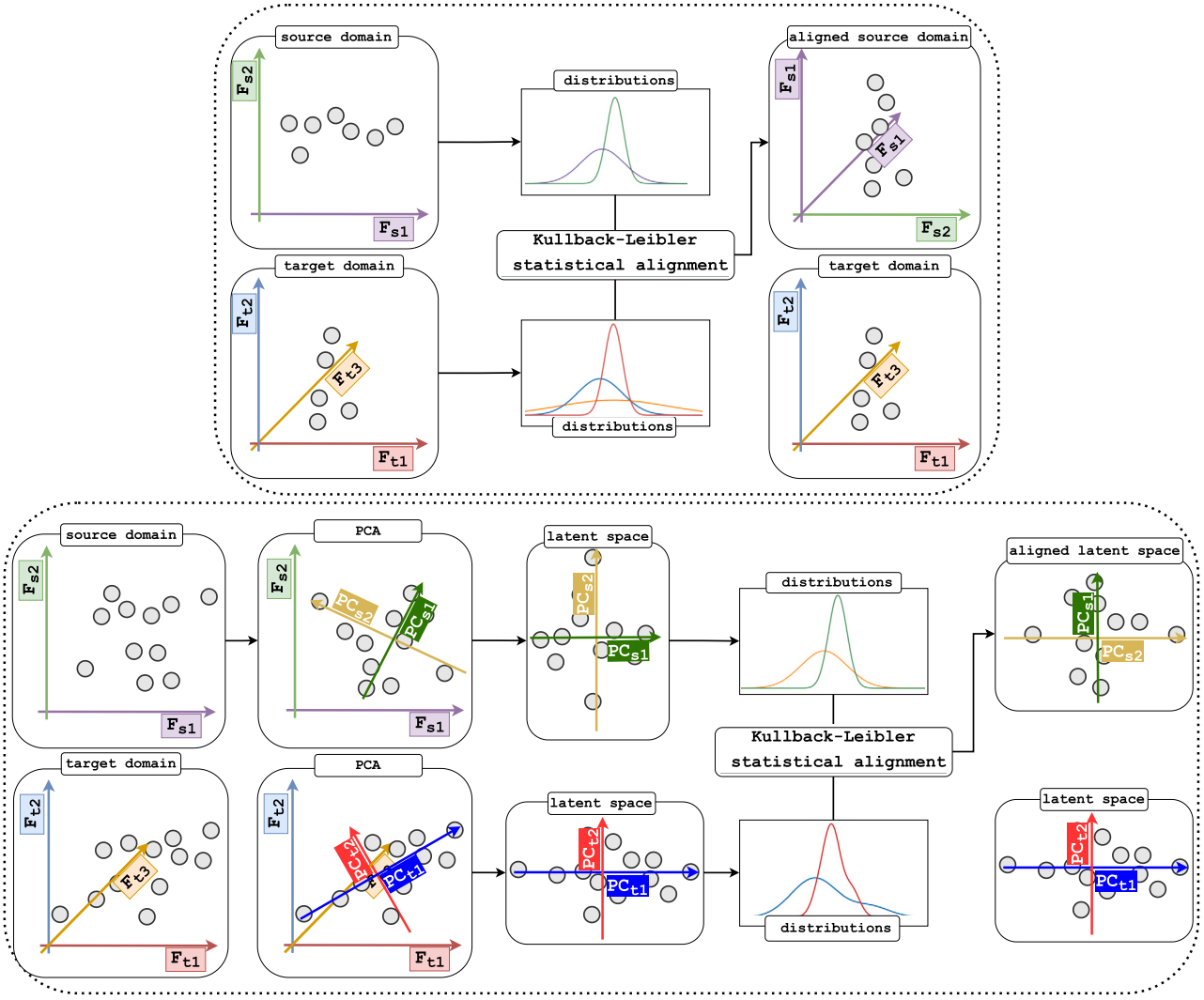

Fig. 1. A graphical representation of the feature alignment performed by STEAL, with the source domain represented in a 2-dimensional feature space and the target domain represented in a 3-dimensional feature space. On the top, we can observe that the source feature F_{s_1} has been selected twice, for the target attributes F_{t_2} and F_{t_3}. In the bottom, we can observe the case in which the PCA is applied to reduce the dimensionality of both source and target features spaces (note that in this specific toy example, it would not be necessary), before proceeding with the alignment of the features.

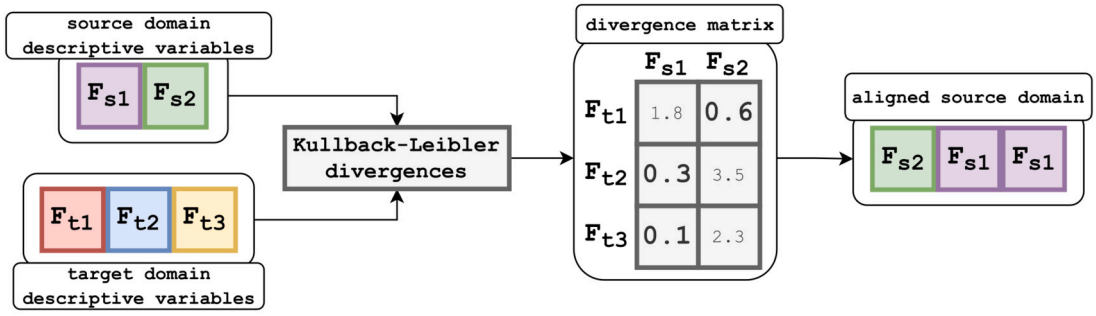

Fig. 2. A graphical focus on the feature alignment between the source and the target domains. The central divergence matrix represents the computed KL divergences between all the possible target-source feature pairs, from which those with the lowest divergence (row-wise) are selected.

of each distinct value is then computed for each target (Algorithm 3, lines 5-7) and source (Algorithm 3, lines 8-10) feature, which is then exploited to estimate the probabilities used in 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 feature (Algorithm 3, lines 11-24). Finally, we compute the KL divergence for each possible target-source pair of features (Algorithm 3, lines 25-28), which are then exploited to find the best (i.e., with the minimum divergence) source feature to align with each target feature (Algorithm 3, lines 29-35).

We remark that the proposed approach implicitly performs a feature selection on the source domain, since some of the features could also

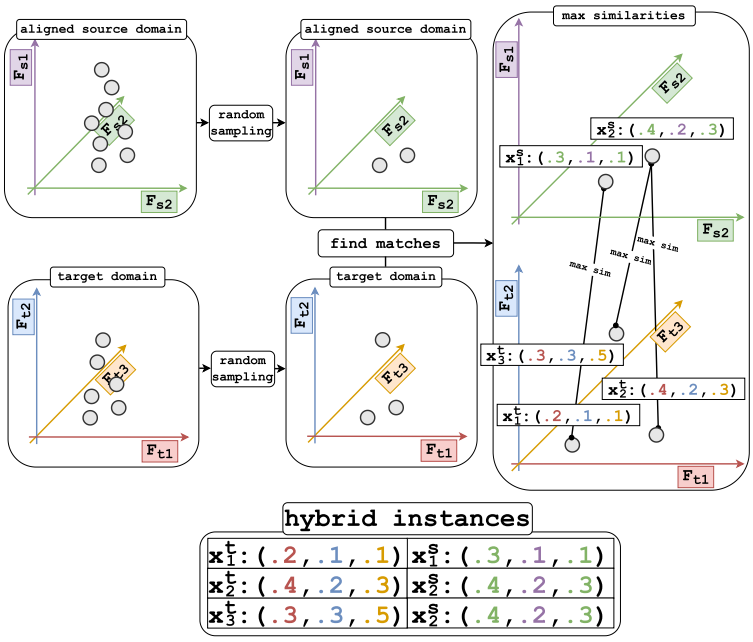

Fig. 3. A graphical representation of the instance matching step performed by STEAL.

have no match with 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.

3.2. Stage 2 - source-target instance matching

After the first stage, source instances are represented in a feature space having the same dimensionality of the target domain, where the i-th feature of the source domain is the most proper representative of the i-th feature of the target domain, because it provides the minimum loss of information, according to the KL-divergence. Therefore, 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: the features of the most similar source instance are concatenated to each target instance. The similarity function used for this purpose is, coherently with the clustering phase that of Equation (1), that is based on the Euclidean distance. In this way, STEAL augments the descriptive features of target instances with those coming from aligned, best-matching, source instances.

The approach followed in this stage is depicted in Fig. 3, while a formal description using the MapReduce paradigm is reported in Algorithm 4. Specifically, we compute the Cartesian product between multiple subsets (randomly selected, with replacement) of target and source instances, and *reduce* the resultset by taking the source instances with the maximum similarity (Algorithm 4, lines 7-14). The adoption of multiple random subsets, instead of working on the whole set of instances, provides some relevant advantages: *i)* it 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, and *ii)* it reduces the computational costs of the whole algorithm (see Section 4).

At the end of this step, we obtain a single target-source hybrid training set by merging the target-source concatenated instances constructed from each sample (Algorithm 4, line 15).

In Algorithm 4, the number of samples, the percentage of selected target instances per sample and the percentage of selected source instances per sample are governed by the parameters $numSamples$, $instRatio_t$ and $instRatio_s$, respectively.

3.3. Stage 3 - distributed model training

In this section, we briefly describe the method adopted by STEAL to learn a predictive model from the training set constructed during the previous stages. Specifically, we considered the distributed variant of Random Forests (RF) [5] available in Apache Spark.[5] This choice is due to the fact that RF generally exhibit a high accuracy and efficiency for both classification and regression tasks, and to the possibility to distribute both the training and the prediction phases in a natural way.

It is worth mentioning that an unseen target instance would be described in the p_t-dimensional feature space, while the model is learned from the hybrid training set described in the $2p_t$-dimensional feature space. Therefore, in order to apply the learned model to predict the output value for a new target instance, we simply concatenate its feature vector twice. This step is coherent with the approach followed in the instance-matching step, since the most similar feature vector of a given target instance would be itself, i.e., $sim(x_i^t, x_i^t) = 1$, for any target instance x_i^t.

The general workflow followed to learn a RF in a distributed fashion is detailed by [8]. The random selection of the features and instances for each tree follows the standard approach adopted by Random Forests [5]. On the other hand, the algorithm distributes the workload for the identification of the best feature and split (i.e., a discrete value for categorical features, or a threshold for continuous numerical features), for each node of each tree of the forest. This process is based on the optimization of the node impurity in terms of Gini index (for classification tasks) or variance (for regression tasks).

[5] spark.apache.org/docs/latest/ml-classification-regression.html#random-forest-classifier.

Algorithm 4: instanceMatching(X_t, X_s).

Data:

· X_t: RDD. Target domain instances (in the $\langle instance\_id, feature\_vector\rangle$ form)
· X_s: RDD. Aligned source domain instances (in the $\langle instance\_id, feature\_vector\rangle$ form)

Result:

· *trainingSet*: RDD. Training set in the $2p_t$-dimensional feature space, obtained by concatenating matched target-source instances

```
1  begin
2      trainingSet ← ∅
3      for i ← 1 to numSamples do
           // Random sampling
4          targetSample ← X_t.sample(instRatio_t)
5          sourceSample ← X_s.sample(instRatio_s)
           // Cartesian product between the sampled target and
              source instances
6          cartesian ← targetSample.cartesian(sourceSample)
           // Pairwise similarity, computed through Equation (1)
              between the sampled target and source instances.
              Concatenation of the target instance with the
              source instance having the maximum similarity
7          matchedInst ← cartesian
8              .map{case (⟨tID, tFeat⟩, ⟨sID, sFeat⟩) →
                   ⟨tID, ⟨tFeat, sFeat, sim(tFeat, sFeat)⟩⟩} // Equation
                   (1)
9              .reduceByKey{case(⟨tFeat1, sFeat1, sim1⟩,⟨tFeat2, sFeat2,
                sim2⟩) →
10                 if(sim1 > sim2)
11                     ⟨tFeat1, sFeat1, sim1⟩
12                 else
13                     ⟨tFeat2, sFeat2, sim2⟩}
14             .map{case (tID, ⟨tFeat, sFeat, sim⟩) →
                   ⟨tID, ⟨tFeat, sFeat⟩⟩}
           // Append the sampled matched instances to the
              training set
15         trainingSet ← trainingSet.union(matchedInst)
16     return trainingSet
```

Algorithm 5: bestSplit($featIndex, binAggr$).

Data:

· $featIndex \in \mathbb{N}$. The index of the feature for which we want to find the best split
· *binAggr*. Data structure containing, for each bin, the cumulative statistics for that bin plus all preceding bins.

Result:

· $bestSplit \in \mathbb{N}$. Index of the best split

```
1  begin
2      numSplits ← binAggr.getNumSplits(featIndex)
3      bestSplit ← Range(0, numSplits)
4          .map{case splitIdx→{
5              leftChildStats ← binAggr.getStats(featIndex, splitIdx)
6              rightChildStats ←
                 binAggr.getStats(featIndex, numSplits)
               // Subtract the left child stats from the right
                  child stats
7              rightChildStats.subtract(leftChildStats)
8              gain ←
                 calculateImpurity(leftChildStats, rightChildStats)
9              ⟨splitIdx, gain⟩
10         }
11         .reduce{case(⟨splitIdx1, gain1⟩, ⟨splitIdx2, gain2⟩) → {
12             if(gain1 > gain2)
13                 ⟨splitIdx1, gain1⟩
14             else
15                 ⟨splitIdx2, gain2⟩
16     } return bestSplit
```

In Algorithm 5, we report the pseudo-code description of the distributed identification of the best split for a given feature. In Algorithm 5, a *bin* represents a distinct discrete value for categorical attributes or a range of values for continuous numerical attributes, after applying a discretization, while the data structure *binAggr* stores, for each bin, the cumulative statistics for that bin together with all preceding bins, and is adopted to efficiently compute, in an incremental fashion, the impurity of the nodes. Specifically, the algorithm returns the index of the split that leads to the maximum gain in terms of reduction of impurity (Algorithm 5, lines 11-16).

In the distributed implementation, both data partitioning and task partitioning are exploited. In data partitioning, a vertical method is employed, where the training dataset is divided into multiple sets of features. Each feature set is then linked with the target variable and distributed across the Spark cluster. To handle the issue of increasing data volume, a data multiplexing technique is introduced: instead of copying the sampled data, only their indices are recorded in a Data-Sampling-Index (DSI) table. This DSI table, along with the feature subsets, is then allocated to all the computational nodes. During the training process of each decision tree, the relevant data is retrieved from the corresponding feature subset via the DSI table. As for task partitioning, *numTrees* decision trees are concurrently built at the primary level of the training process. Additionally, the descriptive variables within each decision tree are processed simultaneously, a process carried out at the second level of the training process, encompassing impurity computing and node-splitting tasks. Additional details can be found in the work by [8].

4. Time complexity analysis

In this section, we briefly discuss the time complexity of STEAL. Let:

- $n_t \in \mathbb{N}$ and $n_s \in \mathbb{N}$ be the number of instances of the target domain and of the source domain, respectively;
- $p_t \in \mathbb{N}$ and $p_s \in \mathbb{N}$ be the number of features of the target domain and of the source domain, respectively;
- $numSamples \in \mathbb{N}$ be the number of samples of the instance matching stage;
- $instRatio_t \in [0, 1]$ and $instRatio_s \in [0, 1]$ be the ratio of sampled instances per sample, of the target domain and of the source domain, respectively;
- $numTrees \in \mathbb{N}$ be the number of trees of the random forest.

In the first stage, STEAL computes the KL divergence for each possible pair of target-source features, namely $p_t \cdot p_s$ KL divergences.

The computation of KL divergences is based on the identification of common values of target-source features, which is achieved through a join operation (see Algorithm 4, line 4). This join operation costs $O(n_t + n_s)$, since Apache Spark exploits the construction of ordered structures, that costs $O(n_t \cdot \log n_t)$ and $O(n_s \cdot \log n_s)$ respectively for the target domain and the source domain. Therefore, assuming that the ordering step is purposely performed to optimize the join operation, the total time complexity of the latter is $O(n_t \cdot \log n_t + n_s \cdot \log n_s + (n_t + n_s)) \approx O(n_t \cdot \log n_t + n_s \cdot \log n_s)$.

Although we perform a single distributed join operation, in principle its cost should be considered for each pair of target-source feature. Therefore, the time complexity for the computation of $p_t \cdot p_s$ KL divergences is $O(p_t \cdot p_s \cdot (n_t \cdot \log n_t + n_s \cdot \log n_s))$.

In the second stage, STEAL builds *numSamples* random samples, each consisting of $instRatio_t \cdot n_t$ instances from the target domain, and

$instRatio_s \cdot n_s$ from the source domain. The similarity between each sampled target-source pair of instances is then computed using Equation (1), which time complexity is $O(p_t)$. Therefore, the final time complexity of the second stage is $O(numSamples \cdot n_t \cdot instRatio_t \cdot n_s \cdot instRatio_s \cdot p_t)$. Since $numSamples$ is a constant value, and $instRatio_t$ and $instRatio_s$ are constant values in $[0, 1]$, in the worst case the time complexity of the second stage can be approximated to $O(n_t \cdot n_s \cdot p_t)$.

Finally, the time complexity of the third stage depends on the complexity of training a model based on Random Forests. In the worst case, the number of instances corresponds to n_t, while the number of features is $2p_t$, which is due to the feature augmentation obtained through instance matching. Since each tree in Random Forests is grown up to the maximum height, i.e., $\log n_t$, the complexity of the training $numTrees$ trees is $O(numTrees \cdot 2p_t \cdot n_t \cdot \log n_t)$, that, since $numTrees$ is a constant value, can be approximated to $O(p_t \cdot n_t \cdot \log n_t)$.

In summary, the STEAL time complexity can be approximated as the sum of the complexity of the three stages, namely:

$O(p_t \cdot p_s \cdot (n_t \cdot \log n_t + n_s \cdot \log n_s)) +$
$O(n_t \cdot n_s \cdot p_t) +$
$O(n_t \cdot \log n_t \cdot p_t)$.

Since, in general, we can assume that $\log n_t < n_s$ and $\log n_s < n_t$, the worst case time complexity of STEAL corresponds to $O(n_t \cdot n_s \cdot p_t \cdot p_s)$.

5. Experiments

In this section, we describe our experimental evaluation aiming to assess the effectiveness and the efficiency of the proposed method STEAL. We conducted our experiments in three relevant application domains 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);
- in medicine, and specifically in the prediction of cerebral stroke for hospital patients (target domain), using also the knowledge related to sepsis for hospital patients (source domain);
- in the energy sector, and specifically in the prediction of the energy consumption of a set of customers of a given area (target domain), exploiting also the knowledge related to a set of customers of another area (source domain).

We compared the results achieved by STEAL with some state-of-the-art transfer learning competitors introduced in Section 2, that are able to work with heterogeneous feature spaces, namely **TJM** [24], **JGSA** [49], **BDA** [40], and **JDOT** [10]. Note that they still require that source and target feature spaces have the same dimensionality.

Moreover, we also evaluated the results obtained by some baseline approaches:

- **T (no transfer)**, that is a predictive model learned only from the dataset of the target domain. The comparison with this baseline allows us to specifically evaluate the contribution coming from the exploitation of the source domain.
- **S (optimal feature alignment)**, that is a predictive model trained only from the source domain dataset, assuming that the optimal feature alignment is provided by a domain expert or, in any case, known beforehand, and not automatically identified as done by STEAL. The comparison with this baseline allows us to assess the effectiveness of the combined exploitation of the source and target data performed by STEAL.
- **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 beforehand. This baseline allows us to understand if the approach adopted by STEAL to automatically align target and source features is effective and can fruitfully be adopted when the true cross-domain feature matching is not known a priori.

In the cases in which baselines based on the optimal feature alignment are applicable, further details will be provided in the next subsections.

All the experiments were run on a computational cluster consisting of 3 nodes, each equipped with a 6-cores Intel i7-8700 CPU and 64 GB RAM.

5.1. Gene regulatory network reconstruction

The reconstruction of the human gene regulatory network consists in the identification of currently unknown gene regulatory activities. A gene regulatory network is composed by a set of genes (represented as nodes) and a set of known regulations (represented as edges). The reconstruction of a gene regulatory network can be performed by resorting to link prediction methods [25], able to work in the PU learning setting. Indeed, observed links can be considered positive examples, but since the goal is to infer the existence of additional links among the non-observed ones, assuming that they are negative examples is conceptually wrong.

In this experiment, we used the dataset defined by [30], which consists of two gene regulatory networks, one for the human organism (target domain) and one for the mouse organism (source domain). For both organisms, each gene is represented through multiple expression levels measured in different samples related to 6 organs (brain, bone marrow, heart, liver, lung, skin). We considered both the available versions of the dataset, i.e., the heterogeneous version (**HET**), where human and mouse genes are described by 174 and 161 expression levels, respectively, and the homogeneous version (**HOM**) obtained by averaging the expression levels per organ. Note that, for the HET version of this dataset, the optimal feature alignment is not available, while for the HOM version the optimal feature alignment can be defined according to some domain knowledge about the organs. Specifically, the average expression level associated with a human gene for a given organ is aligned with the average expression level associated with a mouse gene for the same organ.

While we encountered no issue in running our experiments on STEAL, we faced some out-of-memory errors with TJM, JGSA, BDA, and JDOT on our computational cluster. Therefore, in order to compare STEAL not only with baseline approaches, but also with state-of-the-art methods, we also ran the experiments on a reduced version of the homogeneous dataset (**HOM-RED**) consisting of 2% randomly selected instances.

Note that each training instance is actually an edge between two genes. Therefore, the training instances have a dimensionality of 348 and 322 in the heterogeneous dataset, for the human and the mouse organisms respectively, obtained by concatenating the features of the involved genes. On the other hand, the dimensionality of the training instances of the homogeneous dataset is 12 for both organisms. Detailed quantitative information of the considered datasets is shown in Table 1.

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, coherently with the transductive learning setting. 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. Moreover, we computed the area under the recall@k curve (AUR@K). Note that other well-known measures, such as the Area Under ROC curve (AUROC) and the Area Under Precision-Recall curve (AUPR), result to be inapplicable and unfair in the positive-unlabeled setting since they introduce wrong biases in the ground truth.

Table 1
Quantitative information of the considered datasets in the biological domain.

	HET		HOM		HOM-RED	
	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

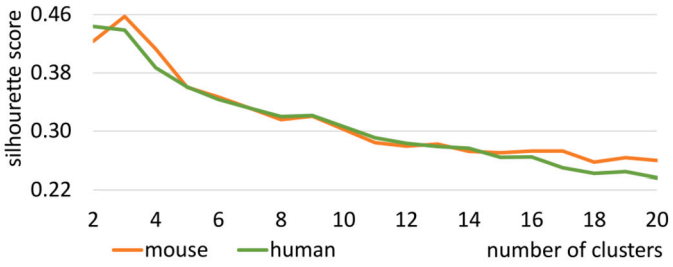

Fig. 4. Silhouette scores for the human and mouse positive interactions.

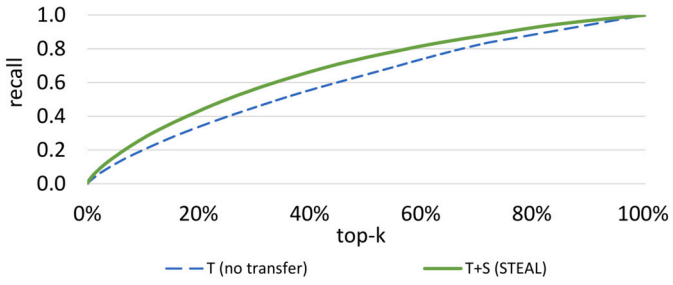

Fig. 5. Recall@k results obtained on the HET dataset by STEAL and baseline approaches.

In STEAL, we performed the estimation of the label confidence (as detailed in Section 3.1) by identifying 3 clusters for the mouse positive instances and 2 clusters for the human positive instances. Such values were identified through the silhouette cluster analysis, which result is depicted in Fig. 4.

As regards the hyperparameters of the instance matching phase, 10 samples were considered (i.e., $numSamples = 10$), each involving 10% random training examples from the source and from the target domains (i.e., $instRatio_t = 10\%$ and $instRatio_s = 10\%$). The rationale behind the choice of specific values of such parameters can be analogous to that usually adopted in ensemble approaches (see, for example, the number of trees and the number of randomly selected instances for each tree, in Random Forest). The actual values we adopted in our experimental evaluation come from some preliminary experiments that we performed, which have shown that increasing the value of such parameters do not generally provide significant advantages but introduces additional computational costs, as it could be expected considering the time complexity analysis reported in Section 4.

For the dataset HET, we ran STEAL with the distributed PCA, that led to identify 18 new features.

Since JGSA, TJM, BDA, and JDOT are not able to naturally work in the PU learning setting, we fed them with the labels and scores estimated by STEAL according to Stage 1. We optimized their hyperparameter d, i.e. the *subspace bases* to identify between the target and source domains, which best value resulted to be, after a grid search in the interval $[1, 12]$, $d = 12$ for JGSA and $d = 6$ for BDA and TJM. Regarding JDOT, we considered the default configuration of its parameters ($\lambda = 1.0, \gamma = 0.1, \alpha = 0.25$) that falls within the range of values suggested by [10].

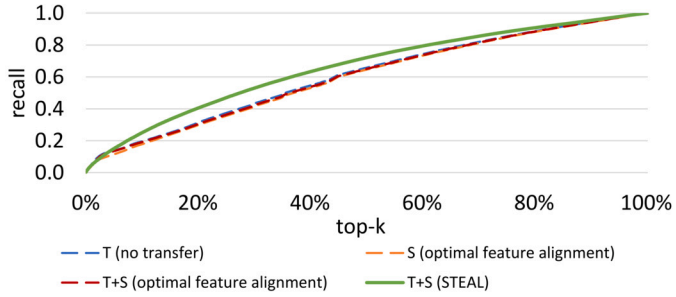

Fig. 6. Recall@k results obtained on the HOM dataset by STEAL and baseline approaches.

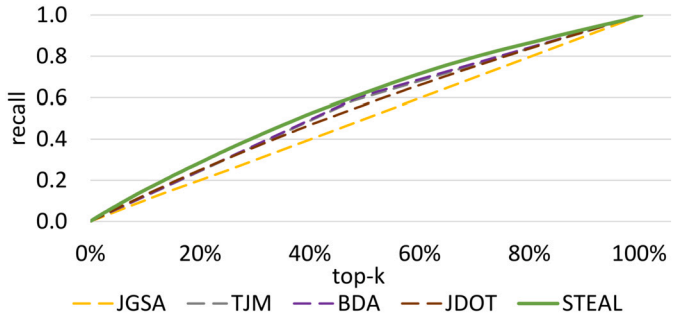

Fig. 7. Recall@k results obtained on the HOM-RED dataset by STEAL and its competitors.

In Figs. 5, 6, 7, we plot the recall@k curves measured on the datasets HET, HOM and HOM-RED, respectively, by varying the value of k. The x-axis represents the ratio between the considered value of k and the cardinality of the whole set of instances in the test set. By observing the figures, it is clear that the line representing STEAL always lies over the others, independently on the considered value of k. This result clearly indicates that STEAL is able to take advantage of knowledge coming from the source domain independently on the number of retrieved instances. In Tables 2, 3, and 4, we report the results in terms of AUR@K, that allow us to assess the differences between the considered approaches with a single quantitative value, that is independent on the value of k. From the results in Tables 2 and 3, 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 27.6% in the homogeneous setting and by 11.3% 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 in the homogeneous setting (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) for the HOM dataset), does not provide as much as benefits as STEAL.

Table 2
Gene network reconstruction results in terms of AUR@K, and relative improvement over T (no transfer), obtained by STEAL on the full heterogeneous datasets (HET).

Method	AUR@K	Impr. over T
T (no transfer)	0.610	-
STEAL	0.679	11.3%

Table 3
Gene network reconstruction results in terms of AUR@K, and relative improvement over T (no transfer), obtained by STEAL and baseline approaches on the full homogeneous datasets (HOM).

Method	AUR@K	Impr. over T
T (no transfer)	0.533	-
S (optimal feature alignment)	0.544	2.1%
T+S (optimal feature alignment)	0.551	3.4%
STEAL	0.680	27.6%

Table 4
Gene network reconstruction results in terms of AUR@K, obtained by STEAL and its competitors on the reduced homogeneous dataset (HOM-RED).

Method	HOM-RED
JGSA	0.500
TJM	0.554
BDA	0.558
JDOT	0.540
STEAL	0.589

In Table 4 we show a comparison between STEAL and its transfer learning competitors on the reduced homogeneous dataset. The results show that the way STEAL exploits knowledge from the source domain is more effective than the competitors, with an advantage ranging from 5.5% (over BDA) to 17.8% (over JGSA). This is motivated by the combined effect of feature alignment and instance matching that we implement in STEAL.

As an additional analysis, we evaluated the effect on STEAL of the number of available labeled instances. We performed this analysis on the HOM dataset, considering different percentages of labeled instances, namely, 20%, 40%, 60% and 80%, compared with the original scenario in which all the labeled instances (100%) are used. By decreasing the number of labeled instances considered, we introduce/strengthen the unbalancing between the number of labeled instances and the number of unlabeled instances. In this way, we can assess the robustness of STEAL to such critical scenarios. The results are reported in Fig. 8 in terms of recall@k and, in Table 5, in terms of AUR@K. From the figure, we can observe that STEAL provides robust results, also when we only consider 20% of the labeled instances. Specifically, we can observe a drop of only 7.42% in terms of AUR@K when we consider 20% of the labeled instances instead of 100%, which can be considered negligible. If we compare these results with those reported in Table 3, we can see that even if we reduce the percentage of labeled examples provided to STEAL to 20%, it still outperforms baseline transfer learning approaches that use 100% of labeled instances.

5.2. Prediction of cerebral stroke in hospital patients

Cerebral stroke is related to abnormal metabolic indicators for both hemorrhagic stroke and ischemic stroke [7]. The DALYs value (Disability Adjusted of Life Years - a measure of the overall severity of an illness,

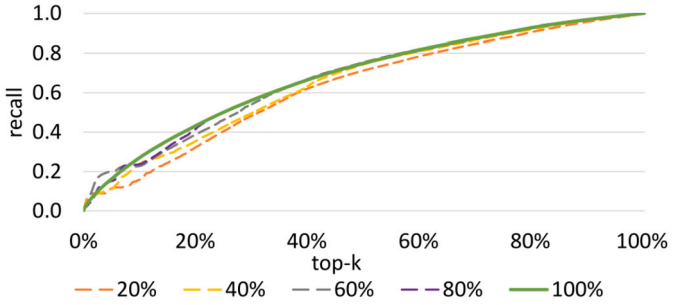

Fig. 8. Recall@k results obtained on the HOM dataset by STEAL, using different percentages of (positive) labeled instances.

Table 5
Gene network reconstruction results in terms of AUR@K, obtained by STEAL with different percentage of (positive) labeled instances considered.

	HOM
STEAL - 20%	0.633
STEAL - 40%	0.654
STEAL - 60%	0.673
STEAL - 80%	0.678
STEAL - 100%	0.680

expressed as the number of years lost to illness, disability, or premature death) caused by cerebral stroke ranks second only after ischemic heart disease [13,14]. Therefore, predicting the relapses of cerebral stroke is fundamental in medicine. In this context, we adopt STEAL to tackle the problem of small (and imbalanced) cerebral stroke data (target domain) by exploiting the knowledge on another medical domain, related to the survival to sepsis [9] (source domain), where the instances are described through a different set of features.

The considered stroke dataset [13,14] consists of 43,400 instances and 11 features. We pre-processed the dataset by removing attributes with more than 30% missing values, by converting categorical descriptive variables into numerical ones through one-hot encoding, and by removing 1,449 instances with missing values. After these steps, we obtained 41,288 instances representing patients who have not relapsed to stroke (the majority class), and 643 representing patients who had a recurrent stroke (the minority class).

The source domain dataset [9] is described through 4 features, and the task is to predict if the patient survived or not, due to sepsis. The dataset consists of 129,392 patients, where 11,735 died because of sepsis, while the remaining 117,657 survived.

The target variable of the two datasets was aligned so that the label that indicates the relapse to stroke in the cerebral stroke dataset corresponds to the label indicating people who did not survive in the sepsis dataset. Contrary to the application domain considered in Section 5.1, where the task from a machine learning viewpoint is link prediction for network reconstruction, in this case we are solving a binary classification task. Therefore, as evaluation measures, in this case we considered the accuracy and the averaged precision, recall, and F1-score, computed on the basis of a stratified 10-fold cross validation. Precision, recall and F1-score are calculated in two different ways: *i)* by means of an arithmetic average over classes and folds (called “macro” in our tables) and *ii)* by means of a weighted average over classes according to class cardinalities, subsequently averaged over folds (called “weighted” in our tables).

Also for these datasets, the competitor systems TJM, JGSA, BDA, and JDOT ran out of memory on our computational cluster. Therefore, we ran further experiments on a reduced version built by: *i)* imposing a balanced class distribution (obtained by downsampling the majority class); *ii)* further reducing the source domain dataset through a 10%

Table 6
Quantitative information of the considered datasets in the medical domain.

	FULL		REDUCED	
# instances	Stroke	Sepsis	Stroke	Sepsis
relapsed_stroke/died	643	11,735	643	1,173
single_stroke/survived	41,288	117,657	643	1,173
total	41,931	129,392	1,286	2,346

stratified random sampling. Moreover, since competitors require that source and target domains have the same number of features, we duplicated multiple times each descriptive feature of the source domain until we obtained a new descriptive space with the same dimensionality of that of the target domain, i.e., 13. Although this procedure may redundancies, it actually allows the competitors to work with a mismatching number of features between the source and the target domains, as well as to select multiple times the same feature from the source domain, as naturally done by STEAL.

An overview of the quantitative information of the considered full and reduced datasets is reported in Table 6.

Since the considered datasets are fully labeled, in this case, the steps adopted by STEAL to work in the PU learning settings were not necessary. Considering the relatively low amount of features, we also did not apply the PCA, namely, we directly worked on the original features. As regards the instance matching, we adopted the same parameters as in the previous experiments, namely, 10 samples each involving 10% random training examples from the source and from the target domains.

We optimized the competitors' number of subspace bases d through a grid search in the interval $[1, 13]$. The best value resulted to be $d = 6$ for JGSA, BDA, TJM and the default configuration for JDOT.

The results reported in Table 7 for the full dataset show that the proposed method STEAL achieved an improvement of 4.4%, 2.3% and 2.4% in terms of accuracy, macro F1-score and weighted F1-score, respectively w.r.t. the no transfer baseline.[6] There are two measures in which the baseline outperforms STEAL, namely, the macro recall and the weighted precision. However, in both cases, the advantage in terms of macro precision and weighted recall still leads to overall improvements in terms of macro F1-score and weighted F1-score, respectively.

Moreover, the results in Table 8 for the reduced dataset show that STEAL resulted the top-ranked when compared with the considered competitors. BDA resulted the second best method, while TJM the third in the ranking. Noteworthy, STEAL achieved an improvement of 13.9%, 13.9% and 13.8%, in terms of accuracy, macro F1-score and weighted F1-score, respectively, over the second best method BDA. It is noteworthy that, in this case, the advantage in terms of macro F1-score and weighted F1-score derives from advantages both in terms of precision and recall, that is, the improvement in one measure is not obtained at the price of penalizing the other.

5.3. Energy consumption prediction

The electrical infrastructure is continuously subject to evolutions, due to the increasing energy demand and the advent of new technologies such as photovoltaic/wind power plants, and charging stations. Moreover, the energy consumption is strongly connected to CO2 emissions, that led to push towards the continuous analysis of data generated by the electrical infrastructure also for ecological purposes. For this reason, several studies work in the direction of identifying predictive models to predict the customer energy consumption [1], to plan maintenance operations and to reduce the environmental impact, as well as for economic reasons, such as cost estimates, energy conservation and management.

In this context, we evaluated the possible benefits of adopting the proposed transfer learning method to learn more accurate models to predict the monthly energy consumption of 76 customers residing in a given area (target), exploiting the knowledge related to 77 customers residing in another area (source). The observations coming from the target and from the source domains are described by the same set of features (homogeneous setting). Specifically, each observation is described by 41 features, namely the customer id, the latitude and the longitude, the year and the month associated to the consumption value to predict, and 12 triplets (year, month and consumption) associated with the previous 12 monthly intervals. Note that, since in this application domain we work on homogeneous domains, the optimal feature alignment is simply the 1-to-1 correspondence between the same features in the two domains.

The temporal period of the dataset ranges from 2010 to 2019. Therefore, we adopted a 8-fold cross validation setting for time series, that assures that no future observations are used during the training of a predictive model for a given fold. The detailed quantitative information of the considered dataset is reported in Table 9. Also for this dataset, we have fully labeled instances and a relatively small feature set. Therefore, we considered the same setting as the experiments of Section 5.2. We optimized the competitors' subspace bases d through a grid search in the interval $[1, 40]$. Note that, for this dataset, BDA and TJM were not able to scale over 24 subspace bases. The best value resulted to be $d = 20$ for JGSA, $d = 21$ for BDA, $d = 24$ for TJM, and the default configuration for JDOT.

As evaluation measures, since in this case we are solving a regression task, we considered the Root Mean Squared Error (RMSE) and the R^2 measures.

From the results reported in Table 10, we can observe that STEAL strongly outperforms all the transfer learning competitors. Specifically, JGSA, TJM, BDA, and JDOT obtained worse results with respect to considering the target domain only (see the results obtained by T (no transfer)), with JGSA obtaining poor results even in comparison with a predictor that always returns the mean of the target attribute (see the negative R^2 value). On the other hand, STEAL was able to profitably exploit data coming from both domains. The results actually exhibit the occurrence of negative transfer phenomena on the predictions provided by competitors. This confirms that the adoption of a transfer learning approach does not necessarily imply better predictive performances with respect to the classical inductive learning counterpart, further emphasizing the benefits of the multi-stage approach implemented in STEAL.

It is noteworthy that the baseline T+S (optimal feature alignment) in this case achieved the best overall results. This is due to the fact that the dataset is purely homogeneous, namely, the features in the source and in the target datasets are exactly the same, and the true mapping for them is known a priori. Although this represents an ideal condition, almost never applicable in real-world scenarios, the comparison with this baseline allows us to assess the effectiveness of the feature alignment step performed by STEAL. In particular, STEAL obtained R^2 and RMSE results that are only 2.5% and 3.5%, respectively, worse than that achieved by T+S (optimal feature alignment). This means that STEAL, with a data-driven feature alignment, was able to get very close to the best possible achievable results, without exploiting any prior knowledge on the true mapping between the source and the target features. Since this prior knowledge is used by T+S (optimal feature alignment), the direct comparison between STEAL and T+S (optimal feature alignment) can be considered unfair, in favor of T+S (optimal feature alignment).

In Fig. 9, we show the trend of the measured R^2 over the different folds for the best performing methods. We remind that the folds represent different time spans and fold $j + 1$ refers to more training data than fold j. From the figure we can see that STEAL needs sufficient data to identify the correct feature alignment, but once the best feature

[6] Note that the average F1 is not obtained as direct computation from average precision and average recall according to the classical F1 formula. On the contrary, the average F1 is computed by averaging single F1-scores over classes (not weighted/weighted) and folds.

Table 7
Cerebral stroke prediction results obtained by STEAL and the no transfer baseline on the full dataset.

Method	Acc	Macro Prec	Macro Rec	Macro F1-score	Weighted Prec	Weighted Rec	Weighted F1-score
T (no transfer)	0.902	0.526	0.647	0.528	0.975	0.902	0.935
STEAL	0.942	0.530	0.597	0.540	0.974	0.942	0.957

Table 8
Cerebral stroke prediction results obtained by STEAL and its competitors on the reduced dataset.

Method	Acc	Macro Prec	Macro Rec	Macro F1-score	Weighted Prec	Weighted Rec	Weighted F1-score
JGSA	0.575	0.589	0.575	0.557	0.589	0.575	0.557
TJM	0.655	0.656	0.655	0.654	0.656	0.655	0.654
BDA	0.674	0.675	0.674	0.673	0.675	0.674	0.674
JDOT	0.551	0.551	0.551	0.550	0.551	0.551	0.550
STEAL	0.768	0.773	0.768	0.767	0.773	0.768	0.767

Table 9
Quantitative information of the considered datasets in the energy field.

Fold	Training period	Testing period	Source Training instances	Target Training instances	Testing instances
1	2010-2011	2012-2019	924	912	14,688
2	2010-2012	2013-2019	1,848	1824	12,852
3	2010-2013	2014-2019	2,772	2,736	11,016
4	2010-2014	2015-2019	3,696	3,648	9,180
5	2010-2015	2016-2019	4,620	4,560	7,344
6	2010-2016	2017-2019	5,544	5,472	5,508
7	2010-2017	2018-2019	6,468	6,384	3,672
8	2010-2018	2019	7,392	7,296	1,836

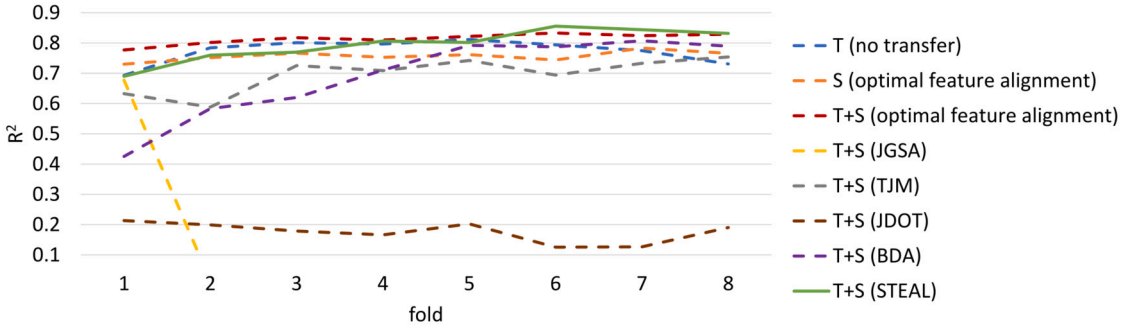

Fig. 9. Trend of the R^2 results obtained by the best performing methods, over the different folds of the dataset on the energy consumption prediction.

Table 10
Energy consumption prediction results, in terms of average RMSE and R^2 results obtained by STEAL, baseline approaches and transfer learning competitors. Note that the results obtained by the baseline T+S (optimal feature alignment) are achievable in *ideal* conditions, namely with purely homogeneous datasets, when the true mapping between source and target features is known a priori.

Method	RMSE	R^2
T (no transfer)	154.622	0.723
S (optimal feature alignment)	159.418	0.707
T+S (optimal feature alignment)	142.928	0.764
JGSA	447.778	-1.809
TJM	174.461	0.647
BDA	174.213	0.639
JDOT	275.516	0.125
STEAL	148.008	0.745

alignment is identified, the method is able to fully exploit the instance matching and improve over both competitors and baselines, including T+S (optimal feature alignment) for which the comparison is, anyway, unfair.

5.4. Scalability analysis

We recall that among the peculiarities of STEAL, there is the possibility of distributing the workload over multiple computational nodes. However, distributed algorithms may generally suffer from overheads and delays introduced by communication activities among computational nodes. For this reason, in this section we report a specific analysis on the STEAL scalability, performed on a dataset of 10 millions instances synthetically generated through a "sampling with replacement" procedure from the dataset for energy consumption prediction.

We measured the running times and computed the speedup factor to evaluate the ability of STEAL to exploit additional computational nodes. The results of this analysis are depicted in Fig. 10. In particular, we show the trend for the running time and for the speedup factor with increasing sizes of the dataset, and with a different number of computational nodes in the cluster (1, 2 or 3). The result shows that STEAL is able to take advantage of possible additional computational nodes. Specifically, even if with datasets of low-medium size ($< 200,000$ instances) the communication overhead is observable, when the dataset size increases it becomes negligible and the measured speedup factors quickly converge to ideal values, namely 2 and 3, respectively, with 2 and 3 computational nodes. This result proves the near-optimal scala-

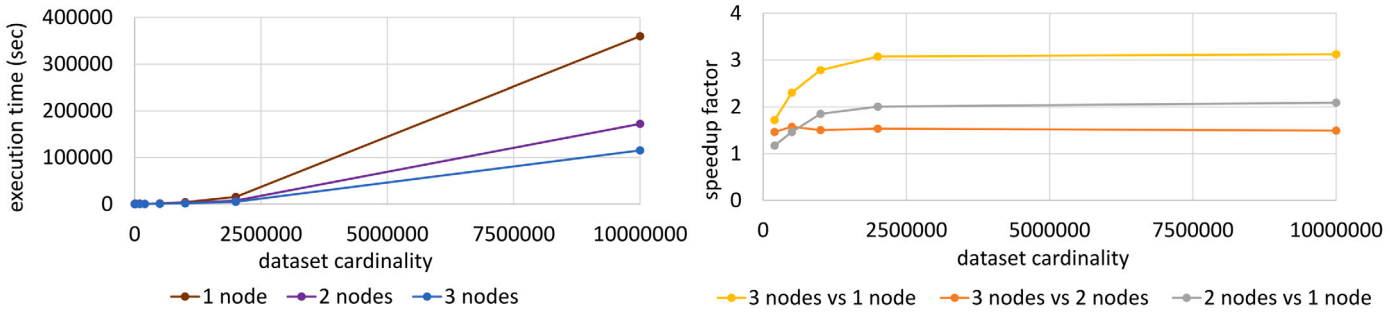

Fig. 10. STEAL running times and speedup factors, measured on a dataset of 10 millions instances.

bility of STEAL, that can in principle analyze arbitrary-sized datasets by simply attaching additional nodes to the computational cluster, without introducing observable communication delays.

6. Conclusions

In this paper, we proposed a novel transfer learning method, called STEAL, that is able to work with heterogeneous feature spaces, thanks to a feature alignment step based on the KL divergence. Contrary to existing heterogeneous transfer learning methods, STEAL can also work in the PU learning setting and can distribute the workload over multiple computational nodes, since it is completely designed using the MapReduce paradigm and implemented in Apache Spark.

Our experimental evaluation, conducted in three different application domains, demonstrated the superiority of STEAL over 4 state-of-the-art transfer learning competitors, and over the considered baseline approaches. Finally, our scalability analysis on a dataset for which it was not possible to run competitors, revealed the capability of STEAL to fully exploit multiple computational nodes without incurring in computational bottlenecks.

It is important to note that, while STEAL overcomes the limitations of existing transfer learning approaches, it can currently exploit one single source domain, and can solve classical tasks, like classification, regression and link prediction, in a single-target setting. Therefore, our future research activities will be devoted to such aspects. In particular, we will study the possibility to extend STEAL to make it able to capture and exploit the knowledge from multiple source domains. This can be achieved by *i)* extending the feature alignment step, through the identification of the best matching feature from all the source domains, for each feature of the target domain, and by *ii)* extending the instance matching step so that each instance of the target domain is augmented with the most promising instance, among those of all the available source domains. Moreover, we will evaluate the possibility to make STEAL able to solve more complex learning tasks, such as multi-target regression and multi-label classification. In these cases, additional challenges may be raised by the heterogeneity of the output spaces: the source domain(s) and the target domain may differ also in terms of number of target variables (in the case of multi-target regression tasks) and in terms of number of labels (in the case of multi-label classification tasks). This additional challenge may require the design of additional steps to perform an *alignment* also among the output spaces, in addition to that already performed by STEAL among the feature spaces.

CRediT authorship contribution statement

Paolo Mignone: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft. **Gianvito Pio:** Conceptualization, Formal analysis, Investigation, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing. **Michelangelo Ceci:** Conceptualization, Funding acquisition, Methodology, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The system STEAL and all the detailed results are publicly available at: https://figshare.com/articles/software/STEAL/19482533.

Acknowledgements

Dr. Paolo Mignone acknowledges the support of Apulia Region through the 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) in the context of “Research for Innovation - REFIN”. This work was also partially supported by the European Union - NextGenerationEU through the Italian Ministry of University and Research, Projects: *i)* PRIN 2022 “BAPHERD: Big Data Analytics Pipeline for the Identification of Heterogeneous Extracellular non-coding RNAs as Disease Biomarkers” grant n. 2022XABBMA CUP: H53D23003690006; *ii)* FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI, CUP: H97G22000210007; *iii)* CN3 RNA - “National Center for Gene Therapy and Drugs based on RNA Technology”, CUP: H93C22000430007.

Appendix A. Distributed computation in apache spark

Apache Spark is based on the MapReduce distributed computing model to process large-scale datasets efficiently across a cluster of machines. The core abstraction in Spark is the Resilient Distributed Dataset (RDD), which represents a distributed collection of elements partitioned across the nodes in the cluster. Spark provides a set of transformations and actions that operate on RDDs, allowing users to perform complex data processing tasks in a distributed manner. However, while the distribution is performed behind the scenes by Spark, the design of new algorithms cannot be based on common control structures, but needs to be specifically based on the MapReduce paradigm, properly feeding the operators with specific functions that also need to satisfy some strict properties.

In the following, we report a brief description of the main operators, that are adopted in all our algorithms described in Section 3.

The **map** operator returns a new RDD obtained by applying a given function (provided as input) to each element of the input RDD. This operator is fully distributed across the computational nodes of the cluster, since each node processes data locally on the RDD partitions it holds. The map operator is often used to transform/prepare RDD data so that they are more suitable for subsequent operations.

The **reduce** operator aggregates the elements of the RDD into a single result, based on a given input function. The input function has to

take two arguments of the type of elements of the RDD and has to return a value of the same type, specifying how two values of the RDD (that may be original values of the RDD, or partial aggregations of other values, obtained through the same function) have to be aggregated. Therefore, the function needs to be *associative* and *commutative*, because the computational nodes need to be able to compute it independently on the order of analysis of the elements they handle.

The **cartesian** operator returns the cartesian product of the elements taken from two input RDDs. Since it needs to compute every possible pairs of elements taken from the two RDDs, this operation is generally expensive, due to the huge amount of communications necessary among the computational nodes. However, there are several cases in which its usage cannot be avoided.

The **union**, **intersect** and **subtract** operators perform the well-known set operations among two RDDs. Therefore, the structure of the involved RDDs needs to be the same. The union operator is generally quite efficient, since it does not remove duplicates and it can be performed without exchanging data among the computational nodes: all the nodes compute a partition of the resulting RDD by merging the respective partitions they handle of the input RDDs. On the other hand, the operators intersect and subtract require exchanging data among the computational nodes.

Finally, it is worth mentioning some operators that work specifically on the so-called paired RDDs, which store *key-value* pairs. This is the case of the **join** operator, that returns an RDD containing all the pairs of elements coming from the input RDDs such that their keys correspond. To execute this operator, Spark distributes input RDDs among the nodes in the cluster on the basis of the keys, namely, key-matching pairs are sent to the same node. This may involve significant data movements across the cluster.

Another relevant operator that can be applied to paired-RDDs is **reduceByKey**. This operator acts as the reduce operator, i.e., it aggregates the values according to a given (associative and commutative) function, but for each group of values having the same key. It can be considered as a customizable "GroupBy" aggregation function like those usually available in the SQL language.

Several other operators are available in Spark, but providing a comprehensive overview is out of the scope of this paper. For further details, the reader can refer to the official documentation: https://spark.apache.org/docs/latest/rdd-programming-guide.html.

References

[1] K. Amasyali, N.M. El-Gohary, A review of data-driven building energy consumption prediction studies, Renewable & Sustainable Energy Reviews 81 (2018) 1192–1205, https://doi.org/10.1016/j.rser.2017.04.095.

[2] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms SODA '07, Society for Industrial and Applied Mathematics, USA, 2007, pp. 1027–1035.

[3] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, S. Vassilvitskii, Scalable k-means++, Proceedings of the VLDB Endowment 5 (2012) 622–633, https://doi.org/10.14778/2180912.2180915.

[4] S. Ban, H.S. Nam, E. Park, Detecting paralysis of stroke symptom in video: transfer learning with gated recurrent unit using public big data of facial images, in: 2022 IEEE International Conference on Big Data (Big Data), 2022, pp. 6587–6589.

[5] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32, https://doi.org/10.1023/A:1010933404324.

[6] Z. Cao, Y. Zhou, A. Yang, S. Peng, Deep transfer learning mechanism for fine-grained cross-domain sentiment classification, Connection Science (2021) 1–18, https://doi.org/10.1080/09540091.2021.1912711.

[7] J. Chen, Y. Chen, J. Li, J. Wang, Z. Lin, A.K. Nandi, Stroke risk prediction with hybrid deep transfer learning framework, IEEE Journal of Biomedical and Health Informatics 26 (2022) 411–422, https://doi.org/10.1109/JBHI.2021.3088750.

[8] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, K. Li, A parallel random forest algorithm for big data in a spark cloud computing environment, IEEE Transactions on Parallel and Distributed Systems 28 (2017) 919–933, https://doi.org/10.1109/TPDS.2016.2603511.

[9] D. Chicco, G. Jurman, Survival prediction of patients with sepsis from age, sex, and septic episode number alone, Scientific Reports 10 (2020), https://doi.org/10.1038/s41598-020-73558-3.

[10] N. Courty, R. Flamary, A. Habrard, A. Rakotomamonjy, Joint distribution optimal transportation for domain adaptation, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 3733–3742.

[11] O. Day, T.M. Khoshgoftaar, A survey on heterogeneous transfer learning, Journal of Big Data 4 (2017) 29, https://doi.org/10.1186/s40537-017-0089-0.

[12] L. Duan, D. Xu, I.W. Tsang, Learning with augmented features for heterogeneous domain adaptation, in: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012, 2012.

[13] T.L.W. Fan, C. Wu, Data for: a hybrid machine learning approach to cerebral stroke prediction based on imbalanced medical-datasets, https://data.mendeley.com/datasets/x8ygrw87jw/1, 2019, https://doi.org/10.17632/x8ygrw87jw.1.

[14] T.L.W. Fan, C. Wu, A hybrid machine learning approach to cerebral stroke prediction based on imbalanced medical dataset, Artificial Intelligence in Medicine 101 (2019) 101723, https://doi.org/10.1016/j.artmed.2019.101723.

[15] Y. Gao, Y. Ruan, C. Fang, S. Yin, Deep learning and transfer learning models of energy consumption forecasting for a building with poor information data, Energy and Buildings 223 (2020) 110156, https://doi.org/10.1016/j.enbuild.2020.110156.

[16] M. Harel, S. Mannor, Learning from multiple outlooks, in: Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011, 2011, pp. 401–408.

[17] A. Hooshmand, R. Sharma, Energy predictive models with limited data using transfer learning, in: Proceedings of the Tenth ACM International Conference on Future Energy Systems e-Energy '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 12–16.

[18] K. Kimura, T. Yoshida, Topic graph based transfer learning via generalized kl divergence based nmf, in: 2011 IEEE International Conference on Granular Computing, 2011, pp. 330–335.

[19] B. Kulis, K. Saenko, T. Darrell, What you saw is not what you get: domain adaptation using asymmetric kernel transforms, in: The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011, 2011, pp. 1785–1792.

[20] W. Li, L. Duan, D. Xu, I.W. Tsang, Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (2014) 1134–1148, https://doi.org/10.1109/TPAMI.2013.167.

[21] J. Liang, Z. Liu, J. Zhou, X. Jiang, C. Zhang, F. Wang, Model-protected multi-task learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2022) 1002–1019, https://doi.org/10.1109/tpami.2020.3015859.

[22] P.-J. Lin, X. Zhai, W. Li, T. Li, D. Cheng, C. Li, Y. Pan, L. Ji, A transferable deep learning prognosis model for predicting stroke patients' recovery in different rehabilitation trainings, IEEE Journal of Biomedical and Health Informatics 26 (2022) 6003–6011, https://doi.org/10.1109/JBHI.2022.3205436.

[23] S. Lin, J. Lv, Z. Yang, Q. Li, W.-S. Zheng, Heterogeneous graph driven unsupervised domain adaptation of person re-identification, Neurocomputing 471 (2022) 1–11, https://doi.org/10.1016/j.neucom.2021.11.009.

[24] 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.

[25] L. Lu, T. Zhou, Link prediction in complex networks: a survey, Physica A: Statistical Mechanics and its Applications 390 (2011), https://doi.org/10.1016/j.physa.2010.11.027.

[26] K.V. Mardia, J.T. Kent, J.M. Bibby, Multivariate Analysis, Academic Press London, New York, 1979.

[27] P. Mignone, G. Pio, Positive unlabeled link prediction via transfer learning for gene network reconstruction, in: M. Ceci, N. Japkowicz, J. Liu, G.A. Papadopoulos, Z.W. Raś (Eds.), Foundations of Intelligent Systems, Springer International Publishing, Cham, 2018, pp. 13–23.

[28] P. Mignone, G. Pio, M. Ceci, Distributed heterogeneous transfer learning for link prediction in the positive unlabeled setting, in: 2022 IEEE International Conference on Big Data (Big Data), 2022, pp. 5536–5541.

[29] 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, https://doi.org/10.1038/s41598-020-78033-7.

[30] P. Mignone, G. Pio, D. D'Elia, M. Ceci, Exploiting transfer learning for the reconstruction of the human gene regulatory network, Bioinformatics 36 (2019) 1553–1561, https://doi.org/10.1093/bioinformatics/btz781.

[31] J. Nam, S. Kim, Heterogeneous defect prediction, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015, 2015, pp. 508–519.

[32] S. Niu, Y. Liu, J. Wang, H. Song, A decade survey of transfer learning (2010-2020), IEEE Transactions on Artificial Intelligence 1 (2020) 151–166, https://doi.org/10.1109/TAI.2021.3054609.

[33] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (2010) 1345–1359, https://doi.org/10.1109/TKDE.2009.191.

[34] 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, https://doi.org/10.1093/bioinformatics/btab647.

[35] P. Prettenhofer, B. Stein, Cross-language text classification using structural correspondence learning, in: ACL 2010, Proceedings of the 48th Annual Meeting of

the Association for Computational Linguistics, July 11–16, 2010, Uppsala, Sweden, 2010, pp. 1118–1127.
[36] G. Qi, C.C. Aggarwal, T.S. Huang, Towards semantic knowledge propagation from text corpus to web images, in: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011, 2011, pp. 297–306.
[37] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65, https://doi.org/10.1016/0377-0427(87)90125-7.
[38] X. Shi, Q. Liu, W. Fan, P.S. Yu, R. Zhu, Transfer learning on heterogenous feature spaces via spectral transformation, in: 2010 IEEE International Conference on Data Mining, 2010, pp. 1049–1054.
[39] C. Wang, S. Mahadevan, Heterogeneous domain adaptation using manifold alignment, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16–22, 2011, 2011, pp. 1541–1546.
[40] J. Wang, Y. Chen, S. Hao, W. Feng, Z. Shen, Balanced distribution adaptation for transfer learning, in: ICDM 2017, 2017, pp. 1129–1134.
[41] Q. Wang, T.P. Breckon, Cross-domain structure preserving projection for heterogeneous domain adaptation, Pattern Recognition 123 (2022) 108362, https://doi.org/10.1016/j.patcog.2021.108362.
[42] 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 Bioinformatics 22-S (2021) 274, https://doi.org/10.1186/s12859-021-04190-9.
[43] Y. Wang, R. Xiao, J. Qi, C. Tao, Cross-sensor remote-sensing images scene understanding based on transfer learning between heterogeneous networks, IEEE Geoscience and Remote Sensing Letters 19 (2022) 1–5, https://doi.org/10.1109/LGRS.2021.3116601.
[44] K.R. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning, Journal of Big Data 3 (2016) 9, https://doi.org/10.1186/s40537-016-0043-6.
[45] J. Wishart, The generalised product moment distribution in samples from a normal multivariate population, Biometrika 20A (1928) 32–52, https://doi.org/10.1093/biomet/20a.1-2.32.
[46] H. Wu, Q. Wu, M.K. Ng, Knowledge preserving and distribution alignment for heterogeneous domain adaptation, ACM Transactions on Information Systems 40 (2022) 1–29, https://doi.org/10.1145/3469856.
[47] B. Zhang, W. Zuo, Learning from positive and unlabeled examples: a survey, in: International Symposium on Information Processing, ISIP 2008 / International Pacific Workshop on Web Mining, and Web-Based Application, WMWA 2008, Moscow, Russia, 23-25 May 2008, 2008, pp. 650–654.
[48] H. Zhang, J.S. Polson, K. Nael, N. Salamon, B. Yoo, S. El-Saden, F. Scalzo, W. Speier, C.W. Arnold, Intra-domain task-adaptive transfer learning to determine acute ischemic stroke onset time, Computerized Medical Imaging and Graphics 90 (2021) 101926, https://doi.org/10.1016/j.compmedimag.2021.101926.
[49] J. Zhang, W. Li, P. Ogunbona, Joint geometrical and statistical alignment for visual domain adaptation, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January, 2017, pp. 5150–5158.
[50] J.T. Zhou, S.J. Pan, I.W. Tsang, Y. Yan, Hybrid heterogeneous transfer learning through deep learning, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada, 2014, pp. 2213–2220.
[51] Y. Zhu, Y. Chen, Z. Lu, S.J. Pan, G. Xue, Y. Yu, Q. Yang, Heterogeneous transfer learning for image classification, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011, 2011.
[52] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proceedings of the IEEE 109 (2021) 43–76, https://doi.org/10.1109/JPROC.2020.3004555.