Source Hypothesis Transfer for Zero-Shot Domain Adaptation

Tomoya Sakai

NEC Corporation
tomoya_sakai@nec.com

Abstract. Making predictions in target unseen domains without training samples is frequent in real-world applications, such as new products' sales predictions. Zero-shot domain adaptation (ZSDA) has been studied to achieve this important but difficult task. An approach to ZSDA is to use multiple source domain data and domain attributes. Several recent domain adaptation studies have mentioned that source domain data are not often available due to privacy, technical, and contractual issues in practice. To address these issues, hypothesis transfer learning (HTL) has been gaining attention since it does not require access to source domain data. It has shown its effectiveness in supervised/unsupervised domain adaptation; however current HTL methods cannot be readily applied to ZSDA because we have no training data (even unlabeled data) for target domains. To solve this problem, we propose an HTL-based ZSDA method that connects multiple source hypotheses by domain attributes. Through theoretical analysis, we derive the convergence rate of the estimation error of our proposed method. Finally, we numerically demonstrate the effectiveness of our proposed HTL-based ZSDA method.

Keywords: Hypothesis transfer learning, Zero-shot domain adaptation, Unseen domains, Domain adaptation

1 Introduction

In real-world applications, training data of our task of interest are not often available. To name a few, sales prediction of new products, preference prediction of new users, and energy consumption prediction of new sites are applications in which labeled training data in a target domain does not exist. The task of making predictions in *unseen* domains (e.g., new products) without any target training data is known as *zero-shot domain adaptation* (ZSDA) [25,26]. To enable ZSDA, these studies proposed an approach that uses multiple source data (i.e., training datasets obtained from multiple domains) and *domain attributes* (i.e., descriptions of domains). For sales prediction of food, domain attributes can be colors, size, and nutritional components. Intuitively, the relation between sales of existing products and domain attributes can be regarded as a clue of estimating sales of new products, since a combination of domain attributes of a new product is new but each element already appears in existing products.

Table 1. Comparison of our method and related work. SHOT [16] does not require source data but cannot be applied to unseen domains. In contrast, MDMT [25] can handle unseen domains but requires source data. Our method addresses both issues.

	Proposed method	SHOT [16]	MDMT [25]
HTL	✓	✓	–
ZSDA	✓	–	✓

Another crucial issue that has emerged is that source domain data are not always available due to legal, technical, and contractual constraints between data owners and data customers [4]. It is common for decision-making rules to be only available, e.g., learned prediction functions are accessible but not source domain data. To handle this situation, *hypothesis transfer learning* (HTL) [14,16] is promising because it does not require source data for training a new model. Since HTL does not require access to source domain data, it secures private information in the source domain data and saves memory and computation time for training a new target model [16].

In the existing ZSDA methods, multiple source domain data are used for training. This is not suitable for applications that are privacy sensitive and require expensive computational resources to store source domain data. The method proposed by Mansour et al. [17] allows us to train a target model from multiple source hypotheses. However, the method requires training data obtained from target domains. Thus, HTL-based ZSDA methods should enable to solve sales prediction of new products while maintaining privacy and reducing storage costs.

In this paper, we propose a ZSDA method that is based on HTL. The main challenge is that we cannot use current HTL methods since target training data are not available in ZSDA. To tackle this challenge, we introduce a new learning objective that connects hypotheses for existing domains through domain attributes to train a prediction model for unseen domains. An advantage of our method is that it can be easily implemented with *Scikit-learn* [20] in Python. Through theoretical analysis, we derive the convergence rate of an estimation error of our method. To the best of our knowledge, this is the first study to present HTL for ZSDA (see also Table 1) and the convergence rate of an estimation error in ZSDA. We then conducted numerical experiments to show that our proposed method achieved comparable or sometimes superior performance to a non-HTL method for ZSDA.

2 Related Work

Domain adaptation without target samples has been studied from several aspects. For example, *domain generalization* (DG) [2, 15] obtains predictions without any sample obtained from target domains. In DG, samples from multiple source domains are assumed available. However, in some applications, it is difficult to assume multiple source domains. To address this issue, *zero-shot deep domain adaptation* [21] (ZDDA) has been proposed.

Compared with DG and ZDDA, our method requires domain attributes, similarly to the methods in [25, 26]. While the requirement might restrict applications of our method, there is a trade-off between having and not having domain attributes. The use of domain attributes incurs annotation cost but enables us to handle a response that depends on both input features and domain attributes. However, the approach without domain attributes cannot handle such a case. ZSDA is beneficial when discriminating domains from input features is difficult or almost impossible.

While several studies have considered making predictions in unseen target domains without any target training data, they relied on the availability of source domain data. From the viewpoint of HTL, the method proposed by Mansour et al. [17] can be regarded as a method from multiple source hypotheses but requires target training data.

3 Problem Setting and Background

In this section, we explain our problem setting and background knowledge.

3.1 Problem Setting

Let a covariate $\boldsymbol{x}^{(t)} \in \mathbb{R}^d$ and its corresponding response $y^{(t)} \in \mathbb{R}$, where t denotes the task index and d is a positive integer. Let us denote a set of seen domain indices by $\mathcal{T}_{\mathrm{S}} = \{1, \dots, T_{\mathrm{S}}\}$, where T_{S} is the number of seen domains. Similarly, let $\mathcal{T}_{\mathrm{U}} := \{T_{\mathrm{S}} + 1, \dots, T_{\mathrm{S}} + T_{\mathrm{U}}\}$ be a set of unseen target domain indices, where T_{U} denotes the number of unseen domains. As a signature of a domain, we assume an m-dimensional vector $\boldsymbol{a}^{(t)} \in \mathbb{R}^m$ is available for each domain and call $\boldsymbol{a}$ domain attributes (domain-attribute vector). Let us define a set of attribute vectors for seen and unseen domain as

$$\mathcal{A}_{\mathrm{S}} := \{\boldsymbol{a}^{(1)}, \dots, \boldsymbol{a}^{(T_{\mathrm{S}})}\},$$
$$\mathcal{A}_{\mathrm{U}} := \{\boldsymbol{a}^{(T_{\mathrm{S}}+1)}, \dots, \boldsymbol{a}^{(T_{\mathrm{S}}+T_{\mathrm{U}})}\},$$

respectively.

Let $h^{(t)} \colon \mathbb{R}^d \to \mathbb{R}$ be a source hypothesis for a domain t and $\boldsymbol{h}_{\mathrm{S}} := (h^{(1)}, \dots, h^{(T_{\mathrm{S}})})^\top$. In this paper, we assume that the (learned) source hypotheses

$$\widehat{\boldsymbol{h}}_{\mathrm{S}} := (\widehat{h}^{(1)}, \dots, \widehat{h}^{(T_{\mathrm{S}})})^\top$$

are available. The source hypotheses $\widehat{\boldsymbol{h}}_{\mathrm{S}}$ can be obtained by supervised learning independently or by *multi-task learning* (MTL) [5] jointly from multiple source domain data. Note that as long as the input-output constraint is satisfied, any class of model (e.g., linear model, tree model, and neural networks) can be used with our method, while neural networks are assumed as a class of hypotheses with SHOT [16].

Our goal is to obtain a prediction of a test sample $\boldsymbol{x}'$ in an unseen target domain $t' \in \mathcal{T}_{\mathrm{U}}$ by using $\boldsymbol{a}^{(t')}$, $\mathcal{A}_{\mathrm{S}}$, and $\widehat{\boldsymbol{h}}_{\mathrm{S}}$ without source domain data.

3.2 Ordinary Supervised Learning

Ordinary supervised learning does not handle unseen domains, but we review the method of standard supervised learning since it can be used for obtaining source hypotheses $\boldsymbol{h}_{\mathrm{S}}$.

Suppose that we have a set of labeled samples for a seen domain t, i.e., source domain data:

$$\mathcal{D}^{(t)} := \{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^{n^{(t)}},$$

where $n^{(t)}$ is the number of labeled samples on t. A simple approach to obtain predictions in a seen domain is to train a predictor with the corresponding labeled samples $\mathcal{D}^{(t)}$. Specifically, for each $t \in \mathcal{T}_{\mathrm{S}}$, a predictor $h^{(t)}$ is trained to minimize the training error plus a regularization functional:

$$\underset{h^{(t)}}{\text{minimize}} \ \frac{1}{n^{(t)}} \sum_{i=1}^{n^{(t)}} \ell\big(h^{(t)}(\boldsymbol{x}_i^{(t)}), y_i^{(t)}\big) + \lambda W\big(h^{(t)}\big), \tag{1}$$

where W is the regularization functional, $\lambda \geq 0$ is the regularization parameter, and $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ is the loss function such as the squared loss: $\ell_{\mathrm{sq}}(y, y') := (y - y')^2$. With the learned source hypotheses $\widehat{\boldsymbol{h}}_{\mathrm{S}}$, we obtain a prediction for seen domains. However, it is not possible to make a prediction in an unseen domain because we only have $\widehat{\boldsymbol{h}}_{\mathrm{S}}$ for seen domains.

3.3 ZSDA with Source Domain Data

An approach to make a prediction in unseen domains is to include attribute vectors into the prediction model. We review one of the state-of-the-art ZSDA methods on the basis of domain attributes [25], the usefulness of which was also investigated, e.g., [10,23,26]. Let $F \colon \mathbb{R}^d \times \mathcal{T} \to \mathbb{R}$ be an *attribute-aware* predictor. An example of F is a bilinear function defined as

$$F(\boldsymbol{x}, t) = \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{a}^{(t)},$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times m}$ is the parameter matrix to be learned.

Suppose we have training data $\mathcal{D} := \{\mathcal{D}^{(t)}\}_{t \in \mathcal{T}_{\mathrm{S}}}$ and a set of seen attributes $\mathcal{A}_{\mathrm{S}}$. We then train F by labeled samples and attribute vectors for all seen domains. That is, we solve the following optimization problem:

$$\underset{F}{\text{minimize}} \ \frac{1}{T_{\mathrm{S}}} \sum_{t \in \mathcal{T}_{\mathrm{S}}} \frac{1}{n^{(t)}} \sum_{i=1}^{n^{(t)}} \ell\big(F(\boldsymbol{x}_i^{(t)}, t), y_i^{(t)}\big) + \widetilde{\lambda} \widetilde{W}(F), \tag{2}$$

where $\widetilde{W}$ is a regularization functional and $\widetilde{\lambda} \geq 0$ is the regularization parameter. After obtaining a learned predictor, denoted as $\widehat{F}$, a prediction of a test sample $\boldsymbol{x}'$ in an unseen domain t' can be obtained by $\widehat{F}(\boldsymbol{x}', t')$.

Although this approach can make predictions for a sample on an unseen domain, it requires access to a source training dataset $\mathcal{D}$, which is not always possible in practice [4].

4 Proposed Method

We explain how to make predictions in unseen domains by using multiple source hypotheses $\widehat{\boldsymbol{h}}_{\mathrm{S}}$.

4.1 Model Collaboration

To obtain predictions in unseen domains, we make the learned predictors take domain attributes into account. While it is difficult to design the predictors to handle domain attributes after their parameters are fixed, our approach can connect the learned predictors with domain attributes.

Our key idea is to make an implicit connection between the learned predictors and another prediction model that can take domain attributes into account. Instead of training the new prediction model, we compute the prediction of input in an unseen domain at *test* time. That is, the computation time of our method is zero until a test sample comes in. Fortunately, this does not cause any problems with ZSDA because we do not know the information of unseen domains in advance.

More specifically, let $\boldsymbol{x}'$ be a test input and $g\colon \mathcal{A} \to \mathbb{R}$ be a prediction function for $\boldsymbol{x}'$. We refer to g as a *fixed-input* model because g is in charge of the prediction of $\boldsymbol{x}'$ only. We then connect g with $\widehat{\boldsymbol{h}}_{\mathrm{S}}$ by minimizing the model collaboration (MC) error defined as

$$\widehat{R}_{\mathrm{MC}}(g) := \frac{1}{T_{\mathrm{S}}} \sum_{t=1}^{T_{\mathrm{S}}} \ell\big(g(\boldsymbol{a}^{(t)}), \widehat{h}^{(t)}(\boldsymbol{x}')\big). \tag{3}$$

In practice, we add a regularization functional $\widetilde{W}$ and solve the optimization problem expressed as

$$\operatorname*{minimize}_{g \in \mathcal{G}} \ \widehat{R}_{\mathrm{MC}}(g) + \widetilde{\lambda}\widetilde{W}(g),$$

where $\widetilde{\lambda}$ is the regularization parameter and $\mathcal{G}$ is a function class, such as linear models, tree models, and neural networks. Minimization of the MC error connects g with $\widehat{\boldsymbol{h}}_{\mathrm{S}}$. From the perspective of generalization, $\widehat{h}^{(t)}$ handles features while $\widehat{g}$ handles domain attributes.

At a glance, this approach might seem heuristic, e.g., one may think the MC error just connects hypotheses $\widehat{\boldsymbol{h}}_{\mathrm{S}}$ and a new model g through a loss function. However, this is a theoretically justified method. In Section 5, we show that the proposed method is theoretically valid.

After solving the above optimization problem, we obtained the learned fixed-input model $\widehat{g}$. The prediction of $\boldsymbol{x}'$ in an unseen domain $\boldsymbol{a}^{(t')}$ is given by $\widehat{g}(\boldsymbol{a}^{(t')})$. Since g is only in charge of $\boldsymbol{x}'$, for another test input $\boldsymbol{x}''$, we retrain g to obtain a prediction. However, as we explain in Section 4.3 and show through experimentation, this computation in an inference phase can be efficiently done.

Our method can be regarded as *transductive* inference [3, 24], where the task is to estimate labels of test instances included in the training samples and

Table 2. Required input to each method. MDMT requires training data $\mathcal{D}$ from multiple source domains. Our method does not require using $\mathcal{D}$ and enables us to make predictions by leveraging source hypotheses $\widehat{\boldsymbol{h}}_{\mathrm{S}}$.

	Proposed Method	MDMT [25]
Training	$\boldsymbol{x}' \in \mathbb{R}^d$, $\mathcal{A}_{\mathrm{S}}$	$\mathcal{D}$, $\mathcal{A}_{\mathrm{S}}$
Inference	$\boldsymbol{a}' \in \mathcal{A}_{\mathrm{U}}$	$\boldsymbol{x}' \in \mathbb{R}^d$, $\boldsymbol{a}' \in \mathcal{A}_{\mathrm{U}}$
Prediction model	$g \colon \mathcal{A} \to \mathbb{R}$	$F \colon \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}$
Source hypotheses	$\widehat{\boldsymbol{h}}_{\mathrm{S}}$	–

abandon the ability of prediction for new test instances in the future. Transductive inference is known as an easier task than *inductive* inference [3, 24]. In this sense, although our method is affected by the accuracy of learned predictors for seen domains, the advantage of transductive inference might neutralize the effect of using learned predictors.

Table 2 summarizes the required input to the proposed method and a (non-HTL) ZSDA method [25], a method of multi-domain and multi-task learning, called as MDMT. A notable difference is that our proposed method does not require source training data $\mathcal{D}$.

4.2 Hyperparameter Tuning

To tune a hyperparameter such as the regularization parameter, we use domain-wise dataset split. For example, if we have 100 domains, by 8:2 domain-wise split, we use 80 domains for training data and 20 for test. Similarly to class-wise cross-validation [22], we can use domain-wise cross-validation. In our implementation with linear ridge regression, we can use a computationally efficient implementation of leave-one-domain-out cross-validation (LOOCV) to tune the regularization parameter.

4.3 Implementation

General Implementation: An example of g is the linear model defined as $g(\boldsymbol{a}^{(t)}) = \boldsymbol{\beta}^\top \boldsymbol{a}^{(t)}$, where $\boldsymbol{\beta} \in \mathbb{R}^m$ is a parameter vector. For the linear model, if we use the squared loss and ℓ_2-regularizer, the optimization problem becomes the linear ridge regression [8] and the solution can be obtained analytically.

A prediction in an unseen domain associated with the attribute vector $\boldsymbol{a}^{(t')}$ can generally be obtained as follows. We first feed $\{\boldsymbol{a}^{(t)}, \widehat{h}^{(t)}(\boldsymbol{x}')\}_{t \in \mathcal{T}_{\mathrm{S}}}$ as training data into a function of a regression method then obtain prediction by feeding $\boldsymbol{a}^{(t')}$ into the trained model. Note that, in the above procedure, we can use any regression method.

Our method can be implemented by a few lines of Python code with the *Scikit-learn* [20] package. We show an example of Python implementation in Code 1. As shown in this code, our method is model-independent. We can thus use any method for both $\widehat{h}^{(t)}$ and g.

```
def predict(x_test, h_S, a_S, a_U):
    """
    @x_test: a test data point (1 times d)
    @h_S: a list of seen predictors (T_S size)
    @a_S: a matrix of domain attributes (T_S times m)
    @a_U: a vector of domain-attributes for target (1 times m
                                             )
    """
    yh_S = [h.predict(x_test) for h in h_S]
    reg = Any_ScikitLearn_Regressor()
    reg.fit(a_S, yh_S)
    return reg.predict(a_U)
```

Code 1. Example of Python implementation

Computationally-Efficient Implementation: An apparent drawback of our method is the necessity of calculation for each test sample. Although we might not need to handle millions of test samples in one second in practice, it is better that the computation time of our method be short.

In this section, we explain a computationally-efficient implementation based on the linear ridge regression with LOOCV [8]. For example, we can use the *RidgeCV* in Scikit-learn as an implementation of the linear ridge regression with LOOCV. In LOOCV, the eigendecomposition of $\boldsymbol{A}_{\mathrm{S}} = (\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(T_{\mathrm{S}})})$ is necessary but only once unless we add new seen domains or change the representation of domain attributes. That is, after eigendecomposition, the computation time of prediction consists of several multiplications of vectors and matrices.

More specifically, the matrix multiplication of $\boldsymbol{A}_{\mathrm{S}}^{\top}\boldsymbol{A}_{\mathrm{S}}$ takes $\mathcal{O}(m^2 T_{\mathrm{S}})$ time. The eigendecomposition of $\boldsymbol{A}_{\mathrm{S}}^{\top}\boldsymbol{A}_{\mathrm{S}}$ takes $\mathcal{O}(m^3)$ time. Once we obtain the eignedecomposition, we can reuse the result for any test point as long as the representation of domain attributes is fixed. If we can compute the eigendecomposition of $\boldsymbol{A}_{\mathrm{S}}^{\top}\boldsymbol{A}_{\mathrm{S}}$ in advance, $\mathcal{O}(m^3 + m^2 T_{\mathrm{S}})$ does not matter in prediction.

Let $\mathcal{O}(H)$ be the computational complexity of computing prediction for an in-service predictor, i.e., inference time. For example, H becomes d if we use the linear models as $\widehat{h}^{(t)}$. Then, obtaining the outputs of all learned predictors for a single test data point requires $\mathcal{O}(HT_{\mathrm{S}})$ time. For each hyperparameter, we can compute the score of LOOCV in $\mathcal{O}(T_{\mathrm{S}}^2)$ time. Let L be the number of candidates of the regularization parameters. The total computation time of the hyperparameter tuning and parameter estimation takes $\mathcal{O}(LT_{\mathrm{S}}^2)$. After we determine the hyperparameter, we then compute the prediction in $\mathcal{O}(mT_{\mathrm{S}})$ time.

5 Theoretical Analysis

Our ultimate goal is to obtain a prediction function that minimizes error to the ground truth function. In contrast, the MC error measures the average loss between a prediction function and learned hypotheses. In this sense, one may think that MC error minimization is just a heuristic. However, it is not. In this section, we investigate the estimation error for evaluating the difference between a minimizer of the MC error and an optimal one that is as close as to the ground truth function, and we elucidate the convergence rate of the estimation error bound.

5.1 Notations and Assumptions

In this analysis, we assume a set of attribute vectors of size T is drawn independently from the distribution with density η:

$$\mathcal{A} = \{\boldsymbol{a}^{(t)}\}_{t=1}^{T} \sim \eta^{T}(\boldsymbol{a}).$$

This assumption would be natural as long as observations of new domains is independent to past observations.

We then define the (expected) risk, i.e, the error over target domains as

$$R_{\mathrm{U}}(g) := \mathrm{E}_{\eta}\left[\ell\big(g(\boldsymbol{a}), f(\boldsymbol{a})\big)\right].$$

Let us define two minimizers as

$$g^{*} = \operatorname*{argmin}_{g\in\mathcal{G}} R_{\mathrm{U}}(g),$$
$$\widehat{g} = \operatorname*{argmin}_{g\in\mathcal{G}} \widehat{R}_{\mathrm{MC}}(g),$$

where $\mathcal{G}$ is a function class. Note that unlike the standard setting, $\widehat{R}_{\mathrm{MC}}(g)$ is not a sample approximation of $R_{\mathrm{U}}(g)$.

In this section, we investigate the estimation error defined as

$$R_{\mathrm{U}}(\widehat{g}) - R_{\mathrm{U}}(g^{*}).$$

More precisely, let us assume that $|g(\boldsymbol{a}) - f(\boldsymbol{a})| \le M$ for all $g \in \mathcal{G}$ and $\boldsymbol{a} \in \mathbb{R}^{m}$, where $f\colon \mathcal{A} \to \mathbb{R}$ is the labeling function. For ℓ, we consider the ℓ_p loss defined as $\ell_p(y, y') = |y - y'|^{p}$ for $p \ge 1$, which includes the squared loss if $p = 2$. Let us also assume that there exists a constant $C_{\boldsymbol{a}} > 0$ such that $\|\boldsymbol{a}\| \le C_{\boldsymbol{a}}$ for all $\boldsymbol{a} \in \mathbb{R}^{m}$, i.e., the attribute vector is bounded.

Let $\mathcal{G}$ be the function class for prediction models. For example, the function class of the linear model can be expressed as $\mathcal{G} = \{\boldsymbol{w}^{\top}\boldsymbol{a} \mid \boldsymbol{w} \in \mathbb{R}^{m};\ \|\boldsymbol{w}\| \le C_{\boldsymbol{w}}\}$. Let $\mathfrak{R}_{T}(\mathcal{G}) := \mathrm{E}_{\mathcal{A}\sim\eta^{T}}\left[\mathrm{E}_{\boldsymbol{\sigma}}\left[\sup_{g\in\mathcal{G}} \frac{1}{T}\sum_{t=1}^{T} \sigma_t g(\boldsymbol{a}^{(t)})\right]\right]$ be the Rademacher complexity, where $\boldsymbol{\sigma} := (\sigma_1, \ldots, \sigma_T)^{\top}$, with σ_is independent uniform random variables taking values in $\{-1, +1\}$.

We then define the empirical risk as

$$\widehat{R}_{\mathrm{S}}(g) := \frac{1}{T_{\mathrm{S}}}\sum_{t=1}^{T_{\mathrm{S}}} \ell\big(g(\boldsymbol{a}^{(t)}), f(\boldsymbol{a}^{(t)})\big).$$

Additionally, let us define the empirical risk for $\boldsymbol{h}_{\mathrm{S}}$ as

$$\xi_{\mathrm{MTL}}(\boldsymbol{h}_{\mathrm{S}}) := \frac{1}{T_{\mathrm{S}}}\sum_{t=1}^{T_{\mathrm{S}}} \ell\big(h^{(t)}(\boldsymbol{x}'), f(\boldsymbol{a}^{(t)})\big).$$

5.2 Results

First, we have the following lemma:

Lemma 1. *For any $\delta > 0$, we have with probability at least $1-\delta$, the following inequality:*

$$\sup_{g\in\mathcal{G}} \left|R_{\mathrm{U}}(g) - \widehat{R}_{\mathrm{MC}}(g)\right| \le \xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}}) + 2pM^{p-1}\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G}) + M^p\sqrt{\frac{\ln(2/\delta)}{2T_{\mathrm{S}}}}. \quad (4)$$

Proof. From the triangle inequality, we have

$$\ell\big(g(\boldsymbol{a}^{(t)}), f(\boldsymbol{a}^{(t)})\big) \le \ell\big(g(\boldsymbol{a}^{(t)}), \widehat{h}^{(t)}(\boldsymbol{x}')\big) + \ell\big(\widehat{h}^{(t)}(\boldsymbol{x}'), f(\boldsymbol{a}^{(t)})\big)$$

for any $\widehat{h}^{(t)}$ at $\boldsymbol{x}'$. Thus,

$$\widehat{R}_{\mathrm{S}}(g) \le \widehat{R}_{\mathrm{MC}}(g) + \xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}}). \quad (5)$$

Similarly,

$$\widehat{R}_{\mathrm{MC}}(g) \le \widehat{R}_{\mathrm{S}}(g) + \xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}}). \quad (6)$$

Next, on the basis of the standard Rademacher complexity analysis (see, e.g., [19, Theorem 10.3]), for any $\delta > 0$, we have with probability at least $1-\delta/2$, the following inequality for all $g \in \mathcal{G}$:

$$R_{\mathrm{U}}(g) - \widehat{R}_{\mathrm{S}}(g) \le 2pM^{p-1}\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G}) + M^p\sqrt{\frac{\ln(2/\delta)}{2T_{\mathrm{S}}}},$$
$$\widehat{R}_{\mathrm{S}}(g) - R_{\mathrm{U}}(g) \le 2pM^{p-1}\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G}) + M^p\sqrt{\frac{\ln(2/\delta)}{2T_{\mathrm{S}}}}.$$

From Eqs. (5) and (6), we thus have with probability at least $1-\delta/2$, the following inequality for all $g \in \mathcal{G}$:

$$R_{\mathrm{U}}(g) - \widehat{R}_{\mathrm{MC}}(g) \le \xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}}) + 2pM^{p-1}\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G}) + M^p\sqrt{\frac{\ln(2/\delta)}{2T_{\mathrm{S}}}},$$
$$\widehat{R}_{\mathrm{MC}}(g) - R_{\mathrm{U}}(g) \le \xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}}) + 2pM^{p-1}\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G}) + M^p\sqrt{\frac{\ln(2/\delta)}{2T_{\mathrm{S}}}},$$

which concludes the proof. □

On the basis of Lemma 3, we have the following estimation error bound:

Theorem 2. *For any $\delta > 0$, we have with probability at least $1-\delta$, the following inequality:*

$$R_{\mathrm{U}}(\widehat{g}) - R_{\mathrm{U}}(g^*) \le 2\xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}}) + 4pM^{p-1}\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G}) + 2M^p\sqrt{\frac{\ln(2/\delta)}{2T_{\mathrm{S}}}} \quad (7)$$

Proof. By definition of $\widehat{g}$, we have $\widehat{R}_{\mathrm{MC}}(\widehat{g}) \leq \widehat{R}_{\mathrm{MC}}(g^*)$. We then derive the upper bound of the estimation error:

$$\begin{aligned} R_{\mathrm{U}}(\widehat{g}) - R_{\mathrm{U}}(g^*) &\leq R_{\mathrm{U}}(\widehat{g}) - \widehat{R}_{\mathrm{MC}}(\widehat{g}) + \widehat{R}_{\mathrm{MC}}(\widehat{g}) - R_{\mathrm{U}}(g^*) \\ &\leq \big(\sup_{g\in\mathcal{G}} R_{\mathrm{U}}(g) - \widehat{R}_{\mathrm{MC}}(g)\big) + \widehat{R}_{\mathrm{MC}}(g^*) - R_{\mathrm{U}}(g^*) \\ &\leq 2\sup_{g\in\mathcal{G}} \big|R_{\mathrm{U}}(\widehat{g}) - \widehat{R}_{\mathrm{MC}}(\widehat{g})\big|. \end{aligned}$$

Combining the above with Lemma 1, we obtain the theorem. □

For $\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G})$, the Rademacher complexity for various models are known to be bounded [1, 19]. To observe the effect of the Rademacher complexity, let us assume the linear model as the function class:[1]

$$\mathcal{G} = \{\boldsymbol{w}^\top \boldsymbol{a} \mid \boldsymbol{w} \in \mathbb{R}^m;\ \|\boldsymbol{w}\| \leq C_{\boldsymbol{w}}\}.$$

From Theorem 2, we then have the following corollary:

Corollary 3. *Assume that the linear model is used as $\mathcal{G}$. For any $\delta > 0$, we have with probability at least $1-\delta$, the following inequality:*

$$R_{\mathrm{U}}(\widehat{g}) - R_{\mathrm{U}}(g^*) \leq 2\xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}}) + \frac{C_{\boldsymbol{a},\boldsymbol{w},\delta}}{\sqrt{T_{\mathrm{S}}}}, \tag{8}$$

where $C_{\boldsymbol{a},\boldsymbol{w},\delta} := 4pM^{p-1}C_{\boldsymbol{a}}C_{\boldsymbol{w}} + M^p\sqrt{2\ln(2/\delta)}$.

Proof. Since the linear models are used for $\mathcal{G}$, we can prove (see [19, Theorem 4.3] for details)

$$\mathfrak{R}_{T_{\mathrm{S}}}(\mathcal{G}) \leq \frac{C_{\boldsymbol{a}}C_{\boldsymbol{w}}}{\sqrt{T_{\mathrm{S}}}}.$$

Plugging the above equation into Eq. (7) and defining $C_{\boldsymbol{a},\boldsymbol{w},\delta} := 4pM^{p-1}C_{\boldsymbol{a}}C_{\boldsymbol{w}} + M^p\sqrt{2\ln(2/\delta)}$, we conclude the proof. □

Corollary 3 shows that the second term on the right-hand side in Eq. (8) decreases in $\mathcal{O}_p(1/\sqrt{T_{\mathrm{S}}})$, meaning that the term becomes small if the number of seen domains T_{S} is large.[2] If we have accurate source hypotheses $\widehat{\boldsymbol{h}}_{\mathrm{S}}$, $\xi_{\mathrm{MTL}}(\widehat{\boldsymbol{h}}_{\mathrm{S}})$ will be also small. Thus, minimizing $\widehat{R}_{\mathrm{MC}}$ leads to a smaller estimation error in Eq. (8).

It should be noted that in a standard supervised learning setting, the estimation error converges with $\mathcal{O}_p(1/\sqrt{N})$, where N is the number of training samples. The convergence rate is known as optimal under certain mild conditions in empirical risk minimization [18]. Since our analysis is also based on a tool for empirical risk minimization, the connection indicates that $\mathcal{O}_p(1/\sqrt{T_{\mathrm{S}}})$ convergence derived from our analysis is optimal under mild conditions.

[1] The linear-in-parameter model and kernel model can be handled in a similar manner.
[2] $\mathcal{O}_p$ denotes the order in probability.

Table 3. Statistics of datasets. T denotes number of domains.

Dataset	n	d	m	T
Synth (T)	$100T$	10	20	T
Coffee	1,161	63	36	23
School	4,593	21	6	23
Book	7,282	64	77	169
Wine	32,906	751	8	75
Sushi	50,000	31	15	100

6 Experiments

We evaluated our proposed method on various synthetic and benchmark datasets. We used a PC equipped with Intel Xeon Gold 6142 and NVIDIA Quadro RTX 5000.

6.1 Datasets

Synthetic datasets: We generated a synthetic dataset, called Synth (T). By varying the number of seen domains T, we confirmed the performance change of ZSDA methods in terms of T. We generated Synth (T) on the basis of the following procedure:

1. Prepare the Gaussian basis functions $\{\phi_\ell(\boldsymbol{x}) = \exp(-\|\boldsymbol{x} - \boldsymbol{x}_\ell\|^2)\}_{\ell=1}^{b}$;
2. Create b-dimensional parameter vector, $\boldsymbol{w}^{(t)} \in \mathbb{R}^b$, each element of which is drawn from the standard normal distribution $\mathcal{N}(0, 1^2)$;
3. Make an m-dimensional attribute vector by $\boldsymbol{a}^{(t)} = \boldsymbol{Q}\boldsymbol{w}^{(t)}$, where $\boldsymbol{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_b) \in \mathbb{R}^{m\times b}$ and $\{\boldsymbol{q}_i \in \mathbb{R}^m\}_{i=1}^{b}$ is the set of m-dimensional orthonormal vectors;
4. Generate feature vectors, each element of which is drawn from the uniform distribution $\mathcal{U}(0, 1)$;
5. Observe the paired samples of size $n^{(t)}$ from $y^{(t)} = \boldsymbol{w}^{(t)\top}\boldsymbol{\phi}(\boldsymbol{x}) + 0.1\varepsilon$, where ε is drawn from the standard normal distribution.

We set $d = 10$, $m = 20$, $b = 20$, and $n^{(t)} = 100$ (thus, $n = 100T$).

Benchmark datasets: We used the goodbooks-10k (Book), coffee quality (Coffee), School [7], SUSHI preference [11] (Sushi), and wine reviews (Wine) datasets as benchmark datasets.[3] Table 3 summarizes the statistics of the datasets used in our experiments.

The Coffee dataset consists of reviews of the coffee beans for several farms. We used "Country of Origin", "Certification Body", and "Altitude" as features of

[3] Book: https://github.com/zygmuntz/goodbooks-10k. Coffee: https://github.com/jldbc/coffee-quality-database. Sushi: http://www.kamishima.net/sushi/. Wine: https://www.kaggle.com/zynicide/wine-reviews.

farms and “Species”, “Processing Method”, and “Variety” as domain attributes of the coffee beans. The “Total Cup Points” was used as the score. We removed coffee beans that received less than ten reviews.

The Book dataset is a collection of book ratings from readers. We used “Age” and “Country“ as features of readers and the tags of books annotated by users in the book-rating platform as domain attributes. We manually extracted book tags that are likely to be relevant to rating

The School dataset contains examination scores of 15,362 students from 139 schools. Similarly to Yang and Hospedales [25], we chose the school in which each year group had more than 50 students. After preprocessing, we have 4,593 samples and 23 domains. We used the school gender (Mixed, Male, and Female) and school denomination (Maintained, Church of England, and Roman Catholic) as domain attributes.

In the Wine dataset, we had 32,906 wine reviews after preprocessing. We used “Variety”, “Country”, and “Price” as features, and color and taste information extracted from the description as domain attributes.

The SUSHI dataset consists of ratings of 100 types of sushi from 5,000 people. We used the information of users as features and the information of sushi as domain attributes.

6.2 Setting

We used the 8:2 domain-wise dataset split to create data for seen and unseen domains. We left the data for unseen domains for evaluation of our method’s performance. We further split the data for seen domains into 80% training and 20% test data.

For our proposed method, we used the linear ridge regression (Ridge) [8] as the fixed-input predictor g. As the trained predictor for a seen domain $\widehat{h}^{(t)}$, we used two methods: Ridge and LightGBM [12]. The former is denoted as Ridge+MC and the latter as LGBM+MC.

As a baseline, we used MDMT [25]. For the attribute-aware predictor F, we used $F(\boldsymbol{x}, t) = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{W} \boldsymbol{a}^{(t)} + c$, where $\boldsymbol{W} \in \mathbb{R}^{r \times m}$, $c \in \mathbb{R}$ is the intercept and $\boldsymbol{\phi} \colon \mathbb{R}^d \to \mathbb{R}^r$ is a feature transformation.

We prepared two architectures, i.e., MDMT1 and MDMT2. For MDMT1, $\boldsymbol{\phi}$ is the identity transformation and $r = d$, meaning that F is a bilinear function, and $\boldsymbol{W}$ and c are the parameters to be learned. For MDMT2, $\boldsymbol{\phi}$ is the two-layer neural network consisting of a $d \times r$ linear layer, batch normalization [9], and rectified linear unit (ReLU) activation [6]. We thus learn $\boldsymbol{W}$, c, and $\boldsymbol{\phi}$ in MDMT2. For both MDMT1 and MDMT2, we used Adam optimizer [13] to solve the optimization problem in Eq. (2). The number of epochs was set to 500.

Table 4. Average and standard error of relative mRMSE (α/β) over 20 trials, where α and β are mRMSE$_{\text{U}}$ of proposed and MDMT, respectively. When α/β was less than 1, our proposed method was more accurate than MDMT. Even though our method does not use source domain data, its performance was often comparable or sometimes superior to that of MDMT.

Dataset	Ridge+MC vs		LGBM+MC vs	
	MDMT1	MDMT2	MDMT1	MDMT2
Synth (50)	1.01 ±0.03	1.00 ±0.04	1.00 ±0.03	0.99 ±0.04
Synth (100)	0.91 ±0.02	0.96 ±0.04	1.01 ±0.02	1.07 ±0.04
Coffee	0.06 ±0.01	0.10 ±0.00	0.06 ±0.01	0.10 ±0.01
School	0.51 ±0.07	0.73 ±0.03	0.49 ±0.06	0.72 ±0.03
Book	0.98 ±0.00	0.92 ±0.01	0.99 ±0.00	0.92 ±0.00
Wine	1.00 ±0.01	0.96 ±0.02	0.93 ±0.01	0.89 ±0.02
Sushi	0.98 ±0.00	0.83 ±0.01	0.98 ±0.00	0.83 ±0.01

6.3 Evaluation Measure

To evaluate the performance of our proposed method, we first defined the mean of the *root-mean-square error* (mRMSE) over *unseen* domains:

$$\text{mRMSE}_{\text{U}} := \frac{1}{T_{\text{U}}} \sum_{t \in \mathcal{T}_{\text{U}}} \sqrt{\text{MSE}(t)},$$

where $\text{MSE}(t) := (1/n^{(t)}) \sum_{i=1}^{n^{(t)}} (y_i^{(t)} - \widehat{y}_i^{(t)})^2$ and $\widehat{y}_i^{(t)}$ is the prediction of a test sample $\boldsymbol{x}_i^{(t)}$ in t.

To measure the performance of our proposed method, we used *relative* mRMSE. Specifically, let α and β be the mRMSE$_{\text{U}}$ of our method and MDMT, respectively. We reported α/β (smaller is better), the relative mRMSE. When α/β was close to one, our method performed comparably with the baseline that can access source domain data.

6.4 Results

Theory and Practice: We first show the mRMSE$_{\text{U}}$ of the proposed method as a function of the number of seen domains T_{S} on Synth (T_{S}). In the theoretical analysis discussed in Section 5, we proved that the generalization error in terms of domains converges at the rate of $\mathcal{O}(1/\sqrt{T_{\text{S}}})$. This theoretical result indicates that the expected error over unseen domains decreases with the number of seen domains.

Figure 2 shows the mRMSE$_{\text{U}}$ of our method and the curve of $1/\sqrt{T_{\text{S}}}$ plus a constant.[4] The results indicate that the mRMSE$_{\text{U}}$ of our method decreased

[4] The constant is calculated such that the value of the curve at $T_{\text{S}} = 200$ is equivalent to that of Ridge+MC.

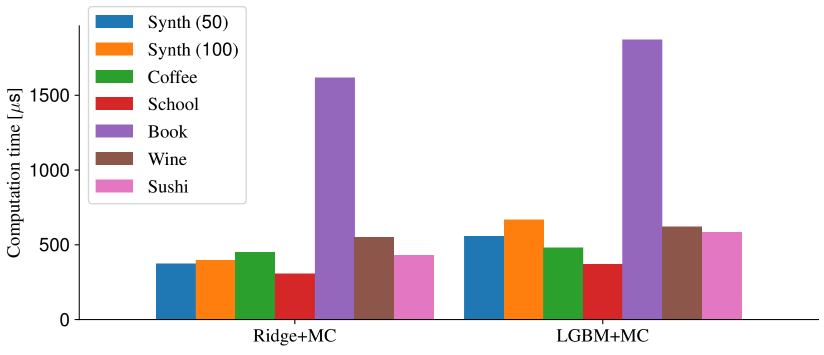

Fig. 1. Average computation time of Ridge+MC and LGBM+MC over 10 trials.

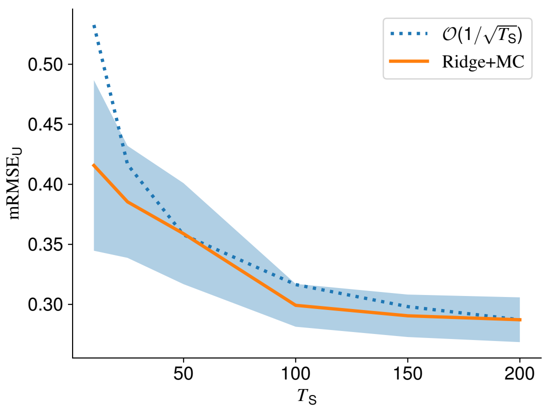

Fig. 2. Average and standard deviation of mRMSE$_U$ of Ridge+MC over 100 trials. Curve $\mathcal{O}(1/\sqrt{T_S})$, derived from our theoretical analysis, is $1/\sqrt{T_S}$ plus constant. mRMSE$_U$ of our method decreased as T_S increased and shape was similar to $\mathcal{O}(1/\sqrt{T_S})$.

as the number of seen domains increased. Compared with the curve $\mathcal{O}(1/\sqrt{T_S})$, the mRMSE$_U$ of our method behaved similarly, indicating that our theoretical analysis can be a guideline on how many seen domains are necessary to obtain a certain performance improvement.

Prediction Performance: Table 4 lists the average and standard errors of relative mRMSE (α/β) over 20 trials, where α and β are the mRMSE$_U$ of the proposed method and MDMT, respectively. For example, α is the mRMSE$_U$ of Ridge+MC and β is that of MDMT1. When α/β was less than 1, our proposed method was more accurate than MDMT. Table 4 shows that even though our method does not use source domain data, its performance was often comparable or sometimes superior to that of MDMT. We thus conclude that HTL for ZSDA is possible with our method.

Computation Time: Since the training time of our method is zero (see Section 4), we are thus interested in the inference time. Figure 1 shows the average inference times (in microseconds). The inference time of our method was less than 1 ms (i.e., 1000 μs) except for the Book dataset. Since the total number of domains of the Book dataset is 169, which is slightly larger than the other datasets, the computation time of our method took slightly longer. This is because the computational complexity of our method depends on the number of seen domains, as discussed in Section 4.3.

In summary, even if our method requires a certain amount of computation time in an inference phase, it is computationally efficient.

7 Conclusions

We proposed a hypotheses transfer method for zero-shot domain adaptation that can work with source hypotheses without accessing source domain data. We argued that our method can be easily implemented with Scikit-learn in Python. When linear models are used, we can make predictions very efficiently, as confirmed from both computational complexity analysis and experiments. We investigated the estimation error bound of our proposed method and revealed that it is theoretically valid. Finally, through numerical experiments, we demonstrated the effectiveness of our proposed method.

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

References

1. Bartlett, P.L., Mendelson, S.: Rademacher and Gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research **3**, 463–482 (2002)
2. Blanchard, G., Lee, G., Scott, C.: Generalizing from several related classification tasks to a new unlabeled sample. In: Advances in Neural Information Processing Systems. pp. 2178–2186 (2011)
3. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. The MIT Press (2006)
4. Chidlovskii, B., Clinchant, S., Csurka, G.: Domain adaptation in the absence of source domain data. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 451–460 (2016)
5. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2004)
6. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 315–323 (2011)
7. Goldstein, H.: Multilevel modelling of survey data. Journal of the Royal Statistical Society. Series D (The Statistician) (1991)

8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer-Verlag, New York (2009)
9. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning. vol. 37, pp. 448–456 (2015)
10. Ishii, M., Takenouchi, T., Sugiyama, M.: Zero-shot domain adaptation based on attribute information. In: Proceedings of The Eleventh Asian Conference on Machine Learning. pp. 473–488 (2019)
11. Kamishima, T., Akaho, S.: Efficient clustering for orders. Mining Complex Data pp. 261–279 (2009)
12. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30. pp. 3146–3154 (2017)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 32nd International Conference on Machine Learning (2015)
14. Kuzborskij, I., Orabona, F.: Stability and hypothesis transfer learning. In: Proceedings of the 30th International Conference on Machine Learning. pp. 942–950 (2013)
15. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5542–5550 (2017)
16. Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In: Proceedings of the 37th International Conference on Machine Learning. pp. 6028–6039 (2020)
17. Mansour, Y., Mohri, M., Rostamizadeh, A.: Domain adaptation with multiple sources. In: Advances in Neural Information Processing Systems. pp. 1041–1048 (2009)
18. Mendelson, S.: Lower bounds for the empirical minimization algorithm. IEEE Transactions on Information Theory **54**(8), 3797–3803 (2008)
19. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of Machine Learning. MIT Press (2012)
20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research (2011)
21. Peng, K.C., Wu, Z., Ernst, J.: Zero-shot deep domain adaptation. In: Proceedings of the European Conference on Computer Vision. pp. 764–781 (2018)
22. Romera-Paredes, B., Torr, P.: An embarrassingly simple approach to zero-shot learning. In: Proceedings of the 32nd International Conference on Machine Learning (2015)
23. Sakai, T., Ohasaka, N.: Predictive optimization with zero-shot domain adaptation. In: Proceedings of the 2021 SIAM International Conference on Data Mining. pp. 369–377 (2021)
24. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag New York (1995)
25. Yang, Y., Hospedales, T.M.: A unified perspective on multi-domain and multi-task learning. In: Proceedings of 3rd International Conference on Learning Representations (2015)
26. Yang, Y., Hospedales, T.M.: Zero-shot domain adaptation via kernel regression on the Grassmannian. In: International Workshop on Differential Geometry in Computer Vision for Analysis of Shapes, Images and Trajectories (2015)