Semi-Supervised Domain Adaptation with Non-Parametric Copulas

David Lopez-Paz
MPI for Intelligent Systems
dlopez@tue.mpg.de

José Miguel Hernández-Lobato
University of Cambridge
jmh233@cam.ac.uk

Bernhard Schölkopf
MPI for Intelligent Systems
bs@tue.mpg.de

Abstract

A new framework based on the theory of copulas is proposed to address semi-supervised domain adaptation problems. The presented method factorizes any multivariate density into a product of marginal distributions and bivariate copula functions. Therefore, changes in each of these factors can be detected and corrected to adapt a density model accross different learning domains. Importantly, we introduce a novel vine copula model, which allows for this factorization in a non-parametric manner. Experimental results on regression problems with real-world data illustrate the efficacy of the proposed approach when compared to state-of-the-art techniques.

1 Introduction

When humans address a new learning problem, they often use knowledge acquired while learning different but related tasks in the past. For example, when learning a second language, people rely on grammar rules and word derivations from their mother tongue. This is called *language transfer* [19]. However, in machine learning, most of the traditional methods are not able to exploit similarities between different learning tasks. These techniques only achieve good performance when the data distribution is stable between training and test phases. When this is not the case, it is necessary to a) collect and label additional data and b) re-run the learning algorithm. However, these operations are not affordable in most practical scenarios.

Domain adaptation, *transfer learning* or *multitask learning* frameworks [17, 2, 5, 13] confront these issues by first, building a notion of *task relatedness* and second, providing mechanisms to *transfer knowledge* between similar tasks. Generally, we are interested in improving predictive performance on a *target task* by using knowledge obtained when solving another related *source task*. Domain adaptation methods are concerned about *what* knowledge we can share between different tasks, *how* we can transfer this knowledge and *when* we should do it or not to avoid additional damage [4].

In this work, we study semi-supervised domain adaptation for regression tasks. In these problems, the object of interest (the mechanism that maps a set of inputs to a set of outputs) can be stated as a conditional density function. The data available for solving each learning task is assumed to be sampled from modified versions of a common multivariate distribution. Therefore, we are interested in sharing the "common pieces" of this generative model between tasks, and use the data from each individual task to detect, learn and adapt the varying parts of the model. To do so, we must find a decomposition of multivariate distributions into simpler building blocks that may be studied separately across different domains. The theory of copulas provides such representations [18].

Copulas are statistical tools that factorize multivariate distributions into the product of its marginals and a function that captures any possible form of dependence among them. This function is referred to as the copula, and it links the marginals together into the joint multivariate model. Firstly intro-

duced by Sklar [22], copulas have been successfully used in a wide range of applications, including finance, time series or natural phenomena modeling [12]. Recently, a new family of copulas named *vines* have gained interest in the statistics literature [1]. These are methods that factorize multivariate densities into a product of marginal distributions and bivariate copula functions. Each of these factors corresponds to one of the building blocks that we assume either constant or varying across different learning domains.

The contributions of this paper are two-fold. First, we propose a non-parametric vine copula model which can be used as a high-dimensional density estimator. Second, by making use of this method, we present a new framework to address semi-supervised domain adaptation problems, which performance is validated in a series of experiments with real-world data and competing state-of-the-art techniques.

The rest of the paper is organized as follows: Section 2 provides a brief introduction to copulas, and describes a non-parametric estimator for the bivariate case. Section 3 introduces a novel non-parametric vine copula model, which is formed by the described bivariate non-parametric copulas. Section 4 describes a new framework to address semi-supervised domain adaptation problems using the proposed vine method. Finally, section 5 describes a series of experiments that validate the proposed approach on regression problems with real-world data.

2 Copulas

When the components of $\mathbf{x} = (x_1, \ldots, x_d)$ are jointly independent, their density function $p(\mathbf{x})$ can be written as

$$p(\mathbf{x}) = \prod_{i=1}^{d} p(x_i)\,. \tag{1}$$

This equality does not hold when $x_1, \ldots, x_d$ are not independent. Nevertheless, the differences can be corrected if we multiply the right hand side of (1) by a specific function that fully describes any possible dependence between $x_1, \ldots, x_d$. This function is called the *copula* of $p(\mathbf{x})$ [18] and satisfies

$$p(\mathbf{x}) = \prod_{i=1}^{d} p(x_i)\ \underbrace{c(P(x_1), ..., P(x_d))}_{\text{copula}}\,. \tag{2}$$

The copula c is the joint density of $P(x_1), \ldots, P(x_d)$, where $P(x_i)$ is the marginal cdf of the random variable x_i. This density has uniform marginals, since $P(z) \sim \mathcal{U}[0, 1]$ for any random variable z. That is, when we apply the transformation $P(x_1), \ldots, P(x_d)$ to $x_1, \ldots, x_d$, we are eliminating all information about the marginal distributions. Therefore, the copula captures any distributional pattern that does not depend on their specific form, or, in other words, all the information regarding the dependencies between $x_1, \ldots, x_d$. When $P(x_1), \ldots, P(x_d)$ are continuous, the copula c is unique [22]. However, infinitely many multivariate models share the same underlying copula function, as illustrated in Figure 1. The main advantage of copulas is that they allow us to model separately the marginal distributions and the dependencies linking them together to produce the multivariate model subject of study.

Given a sample from (2), we can estimate $p(\mathbf{x})$ as follows. First, we construct estimates of the marginal pdfs, $\hat{p}(x_1), \ldots, \hat{p}(x_d)$, which also provide estimates of the corresponding marginal cdfs, $\hat{P}(x_1), \ldots, \hat{P}(x_d)$. These cdfs estimates are used to map the data to the d-dimensional unit hypercube. The transformed data are then used to obtain an estimate $\hat{c}$ for the copula of $p(\mathbf{x})$. Finally, (2) is approximated as

$$\hat{p}(\mathbf{x}) = \prod_{i=1}^{d} \hat{p}(x_i)\, \hat{c}(\hat{P}(x_1), ..., \hat{P}(x_d)). \tag{3}$$

The estimation of marginal pdfs and cdfs can be implemented in a non-parametric manner by using unidimensional kernel density estimates. By contrast, it is common practice to assume a parametric model for the estimation of the copula function. Some examples of parametric copulas are Gaussian, Gumbel, Frank, Clayton or Student copulas [18]. Nevertheless, real-world data often exhibit complex dependencies which cannot be correctly described by these parametric copula models. This lack of flexibility of parametric copulas is illustrated in Figure 2. As an alternative, we propose

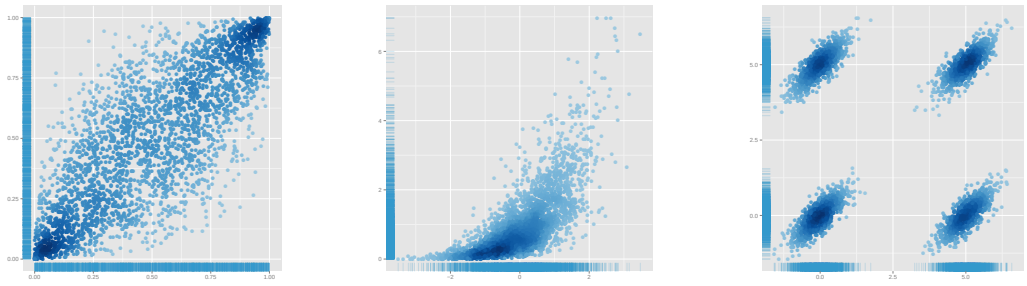

Figure 1: Left, sample from a Gaussian copula with correlation $\rho = 0.8$. Middle and right, two samples drawn from multivariate models with this same copula but different marginal distributions, depicted as rug plots.

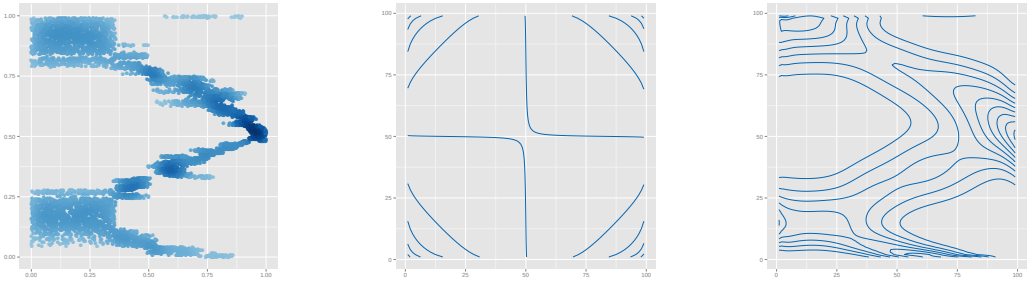

Figure 2: Left, sample from the copula linking variables 4 and 11 in the WIRELESS dataset. Middle, density estimate generated by a Gaussian copula model when fitted to the data. This technique is unable to capture the complex patterns present in the data. Right, copula density estimate generated by the non-parametric method described in section 2.1.

to approximate the copula function in a non-parametric manner. Kernel density estimates can also be used to generate non-parametric approximations of copulas, as described in [8]. The following section reviews this method for the two-dimensional case.

2.1 Non-parametric Bivariate Copulas

We now elaborate on how to non-parametrically estimate the copula of a given bivariate density $p(x, y)$. Recall that this density can be factorized as the product of its marginals and its copula

$$p(x, y) = p(x)\, p(y)\, c(P(x), P(y)). \tag{4}$$

Additionally, given a sample $\{(x_i, y_i)\}_{i=1}^n$ from $p(x, y)$, we can obtain a pseudo-sample from its copula c by mapping each observation to the unit square using estimates of the marginal cdfs, namely

$$\{(u_i, v_i)\}_{i=1}^n := \{(\hat{P}(x_i), \hat{P}(y_i))\}_{i=1}^n. \tag{5}$$

These are approximate observations from the uniformly distributed random variables $u = P(x)$ and $v = P(y)$, whose joint density is the copula function $c(u, v)$. We could try to approximate this density function by placing Gaussian kernels on each observation u_i and v_i. However, the resulting density estimate would have support on $\mathbb{R}^2$, while the support of c is the unit square. A solution is to perform the density estimation in a transformed space. For this, we select some continuous distribution with support on $\mathbb{R}$, strictly positive density ϕ, cumulative distribution Φ and quantile function Φ^{-1}. Let z and w be two new random variables given by $z = \Phi^{-1}(u)$ and $w = \Phi^{-1}(v)$. Then, the joint density of z and w is

$$p(z, w) = \phi(z)\, \phi(w)\, c(\Phi(z), \Phi(w))\,. \tag{6}$$

The copula of this new density is identical to the copula of (4), since the performed transformations are marginal-wise. The support of (6) is now $\mathbb{R}^2$; therefore, we can now approximate it with

Gaussian kernels. Let $z_i = \Phi^{-1}(u_i)$ and $w_i = \Phi^{-1}(v_i)$. Then,

$$\hat{p}(z, w) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(z, w | z_i, w_i, \mathbf{\Sigma}), \tag{7}$$

where $\mathcal{N}(\cdot, \cdot | \nu_1, \nu_2, \mathbf{\Sigma})$ is a two-dimensional Gaussian density with mean (ν_1, ν_2) and covariance matrix $\mathbf{\Sigma}$. For convenience, we select ϕ, Φ and Φ^{-1} to be the standard Gaussian pdf, cdf and quantile function, respectively. Finally, the copula density $c(u, v)$ is approximated by combining (6) with (7):

$$\hat{c}(u, v) = \frac{\hat{p}(\Phi^{-1}(u), \Phi^{-1}(v))}{\phi(\Phi^{-1}(u))\phi(\Phi^{-1}(v))} = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathcal{N}(\Phi^{-1}(u), \Phi^{-1}(v) | \Phi^{-1}(u_i), \Phi^{-1}(v_i), \mathbf{\Sigma})}{\phi(\Phi^{-1}(u))\phi(\Phi^{-1}(v))} \,. \tag{8}$$

3 Regular Vines

The method described above can be generalized to the estimation of copulas of more than two random variables. However, although kernel density estimates can be successful in spaces of one or two dimensions, as the number of variables increases, this methods start to be significantly affected by the curse of dimensionality and tend to overfit to the training data. Additionally, for addressing domain adaptation problems, we are interested in factorizing these high-dimensional copulas into simpler building blocks transferrable accross learning domains. These two drawbacks can be addressed by recent methods in copula modelling called *vines* [1]. Vines decompose any high-dimensional copula density as a product of bivariate copula densities that can be approximated using the non-parametric model described above. These bivariate copulas (as well as the marginals) correspond to the simple building blocks that we plan to transfer from one learning domain to another. Different types of vines have been proposed in the literature. Some examples are *canonical vines*, *D-vines* or *regular* vines [16, 1]. In this work we focus on regular vines (R-vines) since they are the most general models.

An R-vine $\mathcal{V}$ for a probability density $p(x_1, \ldots, x_d)$ with variable set $\mathbf{V} = \{1, \ldots d\}$ is formed by a set of undirected trees $T_1, \ldots, T_{d-1}$, each of them with corresponding set of nodes V_i and set of edges E_i, where $V_i = E_{i-1}$ for $i \in [2, d-1]$. Any edge $e \in E_i$ has associated three sets $C(e), D(e), N(e) \subset \mathbf{V}$ called the conditioned, conditioning and constraint sets of e, respectively. Initially, T_1 is inferred from a complete graph with a node associated with each element of $\mathcal{V}$; for any $e \in T_1$ joining nodes V_j and V_k, $C(e) = N(e) = \{V_j, V_k\}$ and $D(e) = \{\emptyset\}$. The trees $T_2, ..., T_{d-1}$ are constructed so that each $e \in E_i$ is formed by joining two edges $e_1, e_2 \in E_{i-1}$ which share a common node, for $i \geq 2$. The new edge e has conditioned, conditioning and constraint sets given by $C(e) = N(e_1) \Delta N(e_2)$, $D(e) = N(e_1) \cap N(e_1)$, $N(e) = N(e_1) \cup N(e_2)$, where Δ is the symmetric difference operator. Figure 3 illustrates this procedure for an R-vine with 4 variables.

For any edge $e(j, k) \in T_i$, $i = 1, \ldots, d-1$ with conditioned set $C(e) = \{j, k\}$ and conditioning set $D(e)$ let $c_{jk|D(e)}$ be the value of the copula density for the conditional distribution of x_j and x_k when conditioning on $\{x_i : i \in D(e)\}$, that is,

$$c_{jk|D(e)} := c(P_{j|D(e)}, P_{k|D(e)} | x_i : i \in D(e)), \tag{9}$$

where $P_{j|D(e)} := P(x_j | x_i : i \in D(e))$ is the conditional cdf of x_j when conditioning on $\{x_i : i \in D(e)\}$. Kurowicka and Cooke [16] indicate that *any* probability density function $p(x_1, \ldots, x_d)$ can then be factorized as

$$p(\mathbf{x}) = \prod_{i=1}^{d} p(x_i) \prod_{i=1}^{d-1} \prod_{e(j,k) \in E_i} c_{jk|D(e)} \,, \tag{10}$$

where $E_1, \ldots, E_{d-1}$ are the edge sets of the R-vine $\mathcal{V}$ for $p(x_1, \ldots, x_d)$. In particular, each of the edges in the trees from $\mathcal{V}$ specify a different conditional copula density in (10). For d variables, the density in (10) is formed by $d(d-1)/2$ factors. Changes in each of these factors can be detected and independently transferred accross different learning domains to improve the estimation of the target density function.

The definition of $c_{jk|D(e)}$ in (9) requires the calculation of conditional marginal cdfs. For this, we use the following recursive identity introduced by Joe [14], that is,

$$P_{j|D(e)} = \frac{\partial \, C_{jk|D(e)\backslash k}}{\partial P_{k|D(e)\backslash k}} \,, \tag{11}$$

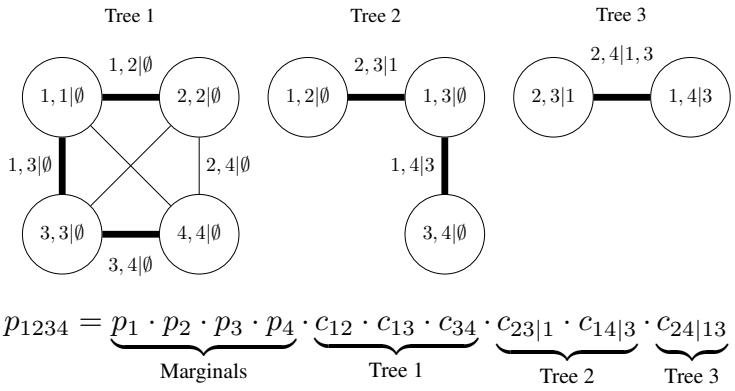

Figure 3: Example of the hierarchical construction of a R-vine copula for a system of four variables. The edges selected to form each tree are highlighted in bold. Conditioned and conditioning sets for each node and edge are shown as $C(e)|D(e)$. Later, each edge in bold will correspond to a different bivariate copula function.

which holds for any $k \in D(e)$, where $D(e) \setminus k = \{i : i \in D(e) \wedge i \neq k\}$ and $C_{jk|D(e)\setminus k}$ is the cdf of $c_{jk|D(e)\setminus k}$.

One major advantage of vines is that they can model high-dimensional data by estimating density functions of only one or two random variables. For this reason, these techniques are significantly less affected by the curse of dimensionality than regular density estimators based on kernels, as we show in Section 5. So far Vines have been generally constructed using parametric models for the estimation of bivariate copulas. In the following, we describe a novel method for the construction of non-parametric regular vines.

3.1 Non-parametric Regular Vines

In this section, we introduce a vine distribution in which all participant bivariate copulas can be estimated in a non-parametric manner. Todo so, we model each of the copulas in (10) using the non-parametric method described in Section 2.1. Let $\{(u_i, v_i)\}_{i=1}^n$ be a sample from the copula density $c(u, v)$. The basic operation needed for the implementation of the proposed method is the evaluation of the conditional cdf $P(u|v)$ using the recursive equation (11). Define $w = \Phi^{-1}(v)$, $z_i = \Phi^{-1}(u_i)$ and $w_i = \Phi^{-1}(v_i)$. Combining (8) and (11) we obtain

$$\begin{aligned}
\hat{P}(u|v) &= \int_0^u \hat{c}(x, v)\, dx \\
&= \frac{1}{n\phi(w)} \sum_{i=1}^{n} \int_0^u \frac{\mathcal{N}(\Phi^{-1}(x), w|z_i, w_i, \mathbf{\Sigma})}{\phi(\Phi^{-1}(x))}\, dx \\
&= \frac{1}{n\phi(w)} \sum_{i=1}^{n} \mathcal{N}(w|w_i, \sigma_w^2)\, \Phi\left[\frac{\Phi^{-1}(u) - \mu_{z_i|w_i}}{\sigma^2_{z_i|w_i}}\right], \qquad (12)
\end{aligned}$$

where $\mathcal{N}(\cdot|\mu, \sigma^2)$ denotes a Gaussian density with mean μ and variance σ^2, $\mathbf{\Sigma} = \begin{pmatrix} \sigma_z^2 & \gamma \\ \gamma & \sigma_w^2 \end{pmatrix}$ the kernel bandwidth matrix, $\mu_{z_i|w_i} = z_i + \frac{\sigma_z}{\sigma_w}\gamma(w - w_i)$ and $\sigma^2_{z_i|w_i} = \sigma_z^2(1 - \gamma^2)$.

Equation (12) can be used to approximate any conditional cdf $P_{j|D(e)}$. For this, we use the fact that $P(x_j|x_i : i \in D(e)) = P(u_j|u_i : i \in D(e))$, where $u_i = P(x_i)$, for $i = 1, \ldots, d$, and recursively apply rule (11) using equation (12) to compute $\hat{P}(u_j|u_i : i \in D(e))$.

To complete the inference recipe for the non-parametric regular vine, we must specify how to construct the hierarchy of trees $T_1, \ldots, T_{d-1}$. In other words, we must define a procedure to select the edges (bivariate copulas) that will form each tree. We have a total of $d(d-1)/2$ bivariate copulas

which should be distributed among the different trees. Ideally, we would like to include in the first trees of the hierarchy the copulas with strongest dependence level. This will allow us to prune the model by assuming independence in the last $k < d$ trees, since the density function for the independent copula is constant and equal to 1. To construct the trees $T_1, \ldots, T_{d-1}$, we assign a weight to each edge $e(j,k)$ (copula) according to the level of dependence between the random variables x_j and x_k. A common practice is to fix this weight to the empirical estimate of Kendall's' τ for the two random variables under consideration[1][1]. Given these weights for each edge, we propose to solve the edge selection problem by obtaining $d-1$ maximum spanning trees. *Prim's Algorithm* [20] can be used to solve this problem efficiently.

4 Domain Adaptation with Regular Vines

In this section we describe how regular vines can be used to address domain adaptation problems in the non-linear regression setting with continuous data. The proposed approach could be easily extended to other problems such as density estimation or classification. In regression problems, we are interested in inferring the *mapping mechanism* or conditional distribution with density $p(y|\mathbf{x})$ that maps one feature vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ into a target scalar value $y \in \mathbb{R}$. Rephrased into the copula framework, this conditional density can be expressed as

$$p(y|\mathbf{x}) \propto p(y) \prod_{i=1}^{d} \prod_{e(j,k)\in E_i} c_{jk|D(e)} \tag{13}$$

where $E_1, \ldots, E_d$ are the edge sets of an R-vine for $p(\mathbf{x}, y)$. Note that the normalization of the right part of (13) is relatively easy since y is scalar.

In the classic domain adaptation setup we usually have large amounts of data for solving a source task characterized by the density function $p_s(\mathbf{x}, y)$. However, only a partial or reduced sample is available for solving a target task with density $p_t(\mathbf{x}, y)$. Given the data available for both tasks, our objective is to build a good estimate for the conditional density $p_t(y|\mathbf{x})$. To address this domain adaptation problem, we assume that p_t is a modified version of p_s. In particular, we assume that p_t is obtained in two steps from p_s. First, p_s is expressed using an R-vine representation as in (10) and second, some of the factors included in that representation (marginal distributions or pairwise copulas) are modified to derive p_t. All we need to address the adaptation across domains is to reconstruct the R-vine representation of p_s using data from the source task, and then identify which of the factors have been modified to produce p_t. These factors are corrected using data from the target task. In the following, we describe how to identify and correct these modified factors.

Marginal distributions can change between source and target tasks (also known as *covariate shift*). In this case, $P_s(x_i) \neq P_t(x_i)$, for $i = 1, \ldots, d$, or $P_s(y) \neq P_t(y)$, and we need to re-generate the estimates of the affected marginals using data from the target task. Additionally, some of the bivariate copulas $c_{jk|D(e)}$ may differ from source to target tasks. In this case, we also re-estimate the affected copulas using data from the target task. Simultaneous changes in both copulas and marginals can occur. However, there is no limitation in updating each of the modified components separately. Finally, if some of the factors remain constant across domains, we can use the available data from the target task to improve the estimates obtained using only the data from the source task. Note that we are addressing a more general problem than *covariate shift*. Besides identifying and correcting changes in marginal distributions, we also consider changes in any possible form of dependence (conditional distributions) between random variables.

For the implementation of the strategy mentioned above, we need to identify when two samples come from the same distribution or not. For this, we propose to use the non-parametric two-sample test *Maximum Mean Discrepancy* (MMD) [10]. MMD will return low p-values when two samples are unlikely to have been drawn from the same distribution. Specifically, given samples from two distributions P and Q, MMD will determine $P \neq Q$ if the distance between the embeddings of the empirical distributions for these two samples in a RKHS is significantly large.

[1]We have tried more general dependence measures such as the HSIC (*Hilbert-Schmidt Independence Criterion*) without observing gains that justify the increase of computational costs.

Table 1: Average TLL obtained by NPRV, GRV and KDE on six different UCI datasets.

Dataset	**Auto**	**Cloud**	**Housing**	**Magic**	**Page-Blocks**	**Wireless**
No. of variables	**8**	**10**	**14**	**11**	**10**	**11**
KDE	1.32 ± 0.06	3.25 ± 0.10	1.96 ± 0.17	1.13 ± 0.11	1.90 ± 0.13	0.98 ± 0.06
GRV	1.84 ± 0.08	$\mathbf{5.00 \pm 0.12}$	1.68 ± 0.11	2.09 ± 0.08	4.69 ± 0.20	0.36 ± 0.08
NPRV	$\mathbf{2.07 \pm 0.07}$	4.54 ± 0.13	$\mathbf{3.18 \pm 0.17}$	$\mathbf{2.72 \pm 0.17}$	$\mathbf{5.64 \pm 0.14}$	$\mathbf{2.17 \pm 0.13}$

Semi-supervised and unsupervised domain adaptation: The proposed approach can be easily extended to take advantage of additional unlabeled data to improve the estimation of our model. Specifically, extra unlabeled target task data can be used to refine the factors in the R-Vine decomposition of p_t which do not depend on y. This is still valid even in the limiting case of not having access to labeled data from the target task at training time (*unsupervised domain adaptation*).

5 Experiments

To validate the proposed method, we run two series of experiments using real world data. The first series illustrates the accuracy of the density estimates generated by the proposed non-parametric vine method. The second series validates the effectiveness of the proposed framework for domain adaptation problems in the non-linear regression setting. In all experiments, kernel bandwidth matrices are selected using Silverman's rule-of-thumb [21]. For comparative purposes, we include the results of different state-of-the-art domain adaptation methods whose parameters are selected by a 10-fold cross validation process on the training data.

Approximations: A complete R-Vine requires the use of conditional copula functions, which are challenging to learn. A common approximation is to ignore any dependence between the copula functional form and its set of conditioning variables. Note that the copula functions arguments remain to be conditioned cdfs. Moreover, to avoid excesive computational costs, we consider only the first tree ($d - 1$ copulas) of the R-Vine, which is the one containing the most amount of dependence between the distribution variables. Increasing the number of considered trees did not lead to significant performance improvements.

5.1 Accuracy of Non-parametric Regular Vines for Density Estimation

The density estimates generated by the new non-parametric R-vine method (NPRV) are evaluated on data from six normalized UCI datasets [9]. We compare against a standard density estimator based on Gaussian kernels (KDE), and a parametric vine method based on bivariate Gaussian copulas (GRV). From each dataset, we extract 50 random samples of size 1000. Training is performed using 30% of each random sample. Average test log-likelihoods and corresponding standard deviations on the remaining 70% of the random sample are summarized in Table 1 for each technique. In these experiments, NPRV obtains the highest average test log-likelihood in all cases except one, where it is outperformed by GRV. KDE shows the worst performance, due to its direct exposure to the curse of dimensionality.

5.2 Comparison with other Domain Adaptation Methods

NPRV is analyzed in a series of experiments for domain adaptation on the non-linear regression setting with real-world data. Detailed descriptions of the 6 UCI selected datasets and their domains are available in the supplementary material. The proposed technique is compared with different benchmark methods. The first two, GP-SOURCE and GP-ALL, are considered baselines. They are two gaussian process (GP) methods, the first one trained only with data from the source task, and the second one trained with the normalized union of data from both source and target problems. The other five methods are considered state-of-the-art domain adaptation techniques. DAUME [7] performs a feature augmentation such that the kernel function evaluated at two points from the same

Table 2: Average NMSE and standard deviation for all algorithms and UCI datasets.

Dataset	**Wine**	**Sarcos**	**Rocks-Mines**	**Hill-Valleys**	**Axis-Slice**	**Isolet**
No. of variables	**12**	**21**	**60**	**100**	**386**	**617**
GP-Source	0.86 ± 0.02	1.80 ± 0.04	0.90 ± 0.01	1.00 ± 0.00	1.52 ± 0.02	1.59 ± 0.02
GP-All	0.83 ± 0.03	1.69 ± 0.04	1.10 ± 0.08	0.87 ± 0.06	1.27 ± 0.07	1.58 ± 0.02
Daume	0.97 ± 0.03	0.88 ± 0.02	0.72 ± 0.09	0.99 ± 0.03	0.95 ± 0.02	0.99 ± 0.00
SSL-Daume	0.82 ± 0.05	0.74 ± 0.08	0.59 ± 0.07	0.82 ± 0.07	0.65 ± 0.04	0.64 ± 0.02
ATGP	0.86 ± 0.08	0.79 ± 0.07	$\mathbf{0.56 \pm 0.10}$	0.15 ± 0.07	1.00 ± 0.01	1.00 ± 0.00
KMM	1.03 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
KuLSIF	0.91 ± 0.08	1.67 ± 0.06	0.65 ± 0.10	0.80 ± 0.11	0.98 ± 0.07	0.58 ± 0.02
NPRV	$\mathbf{0.73 \pm 0.07}$	$\mathbf{0.61 \pm 0.10}$	0.72 ± 0.13	$\mathbf{0.15 \pm 0.07}$	0.38 ± 0.07	0.46 ± 0.09
UNPRV	0.76 ± 0.06	0.62 ± 0.13	0.72 ± 0.15	0.19 ± 0.09	$\mathbf{0.37 \pm 0.07}$	$\mathbf{0.42 \pm 0.04}$
Av. Ch. Mar.	10	1	38	100	226	89
Av. Ch. Cop.	5	8	49	34	155	474

domain is twice larger than when these two points come from different domains. SSL-DAUME [6] is a SSL extension of DAUME which takes into account unlabeled data from the target domain. ATGP [4] models the source and target task data using a single GP, but learns additional kernel parameters to correlate input vectors between domains. This method outperforms others like the one proposed by Bonilla et al. [3]. KMM [11] minimizes the distance of marginal distributions in source and target domains by matching their means when mapped into an universal RKHS. Finally, KULSIF [15] operates in a similar way as KMM. Besides NPRV, we also include in the experiments its fully unsupervised variant, UNPRV, which ignores any labeled data from the target task.

For training, we randomly sample 1000 data points for both source and target tasks, where all the data in the source task and 5% of the data in the target task are labeled. The test set contains 1000 points from the target task. Table 2 summarizes the average test normalized mean square error (NMSE) and corresponding standard deviation for each method in each dataset across 30 random repetitions of the experiment. The proposed methods obtain the best results in 5 out of 6 cases. Notably, UNPRV (Unsupervised NPRV), which ignores labeled data from the target task, also outperforms the other benchmark methods in most cases. Finally, the two bottom rows in Table 2 show the average number of marginals and bivariate copulas which are updated in each dataset during the execution of NPRV, respectively.

Computational Costs: Running NPRV requires to fill in a weight matrix of size $\mathcal{O}(d^2)$ with the empirical estimates of Kendall's τ for any two random variables. The computation of each of these estimates can be done efficiently with cost $\mathcal{O}(n \log n)$, where n is the number of available data points. Therefore, the final training cost of NPRV is $\mathcal{O}(d^2 n \log n)$. In practice, we obtain competitive training times. Training NPRV for the *Isolet* dataset took about 3 minutes on a regular laptop computer. Predictions made by a single level NPRV have cost $\mathcal{O}(nd)$. Parametric copulas may be used to reduce the computational demands.

6 Conclusions

We have proposed a novel non-parametric domain adaptation strategy based on copulas. The new approach works by decomposing any multivariate density into a product of marginal densities and bivariate copula functions. Changes in these factors across different domains can be detected using two sample tests, and transferred across domains in order to adapt the target task density model. A novel non-parametric vine method has been introduced for the practical implementation of this method. This technique leads to better density estimates than standard parametric vines or KDE, and is also able to outperform a large number of alternative domain adaptation methods in a collection of regression problems with real-world data.

References

[1] K. Aas, C. Czado, A. Frigessi, and H. Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2):182–198, 2006.

[2] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, 2010.

[3] E. Bonilla, K. Chai, and C. Williams. Multi-task gaussian process prediction. *NIPS*, 2008.

[4] B. Cao, S. Jialin, Y. Zhang, D. Yeung, and Q. Yang. Adaptive transfer learning. *AAAI*, 2010.

[5] C. Cortes and M. Mohri. Domain adaptation in regression. In *Proceedings of the 22nd international conference on Algorithmic learning theory*, ALT'11, pages 308–323, Berlin, Heidelberg, 2011. Springer-Verlag.

[6] H. Daumé, III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59, 2010.

[7] H. Daumé III. Frustratingly easy domain adaptation. *Association of Computational Linguistics*, pages 256–263, 2007.

[8] J. Fermanian and O. Scaillet. The estimation of copulas: Theory and practice. *Copulas: From Theory to Application in Finance*, pages 35–60, 2007.

[9] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[10] A. Gretton, K. Borgwardt, M. Rasch, B. Scholkopf, and A. Smola. A kernel method for the two-sample-problem. *NIPS*, pages 513–520, 2007.

[11] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schoelkopf. Correcting sample selection bias by unlabeled data. *NIPS*, pages 601–608, 2007.

[12] P. Jaworski, F. Durante, W.K. Härdle, and T. Rychlik. *Copula Theory and Its Applications*. Lecture Notes in Statistics. Springer, 2010.

[13] S. Jialin-Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[14] H. Joe. Families of m-variate distributions with given margins and $m(m-1)/2$ bivariate dependence parameters. *Distributions with Fixed Marginals and Related Topics*, 1996.

[15] T. Kanamori, T. Suzuki, and M. Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3):335–367, 2012.

[16] D. Kurowicka and R. Cooke. *Uncertainty Analysis with High Dimensional Dependence Modelling*. Wiley Series in Probability and Statistics, 1st edition, 2006.

[17] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[18] R. Nelsen. *An Introduction to Copulas*. Springer Series in Statistics, 2nd edition, 2006.

[19] S. Nitschke, E. Kidd, and L. Serratrice. First language transfer and long-term structural priming in comprehension. *Language and Cognitive Processes*, 5(1):94–114, 2010.

[20] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[21] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability. Chapman and Hall, 1986.

[22] A. Sklar. Fonctions de repartition à n dimension set leurs marges. *Publ. Inst. Statis. Univ. Paris*, 8(1):229–231, 1959.