Heterogeneous Transfer Learning from a Partial Information Decomposition perspective

Gabriele Gianini[1][0000−0001−5186−0199], Annalisa Barsotti[1][0000−0002−8279−729X], Corrado Mio[2][0000−0002−1087−4866], and Jianyi Lin[3][0000−0002−3299−448X]

[1] Università degli Studi di Milano, via Celoria 18, 20133, Milano, MI - Italy
{gabriele.gianini,annalisa.barsotti}@unimi.it
[2] EBTIC, Khalifa University of Science and Technology, Abu Dhabi, UAE
corrado.mio@ku.ac.ae
[3] Università Cattolica del Sacro Cuore, Milano, Italy
jianyi.lin@unicatt.it

Abstract. Transfer Learning (TL) encompasses a number of Machine Learning Techniques that take a pre-trained model aimed at solving a task in a Source Domain and try to reuse it to improve the performance of a related task in a Target Domain An important issue in TL is that the effectiveness of those techniques is strongly dataset-dependent. In this work, we investigate the possible structural causes of the varying performance of Heterogeneous Transfer Learning (HTL) across domains characterized by different, but overlapping feature sets (this naturally determine a partition of the features into Source Domain specific subset, Target Domain specific subset, and shared subset). To this purpose, we use the Partial Information Decomposition (PID) framework, which breaks down the multivariate information that input variables hold about an output variable into three kinds of components: Unique, Synergistic, and Redundant. We consider that each domain can hold the PID components in implicit form: this restricts the information directly accessible to each domain. Based on the relative PID structure of the above mentioned feature subsets, the framework is able to tell, in principle: 1) which kind of information components are lost in passing from one domain to the other, 2) which kind of information components are at least implicitly available to a domain, and 3) what kind information components could be recovered through the bridge of the shared features. We show an example of a bridging scenario based on synthetic data.

Keywords: Heterogeneous Transfer Learning; Partial Information Decomposition; Transferable Information Components.

1 Introduction

The expression Transfer Learning (TL) covers a wide collection of Machine Learning (ML) methods aimed at reusing knowledge across domains. In machine learning, most often it refers to the reuse of a pre-trained model aimed at solving a task in a Source Domain to improve the performance of a related task in a Target Domain [7].

For instance, one can learn to classify instances from a low cardinality dataset of the target domain by leveraging the knowledge gained from the instances from a large dataset in a similar domain: this first case is an example of homogeneous Transfer Learning. Furthermore, new and potentially useful knowledge can be imported into a target domain from a source domain characterized by different variables when the two sets of variables do not completely overlap: this case is an example of Heterogeneous Transfer Learning (HTL) [1].

The landscape of specific problems and approaches in TL is varied. A significant number of techniques have been proposed for HTL, including Heterogeneous Feature Augmentation [2] and Heterogeneous Max-Margin Classifier Adaptation [4]), and Sparse Heterogeneous Feature Representation (SHFR) for multiclass Heterogeneous Domain Adaptation (HDA) [6], however, it has been reported that their effectiveness is strongly dataset dependent. In this work, we are interested in understanding the prerequisites necessary for a successful transfer and the related challenges in the case of HTL with partial feature set superposition.

To this end, we propose an approach based on the Partial Information Decomposition (PID) framework, proposed by Williams and Beer in 2010 [5]. The framework decomposes the multivariate information that a set of input variables holds about an output variable into Unique, Synergistic, and Redundant components. In addition, we consider the degree of accessibility to information components (such degree can depend upon representational issues and noise).

We point to a naming-related issue in discussing TL and PID together in a ML context: the terms "source" and "target" can take two different meanings. In the TL context one speaks of *Source Domain* and *Target Domain*: they refer to the spaces *from* which and *to* which, respectively, information has to be moved (the information is typically about joint distributions of predictor and predicted variables). In the PID context one speaks of *information source* to refer to a predictor variable or a set of predictor variables. In ML the term *target* refers to predicted variables (e.g. class labels). To avoid confusion, along this work, we always add a specification to the terms "source" and "target" and use

capitalization to mark the distinction. For example: we mention Source Domain variables and Target Domain variables; the Source Domain variables include input features features and target/output variables; furthermore, within each Domain it can be useful to distinguish among a number of information sources, i.e. subsets of input variables providing information about a target variable.

In HTL with partial feature set superposition it is useful to distinguish the following three subsets of features: Source-Domain-specific features (we identify this set of features, available only to the Source Domain, with information source α), target-domain-specific features (we identify this set of features, available only to the Target Domain, with information source γ) and shared features (we identify this non-empty set with information source β). In other words the Source Domain holds the information sources α and β, whereas the Target Domain holds the information sources β and γ.

The functional dependence relationships among these three information sources and with the target/output variable determine the potential usefulness or irrelevance of the transfer schemes. Based on the correlation structure of the three feature sets our framework is able to tell in principle:

- what PID components could be readily *available* in the Source Domain to transfer to the Target Domain and what could be *accessible* in practice
- which part of that information is it actually needed in the Target Domain to improving the classification/regression task performance
- and, for the information that would be needed, but is not accessible, whether a surrogate/proxy could be transferred exploiting the shared features.

In this way, one can analyze the different transfer learning schemes and tell if they have the potential to exploit the synergistic information when present. We study in particular the most relevant transfer scheme, based on *bridging* through the shared features and show an example of this approach using synthetic data.

Notice that for the sake of illustration we limit our examples to the case of a single-bit output variable (which could represent a dichotomous label classifier): the two domains (Source Domain and Target Domain) can hold shares of information about this output. Similarly, for the sake of clarity in the theoretical discussion of the framework, we consider only feature sets consisting of a single one-bit variable (α, β and γ each consisting of one bit): the Source Domain holds the two bits α and β, and the target domain holds the two bits β and γ. This structure can, in principle, be extrapolated to any number of bits in the input domain variables and any number of bits in the output variable.

The paper is structured as follows. After recalling in Section 2, definitions and concepts related to Transfer Learning in general and to Heterogeneous Transfer Learning in particular, we outline, in Section 3 the theory of Partial Information Decomposition. Then in Section 4 we present our framework for the application of PID to HTL. Finally, in Section 5 we demonstrate the result of some experimentation using synthetic data. A summary of the findings and an outlook for future work conclude the paper in Section 6.

2 Transfer Learning

Sometimes, the knowledge acquired from solving a task on a dataset can be reused to solve the same or a related task on a different dataset: this is called Transfer Learning.

Domain. A Domain $\mathcal{D} = \{\mathcal{X}, P\}$ is defined by a space and a probability distribution on its samples. The space is denoted by $\mathcal{X}$ (for example, it could be $\mathbb{R}^n$) and called *feature space*. A Source and a Target Domain are different when they have different feature spaces ($\mathcal{X}^S \neq \mathcal{X}^T$) or different probability distributions ($P^S(X) \neq P^T(X)$).

Task. In supervised learning, a task $\mathcal{T} = \{\mathcal{Y}, f\}$ consists of a label space $\mathcal{Y}$ (e.g. $\{0, 1\}$) and a predictive function f, which is expected to be learned from the training data. Two tasks are different if they have two different label spaces $\mathcal{Y}^S \neq \mathcal{Y}^T$ or different or different conditional probability distributions $P^S(y|X) \neq P^T(y|X)$.

Transfer Learning. The Transfer Learning problem consists of finding approaches to exploit the knowledge *implemented* in a Source Domain and a task $(\mathcal{D}^{\mathcal{S}}, \mathcal{T}^{\mathcal{S}})$ to improve the performance of the learned decision function f^T for a task $\mathcal{T}^T$ in a target domain and task $(\mathcal{D}^{\mathcal{T}}, \mathcal{T}^{\mathcal{T}})$.

Homogeneous vs. Heterogeneous Transfer Learning. There are several categorizations of transfer learning problems. One of them is based on the consistency between Source Domain and Target Domain feature spaces and label spaces. If $\mathcal{X}^S = \mathcal{X}^T$ and $\mathcal{Y}^S = \mathcal{Y}^T$ – that is, if the feature spaces in the Source and Target Domains use the same attributes and labels – the scenario is called *homogeneous transfer learning*. Otherwise, if $\mathcal{X}^S \neq \mathcal{X}^T$ and/or $\mathcal{Y}^S \neq \mathcal{Y}^T$, the scenario is called *heterogeneous transfer learning*.

Setup of the present work. In this work we assume that $\mathcal{Y}^S = \mathcal{Y}^T$ but $\mathcal{X}^S \neq \mathcal{X}^T$ and that the two different set of features partially overlap: $\mathcal{X}_\cap \equiv (\mathcal{X}^S \cap \mathcal{X}^T) \neq \emptyset$.

3 Partial Information Decomposition (PID)

The goal of Partial Information Decomposition (PID), first proposed by Williams and Beer in 2010 [5], is to break down the multivariate mutual information that a set of source variables provides about a target output variable, into its simplest components – non-negative terms called sometimes *information atoms*. For example, in a two-source setting, some information about the output variable might only be found in a certain source, while other information might be shared by the two sources, and still other information might be made synergistically accessible only by combining both sources. This framework can be used in our setting since also in classification and regression it is crucial to understand how information about the target variable is distributed among the different sources, which are given by the feature variables.

3.1 PID for two information sources

Assume we have two variables, called X_1 and X_2, and assume that to each item described by combination of their values (x_1, x_2) a function f (that we want to learn) assigns a Boolean label $y \in Y = \{0, 1\}$. We can break down the information about the target into redundant, unique and synergistic

- *Redundant information* ($Rdn_{12} \equiv Rdn(Y : X_1, X_2)$: the same information is present both in source 1 and source 2, each source holds one bit of information about the label),
- *Unique information* ($Unq_1 \equiv Unq(Y : X_1)$: information present in 1 only (not available to 2), i.e. 1 holds a bit of information; $Unq_2 \equiv Unq(Y : X_2)$: information present in 2 only (not available to 1), i.e. 2 holds a bit of information),
- *Synergistic information* ($Syn_{12} \equiv Syn(Y : X_1, X_2)$: the information can only be provided by source 1 and source 2 jointly, they cannot provide that information individually, i.e. individually they hold zero bits of information, jointly they hold one). A prototypical example of this is the XOR function.

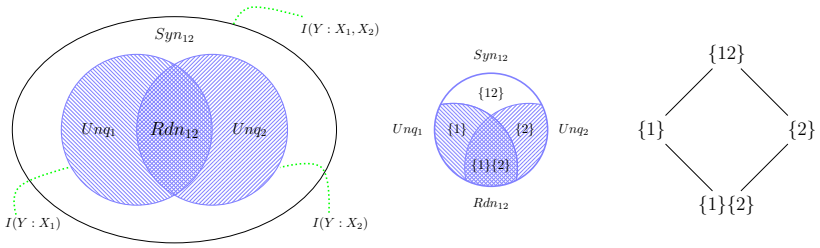

Fig. 1: Partial Information diagrams for two information sources 1 and 2. Left: the Venn diagram. Center: its compact representation. Right: the corresponding lattice of coalitions. In the center and right diagrams {1} and {2} are singletons and represent the unique information held by the sources about the target variable; whereas {1}{2} (the shorthand for {{1}{2}}) represents the two singleton sources that can provide the same information, i.e. redundant information; finally, {12} represents the *coalition* of both information sources.

In other words, the information that source 1 alone holds about the target is composed by the unique part and the redundant part, similarly for source 2:

$$I(Y : X_1) = Rdn_{12} + Unq_1 \qquad I(Y : X_2) = Rdn_{12} + Unq_2 \tag{1}$$

The information that the two sources hold together contains all the four terms

$$I(Y : X_1, X_2) = Rdn_{12} + Unq_1 + Unq_2 + Syn_{12} \tag{2}$$

Figure 1 illustrates this decomposition. The above equations form an underdetermined system of three equations with four unknowns: the PI decomposition alone does not provide a method to work out the PI terms. To the latter purpose one needs to specify one of the four variables in the system, e.g. by postulating a formula to compute either Rdn or Syn. A number of proposals for defining the PID terms have been advanced (see Kolchinsky [3] for a list of pointers). This problem, however, is out of the scope of the present work: we use the PID framework to assess the transfer learning schemas and do not take a quantitative approach, thus we do not need to measure the precise size of PID components.

3.2 The PID for three information sources

Whereas the PID lattice for two information sources consists of 4 nodes, the PID lattice for three sources consists of 18 nodes (see Figure 2). The singletons nodes {1}, {2}, and {3} represent the unique information held by the sources about

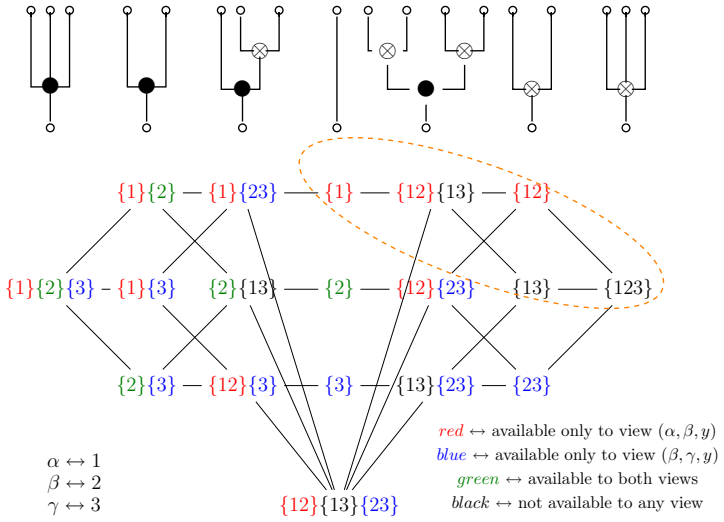

Fig. 2: The 18 node three-variate redundancy lattice. Above the lattice are represented for illustrative purposes, the Boolean circuit prototypes for the components for the setting where each input source provides a single bit and the output consists of a single bit: the solid circle • represents redundancy, and the symbol ⊗ represents synergy. The sources 1, 2, and 3, in the discussion of Section 4 correspond, respectively, to the variables α, β and γ, whereas in the example of Section 5 they correspond, respectively, to the variables κ^α, κ^β and κ^γ. To the benefit of the discussion in Section 4, the colors denote the availability of the Partial Information components to the different views (Source Domain view, Target Domain view, and all-encompassing view). The dashed ellipse indicates the Partial Information components not available to the Target Domain view.

the target; the nodes {12}, {13}, {23}, and {123} that represent synergistic information; the remainder are nodes that represent redundant information shared by two singletons, such as with {1}{2} – by a singleton and a coalition, such as in {1}{23} – by two coalitions, such as in {12}{13} – or by three coalitions, such as in {12}{13}{23}.

One can look at the three information source diagram as a breakdown of the two information source PID components according to their relationship with an extra source. Notice that in the two-source lattice and in the three-source lattice the same symbol denotes two different kinds of information. For example, in the two-source diagram the term {12} denotes the *whole* synergistic information of source 1 and 2, whereas in the three-source diagram the term {12} denotes the part of synergistic information of 1 and 2 that has no redundancies, i.e.

that cannot be found elsewhere. To avoid confusion from now on we denote the two-source diagram component by a prime, for example

$$\{12\}' = \{12\} + \{12\}\{3\} + \{12\}\{13\} + \{12\}\{23\} + \{12\}\{13\}\{23\} \tag{3}$$

4 Heterogeneous Transfer Learning meets PID

In HTL with overlapping feature sets we can regard the sets

$$\alpha = (\mathcal{X}^S \setminus \mathcal{X}^T) \qquad \beta = (\mathcal{X}^S \cap \mathcal{X}^T) \qquad \gamma = (\mathcal{X}^T \setminus \mathcal{X}^S)$$

as three sources of information about the target variable Y:

- the Source Domain view (X^S, Y) includes the information sources α and β,
- the Target Domain view (X^T, Y) includes the information sources β and γ.

We denote the Source Domain view by (α, β, y), the Target Domain view by (β, γ, y), and for the sake of comparison the all-encompassing view by $(\alpha, \beta, \gamma, y)$.

If we map the information sources α, β and γ respectively to 1, 2, and 3, i.e.

$$1 \leftrightarrow \alpha \qquad 2 \leftrightarrow \beta \qquad 3 \leftrightarrow \gamma \tag{4}$$

we have that

- the Source Domain view holds the components $\{1\}'$, $\{2\}'$, $(\{1\}\{2\})'$, $\{12\}'$;
- the Target Domain view holds the components $\{2\}'$, $\{3\}'$, $(\{2\}\{3\})'$, $\{23\}'$;
- the all-encompassing view holds all the 18 components shown in Figure 2.

Notice the following key points

- the sole fact that the information is held by a source, does not grant that it can be readily used for the prediction: the information has obviously to be learned from the data before being used for prediction, and the process of learning can fail for several reasons: this happens for instance when the model chosen for learning is not suitable for the task.[4]
- *when an information component is not held by a two-source view – or when is held by the view, but is hard to learn – it still can, in principle, be at least partially recovered by that view from the redundant components* (redundant with the missing information): this can be considered a sort of *bridging*.

[4] For example, if a full dataset defining the XOR function in a Cartesian plane is available to an information source, say β, the attempt to learn the corresponding classifier using a straight line as class separation boundary is bound to fail.

In this capability lies the specificity of Heterogeneous Transfer Learning: the process of HTL can involve at the same time domain adaptation (as in Homogeneous TL) over the shared features, and bridging of information through exploitation of PID redundancies. A bridging example is given in the next Section.[5]

For example, by construction the Target Domain view (β, γ, y) does not directly hold the synergistic information of the two sources 1 and 2 (i.e. $\{12\}'$); nonetheless $\{12\}'$ can be broken into parts (see equation (3)), some of which are redundant with components actually held by the Target Domain view:

- the part $\{12\}\{3\}$ can in principle be recovered from $\{3\}'$
- the part $\{12\}\{23\}$ can in principle be recovered from $\{23\}'$
- the part $\{12\}\{13\}\{23\}$ can also in principle be recovered from $\{23\}'$
- the part $\{12\}\{13\}$ is not accessible to the Target view
- the part $\{12\}$ is equally not accessible to the Target view

The first three components can act as *bridges* between views.

Overall, looking at Figure 2 one can see that with respect to bridging, from the stand point of the Target Domain view, there are only three kinds of component: those potentially available in full (i.e. $\{2\}'$, $\{3\}'$, and $\{23\}'$, those not available and not recoverable, due to lack of redundancy (within the ellipse), and those not available but potentially recoverable, thanks to redundancy.

Recoverable information. The components that by construction are not immediately available to the Target Domain view are

$\{1\}'$ that can be partially recovered from $\{1\}\{2\}$, $\{1\}\{23\}$, $\{1\}\{2\}\{3\}$, $\{1\}\{3\}$;
$\{12\}'$ that can be partially recovered from $\{12\}\{23\}$, $\{12\}\{3\}$, $\{12\}\{13\}\{23\}$;
$\{13\}'$ that can be partially recovered from $\{2\}\{13\}$, $\{13\}\{23\}$, $\{12\}\{13\}\{23\}$.

On the contrary, $\{123\}$, i.e. the synergistic information in the coalition of the three sources is completely lost from the point of view of the Target Domain.

5 An illustrative example of bridging

We provide an illustrative example of the bridging discussed above.

Notice that that, in the example, the mapping between the sources α, β, γ, and the elements of Figure 2 (where each source was represented as a single

[5] In that example we will bridge from α to β information that is synergistic to γ for the prediction of the target/output variable, so that the Target Domain view can exploit the synergy between the available γ and the non-available α.

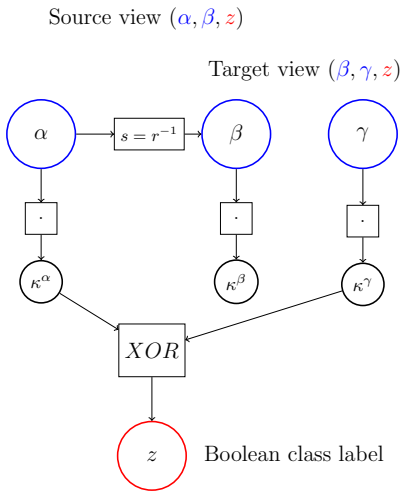

Fig. 3: Schematic view of the feature dependencies: α and γ are independent, so are their Boolean labels (unavailable to the observers), which are combined by a XOR to yield the labels z. Thus α and β can synergistically predict z. However, neither the source view nor the target view can observe the two feature sets at the same time. The Source Domain observer can only see α, β and the label z, while the Target Domain observer can only see β,γ, and the label z.

bit) is not direct: to allow a pictorial representation in the Cartesian plane, each source consists of two variables, i.e. two coordinates, each defined by several bits; however each point in the Cartesian plane (each coordinate pair) is associated with a Boolean class κ^{σ} with $\sigma \in \{\alpha, \beta, \gamma\}$: thus the relationship with the elements of the diagram in Figure 2 is defined by

$$1 \leftrightarrow \kappa^{\alpha} \qquad 2 \leftrightarrow \kappa^{\beta} \qquad 3 \leftrightarrow \kappa^{\gamma} \tag{5}$$

Also notice that in the following for notation convenience the target/output variable about which the domains hold some information is denoted by z. The reason for the choice will be apparent from the data description.

5.1 A Source-view and Target-view model

We generate the data according to the following stylized scenario graphically illustrated in Figure 3:

– the Cartesian plain points of α and γ are generated independently from one another (they also form two tables with an equal number of rows);

– generation of the label z:
  - the Boolean labels κ^α, κ^β, and κ^γ are assigned to the points of the α, β and γ datasets respectively based on their geometrical position (further details later);
  - the labels κ^α, κ^β, and κ^γ are not included in the source or target views, but are combined by an XOR function to yield the Boolean label $z \in \mathcal{Y}$, i.e.

$$z = XOR(\,\kappa^\alpha,\ \kappa^\gamma\,)$$

    (κ^β is not used): this makes α and γ *synergetic* for the prediction of z: the source-specific features alone cannot predict the target better than a static dummy classifier issuing always the majority class label, the same holds for the target-specific features;[6]
– then each point of β is generated from a point of α through a deterministic transform – consisting in a non-linear deformation – and some added noise;
– finally we prepared a single, all-encompassing dataset, from which we later extracted the Source view and the Target view, as illustrated in Figure 4.
  - the Source Domain dataset (α, β, z) is built by the columns of the variable sets α and β, plus the target z, and using the first half of the rows;
  - similarly the Target Domain dataset (β, γ, z) is built by the columns of the variable sets β and γ, and the target z and with the reminder rows.

The feature sets α, β and γ are defined by two variables each: $\alpha = \{A, B\}$, $\beta = \{C, D\}$, $\gamma = \{E, F\}$, and the class label variable is the Boolean variable Z.

Generation of the dataset with the desired features. We generated the features as follows (hereafter n_0 and n_1 indicate, respectively, the size of the class 0 and class 1 portions of the dataset, and $n = n_0 + n_1$ while $\mathcal{U}(a, b)$ denotes a random uniform over the interval $[a, b]$.

$$r_i \sim \mathcal{U}(0, a), \quad i = 1, \ldots, n_0 \qquad r_i \sim \mathcal{U}(b, c), \quad i = n_S + 1, \ldots, n_S + n_T$$
$$\kappa_i^\alpha = 0, \quad i = 1, \ldots, n_0 \qquad \kappa_i^\alpha = 1, \quad i = n_0 + 1, \ldots, n_0 + n_1$$

[6] It is apparent that in this scenario α and γ can, together, predict the target variable z. However, neither the Source Domain nor the Target Domain encompass both feature sets. However, since α can in principle be partially recovered from β, there is room for improving the target domain prediction, with respect to those of a model learned solely on the basis of view (β, γ, z).

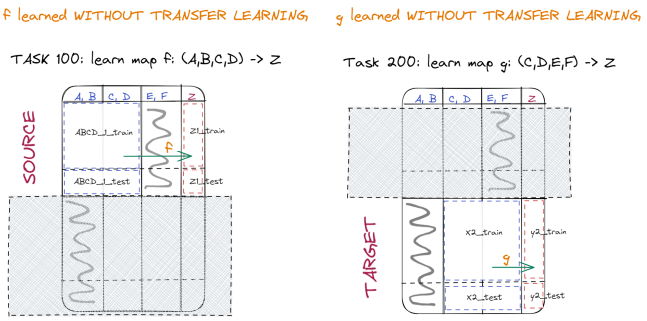

Fig. 4: Schematic view of the construction of the Source Domain side dataset (left) and Target Domain side dataset (right): the wiggles in the upper and the lower part of the table indicate columns that are not available to the Source side or to the Target side respectively. From the Target-side dataset one can learn a classifier g, that does not have optimal performance, since it misses some synergistic features, available only at the Source side.

and $\theta_i = \mathcal{U}(0, 2\pi)$, for $i = 1, \ldots, n$. With $a = 4^2, b = 6^2, c = 8^2$ this generates the radial coordinates of random points within, respectively, an inner disc (class 0) and an outer annulus (class 1). Then we put them in polar coordinates and add some Gaussian noise

$$\begin{aligned} x_i &= r_i \cos(\theta_i) + \varepsilon_x \qquad \text{where } \varepsilon_x \sim \mathcal{N}(0, (x_{max} - x_{min})/d) \\ y_i &= r_i \sin(\theta_i) + \varepsilon_y \qquad \text{where } \varepsilon_y \sim \mathcal{N}(0, (y_{max} - y_{min})/d) \end{aligned}$$

with $d = 15$. Then we set two other Cartesian coordinate variables $u_i \sim \mathcal{U}(-\ell, +\ell)$, and $v_i \sim \mathcal{U}(-\ell, +\ell)$ for $i = 1, \ldots, n$ (with $\ell = 10$), and set the class of the corresponding points to the Boolean value $\kappa_i^\gamma = (1 + \text{sgn}(u_i)\,\text{sgn}(v_i))/2$.

The class for the point $(r_i, \theta_i, x_i, y_i, u_i, v_i)$ was set to $z = XOR(\,\kappa^\alpha,\ \kappa^\gamma\,)$

Finally, we set $A = r$, $B = \theta$, $C = x$, $D = y$, $E = u$, $F = v$ and $Z = z$. The result is graphically illustrated in Figure 5.

5.2 Outcomes

Notation. Let $a_c(w)$ indicate the accuracy of a classifier c based on a given view w: for example, $a_{SVM}((\alpha, \beta))$ denotes the accuracy of an SVM classifier trained on view (α, β); we are going to use the Support Vector Machine classifiers $(c = SVM)$, and Random Forest $(c = RF)$.

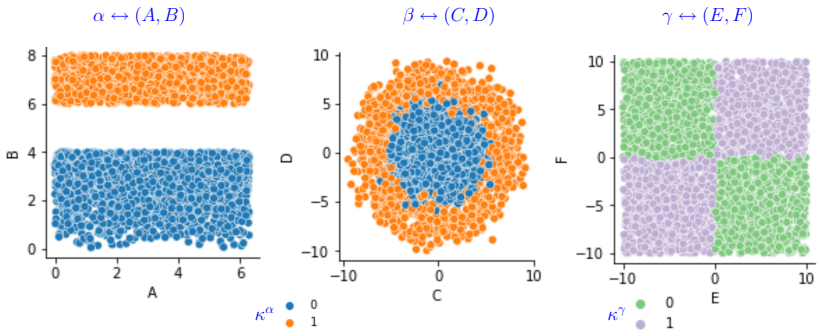

Fig. 5: Scatter-plots of the synthetic dataset. The colors correspond to the Boolean values of the class κ^{α} and κ^{γ} (see legend) of the 2D points; the class label for the 6D points are obtained by $Z = XOR(\kappa^{\alpha}, \kappa^{\gamma})$.

In the following we are going to demonstrate that using either $c = SVM$ or $c = RF$, without transfer learning we have

$$a_c(\alpha, \gamma) > a_c(\alpha, \beta, \gamma) > a_c(\beta, \gamma) > a_c(\alpha, \beta) \tag{6}$$

This happens because (α, γ) holds the synergistic information without the noise that β contributes to (α, β, γ); furthermore (β, γ) contains the synergistic information but only in noisy form, while (α, β), does not hold that information and cannot predict the output variable.

The performance of the prediction using the Target Domain data set (β, γ, z), i.e., with columns (C, D, E, F), is far from optimal. The SVM classifier obtained optimizing the model parameters (through random search) yields a test accuracy of 0.770 and a test precision of 0.726, against a dummy classifier accuracy and precision of 0.580 (reflecting the proportion of the classes: the dummy classifier always bets on the same class). Thus, the improvement over the Dummy classifier is of only 0.190 for the accuracy and 0.164 for the precision. From now on we quote the results in terms of improvement with respect to the dummy classifier, which in this dataset has by construction $accuracy = precision = 0.580$ and $recall = 1$.

However, there is room for increasing the performance. For the sake of comparison we trained an SVM (again optimizing the hyperparameters by random search) on the whole feature set $(\alpha, \beta, \gamma) \leftrightarrow (A, B, C, D, E, F)$ and obtained an improvement over the dummy dataset of 0.453 in accuracy and of 0.422 in precision. The performance gets even better if we remove the redundant (in principle)

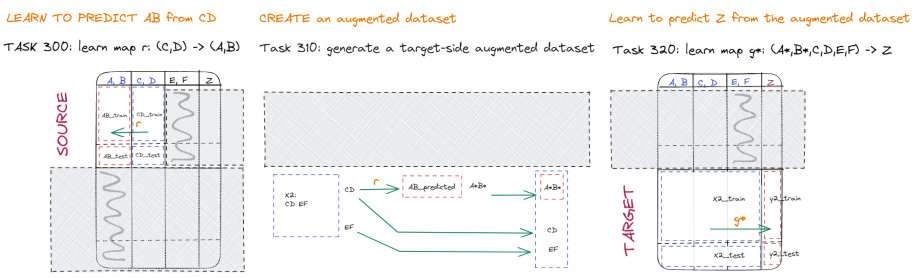

Fig. 6: Schematic view of the transfer learning approach. The shaded areas refer to rows or parts of the columns that are not involved in the specific phase of the procedure.

but (in practice) skewed and noisy columns $\beta \leftrightarrow (CD)$: In fact, the feature set $(\alpha, \gamma) \leftrightarrow (A, B, E, F)$ fares an increase of 0.463 and 0.439 respectively in accuracy and precision w.r.t. the dummy classifier.

The Transfer Learning approach. To transfer synergetic information from the source data set to the Target Domain side, we use the procedure shown in Figure 6.

- First, using the Source Domain dataset, we learn an SVM regressor r mapping (C, D) onto (A, B) (i.e., we learn the *bridge*).
- Then, using the Target Domain dataset we run the regressor r and map the features (C, D) onto estimates of (A, B), which we denote by (A^*, B^*).
- At this point we augment the target dataset with the new predicted features, obtaining the dataset with features (A^*, B^*, C, D, E, F)
- and finally use the augmented dataset to train the new SVM classifier g^*

The SVM regressor r (obtained by optimizing the hyper-parameters by random search), is set to play the role of an effective bridge between the domains: evaluating the effectiveness of r in predicting (A, B) based on (C, D), one finds that the regressor is able to explain a considerable part of the variance. If we predict the feature $B*$ and the feature A^* separately, the proportion of explained variance (defined as $1 -$ (variance of residuals/(total variance)) for the feature B^* is 0.81, while for the feature A^* it is 0.60.

Performance outcomes with Transfer Learning. The outcomes of the process in the synthetic data set support the possibility of transferring some information from source-only features to the target side classifier. In fact, the SVM

Table 1: Results of SVM and RF classifiers on different feature sets. The Extra Accuracy and the Extra Precision, with respect to the ones of the dummy classifier (always issuing the same label) are reported. The comparison between the performances of the feature sets CDEF and A*B*CDEF, highlighted in boldface, shows that the transfer learning process has been effective.

	Feature Sets	Features	Dummy Classifier Accuracy = = Precision	SVM Classifier Extra Accuracy	SVM Classifier Extra Precision	RF Classifier Extra Accuracy	RF Classifier Extra Precision
ALL rows	(α, β, γ)	ABCDEF	0.498	0.453	0.422	0.495	0.498
	(α, γ)	AB EF		0.463	0.439	0.498	0.498
Source rows	(α, β)	ABCD	0.544	-0.055	0.243	-0.050	-0.014
Target rows	(β, γ)	**CDEF**	0.580	**0.190**	**0.146**	**0.275**	**0.253**
		B*CDEF		0.250	0.254	0.320	0.337
	$(\alpha^*, \beta, \gamma)$	**A*B*CDEF**		**0.410**	**0.408**	**0.420**	**0.420**

classifier g^* has an improvement over the dummy classifier of 0.410 for accuracy and of 0.408 for precision. A more comprehensive account is reported in Table 1. One can observe that in terms of accuracy a_c and in terms of precision p_c, for both the SVM classifier and the RF classifier we have

$$a_c(\alpha^*, \beta, \gamma) > a_c(\beta, \gamma)$$

$$p_c(\alpha^*, \beta, \gamma) > p_c(\beta, \gamma)$$

In other words the transfer learning process, when using SVMs or RFs is effective.

In addition, the analysis of feature importance confirms the effectiveness of this TL scheme on this data set. Using the Shapley Value of the features and their permutation importance (results not reported here), we find that the reconstructed features, in particular $B*$ are among the most impacting on precision.

On the other hand, trying the same transfer learning schema with Logistic Regressor classifiers fails completely. This model is not able to predict the correct class better than the Dummy classifier, no matter which set of features is provided for training.

6 Discussion and Conclusion

In Transfer Learning, not only the various sources may refer to differently distributed populations, but also the corresponding feature sets can overlap only partially. Each source will have a number of unique features: leaving them out of the analysis means missing the opportunity to exploit their synergistic effects.

In this work, we provided a framework based on Partial Information Decomposition to analyze the kinds of information that can be transferred from the Source Domain to the Target Domain, when their feature sets overlap. Some information component from the source is irremediably lost to the Target Domain, some is implicitly available, some can be recovered using a bridging approach based on the shared features. We demonstrate the latter case with a numerical example based on synthetic data. We outline a pattern where heterogeneous transfer learning can be useful and where a specific heterogeneous transfer learning schema can be effective.

This HTL approach contrasts with other approaches that focus on domain adaptation over shared features, such as that in [2], which works by using a symmetric transformation that maps the feature spaces of the source and target data to a common subspace using projection matrices. Such a method to incorporate the original features of the data into the transformed data in the common subspace uses two feature mapping functions that involve plugging zeros for the nonshared features (that is, the Source Domain specific features and the Target Domain specific features) to match the domain dimensions. This operation does not preserve domain-specific information and hinders the exploitation of bridging and possible synergies.

We plan to extend the present work by experimenting with different dependency patterns and extending the analysis to real-world datasets.

Acknowledgements

The work was partially supported by the project MUSA – Multilayered Urban Sustainability Action – project, funded by the European Union – NextGenerationEU, (CUP G43C22001370007, Code ECS00000037). The work was also partially supported by the project SERICS (PE00000014) under the NRRP MUR program funded by the EU under NextGenerationEU.

References

1. Day, O., Khoshgoftaar, T.M.: A survey on heterogeneous transfer learning. Journal of Big Data **4**, 1–42 (2017)
2. Duan, L., Xu, D., Tsang, I.: Learning with augmented features for heterogeneous domain adaptation. arXiv preprint arXiv:1206.4660 (2012)
3. Kolchinsky, A.: A novel approach to the partial information decomposition. Entropy **24**(3) (2022). https://doi.org/10.3390/e24030403, https://www.mdpi.com/1099-4300/24/3/403
4. Mozafari, A.S., Jamzad, M.: A svm-based model-transferring method for heterogeneous domain adaptation. Pattern Recognition **56**, 142–158 (2016). https://doi.org/https://doi.org/10.1016/j.patcog.2016.03.009
5. Williams, P.L., Beer, R.D.: Nonnegative decomposition of multivariate information. arXiv: Information Theory (2010)
6. Zhou, J.T., Tsang, I.W., Pan, S.J., Tan, M.: Multi-class heterogeneous domain adaptation. Journal of Machine Learning Research **20**(57), 1–31 (2019), http://jmlr.org/papers/v20/13-580.html
7. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proceedings of the IEEE **109**(1), 43–76 (2020)