Transfer learning with model transformation

Shunya Minami

Department of Statistical Science
School of Multidisciplinary Sciences
The Graduate University for Advanced Studies, SOKENDAI

This dissertation is submitted for the degree of
Doctor of Philosophy

March 2023

Abstract

The effective training of statistical models with a limited sample size is critical for the success of statistics and machine learning in practical applications. A methodology called transfer learning has been actively studied to address this challenge. Transfer learning imitates the learning process of human intelligence, where predictions are made with little experience by reusing the knowledge gained in the past. While end-to-end model development with insufficient data is challenging, models from related domains, which are trained with sufficient data, can be adapted to compensate for the data inadequacy. Transfer learning has been applied in various fields, such as image recognition, natural language processing, and materials science, as an effective solution to the small data problem.

This thesis focuses on supervised transfer learning in regression tasks and proposes frameworks to estimate the shift from the source domain to the target domain. Based on statistical approaches, general frameworks are developed that encompass some existing procedures such as transfer learning based on Bayesian inference, density-ratio estimation, neural networks, and variable transformation. Furthermore, theoretical analyses are conducted to clarify some characteristic features, including the preference of the hyperparameters and learning efficiency of the developed framework. We demonstrate the practical benefits of the generalized frameworks through several case studies.

Acknowledgements

I am grateful to everyone who has supported me during my Ph.D. course. First, I would like to express my gratitude to my supervisor, Prof. Ryo Yoshida. In addition to guiding me in conducting my research, he has been a role model to me as a professional researcher. I feel the time spent under his guidance worthwhile for my future career as a researcher.

In addition, I would like to thank my sub-supervisor, Prof. Stephen Wu. His kindness and attentiveness were instrumental in ensuring the success of my research. I would also like to express my gratitude to Prof. Kenji Fukumizu. He was an excellent collaborator during my research. I learned extensively from his valuable and insightful feedback.

I would like to offer my special thanks to Prof. Song Liu. He not only advised me as a collaborator but also facilitated my stay for nearly a month at the University of Bristol. My time in Bristol was very stimulating from an academic perspective.

I am indebted to the members of my dissertation committee, Prof. Hironori Fujisawa, Prof. Hideitsu Hino, and Prof. Kota Matsui, who provided valuable feedback on my doctoral thesis.

I am deeply grateful to the members of Yoshida Lab, my colleagues, and the staff at The Institute of Statistical Mathematics and The Graduate University for Advanced Studies (SOKENDAI) for their support and encouragement at various times. My deepest gratitude goes to my family for their unwavering support and understanding throughout my Ph.D. course.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is used to recognize patterns in data. Although it has achieved great success in a wide range of fields, its effectiveness depends on the availability of a huge amount of data. For example, BERT (Devlin et al., 2019), which recorded the highest accuracy for various natural language processing tasks at the time of its development, used data sets containing more than three billion words for training. An image classification model is generally trained on more than ten million images (Deng et al., 2009; Thomee et al., 2016). Thus, it is essential to collect a huge amount of training data to obtain a well-performing machine learning model.

However, in many real-world problems, it is challenging to collect a sufficient amount of data. For example, in scientific fields such as materials science, we can obtain extremely limited amounts of data owing to several obstacles, e.g., cost of data collection, diverse demands of researchers, and constraints related to strict information confidentiality. For further progress of machine learning, it is necessary to overcome the bottleneck of limited data supply.

Transfer learning (TL) (Pan & Yang, 2009; Yang et al., 2020) has received considerable attention as a methodology for solving the small data problem. TL refers to the problem of applying the knowledge learned in one or more tasks to efficiently develop an effective model for a new task (Silver et al., 2005), To obtain a well-performing model in a new domain, called the *target domain*, we reuse the knowledge such as parameters of the models, encoded features, hyperparameters, and samples from the related domains, called the *source domains*.

Although concepts such as TL have been extensively studied, the potential of TL began to be noticed after the success of deep learning. By training a deep neural network model using large amounts of data, we can obtain a high-performing model that can occasionally outperform human capability. Such models are trained specifically for a particular prediction task, which begets a discussion of their performance on other similar prediction tasks (Donahue et al., 2014; Agrawal et al., 2014). Various

frameworks and theories of TL have been studied. In practical applications, TL has been shown to be an effective technique for overcoming the small data problem, not only in computer vision (Krizhevsky et al., 2012; Csurka, 2017) and natural language processing (Ruder et al., 2019; Devlin et al., 2019) but also in various other scientific fields, such as materials science (Yamada et al., 2019; Wu et al., 2019; Ju et al., 2021) and biology (Sevakula et al., 2019).

The success of TL depends on the specific knowledge to be transferred. Typically, the parameters or intermediate representations of pre-trained source models are reused (Yosinski et al., 2014). These procedures are called *parameter transfer* and *feature extraction*, respectively. Parameter transfer uses the parameters of the pre-trained model as the initial values for training for a target task. In Bayesian statistics, this approach corresponds to adopting the source parameters into a prior distribution. Feature extraction uses the features encoded in the intermediate layers of the source model as inputs for a target task. Hypothesis transfer learning (HTL) is a special case of feature extraction, wherein the output of the source model is used to predict the domain shift (Kuzborskij & Orabona, 2013; 2017; Du et al., 2017).

Problem settings and our contributions

This thesis focuses on supervised TL in regression problems, in particular, the problem of reusing features obtained from the training in the source domain. We assume that the output of the target domain $y \in \mathcal{Y} \subset \mathbb{R}$ follows $y = f_t(x) + \epsilon$, where $f_t : \mathcal{X} \to \mathbb{R}$ is the true model on the target domain, and the observation noise ϵ has mean zero and variance σ^2. We are given n labeled samples $\{(x_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$ from the target domain and the feature representation $f_s(x) \in \mathcal{F}_s$ from one or more source domains. For example, TL based on feature extraction uses the intermediate representation of a neural network as f_s, whereas HTL uses the output of the source model.

There are several benefits of focusing on the reuse of source features, among which the most notable is applicability. In general, source knowledge is not always available as statistical models, such as neural networks, but may be as physical models, Moreover, source knowledge may not be explicitly represented as a function of the input x. In such cases, approaches relying on a particular statistical model, such as parameter transfer, cannot be used, although feature-based approaches can be used. Another benefit is computational feasibility. In parameter transfer, all parameters of the source model must be stored and the entire model must be retrained. When transferring from extremely large models, such as state-of-the-art neural network models with more than a billion parameters, the computational cost becomes enormous. In contrast, in the case of feature extraction, the computational cost is relatively low because the feature representation is computed only once in advance. For the widespread application of TL, it is essential to develop tractable TL methods such as

feature extraction.

After reviewing several approaches and theoretical results of TL in Chapter 2, this thesis proceeds as follows.

Chapter 3 In Chapter 3, we present a new class of TL that is applicable to any regression model. The proposed class unifies different classes of existing TL methods for regression. To model the shift from a pre-trained source model to a target model, we introduce a density-ratio reweighting function. The density-ratio function is estimated by applying Bayesian inference with a specific prior distribution using a fixed source model. Two hyperparameters and the model for the density-ratio function characterize the proposed class. It can integrate and extend three popular methods of TL within a unified framework, including TL based on cross-domain similarity regularization (Marx et al., 2005; Raina et al., 2006; Kuzborskij & Orabona, 2013; 2017; Jalem et al., 2018), probabilistic TL using density-ratio estimation (Sugiyama et al., 2012; Liu & Fukumizu, 2016), and fine-tuning of pre-trained neural networks (Yosinski et al., 2014; Hinton et al., 2015; Kirkpatrick et al., 2017).

In general, the model transfer operates through a regularization scheme to leverage the transferred knowledge between different tasks. Conventional regularization aims to retain the similarity between the pre-trained and transferred models. This natural idea is referred to as cross-domain similarity regularization. The density-ratio method operates with an opposite learning objective, which we call cross-domain dissimilarity regularization; the discrepancy between two tasks is modeled and inferred, and the transferred model is given as a weighted sum of the pre-trained source model and the newly trained model. These entirely different approaches can be unified within the proposed framework.

To summarize, the features and contributions of Chapter 3 are as follows:

- We propose a novel framework for TL by introducing a density-ratio reweighting function, which is estimated using a Bayesian framework with a specific prior distribution.
- This framework can be applied to any regression model.
- The proposed class, which has two hyperparameters, can unify and hybridize three existing methods of TL, including regularization based on cross-domain similarity and dissimilarity.
- The two hyperparameters and a model for the density-ratio function are selected through cross-validation. With this unified workflow, ordinary supervised learning without transfer can also be selected if the previous learning experience interferes with the learning of the new task.

- The proposed framework can be implemented at no extra cost. With a simple transformation of the output variable, the model can be trained using off-the-shelf libraries for regression, which can implement the squared loss minimization with any regularization scheme. In addition, the method is applicable to scenarios where only the source model is accessible but not the source data, for example, owing to privacy reasons.

The practical benefits of bridging entirely different methods in the unified workflow are demonstrated for a wide range of prediction tasks in science and engineering applications.

Chapter 4 Chapter 4 develops a TL methodology to estimate the cross-domain shifts and domain-specific factors simultaneously and separately by using the given target samples. Based on the HTL procedure, we consider two different transformation functions: one to represent and estimate the domain-specific factors and the other to adapt them to the target domain in combination with the source features. For these transformation functions, we derive the theoretically optimal transformation function class based on the assumptions of invertibility and differentiability as well as consistency, i.e., the optimal predictor does not change during the two transformations. The resulting function class takes the form of an affine coupling $g_1 + g_2 \cdot g_3$ of three functions g_1, g_2, and g_3, where the cross-domain shift is represented by the functions g_1 and g_2, and the domain-specific factors are represented by g_3. These functions can be estimated simultaneously using conventional supervised learning algorithms such as kernel methods or deep neural networks. We refer to this framework as *affine model transfer*.

The affine coupling used in the affine model transfer is the basic model architecture of invertible neural networks and is widely used in several fields, including generative modeling (Dinh et al., 2014; 2017; Kingma & Dhariwal, 2018; Papamakarios et al., 2021). Neural networks based on affine coupling layers have been proven to have universal approximation ability (Teshima et al., 2020), which means that the proposed model class has the potential to represent a broad class of transformation functions.

Furthermore, when we use the intermediate layers of a source neural network as feature representations in the target domain, the affine model transfer is identical to ordinary TL based on feature extraction. We can formulate a wide variety of TL algorithms within the affine model transfer, including neural transfer as a special case.

To summarize, the contributions of Chapter 4 are as follows:

- The affine model transfer is proposed to adapt the source features to the target domain by separately estimating the cross-domain shifts and domain-specific factors.

- Several existing methods of HTL are encompassed in the affine model transfer, including neural network-based TL.
- The affine model transfer is compatible with any type of source model. For example, non-machine learning models, such as physical models, can be used. It can also handle multiple source models without loss of generality.
- For each of the three functions g_1, g_2, and g_3, we provide an efficient and stable estimation algorithm through modeling using the kernel method.
- Two theoretical properties of the affine model transfer are shown: generalization bound and excess risk bound.

Based on several applications, we compare the affine model transfer with other TL algorithms, discuss its strengths and weaknesses, and demonstrate the advantage of the ability to estimate the cross-domain shifts and domain-specific factors separately.

Chapter 3 is based on the following conference paper (Minami et al., 2021):

- Shunya Minami, Song Liu, Stephen Wu, Kenji Fukumizu, and Ryo Yoshida. A general class of transfer learning regression without implementation cost. *Proceedings of AAAI Conference on Artificial Intelligence*, 35:8992–8999, 2021.

Chapter 4 is based on the following paper (Minami et al., 2022):

- Shunya Minami, Kenji Fukumizu, Yoshihiro Hayashi, and Ryo Yoshida. Transfer learning with affine model transformation. *arXiv*, abs/2210.09745, 2022.

Chapter 2

Transfer learning

In this chapter, we review several approaches and theoretical results of TL.

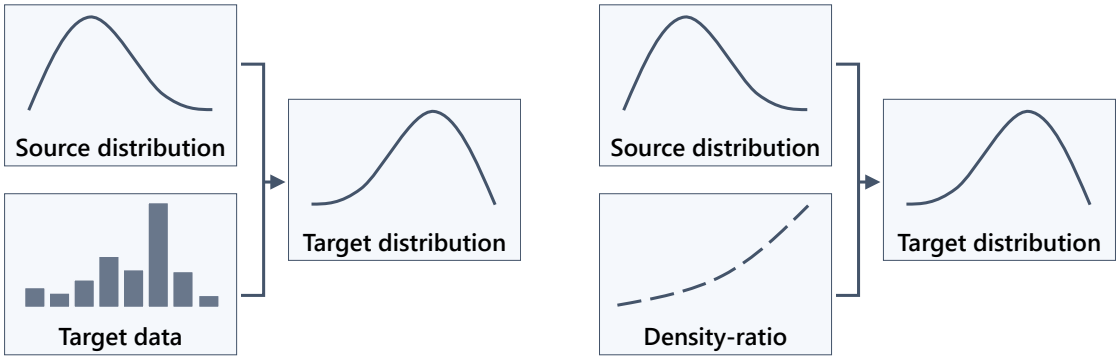
TL has been traditionally classified into two broad categories—supervised TL and unsupervised TL— depending on the availability of the labels of the target samples (Pan & Yang, 2009).

Definition 1 (Supervised transfer learning)**.** Given some source knowledge, supervised transfer learning aims to improve the performance of the predictive model in the target domain by using the source knowledge and labeled data $\{(x_i, y_i)\}_{i=1}^n$ in the target domain.

Definition 2 (Unsupervised transfer learning)**.** Given some source knowledge, unsupervised transfer learning aims to improve the performance of the predictive model in the target domain by using the source knowledge and unlabeled data $\{x_i\}_{i=1}^n$ in the target domain.

Examples of source knowledge in these definitions are samples in the source domain (*instance transfer*), parameters of pre-trained models (*parameter transfer*), and features obtained from training in the source domain (*feature representation transfer*). In Pan & Yang (2009), supervised instance transfer is further classified into two categories. The case where the labels of the source samples are available is referred to as multi-task learning (Caruana, 1997; Zhang & Yang, 2021), and the case where they are not available is referred to as self-taught learning (Raina et al., 2007).

This thesis mainly focuses on supervised TL, and this chapter provides a focused review of its methods and theoretical results. In addition, a few key concepts related to unsupervised TL are introduced.



(a) TL based on Bayesian inference (b) TL based on density-ratio estimation

Figure 2.1: Schematics of two probabilistic transfer learning approaches. (a) In transfer learning based on Bayesian inference, the source knowledge is used in the prior distribution and transferred according to Bayes' theorem with the likelihood using the target samples. (b) Based on density-ratio estimation, the shift from the source distribution is adjusted by estimating the ratio of the densities and weighting the source distribution.

2.1 Methods

We begin with a review of some supervised TL frameworks. The typical approaches in this context are introduced from two perspectives: probabilistic modeling and non-probabilistic modeling.

2.1.1 Probabilistic supervised transfer learning

From the probabilistic viewpoint, the goal of TL is to predict the conditional distribution $p_t(y|x)$ of the target domain by using the conditional distribution $p_s(y|x)$ of the source domain. Therefore, the TL problem reduces to the problem of estimating the shift of the distribution from p_s to p_t.

Bayesian inference

Bayesian inference is one of the most common approaches to adjust the shift of the two distributions. By focusing on the similarity between the source and target domains, the probability distribution is updated using the collected sample set $\mathcal{D}_t$ of the target domain as follows:

$$p(\theta_t|\mathcal{D}_t) \propto p(\mathcal{D}_t|\theta_t)p(\theta_t),$$

where θ_t is the parameter of the target model.

To obtain a point estimate of the parameter θ_t, a maximum a posteriori probability (MAP) estimation is conducted.

$$\hat{\theta}_t = \arg\max_{\theta_t} p(\theta_t|\mathcal{D}_t). \tag{2.1}$$

Given the specific form of the likelihood function and prior distribution, the objective function to be minimized is derived. Typically, for regression problems, the Gaussian distribution is imposed on the likelihood as

$$p(\mathcal{D}_t|\theta_t) \propto \prod_{i=1}^{n} \exp\left\{ -\frac{(y_i - f_t(x_i;\theta_t))^2}{\sigma_1} \right\}, \tag{2.2}$$

where f_t is the target model with a parameter θ_t and $\sigma_1 > 0$ is a variance parameter for the Gaussian distribution.

For the prior, the Gaussian distribution, which is the conjugate prior of the Gaussian distribution, is commonly used, and various methods have been proposed depending on the place of application of the source knowledge in the prior distribution—in the mean parameter (Daumé & Marcu, 2006; Daumé, 2007; Finkel & Manning, 2009), covariance matrix (Raina et al., 2006), or both (Gönen & Margolin, 2014; Shwartz-Ziv et al., 2022). Here, we refer to the use of a Gaussian distribution with the source parameter θ_s as its mean.

$$p(\theta_t) \propto \exp\left\{ -\frac{\|\theta_t - \theta_s\|_2^2}{\sigma_2} \right\}, \tag{2.3}$$

where $\sigma_2 > 0$ is a variance parameter. By substituting Eq. (2.2) and Eq. (2.3) into Eq. (2.1), the optimization problem can be expressed as

$$\hat{\theta}_t = \arg\min_{\theta_t} \sum_{i=1}^{n} \{y_i - f_t(x_i;\theta_t)\}^2 + \lambda_1 \|\theta_t - \theta_s\|_2^2, \tag{2.4}$$

where $\lambda_1 := \sigma_1/\sigma_2$. This objective function is regarded as an ordinary least squares with ℓ_2-regularization. This regularization aims to facilitate the estimation of the target parameter θ_t in the neighborhood of the source parameter θ_s. Because the Gaussian distribution is used for the prior distribution, a variant of ridge regression is derived. When the Laplace distribution is used, a lasso-like regularization is derived by the same procedure (Takada & Fujisawa, 2020).

Eq. (2.4) imposes the same weight on each element of the parameters as the regularization penalty. This assumes that each coefficient changes by the same amount through a domain shift, which is very severe. Elastic weight consolidation (Kirkpatrick et al., 2017) uses the Fisher information matrix to weigh each dimension to capture

the parameter changes correctly. This is expected to prevent the excessive loss of knowledge in the source domain, called catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999).

Another example of the prior distribution is a Gaussian distribution using the output of the source model $f_s(x;\theta_s)$ as its mean.

$$p(\theta_t) \propto \prod_{i=1}^{n} \exp\left\{ -\frac{(f_t(x_i;\theta_t) - f_s(x_i;\theta_s))^2}{\sigma_3} \right\}, \tag{2.5}$$

where $\sigma_3 > 0$ is a variance parameter. Eq. (2.3) focuses on the similarity between the parameters, whereas Eq. (2.5) focuses on the similarity between the outputs. By using Eq. (2.2) and Eq. (2.5), we obtain another optimization problem as follows:

$$\hat{\theta}_t = \arg\min_{\theta_t} \sum_{i=1}^{n} \{y_i - f_t(x_i;\theta_t)\}^2 + \lambda_2 \sum_{i=1}^{n} \{f_s(x_i;\theta_s) - f_t(x_i;\theta_t)\}^2, \tag{2.6}$$

where $\lambda_2 := \sigma_1/\sigma_3$.

By completing the square, the optimization problem Eq. (2.6) is reduced to

$$\hat{\theta}_t = \arg\max_{\theta_t} \sum_{i=1}^{n} \left\{ \frac{y_i + \lambda_2 f_s(x_i;\theta_s)}{1+\lambda_2} - f_t(x_i;\theta_t) \right\}^2. \tag{2.7}$$

This means that we only need to solve the ordinary least squares for the transformed variable $z_i = (y_i + \lambda_2 f_s(x_i;\theta_s))/(1+\lambda_2)$.

Density-ratio estimation

Another probabilistic approach of TL is based on density-ratio estimation (Sugiyama et al., 2012). This approach is often used in conjunction with the covariate shift assumption (Shimodaira, 2000), i.e., the conditional distribution $p(y|x)$ does not change in the source and target domains but only the marginal distribution $p(x)$ shifts. The importance weighting of the log-likelihood is performed by estimating the ratio of the marginal distributions $p_t(x)/p_s(x)$ (Sugiyama et al., 2007b; 2008; Yamada et al., 2013; Lu et al., 2021). These methods are often used in unsupervised TL settings. The details are presented in Section 2.1.3.

Further, some other approaches related to supervised TL settings consider the ratio of the conditional distributions (Liu & Fukumizu, 2016). By using the source conditional distribution $p_s(y|x)$, the target conditional distribution $p_t(y|x)$ can be decomposed as

$$p_t(y|x) = \frac{p_t(y|x)}{p_s(y|x)} \cdot p_s(y|x) = w(y,x) p_s(y|x),$$

where $w(y, x) := p_t(y|x)/p_s(y|x)$. The pre-trained model $p_s(y|x; \theta_s)$ is adopted for the source distribution, and the density-ratio is modeled as $w(y, x; \theta_w)$ and estimated using the target samples. Note that because the product $w(y, x; \theta_w)p_s(y|x; \theta_s)$ is a probability density function, the integral over the entire space should be equal to 1.

$$\int w(u, x; \theta_w)p_s(u|x; \theta_s)\mathrm{d}u = 1.$$

The density-ratio TL of Liu & Fukumizu (2016) is designed to minimize the conditional Kullback–Leibler divergence between the true density $p_t(y|x)$ and density-ratio model $w(y, x; \theta_w)p_s(y|x; \theta_s)$, as follows:

$$\begin{aligned}
&\mathbb{E}_{p_t(x)}\left[\mathrm{KL}(p_t(y|x)||w(y, x; \theta_w)p_s(y|x; \theta_s))\right] \\
&\qquad = \mathbb{E}_{p_t(x)}\left[\mathrm{KL}\left(p_t(y|x)\middle|\middle|\frac{w(y, x; \theta_w)p_s(y|x; \theta_s)}{\int w(u, x; \theta_w)p_s(u|x; \theta_s)\mathrm{d}u}\right)\right] \\
&\qquad = -\int p_t(x)\int p_t(y|x)\log w(y, x; \theta_w)\mathrm{d}y\mathrm{d}x \\
&\qquad\qquad + \int p_t(x)\log\int w(u, x; \theta_w)p_s(u|x; \theta_s)\mathrm{d}u\mathrm{d}x + \text{const.}
\end{aligned} \tag{2.8}$$

The right-hand side represents the cross-entropy with respect to $p_t(y|x)$ and $w(y, x; \theta_w)$ in which the source density $p_s(y|x; \theta_s)$ is omitted as a constant. The second term corresponds to the normalizing constant of the unnormalized target model in the right-hand side of $p_t(y|x; \theta_w) \propto w(y, x; \theta_w)p_s(y|x; \theta_s)$.

Liu & Fukumizu (2016) focused on classification tasks, but this thesis focuses on regression tasks. As an example of modeling for regression tasks, the following Gaussian distributions are adopted:

$$w(y, x; \theta_w) \propto \exp\left\{-\frac{(y - f_w(x; \theta_w))^2}{\sigma}\right\}, \tag{2.9}$$

$$p_s(y|x; \theta_s) \propto \exp\left\{-\frac{(y - f_s(x; \theta_s))^2}{\eta}\right\}. \tag{2.10}$$

Using Eq. (2.9) and Eq. (2.10), we obtain the normalizing constant as

$$\begin{aligned}
&\int w(u,x;\theta_w)p_s(u|x;\theta_s)\mathrm{d}u \\
&\qquad\propto \int \exp\left\{-\frac{(u-f_w(x;\theta_w))^2}{\sigma}-\frac{(u-f_s(x;\theta_s))^2}{\eta}\right\}\mathrm{d}u \\
&\qquad= \int \exp\left\{-\left(\frac{1}{\sigma}+\frac{1}{\eta}\right)\left(u-\frac{\eta f_w(x;\theta_w)+\sigma f_s(x;\theta_s)}{\sigma+\eta}\right)^2\right. \\
&\qquad\qquad\qquad\qquad\left.-\frac{(f_w(x;\theta_w)-f_s(x;\theta_s))^2}{\sigma+\eta}\right\}\mathrm{d}u \\
&\qquad\propto \exp\left\{-\frac{(f_w(x;\theta_w)-f_s(x;\theta_s))^2}{\sigma+\eta}\right\}. \qquad (2.11)
\end{aligned}$$

With this expression, the empirical Kullback–Leibler divergence for the training set $\mathcal{D}_t$ can be written as

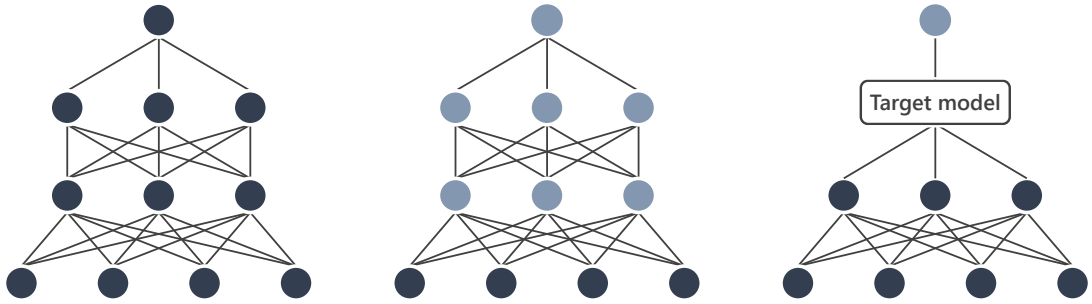
$$\begin{aligned}
&\mathbb{E}_{\hat{p}_t(x)}[\mathrm{KL}(\hat{p}_t(y|x)||p_t(y|x;\theta_w))] \\
&\qquad= -\frac{1}{n}\sum_{i=1}^{n}\Big[\log w(y_i,x_i;\theta_w)-\log\int w(u,x_i;\theta_w)p_s(u|x_i;\theta_s)\mathrm{d}u\Big] \\
&\qquad\propto \frac{1}{n}\sum_{i=1}^{n}\Big[(y_i-f_w(x_i;\theta_w))^2-\rho(f_s(x_i;\theta_s)-f_w(x_i;\theta_w))^2\Big]+\text{const.}, \qquad (2.12)
\end{aligned}$$

where $\rho=\sigma/(\sigma+\eta)\in(0,1)$, and $\hat{p}(x)$ and $\hat{p}_t(y|x)$ are the empirical distributions. All terms irrelevant to θ_w are omitted.

The parameter θ_w in the density-ratio model should be estimated by minimizing Eq. (2.12). For the prediction function, we use the plug-in estimator $\arg\max_y\ p_t(y|x,\hat{\theta}_w)$, which results in

$$\hat{y}(x)=(1-\rho)f_w(x;\hat{\theta}_w)+\rho f_s(x;\theta_s). \qquad (2.13)$$

Eq. (2.7) and (2.12) are similar, but the signs of the regularization parameters are different. The regularization parameter λ_2 in Eq. (2.7) takes the value $\lambda>0$, i.e., a positive value. Hence, f_t is estimated to be close to f_s, as described above. Conversely, the regularization parameter ρ in Eq. (2.12) occurs in the interval $(0,1)$; hence, the second term in Eq. (2.12) serves to move f_w farther from f_s. This is because Bayesian inference focuses on the similarity between the source and target domains, whereas density-ratio estimation focuses on the difference between the two domains.



(a) Source neural network (b) Fine-tuning (c) Feature extraction

Figure 2.2: Schematics of neural network-based transfer learning approaches. (a) From the pre-trained source neural network, (b) fine-tuning retrains the parameters of the source network using the target samples (represented by the gray nodes). (c) In feature extraction, the intermediate representation is extracted and the target model is trained using the extracted representation.

2.1.2 Non-probabilistic supervised transfer learning

Rather than addressing the domain shifts probabilistically, other approaches to transfer knowledge by assuming specific functional forms or functional relationships in the machine learning models of each domain can be adopted.

Neural network-based approach

To date, the most outstanding successes of TL have been achieved by reusing the layers or weights of deep neural networks (Yosinski et al., 2014). Typically, one or more layers in the pre-trained neural network are reused or refined using a limited target dataset by following two standard methods.

Fine-tuning Fine-tuning uses the parameters of the pre-trained neural network as the initial values for training in the target domain. Similar to Bayesian inference, this method is based on the assumption that the optimal parameters of the target model lie in the neighborhood of the parameters of the source model. By tuning the source parameters using samples from the target domain for a few steps, the target model can be obtained more efficiently than by training from scratch.

Feature extraction Given an L-layer neural network model $f_s(x) = f_L \circ f_{L-1} \circ \cdots \circ f_1(x)$ trained in the source domain, TL based on feature extraction uses the sub-model $\phi(x) = f_K \circ f_{K-1} \circ \cdots \circ f_1(x)$ made up of the Kth ($K < L$) layer of the

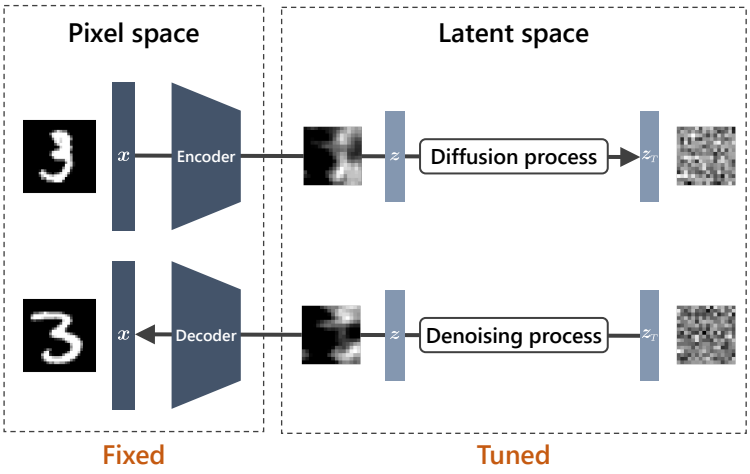


Figure 2.3: Architecture of Stable Diffusion (Rombach et al., 2022). Stable Diffusion maps the input image to the latent space. In the latent space, noises are added and removed using the diffusion model (Ho et al., 2020). For customization, the encoder and decoder are frozen, and only the diffusion model in the latent space is tuned.

pre-trained model as a calculator of the descriptors for the target task. The target model is trained using samples from the target domain and the computed descriptor $\phi(x)$. Because it is operationally intuitive and tractable, feature extraction has been successfully applied in several fields (Wu et al., 2019; Yamada et al., 2019; Ju et al., 2021), and its effectiveness has been established theoretically (Tripuraneni et al., 2020; 2021; Du et al., 2021). Feature extraction is based on the assumption that the source model would internally acquire features in common with the related domains.

Often, these two approaches are used in combination, where some parameters are fixed and the remaining are retrained. For example, Stable Diffusion (Rombach et al., 2022), a powerful image-to-text deep learning model, encourages the combination of feature extraction and fine-tuning for effective customization for various types of image generative models. Stable Diffusion maps images from the pixel space to the latent space, which is the space for the underlying feature representations in the image, and generates data in the latent space corresponding to a given text. Because the mapping from the pixel space to the latent space is trained from a large amount of data, it is assumed that Stable Diffusion ensures a universal mapping to the underlying feature representation of the images. Therefore, during customization, this mapping is frozen and reused, and only the generative networks in the latent space are fine-tuned using the images of the desired domain, thus allowing efficient model training using only a small amount of image data.

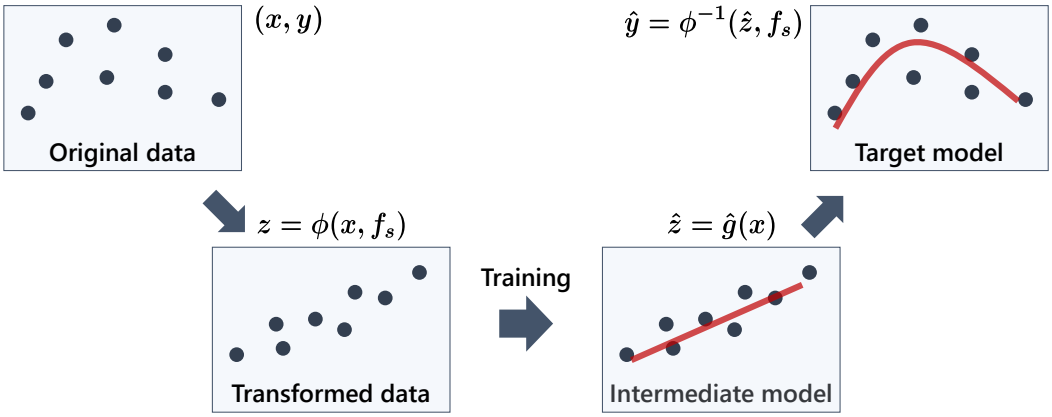


Figure 2.4: Learning procedure of hypothesis transfer learning. The data are transformed from the original data space into another space by using a transformation function ϕ. We train the intermediate model $\hat{g}$ on that space and then use the inverse transformation function ϕ^{-1} to convert the model back to the original space.

Hypothesis transfer learning

If there is a functional relationship between the source and target domains, only the domain-specific factors are additionally learned using the target samples to shift the source models to the target.

Du et al. (2017) provided a TL framework using a transformation function called HTL as follows:

1. With the source features, perform a variable transformation of the observed outputs as $z_i = \phi(y_i, f_s(x_i))$, using the transformation function $\phi : \mathcal{Y} \times \mathcal{F}_s \to \mathbb{R}$.

2. Train an intermediate model $\hat{g}(x)$ to predict the transformed output z for any given x using the transformed sample set $\{(x_i, z_i)\}_{i=1}^n$.

3. Obtain a target model $\hat{f}_t(x) = \phi^{-1}(\hat{g}(x), f_s(x))$ using the inverse of the transformation function.

This class of TL includes several approaches proposed in previous studies. For example, Kuzborskij & Orabona (2013; 2017) proposed a learning algorithm consisting of a linear transformation $\phi = y - \langle \theta, f_s(x) \rangle$ with a pre-defined coefficient θ. In this case, the factors unexplained by the linear combination of the source features are learned with g, and the target output is predicted additively with the common factor $\langle \theta, f_s(x) \rangle$ and the additionally learned g.

Learning the target domain itself is often complex and difficult, especially with a small sample size. The features obtained in the source domain are thought considered

to capture the domain-common patterns, and using these features to transform the objective variables would make the task easier to handle. In other words, the transformations make it possible to solve the task efficiently, even with a small sample size.

2.1.3 Unsupervised transfer learning

Although this thesis focuses on supervised TL, we briefly discuss unsupervised TL, in which no target labels are available; this topic has also been actively studied. To provide an overview of TL methodologies, we mention here some key approaches for unsupervised TL problems.

Covariate shift adaptation

The situation where the conditional distribution of the outputs y for a given input x remains the same for the source and target but the distributions of the inputs (covariates) x are different is called *covariate shift* (Shimodaira, 2000). Denote the source and target input distributions as $p_s(x)$ and $p_t(x)$, respectively, and the conditional distributions of the output as $p_s(y|x)$ and $p_t(y|x)$, respectively. In the covariate shift adaptation, it is assumed that $p_s(x) \neq p_t(x)$ and $p_s(y|x) = p_t(y|x)$ hold for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. This assumption is closely related to the fields of experimental design (Fedorov et al., 1972; Pukelsheim, 2006), active learning (Settles, 2012), and sample selection bias (Heckman, 1979).

By adopting the covariate shift assumption, Shimodaira (2000) proposed the maximum weighted log-likelihood estimation (or importance-weighted empirical risk minimization) as a parameter estimation method, which is expressed as follows:

$$\max_{\theta} \sum_{i=1}^{n} \frac{p_t(x_i^{(s)})}{p_s(x_i^{(s)})} \log p_s(y_i^{(s)}|x_i^{(s)};\theta), \tag{2.14}$$

where $\{(x_i^{(s)}, y_i^{(s)})\}_{i=1}^{n}$ is a set of the observed samples in the source domain. In this maximization problem, the source samples that are not important to the target task are considered to be down-weighted using the density-ratio, and vice versa. This approach is theoretically shown to have some asymptotic properties such as consistency.

In real applications, the density-ratio $p_t(x)/p_s(x)$ in Eq. (2.14) and some hyperparameters need to be determined appropriately. Hence, several model selection criteria such as the information criterion (Shimodaira, 2000; Sugiyama & Klaus-Robert, 2005) and importance-weighted cross validation method (Sugiyama et al., 2007a) have been proposed.

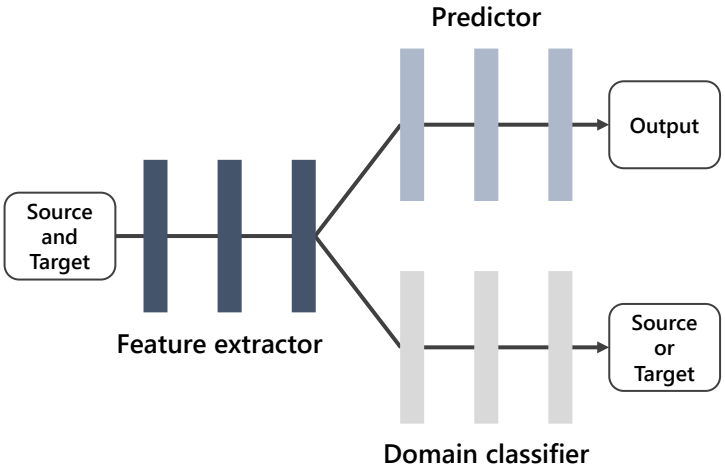


Figure 2.5: Architecture of domain-adversarial neural network (Ganin et al., 2016). Motivated by the theoretical result of Ben-David et al. (2009), the domain-adversarial neural network (DANN) has a GAN-like architecture. DANN maps the inputs to the domain-common representations, which make the domains indistinguishable, and predicts the outputs using these representations.

Domain-adversarial neural networks

In Section 2.1.2, we have introduced two standard neural network-based TL procedures: fine-tuning and feature extraction. Besides these standard procedures, a theoretically motivated methodology for unsupervised TL settings has also been proposed. To obtain a source model that also shows high performance in the target domain, the domain-adversarial neural network (DANN) (Ganin et al., 2016) builds an encoder so that the source and target inputs are indistinguishable. Its model architecture is inspired by the generative adversarial networks (GANs) (Goodfellow et al., 2014) described in Figure 2.5.

The training strategy of DANN is motivated by the theoretical analysis in Ben-David et al. (2009). In Ben-David et al. (2009), it is shown that the error of a given source model in the target domain can be evaluated using the distance between the data-generating distributions of the two domains, called the $\mathcal{H}\Delta\mathcal{H}$-divergence. Intuitively, $\mathcal{H}\Delta\mathcal{H}$-divergence refers to the difficulty in discriminating the data-generating distributions of the two domains. As the two distributions are more difficult to discriminate, the difference in the model error becomes smaller. The details of the theoretical analysis of Ben-David et al. (2009) are presented in Section 2.2.3. DANN is considered an implementation of this theoretical result in combination with the GAN architecture.

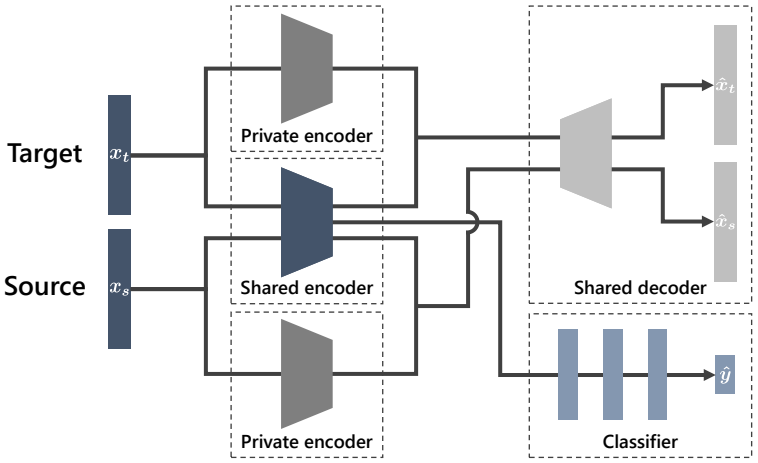


Figure 2.6: Architecture of domain separation networks (Bousmalis et al., 2016). Inputs from the source and target domains are mapped to the domain-invariant feature representations using the shared encoder, with separation of domain-specific factors using the private encoder. Orthogonality constraints are used to ensure that the shared and private encoders capture different aspects of the input. Furthermore, by training the shared decoder to decode the encoded features to the original input, two encoded representations can be guaranteed to contain nearly complete information about the original input. The classifier is trained from the domain-invariant features by using the labeled source samples.

DANN trains encoders such that the domain discriminator would fail. Other strategies for obtaining domain-invariant features have been proposed, such as measuring the distance between distributions using the maximum mean discrepancy (MMD) (Baktashmotlagh et al., 2013; Ghifary et al., 2014; Long et al., 2015) or correlation distance (Sun & Saenko, 2016).

Domain separation networks

Some unsupervised TL approaches, including DANN, aim to train an encoder that maps the inputs from each domain to a domain-invariant feature representation, and then uses it to train a predictor that can generalize it to the target domain by using the source samples. To obtain this domain-invariant feature representation more efficiently, Bousmalis et al. (2016) proposed a domain separation network (DSN), which is a neural network architecture to separate the domain-specific components explicitly from the inputs of both domains. Figure 2.6 shows the network architecture of DSN. In DSN, the domain-common and domain-specific factors are explicitly and

jointly modeled. The mapping to the domain-common representation obtained in this manner is expected to eliminate bias for individual domains and aid the training of domain-invariant predictors. In relation to this approach, we provide a method to model the domain shifts and domain-specific components separately and estimate them simultaneously in Chapter 4.

Gradual domain adaptation

These domain adaptation methods described above can be used only for sufficiently small domain shifts. For example, DANN and DSN train encoders to map inputs to domain-common feature representations. However, it is difficult to obtain the desirable representations when the domains are completely different. To adapt to large domain shifts, another approach called *gradual domain adaptation* (Kumar et al., 2020) is proposed, which gradually adapts a given source model with unlabeled data whose distribution gradually shifts toward the target domain. By using a technique called self-training or pseudo-labeling (Chapelle et al., 2006; Lee et al., 2013), the gradually shifting inputs are pseudo-labeled and the model is trained on these pseudo-labeled data. Kumar et al. (2020) theoretically shows that such gradual domain adaptation is superior to the conventional unsupervised TL methods that adapt directly to the target domain.

2.2 Theoretical analyses

The goal of TL is to obtain high-performing models by using relevant knowledge. As with developing TL algorithms, it is important to theoretically guarantee that these algorithms have the capability to improve the generalization performance. This section introduces some important mathematical tools for theoretical analysis and presents several theoretical results of TL.

Let $(\mathcal{Z}, P)$ be an arbitrary probability space, and set $\{z_i\}_{i=1}^n$ to be independent random variables distributed according to P. For a function $f : \mathcal{Z} \to \mathbb{R}$, define the expectation of f with respect to P and its empirical counterpart as

$$Pf = \mathbb{E}_P f(z), \qquad P_n f = \frac{1}{n} \sum_{i=1}^{n} f(z_i).$$

2.2.1 Generalization error and excess risk

We need a criterion to evaluate the performance of the trained model. By using a loss function ℓ, the expected error of the trained model h is defined as the generalization error as follows:

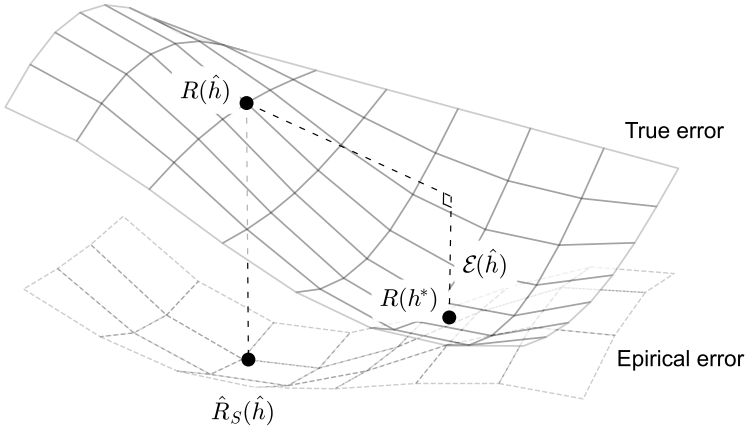


Figure 2.7: Relationship between the empirical error, generalization error, and smallest possible error. The model $\hat{h}$ is trained to minimize the empirical error $\hat{R}_S(\hat{h})$ (on the dashed surface). The generalization error $R(\hat{h})$ is the value of the true error (on the solid surface) of the model. The function that achieves the smallest possible error is denoted as h^*. The difference between the smallest possible error $R(h^*)$ and the generalization error $R(\hat{h})$ is called the excess risk $\mathcal{E}(\hat{h})$.

Definition 3 (Generalization error)**.** The generalization error of the function h is defined as

$$R(h) := P\ell(y, h) = \mathbb{E}_{(x,y)}[\ell(y, h(x))].$$

Because the sample-generating distribution is unknown, the generalization error cannot be accessed directly. Instead, the expectation on the empirical distribution can be measured.

Definition 4 (Empirical error)**.** The empirical error of the function h is defined as:

$$\hat{R}_S(h) := P_n\ell(y, h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(x_i)).$$

One of the principal challenges in statistical learning theory is to evaluate the difference between these two values, i.e., $R(h) - \hat{R}_S(h)$, as a function of the number of samples n.

In addition, it is important to evaluate how close the generalization error is to the smallest possible error. The difference between the generalization error of a given function and the smallest possible error is called the excess risk, which is defined as follows.

Definition 5 (Excess risk)**.** The excess risk of the function h is defined as:

$$\mathcal{E}(h) := R(h) - \inf_h R(h) \ \big(= P(h - h^*) \big),$$

where $h^* = \arg\inf_h R(h)$.

The excess risk refers to how close the trained model h is to the optimal element in the function space. By evaluating this as a function of the number of samples, the efficiency of the model estimation can be assessed.

2.2.2 Complexity of function space

In machine learning models, the learning guarantee derivation strongly relies on the notion of complexity of a function space. Here, we introduce the standard complexity measure: the Rademacher complexity.

$\mathcal{H}$ denotes a function space and let $S = \{z_i\}_{i=1}^n$ be a fixed set of samples drawn from the distribution P. Let $\{\sigma_i\}_{i=1}^n$ be a set of independent uniform random variables taking values in $\{-1, +1\}$.

Global complexity

The following statement is a formal definition of the empirical Rademacher complexity (Mohri et al., 2018).

Definition 6 (Empirical Rademacher complexity)**.** The empirical Rademacher complexity of $\mathcal{H}$ with respect to the sample S is defined as:

$$\hat{\mathfrak{R}}_S(\mathcal{H}) := \mathbb{E}_\sigma \left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(z_i) \right].$$

The Rademacher complexity expresses the richness of a family of functions by measuring the degree to which the output of a function in $\mathcal{H}$ can be correlated on average with random noise.

Taking the expectation with respect to the sample realization, the (global) Rademacher complexity is defined as follows.

Definition 7 (Rademacher complexity)**.** The (global) Rademacher complexity of $\mathcal{H}$ is defined as:

$$\mathfrak{R}(\mathcal{H}) := \mathbb{E}_{S \sim P^n}[\hat{\mathfrak{R}}_S(\mathcal{H})],$$

For regression tasks, the generalization error can be evaluated using the empirical error and Rademacher complexity. The following statement is based on Theorem 11.3 of (Mohri et al., 2018).

Rademacher complexity generalization bounds Assume that a loss function ℓ is upper-bounded by $M > 0$, i.e., $\ell(y, y') < M$ for all $y, y' \in \mathcal{Y}$, and for any $y \in \mathcal{Y}$, $\ell(y, \cdot)$ is μ_ℓ-Lipschitz for some $\mu_\ell > 0$. Then, with probability at least $1 - \delta$,

$$R(h) \leq \hat{R}_S(h) + 2\mu_\ell \mathfrak{R}(\mathcal{H}) + M\sqrt{\frac{\log \frac{1}{\delta}}{n}}, \tag{2.15}$$

$$R(h) \leq \hat{R}_S(h) + 2\mu_\ell \hat{\mathfrak{R}}_S(\mathcal{H}) + 3M\sqrt{\frac{\log \frac{2}{\delta}}{2n}}. \tag{2.16}$$

In addition, Koltchinskii (2001) and Bartlett et al. (2004) have shown the excess risk bound of the following form:

Rademacher complexity excess risk bounds Let $\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{R}_S(h)$ and $h^* = \arg\min_{h \in \mathcal{H}} R(h)$. Then, there exists a constant $C > 0$ such that, for any $\delta > 0$, with probability at least $1 - \delta$,

$$\mathcal{E}(\hat{h}) = R(\hat{h}) - R(h^*) < C\left(\mathfrak{R}(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{n}}\right). \tag{2.17}$$

Computing the (empirical) Rademacher complexity is generally an NP-hard problem (Mohri et al., 2018). Therefore, when evaluating the generalization error of machine learning models, we do not calculate the Rademacher complexity directly but rather bound it. One example is the learning guarantee for kernel methods. The following statement is based on Theorem 6.12 of Mohri et al. (2018).

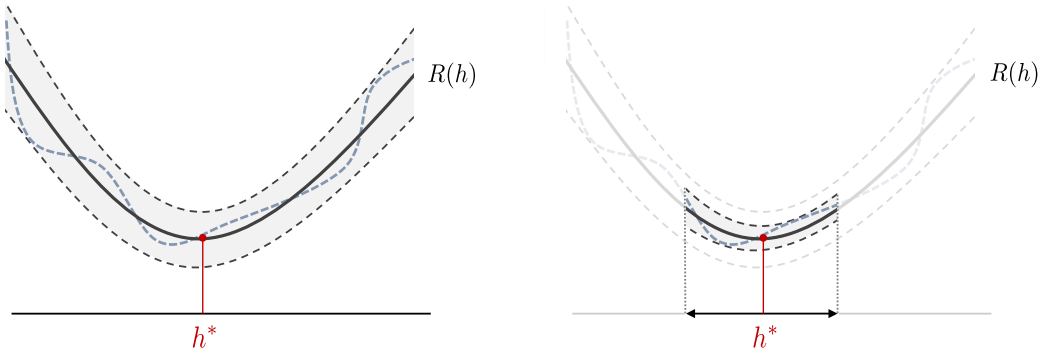
Rademacher complexity bound for kernel methods Let k be a positive definite kernel, and let Φ be the feature mapping associated with k. Assume that there exists $r > 0$ such that $k(x, x) \leq r^2$. Let $\mathcal{H} = \{x \mapsto \langle w, \Phi(x) \rangle : \|w\|_{\mathcal{H}} \leq \Lambda\}$ for some $\Lambda \geq 0$. Then,

$$\hat{\mathfrak{R}}_S(\mathcal{H}) \leq \sqrt{\frac{r^2\Lambda^2}{n}},$$

which results in

$$\mathfrak{R}(\mathcal{H}) \leq \sqrt{\frac{r^2\Lambda^2}{n}}.$$

By combining these bounds with Eq. (2.15)-(2.17), we can see that the convergence rates of the generalization error bounds and excess risk bounds are at most $\mathcal{O}(1/\sqrt{n})$.



(a) Global bound (b) Local bound

Figure 2.8: (a) Bounding the true error $R(h)$ uniformly throughout the entire function space would result in unnecessarily loose bounds (gray area). The blue dashed line is the empirical error function for one sample realization. (b) By restricting the existence area of the function around the optimal function h^*, it is expected that its risk can be evaluated more efficiently and tighter bounds can be derived.

Local complexity

The Rademacher complexity in Definitions 6 and 7 considers the supremum over the entire function space. However, when training the machine learning models in practice, we do not consider the entire space but rather focus on the subspace of functions that are not too complex, such as functions with small norms. In other words, the global Rademacher complexity may overestimate the representational power of the function class. To represent the restricted function class, the *local* Rademacher complexity is defined as follows (Bartlett et al., 2005):

Definition 8 (Local Rademacher complexity)**.** For any $r > 0$, the empirical local Rademacher complexity of $\mathcal{H}$ is defined as

$$\hat{\mathfrak{R}}_S(\mathcal{H} : Ph^2 < r) = \mathbb{E}_\sigma \left[\sup_{h \in \mathcal{H} : Ph^2 < r} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(z_i) \right].$$

The local Rademacher complexity is defined as the expectation of the empirical complexity, as follows:

$$\mathfrak{R}(\mathcal{H} : Ph^2 < r) = \mathbb{E}_{S \sim P^n}[\hat{\mathfrak{R}}_S(\mathcal{H} : Ph^2 < r)].$$

The difference between the global and local Rademacher complexities is the restriction on the function space. $Ph^2 = \mathbb{E}[h(x)^2]$ represents the variance in the

output of the function, and r controls the complexity of the function space. The global Rademacher complexity corresponds to a special case for $r = \infty$.

The following results are from Corollary 5.1 and Corollary 5.3 of Bartlett et al. (2005), and are considered as the localized versions of the generalization bound and excess risk bound described above.

Local Rademacher complexity generalization bounds For a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$, define the loss class $\mathcal{L}_\mathcal{H}$ as

$$\mathcal{L}_\mathcal{H} = \{(x, y) \mapsto \ell(y, h(x)) : h \in \mathcal{H}\}.$$

Define

$$\hat{\psi}_n(r) = 20\hat{\mathfrak{R}}_S\{h \in \mathrm{star}(\mathcal{L}_\mathcal{H}, 0) : P_n h^2 \le 2r\} + \frac{13 \log \frac{1}{\delta}}{n},$$

where $\mathrm{star}(\mathcal{L}_\mathcal{H}, 0)$ is the star-hull of $\mathcal{L}_\mathcal{H}$ around 0, i.e.,

$$\mathrm{star}(\mathcal{L}_\mathcal{H}, 0) = \{\alpha \ell_h : \ell_h \in \mathcal{L}_\mathcal{H}, \alpha \in [0, 1]\}.$$

Let $\hat{r}^*$ be a solution of the equation $r = \hat{\psi}_n(r)$. Then, for any $K > 1$ and all $h \in \mathcal{H}$, with probability at least $1 - 3\delta$,

$$R(h) \le \frac{K}{K-1}\hat{R}_n(h) + 6K\hat{r}^* + \frac{(11 + 5K) \log \frac{1}{\delta}}{n}. \tag{2.18}$$

Local Rademacher complexity excess risk bounds To evaluate the excess risk using the local Rademacher complexity, we make the following assumptions.

- There exists $h^* \in \mathcal{H}$ satisfying $P\ell(y, h) = \inf_{h \in \mathcal{H}} P\ell(y, h)$, and $\hat{h} \in \mathcal{H}$ satisfying $P_n\ell(y, \hat{h}) = \inf_{h \in \mathcal{H}} P_n\ell(y, h)$.
- The loss function ℓ is μ_ℓ-Lipschitz with respect to the first argument.
- There exists a constant $B \ge 1$ such that $P(h - h^*)^2 \le BP(\ell(y, h) - \ell(y, h^*))$.

Assume ψ is a sub-root function[1] for which

$$\psi(r) \ge \mu_\ell B \mathfrak{R}\{h \in \mathcal{H} : \mu_\ell^2 P(h - h^*)^2 \le r\}.$$

Then, for any $x > 0$ and any $r \ge \psi(r)$, with probability at least $1 - \delta$,

$$R(\hat{h}) - R(h^*) \le 705\frac{r}{B} + \frac{(11L + 27B) \log \frac{1}{\delta}}{n}. \tag{2.19}$$

Both Eq. (2.18) and Eq. (2.19) use the fixed points of the local Rademacher complexity. For the kernel method, these values can be evaluated as follows.

[1] A function $\psi : [0, \infty) \to [0, \infty)$ is sub-root if it is nonnegative, non-decreasing and if $r \mapsto \psi(r)/\sqrt{r}$ is non-increasing for $r > 0$. See Bartlett et al. (2005) for the properties of sub-root functions.

Local Rademacher complexity bound and its fixed point for kernel methods Let $\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) with a positive definite kernel k. Define an integral operator T associated with k and an arbitrary distribution P, i.e., $Tf = \int k(\cdot, y)f(y)\mathrm{d}P(y)$. Let $\{\lambda_i\}_{i=1}^{\infty}$ be its eigenvalues arranged in a non-increasing order. Moreover, consider the normalized Gram matrix $\hat{T} = \frac{1}{n}(k(x_i, x_j))_{i,j=1,\ldots,n}$, and let $\{\hat{\lambda}_i\}_{i=1}^{n}$ be its eigenvalues arranged in a non-increasing order. The following results are from Mendelson (2002) and Bartlett et al. (2005). For every $r > 0$,

$$\hat{\mathfrak{R}}_S\{h \in \mathcal{H} : P_n h^2 \leq r\} \leq \left(\frac{2}{n}\sum_{i=1}^{n}\min\{r, \hat{\lambda}_i\}\right)^{\frac{1}{2}}, \tag{2.20}$$

and

$$\mathfrak{R}\{h \in \mathcal{H} : Ph^2 \leq r\} \leq \left(\frac{2}{n}\sum_{i=1}^{\infty}\min\{r, \lambda_i\}\right)^{\frac{1}{2}}.$$

Bartlett et al. (2005) evaluated the fixed point of the upper-bound of Eq. (2.20) as

$$\hat{r}^* \leq \min_{0\leq h\leq n}\left(\frac{h}{n} + \sqrt{\frac{1}{n}\sum_{i>h}\hat{\lambda}_i}\right). \tag{2.21}$$

Eq. (2.21) implies that the order of the fixed point is at most $\mathcal{O}(1/\sqrt{n})$, which corresponds to the basic convergence rate with the global complexity. Furthermore, the order and therefore the convergence rate of the generalization error and excess risk improve in comparison with the above basic rate depending on the decay rate of the eigenvalues of the integral operator T or Gram matrix $\hat{T}$.

2.2.3 Theoretical results for transfer learning

Finally, we provide a brief review of some theoretical results of TL. For more detailed reviews, see Redko et al. (2020); Yang et al. (2020); Zhang & Yang (2021).

Divergence-based analyses

Ben-David et al. (2009) presented the principal investigation for domain adaptation, which pioneered the learning theory of TL. In the unsupervised TL setting, Ben-David et al. (2009) evaluated the performance of the source model on the target domain as follows:

Target error bound of source model (Theorem 2 of Ben-David et al. (2009)) Let $f_s(x), f_t(x)$ be the true functions of the source and target domains, respectively. Define

$$\lambda := \min_{h \in \mathcal{H}} \left[\mathbb{E}_{x \sim \mathcal{D}_s}[|h(x) - f_s(x)|] + \mathbb{E}_{x \sim \mathcal{D}_t}[|h(x) - f_t(x)|] \right].$$

Let $\mathcal{H}$ be a hypothesis space of Vapnik–Chervonenkis (VC) dimension[2] d. If $\mathcal{U}_s, \mathcal{U}_t$ are unlabeled sample sets of size m each, drawn from the source and target distributions $\mathcal{D}_s, \mathcal{D}_t$, respectively, then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, for every $h \in \mathcal{H}$,

$$\begin{aligned} &\mathbb{E}_{x \sim \mathcal{D}_t}[|h(x) - f_t(x)|] \\ &\quad \leq \mathbb{E}_{x \sim \mathcal{D}_s}[|h(x) - f_s(x)|] + \frac{1}{2} \hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_s, \mathcal{U}_t) \\ &\qquad + 4\sqrt{\frac{2d \log(2m) + \log(2/\delta)}{m}} + \lambda. \end{aligned} \tag{2.22}$$

$d_{\mathcal{H}\Delta\mathcal{H}}(\cdot, \cdot)$ is called $\mathcal{H}\Delta\mathcal{H}$-*divergence* and defined as

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) := 2 \sup_{h, h' \in \mathcal{H}} \left| P_{x \sim \mathcal{D}_s}[h(x) \neq h'(x)] - P_{x \sim \mathcal{D}_t}[h(x) \neq h'(x)] \right|,$$

for any distribution $\mathcal{D}_s, \mathcal{D}_t$. $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\cdot, \cdot)$ is its empirical counterpart. $\mathcal{H}\Delta\mathcal{H}$-divergence intuitively represents the difficulty in discriminating the source and target distributions. It takes a minimum value of 0 when the source and target distributions cannot be discriminated using the functions in the hypothesis space $\mathcal{H}$. Further, it takes a maximum value of 2 when they can be completely discriminated. Eq. (2.22) indicates that as the two input distributions $\mathcal{D}_s, \mathcal{D}_t$ are indistinguishable, the performance of the source model in the target domain improves. DANN, introduced in Section 2.1.2, implements this implication using a neural network architecture.

Eq. (2.22) is for the unsupervised TL settings, but it can be extended to include the supervised setting, i.e., the setting for which labeled samples of the target domain are obtained.

Supervised TL bound (Theorem 3 of Ben-David et al. (2009)) In addition to the setting in Eq. (2.22), assume that for a fixed $\beta \in [0, 1]$, we have n labeled samples generated by drawing βn samples from $\mathcal{D}_t$ and $(1 - \beta)n$ samples from $\mathcal{D}_s$ and

[2]The VC dimension is a measure of the complexity of the set of functions for learning a binary classification problem. It is defined as the maximum number of points that the model can scatter. For a more detailed description, see Mohri et al. (2018).

labeling them according to the true functions f_s and f_t, respectively. Consider the following combined error:

$$\hat{\epsilon}_\alpha(h) := \frac{\alpha}{\beta n} \sum_{(x,y):\text{target}} |y_i - h(x_i)| + \frac{1-\alpha}{(1-\beta)n} \sum_{(x,y):\text{source}} |y_i - h(x_i)| \tag{2.23}$$

for some $\alpha \in [0, 1]$. If $\hat{h} \in \mathcal{H}$ is the empirical minimizer of the combined loss $\hat{\epsilon}_\alpha(h)$ and $h^* = \min_{h \in \mathcal{H}} \mathbb{E}_{x \sim \mathcal{D}_t}[|h(x) - f_t(x)|]$ is the target error minimizer, the for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\begin{aligned}
&\mathbb{E}_{x \sim \mathcal{D}_t}[|\hat{h}(x) - f_t(x)|] \\
&\quad \leq \mathbb{E}_{x \sim \mathcal{D}_t}[|h^*(x) - f_t(x)|] \\
&\qquad + 4\sqrt{\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}} \sqrt{\frac{2d \log(2(n+1)) + 2\log(8/\delta)}{n}} \\
&\qquad + 2(1-\alpha)\left(\frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_s, \mathcal{U}_t) + 4\sqrt{\frac{2d\log(2m) + \log(8/\delta)}{m}} + \lambda\right).
\end{aligned} \tag{2.24}$$

α is a parameter that controls the degree of contribution of the source samples to the training of the target model; at $\alpha = 0$, Eq. (2.24) is identical to Eq. (2.22), and at $\alpha = 1$, Eq. (2.24) corresponds to the standard learning bounds. By selecting the optimal α, Eq. (2.24) becomes a tighter bound than these two settings.

The results of Ben-David et al. (2009) have pioneered the theoretical analysis of TL and domain adaptation, resulting in various theoretical results. Many of them differ in the quantification of the differences in the input distributions of the target and source domains, for example, based on the discrepancy distance (Mansour et al., 2009), integral probability metrics (Zhang et al., 2012), based on Wasserstein distance (Courty et al., 2014) and MMD (Redko et al., 2017).

Complexity-based analyses

Rather than evaluating the generalization performance of the target model based on the divergence of the probability distribution, some theoretical analyses are based on the belief that using the source knowledge would efficiently constrict the function space to be explored.

Kuzborskij & Orabona (2017) derived the generalization error bounds and excess risk bound in the linear supervised HTL setting of the following forms:

$$R(h) - \hat{R}_S(h) = \mathcal{O}\left(\frac{R_s + \sqrt{R_s}}{\sqrt{n}} + \frac{R_s + 1}{n}\right),$$
$$R(h) - \inf_{h \in \mathcal{H}} R(h) = \mathcal{O}\left(\frac{\sqrt{R_s} + \sqrt[4]{R_s}}{\sqrt[4]{n}} + \frac{\sqrt[4]{R_s} + \sqrt[8]{R_s}}{\sqrt[4]{n^{1.5}}} + \sqrt{\frac{R_s}{n}} + \frac{1}{n}\right),$$

where R_s is the generalization error in the target domain of the source models. These bounds indicate that when knowledge from the source domain is valuable for the training in the target domain, the convergence rate of the generalization error and excess risk bound improves up to $\mathcal{O}(1/n)$. The magnitude of the metric R_s is considered to represent the transferability between the source and target domains.

Tripuraneni et al. (2020) considered the supervised feature extraction setting, where the domain-common representation is trained through the source tasks and the obtained representation is reused for the training in the target domain. The transferability is measured through the task diversity ν, and it is shown that the convergence rate of the excess risk has the following order:

$$\tilde{\mathcal{O}}\left(\frac{1}{\nu}\left(\sqrt{\frac{C(\mathcal{H}) + tC(\mathcal{F})}{n_s t}}\right) + \sqrt{\frac{C(\mathcal{F})}{n_t}}\right),$$

where $\tilde{O}$ denotes the order ignoring the logarithmic factors, t is the number of source domains, and n_s and n_t are the number of the training samples of the source and target domains, respectively. $C(\cdot)$ captures the complexity of the space. $\mathcal{H}$ is the function space of the task-shared model, and $\mathcal{F}$ is the function space of the task-specific maps. The task diversity ν measures how well the source domain covers the space required for training in the target domain[3]. As the source domains cover a sufficiently large space, ν becomes larger and the learning rate improves.

[3]For a precise definition, see Tripuraneni et al. (2020).

Chapter 3

Bayesian transfer learning with density-ratio modeling

In this chapter, we propose a novel framework that unifies and extends the existing methods of TL for regression. To bridge a pre-trained source model to the model for a target task, we introduce a density-ratio reweighting function, which is estimated using the Bayesian framework with a specific prior distribution. By changing two intrinsic hyperparameters and the choice of the density-ratio model, the proposed method can integrate three popular methods of TL: TL based on cross-domain similarity regularization, a probabilistic TL using density-ratio estimation, and fine-tuning of pre-trained neural networks. Moreover, the proposed method can benefit from its simple implementation without any additional cost; the regression model can be fully trained using off-the-shelf libraries for supervised learning in which the original output variable is simply transformed into a new output variable. We demonstrate its simplicity, generality and applicability using various real-data applications.

3.1 Method

We are given a pre-trained model $f_s(x)$ on the source task, which defines the mapping between any input x to a real-valued output $y \in \mathbb{R}$. Note that f_s is fixed, and we do not consider the error of its training or uncertainty of its prediction. The objective is to transform the given $f_s(x)$ into a target model $f_t(x)$ using n instances from the target domain $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^n$.

Inspired by Liu & Fukumizu (2016), we apply probabilistic modeling to the transition from $f_s(x)$ to $f_t(x)$. As mentioned in Section 2.1.1, with the conditional distribution $p_s(y|x)$ of the source task, the conditional distribution on the target can

be decomposed as

$$p_t(y|x) = w(y,x)p_s(y|x),$$

where $w(y,x) := p_t(y|x)/p_s(y|x)$. Consider that the source distribution is modeled using $p_s(y|x,f_s)$, which involves the pre-trained $f_s(x)$. In addition, the density-ratio function $w(y,x)$ is separately modeled as $w(y,x;\theta_w)$ with an unknown parameter θ_w, which will be associated with a regression model $f_w(x;\theta_w)$. The target model $p_t(y|x;\theta_w)$ is then described as

$$\begin{aligned} p_t(y|x;\theta_w) &= w(y,x;\theta_w)p_s(y|x,f_s) \\ &\text{such that } \forall x : \int w(y,x;\theta_w)p_s(y|x,f_s)\mathrm{d}y = 1, \end{aligned} \tag{3.1}$$

where the normalization constraint is due to the fact that the conditional probability needs to be normalized to 1 over its domain.

We employ Bayesian inference to estimate the unknown θ_w in the density-ratio model $w(y,x;\theta_w)$. The target model $p_t(y|x;\theta_w)$ is used as the likelihood for Bayesian inference, and a prior distribution $p(\theta_w|f_s)$ is placed on θ_w, which depends on the given f_s. The posterior distribution is then expressed as

$$p(\theta_w|\mathcal{D}_t) \propto \prod_{i=1}^{n} p_t(y_i|x_i;\theta_w)p(\theta_w|f_s). \tag{3.2}$$

As in Eq. (2.9) and Eq. (2.10), the following Gaussian models are used for the likelihood function:

$$w(y,x;\theta_w) \propto \exp\left\{-\frac{(y-f_w(x;\theta_w))^2}{\sigma}\right\}, \tag{3.3}$$

$$p_s(y|x,f_s) \propto \exp\left\{-\frac{(y-f_s(x))^2}{\eta}\right\}, \tag{3.4}$$

where $\sigma > 0$ and $\eta > 0$. Thus, as well as Eq. (2.11), the normalization constant in Eq. (3.1) is given as

$$\int w(y,x;\theta_w)p_s(y|x,f_s)\mathrm{d}y \propto \exp\left\{-\frac{(f_s(x)-f_w(x;\theta_w))^2}{\sigma+\eta}\right\},$$

which depends on the proximity of $f_w(x;\theta_w)$ to $f_s(x)$. Further, we regularize the training based on the discrepancy of the two models $f_w(x;\theta_w)$ and $f_s(x)$, which can belong to different classes of regression models. For this, we introduce a prior distribution that implements a function-based regularization as

$$p(\theta_w|f_s) \propto \exp\left\{-\sum_{i=1}^{m}\frac{(f_s(u_i)-f_w(u_i;\theta_w))^2}{\lambda}\right\}, \tag{3.5}$$

where $\lambda \in \mathbb{R}\backslash\{0\}$. The discrepancy is measured by the sum of their squared distances over m input values $\mathcal{U} = \{u_i\}_{i=1}^m$. Hereinafter, we use the n observed inputs in $\mathcal{D}_t$ for $\mathcal{U}$. The posterior distribution involves three hyperparameters (σ, η, λ). Note that λ can be either positive or negative and controls the degree of discrepancy positively or negatively. As described below, this Gaussian-type modeling leads to an analytic workflow that can benefit from less effort for the implementation.

We consider the maximum a posteriori (MAP) estimation of θ_w and a class of prediction functions $\hat{y}(x)$ that is characterized by two hyperparameters τ and ρ:

$$\hat{\theta}_w = \arg\min_{\theta_w} \sum_{i=1}^{n} \left\{ (y_i - f_w(x_i;\theta_w))^2 - \tau (f_s(x_i) - f_w(x_i;\theta_w))^2 \right\}, \tag{3.6}$$

$$\hat{y}(x) = \arg\max_{y} \; p_t(y|x;\hat{\theta}_w) = (1-\rho) f_w(x;\hat{\theta}_w) + \rho f_s(x), \tag{3.7}$$

$$\tau \coloneqq \frac{\sigma}{\sigma+\eta} - \frac{\sigma}{\lambda} \in (-\infty, 1), \quad \rho \coloneqq \frac{\sigma}{\sigma+\eta} \in (0,1).$$

In the training objective Eq. (3.6), the first term measures the goodness-of-fit with respect to $\mathcal{D}_t$. The second term is derived from the normalization term in Eq. (3.1) and the prior distribution Eq. (3.5). It regularizes the training through the discrepancy between $f_w(x;\theta_w)$ and the pre-trained $f_s(x)$. The prediction function Eq. (3.7) corresponds to the mode of the plug-in predictive distribution Eq. (3.1). Note that the original three hyperparameters are reduced to $\tau \in (-\infty, 1)$ and $\rho \in (0,1)$. By varying (τ, ρ) and different models on $f_w(x;\theta_w)$ coupled with the learning algorithms, the resulting method can bridge various methods of TL as described later.

3.2 Implementation cost

By completing the square of Eq. (3.6) with respect to $f_w(x;\theta_w)$, the objective function can be rewritten as a residual sum of squares on a transformed output variable z as

$$\hat{\theta}_w = \arg\min_{\theta_w} \sum_{i=1}^{n} (z_i - f_w(x_i;\theta_w))^2, \quad z_i = \frac{y_i - \tau f_s(x_i)}{1-\tau}.$$

Once the original output y_i is simply converted to z_i with a given $f_s(x)$ and τ, the model can be trained by using a common squared loss minimization library for regression. Any regularization term, such as ℓ_1- or ℓ_2-regularization, can also be added. Therefore, the proposed method can be implemented at essentially no cost. In the applications described later, we utilized ridge regression, random forest regression, and neural networks as $f_w(x;\theta_w)$. We simply used the standard libraries of R language (glmnet (Friedman et al., 2010), ranger (Wright & Ziegler, 2017), and MXNet (Chen et al., 2015)) without any customization or additional coding.

Furthermore, as no source data appear in the objective function, the model is learnable by using only the training instances in the target domain as long as the source model is callable. This separately learnable property will be a great advantage in certain cases, for example, where training the source model from scratch is time-consuming or the source data can not be disclosed.

3.3 Relations to existing methods

By adjusting (τ, ρ) and selecting the appropriate choice of $f_w(x; \theta_w)$, our method can represent different types of TL as described below. Figure 3.1 presents a visual overview of the relationship between the different methods.

Regularization based on cross-domain similarity

One of the most natural ideas for model refinement is to use the similarity with the pretrained $f_s(x)$ as a constraint condition. Several existing studies have incorporated such cross-domain similarity regularization in TL or other related machine learning tasks such as avoiding catastrophic forgetting in continual learning (Kirkpatrick et al., 2017) and knowledge distillation to compress complex neural networks to simpler models (Hinton et al., 2015).

In Section 2.1.1, we have introduced regularization based on Bayesian inference. Similarly, we consider a posterior distribution in Eq. (3.2) and impose the Gaussian distribution on the likelihood $p_t(y|x; \theta_w) = \mathcal{N}(y|f_w(x; \theta_w), \sigma)$. The same prior to Eq. (3.5) is imposed to $p(\theta_w|f_s)$. Then, as described in Section 2.1.1, the MAP estimator for θ_w and the mode of the plug-in predictive distribution are of the following form:

$$\hat{\theta}_w = \arg\min_{\theta_w} \sum_{i=1}^{n} \left\{ (y_i - f_w(x_i; \theta_w))^2 + \frac{\sigma}{\lambda} (f_s(x_i) - f_w(x_i; \theta_w))^2 \right\}, \tag{3.8}$$

$$\hat{y}(x) = f_w(x; \hat{\theta}_w). \tag{3.9}$$

The objective function of our method, Eq. (3.6), can represent the MAP estimation using the objective function in Eq. (3.8) by restricting the hyperparameter τ (or λ) to a negative value, i.e., $\tau = -\sigma/\lambda < 0$. The prediction function in Eq. (3.9) corresponds to $\rho = 0$ in our method. With a negative τ, the model $f_w(x; \theta_w)$ is estimated to be closer to the pre-trained source model. Such a newly trained model $f_w(x; \hat{\theta}_w)$ is directly used as the prediction function without using the source model.

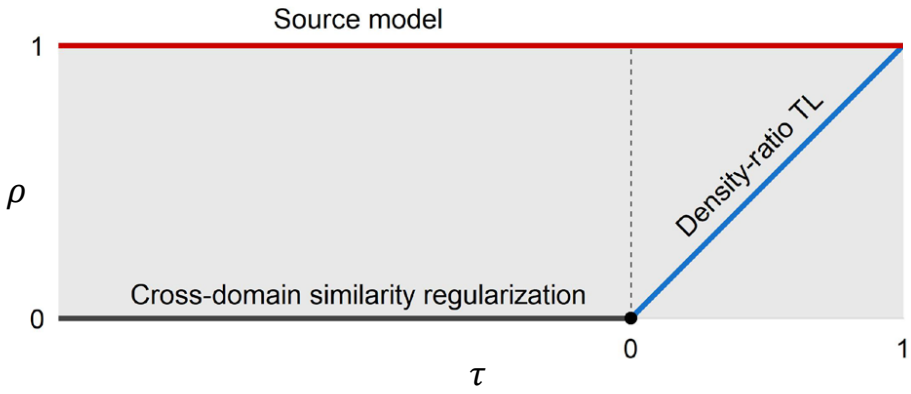


Figure 3.1: Existing methods are mapped onto the hyperparameter space (τ, ρ). The cross-domain similarity regularization corresponds to $\tau < 0$ and $\rho = 0$ (black line). If neural networks are applied to both $f_w(x; \theta_w)$ and $f_s(x)$, this region corresponds to the fine-tuning of neural networks. If $\tau = \rho$ (blue line), the class represents the density-ratio TL. The region with $\tau = \rho = 0$ (black dot) or $\rho = 1$ (red line) represents an ordinal regression without transfer or the case where a source model is directly used as the target, respectively.

Transfer Learning Based on Neural Networks

To our best knowledge, the most powerful and widely used method of TL relies on deep neural networks (Yosinski et al., 2014). When neural networks are used in both $f_w(x; \theta_w)$ and $f_s(x)$ in the objective function Eq. (3.8), the pretrained $f_s(x)$ is fine-tuned to $f_w(x; \theta_w)$ by retaining the cross-domain similarity between their output layers.

Transfer learning based on density-ratio estimation

As mentioned in Section 2.1.1, the density-ratio TL of Liu & Fukumizu (2016) was designed to minimize the conditional Kullback–Leibler divergence

$$\mathbb{E}_{q(x)}[\mathrm{KL}(q(y|x)||p_t(y|x; \theta_w))]$$

between the true density $q(y|x)$ and the transferred model $p_t(y|x; \theta_w)$ based on the density-ratio reweighting as in Eq. (3.1). If the transfer model is parameterized in the same way as Eq. (3.3), the minimization problem of the empirical Kullback–Leibler divergence and the prediction function are given as follows:

$$\hat{\theta}_w = \arg\min_{\theta_w} \sum_{i=1}^{n} \left\{ (y_i - f_w(x_i; \theta_w))^2 - \rho (f_s(x_i) - f_w(x_i; \theta_w))^2 \right\}, \tag{3.10}$$

$$\hat{y}(x) = (1 - \rho) f_w(x) + \rho f_s(x), \tag{3.11}$$

where $\rho \coloneqq \sigma/(\sigma+\eta) \in (0,1)$.

In terms of the proposed class of TL, the method in Liu & Fukumizu (2016) can be considered as a specific choice of $\tau = \rho \in (0,1)$ (the blue line in Figure 3.1). This corresponds to the case where λ in Eq. (3.5) is sufficiently large, i.e., the prior distribution for the parameters of the density-ratio function is uniformly distributed and non-informative. It is noted that the objective function in Eq. (3.10) resembles Eq. (3.8) in cross-domain similarity regularization. These two methods are regularized based on the discrepancy between $f_w(x;\theta_w)$ and $f_s(x)$, but their regularization mechanisms work in the opposite directions: the regularization parameter τ takes a positive value for the method in Liu & Fukumizu (2016), which we call cross-domain dissimilarity regularization, whereas a negative value is used for cross-domain similarity regularization.

Learning without transfer

The proposed family of methods contains two learning schemes without transfer. If the hyperparameters are selected as $\tau = 0$ and $\rho = 0$ (the black dot in Figure 3.1), the density-ratio model $f_w(x;\hat{\theta}_w)$ is estimated without using the source model, and the resulting prediction model becomes $\hat{y}(x) = f_w(x;\hat{\theta}_w)$. This corresponds to an ordinary regression procedure. When negative transfer occurs, i.e., the previous learning experience interferes with learning in the new task, the desirable hyperparameters are approximately $\tau = 0$ and $\rho = 0$. In addition, by setting $\rho = 1$ (the red line in Figure 3.1), the source model alone gives the prediction model as $\hat{y}(x) = f_s(x)$, regardless of $f_w(x;\theta_w)$. By cross-validating the hyperparameters, the proposed framework will automatically determine when not to transfer without using a separate pipeline.

3.4 Selection of hyperparameters

As described above, the proposed method can hybridize various mechanisms of TL by adjusting τ and ρ. The values of the hyperparameters are adjusted through cross-validation. Clearly, the optimal combination of the hyperparameters will differ depending on the cross-domain relationships and choice of the density-ratio model.

Here, we present an expression for the mean squared error (MSE) based on bias-variance decomposition. For simplicity, we restrict $f_w(x;\hat{\theta}_w)$ to occur in the set of all linear predictions taking the form of $f_w(x;\hat{\theta}_w) = x^\top \mathrm{S}\mathbf{z}$. The $n \times n$ smoothing matrix S depends on n samples of p input features $\phi(x_i) \in \mathbb{R}^p$ $(i = 1,\ldots,n)$ with a predefined basis set ϕ, and $\mathbf{z}$ is a vector of n transformed outputs z_i $(i = 1,\ldots,n)$. An example of this class of prediction is kernel ridge regression.

We assume that y follows $y = f_t(x) + \epsilon$, where $f_t(x)$ denotes the true model and the observation noise ϵ has mean zero and variance σ^2. For the prediction function

$\hat{y}(x) = (1-\rho) f_w(x; \hat{\theta}_w) + \rho f_s(x)$, $\mathrm{MSE}(\hat{y}(x)) = \mathbb{E}_{y|x}[y - \hat{y}(x)]^2$ can be expressed as

$$\begin{aligned}\mathrm{MSE}(\hat{y}(x)) = &\left[\frac{\rho-\tau}{1-\tau}\mathrm{D}(x) + \frac{1-\rho}{1-\tau}\mathrm{B}_1(x) - \frac{\tau(1-\rho)}{1-\tau}\mathrm{B}_2(x)\right]^2 \\ &+ \left(\frac{1-\rho}{1-\tau}\right)^2 \mathrm{V}(x) + \sigma^2,\end{aligned} \tag{3.12}$$

where

$$\begin{aligned}\mathrm{D}(x) &= f_t(x) - f_s(x), \\ \mathrm{B}_1(x) &= f_t(x) - x^\top \mathrm{S}\mathbf{f}_t, \\ \mathrm{B}_2(x) &= f_s(x) - x^\top \mathrm{S}\mathbf{f}_s, \\ \mathrm{V}(x) &= \sigma^2 x^\top \mathrm{S}\mathrm{S}^\top x.\end{aligned} \tag{3.13}$$

The derivation is described in detail later. The first term is the squared bias, which consists of three building blocks. $\mathrm{D}(x)$ represents the discrepancy between $f_t(x)$ and $f_s(x)$. $\mathrm{B}_1(x)$ is a bias of the linear estimator $x^\top \mathrm{S}\mathbf{f_t}$ with respect to the true model $f_t(x)$, assuming that n observations $\mathbf{f}_t = [f_t(x_1), \ldots, f_t(x_n)]^\top$ are given for the unknown $f_t(x)$. Likewise, $\mathrm{B}_2(x)$ is the bias of $x^\top \mathrm{S}\mathbf{f}_s$ with respect to $f_s(x)$. The second term corresponds to the variance of $\hat{y}(x)$. This is proportional to $\mathrm{V}(x) = \sigma^2 x^\top \mathrm{S}\mathrm{S}^\top x$. The third term is the variance of the observation noise.

The relative magnitudes of $\mathbb{E}_x[\mathrm{D}(x)^2]$, $\mathbb{E}_x[\mathrm{B}_1(x)^2]$, $\mathbb{E}_x[\mathrm{B}_2(x)^2]$, and $\mathbb{E}_x[\mathrm{V}(x)]$ determine the optimal hyperparameters for the cross-domain similarity regularization, density-ratio TL, and learning without transfer. Let $\mathrm{D} = \mathrm{D}(x)$, $\mathrm{B}_1 = \mathrm{B}_1(x)$, $\mathrm{B}_2 = \mathrm{B}_2(x)$, and $\mathrm{V} = \mathrm{V}(x)$. Consider the expectation of the MSE in Eq. (3.12) with respect to the marginal distribution of x: $\mathbb{E}_{q(x)}[\mathrm{MSE}(\hat{y}(x))]$. Because the expected MSE is quadratic with respect to ρ for any τ, the minimum under the inequality constraint $0 \leq \rho \leq 1$ is achieved by

$$\rho(\tau) = \begin{cases} 0 & \rho_*(\tau) \leq 0 \\ \rho_*(\tau) & 0 < \rho_*(\tau) < 1 \\ 1 & \rho_*(\tau) \geq 1 \end{cases},$$

where $\rho_*(\tau)$ denotes the solution for the unconstrained minimization. Taking the derivative of the expected MSE with respect to ρ, we have the following:

$$\begin{aligned}&\frac{1}{(1-\tau)^2}\mathbb{E}[((\rho-\tau)\mathrm{D} + (1-\rho)\mathrm{B}_1 - \tau(1-\rho)\mathrm{B}_2))(\mathrm{D} - \mathrm{B}_1 + \tau\mathrm{B}_2)] - \frac{1-\rho}{(1-\tau)^2}\mathbb{E}[\mathrm{V}] \\ &= 0.\end{aligned} \tag{3.14}$$

Assuming that $\tau \neq 1$, this leads to an expression for the unconstrained solution as

$$\rho_*(\tau) = \frac{\mathbb{E}[(\tau \mathrm{D} - \mathrm{B}_1 + \tau \mathrm{B}_2)(\mathrm{D} - \mathrm{B}_1 + \tau \mathrm{B}_2)] + \mathbb{E}[\mathrm{V}]}{\mathbb{E}[\mathrm{D} - \mathrm{B}_1 + \tau \mathrm{B}_2]^2 + \mathbb{E}[\mathrm{V}]}. \tag{3.15}$$

Likewise, taking the derivative of the expected MSE with respect to τ, we have

$$\frac{1-\rho}{(1-\tau)^3}\mathbb{E}[((\rho-\tau)\mathrm{D} + (1-\rho)\mathrm{B}_1 - \tau(1-\rho)\mathrm{B}_2))(\mathrm{D} - \mathrm{B}_1 + \mathrm{B}_2)] - \frac{(1-\rho)^2}{(1-\tau)^2}\mathbb{E}[\mathrm{V}] = 0. \tag{3.16}$$

Combining Eq. (3.14) and Eq. (3.16), where $\tau \neq 1$ and $\rho \neq 1$, we obtain

$$(1-\tau)\mathbb{E}[\tau(\mathrm{D} + (1-\rho)\mathrm{B}_2)\mathrm{B}_2 - (1-\rho)\mathrm{B}_1\mathrm{B}_2 + \rho \mathrm{D}\mathrm{B}_2] = 0. \tag{3.17}$$

Thus, the solution is expressed as

$$\tau(\rho) = \frac{(1-\rho)\mathbb{E}[\mathrm{B}_1\mathrm{B}_2] + \rho\mathbb{E}[\mathrm{D}\mathrm{B}_2]}{(1-\rho)\mathbb{E}[\mathrm{B}_2^2] + \mathbb{E}[\mathrm{D}\mathrm{B}_2]}. \tag{3.18}$$

According to the two expressions in Eq. (3.15) and Eq. (3.18), we can investigate the preference in the hyperparameter selection in regard to the bias and variance components in the data generation process.

Consider a case where the source and target models are significantly different by taking the limit $\mathbb{E}[\mathrm{D}^2] \to \infty$. For the expectation of $\mathbb{E}[\mathrm{D}\mathrm{X}]$ for the product of D and any $\mathrm{X} \in \{\mathrm{B}_1, \mathrm{B}_2\}$, it holds that $\mathbb{E}[\mathrm{D}\mathrm{X}]/\mathbb{E}[\mathrm{D}^2] \to 0$ as $\mathbb{E}[\mathrm{D}^2] \to \infty$. This can be explained by considering the Cauchy–Schwarz inequality, as follows:

$$-\mathbb{E}[\mathrm{D}^2]^{\frac{1}{2}}\mathbb{E}[\mathrm{X}^2]^{\frac{1}{2}} \leq \mathbb{E}[\mathrm{D}\mathrm{X}] \leq \mathbb{E}[\mathrm{D}^2]^{\frac{1}{2}}\mathbb{E}[\mathrm{X}^2]^{\frac{1}{2}} \Leftrightarrow -\frac{\mathbb{E}[\mathrm{X}^2]^{\frac{1}{2}}}{\mathbb{E}[\mathrm{D}^2]^{\frac{1}{2}}} \leq \frac{\mathbb{E}[\mathrm{D}\mathrm{X}]}{\mathbb{E}[\mathrm{D}^2]} \leq \frac{\mathbb{E}[\mathrm{X}^2]^{\frac{1}{2}}}{\mathbb{E}[\mathrm{D}^2]^{\frac{1}{2}}}.$$

In the right-hand side, the upper- and lower-bounds go to zero as $\mathbb{E}[\mathrm{D}^2] \to \infty$. Thus, in Eq. (3.15), all terms except those having $\mathbb{E}[\mathrm{D}^2]$, which appear in its numerator and denominator, approach asymptotically to zero, which results in

$$\rho_*(\tau) \to \frac{\tau\mathbb{E}[\mathrm{D}^2]}{\mathbb{E}[\mathrm{D}^2]} = \tau \quad \text{as} \quad \mathbb{E}[\mathrm{D}^2] \to \infty.$$

Furthermore, noting that $\mathbb{E}[\mathrm{D}\mathrm{X}] = \mathcal{O}(\mathbb{E}[\mathrm{D}^2]^{\frac{1}{2}})$, it can be seen that $\tau(\rho)$ in Eq. (3.18) approaches ρ asymptotically.

$$\tau(\rho) \to \frac{\rho\mathbb{E}[\mathrm{D}\mathrm{B}_2]}{\mathbb{E}[\mathrm{D}\mathrm{B}_2]} = \rho \quad \text{as} \quad \mathbb{E}[\mathrm{D}^2] \to \infty.$$

Therefore, when $\mathbb{E}[\mathrm{D}^2]$ dominates the other three quantities, the density-ratio TL ($\tau = \rho$) is preferred. This fact accounts for the experimental observations discussed later.

In contrast, if the source and target models are entirely the same ($\mathbb{E}[\mathrm{D}^2] = 0$), it holds that $\rho_*(\tau) = 1$. Alternatively, if $\mathbb{E}[\mathrm{V}] \to \infty$, $\rho_*(\tau) = 1$. The direct use of the source model as a prediction function tends to be optimal as the similarity between the source and target tasks increases or the variance $\mathbb{E}[\mathrm{V}]$ becomes larger. The conditions under which cross-domain similarity regularization is preferred have not been clarified theoretically or experimentally.

Derivation details of Eq. (3.12)

Here, we explain the derivation of the mean squared error decomposition of the proposed TL procedure, as shown in Eq. (3.12). It is well-known that the mean squared error $\mathrm{MSE}(\hat{y}(x)) = \mathbb{E}[y - \hat{y}(x)]^2$ can be decomposed as

$$\mathbb{E}[y - \hat{y}(x)]^2 = \big(\mathrm{Bias}[\hat{y}(x)]\big)^2 + \mathrm{Var}[\hat{y}(x)] + \sigma^2, \tag{3.19}$$

where $\mathrm{Bias}[\hat{y}(x)] = f_t(x) - \mathbb{E}[\hat{y}(x)]$ and $\mathrm{Var}[\hat{y}(x)] = \mathbb{E}[\hat{y}(x) - \mathbb{E}[\hat{y}(x)]]^2$. For the proposed framework, these terms are written as follows:

$$\begin{aligned}
\mathrm{Bias}[\hat{y}(x)] &= f_t(x) - \mathbb{E}[\hat{y}(x)] \\
&= f_t(x) - (1-\rho)\mathbb{E}[\hat{f}_w(x)] - \rho f_s(x) \\
&= f_t(x) - (1-\rho)\left\{ \frac{f_t(x) - \tau f_s(x)}{1-\tau} - \frac{f_t(x) - \tau f_s(x)}{1-\tau} + \mathbb{E}[\hat{f}_w(x)] \right\} - \rho f_s(x) \\
&= f_t(x) - (1-\rho)\left\{ \frac{f_t(x) - \tau f_s(x)}{1-\tau} - \mathrm{Bias}[\hat{f}_w(x)] \right\} - \rho f_s(x) \\
&= \left\{ 1 - \frac{1-\rho}{1-\tau} \right\} f_t(x) - \left\{ -\frac{\tau(1-\rho)}{1-\tau} + \rho \right\} f_s(x) + (1-\rho)\mathrm{Bias}[\hat{f}_w(x)] \\
&= \frac{\rho - \tau}{1-\tau}(f_t(x) - f_s(x)) + (1-\rho)\mathrm{Bias}[\hat{f}_w(x)],
\end{aligned} \tag{3.20}$$

$$\begin{aligned}
\mathrm{Var}[\hat{y}(x)] &= \mathbb{E}\big[\hat{y}(x) - \mathbb{E}[\hat{y}(x)]\big]^2 \\
&= \mathbb{E}\big[(1-\rho)\hat{f}_w(x) + \rho f_s(x) - \mathbb{E}[(1-\rho)\hat{f}_w(x) + \rho f_s(x)]\}\big]^2 \\
&= \mathbb{E}\big[(1-\rho)(\hat{f}_w(x) - \mathbb{E}[\hat{f}_w(x)])\big]^2 \\
&= (1-\rho)^2 \mathrm{Var}[\hat{f}_w(x)],
\end{aligned} \tag{3.21}$$

where $\text{Bias}[\hat{f}_w(x)] = \frac{f_t(x)-\tau f_s(x)}{1-\tau} - \mathbb{E}[\hat{f}_w(x)]$, and $\mathbb{E}[\hat{f}_w(x) - \mathbb{E}[\hat{f}_w(x)]]^2$. For the linear predictor $f_w(x;\hat{\theta}_w) = x^\top \text{S}\mathbf{z}$, we can proceed as

$$\begin{aligned}
\text{Bias}[\hat{f}_w(x)] &= \frac{f_t(x)-\tau f_s(x)}{1-\tau} - \mathbb{E}\left[x^\top \text{S}\frac{\mathbf{y}-\tau\mathbf{f}_s}{1-\tau}\right] \\
&= \frac{1}{1-\tau}\big[f_t(x) - \mathbb{E}[x^\top \text{S}\mathbf{y}]\big] - \frac{\tau}{1-\tau}\big[f_s(x) - \mathbb{E}[x^\top \text{S}\mathbf{f}_s]\big] \\
&= \frac{1}{1-\tau}\text{B}_1(x) - \frac{\tau}{1-\tau}\text{B}_2(x), \qquad (3.22)
\end{aligned}$$

and

$$\begin{aligned}
\text{Var}[\hat{f}_w(x)] &= \mathbb{E}\big[\hat{f}_w(x) - \mathbb{E}[\hat{f}_w(x)]\big]^2 \\
&= \mathbb{E}\left[x^\top \text{S}\frac{\mathbf{y}-\tau\mathbf{f}_s}{1-\tau} - \mathbb{E}\left[x^\top \text{S}\frac{\mathbf{y}-\tau\mathbf{f}_s}{1-\tau}\right]\right]^2 \\
&= \mathbb{E}\left[\frac{x^\top \text{S}\mathbf{y}}{1-\tau} - \frac{\mathbb{E}[x^\top \text{S}\mathbf{y}]}{1-\tau}\right]^2 \\
&= \left(\frac{1}{1-\tau}\right)^2 x^\top \text{S}\,\mathbb{E}[\mathbf{y} - \mathbb{E}[\mathbf{y}]]^2 \text{S}^\top x \\
&= \left(\frac{1}{1-\tau}\right)^2 \text{V}(x), \qquad (3.23)
\end{aligned}$$

where B_1, B_2 and V are as defined in Eq. (3.13). Substituting Eq. (3.20)-(3.23) into Eq. (3.19) yields Eq. (3.12).

3.5 Experimental results

3.5.1 Illustrative example

Some intrinsic properties of the proposed method are illustrated by presenting numerical examples using artificial data. According to our analysis, the magnitudes of the bias and variance and the hyperparameters that minimize the MSE are interrelated. This link will be demonstrated in this section.

We assumed the true functions on the source and target tasks to be linear as $f_t(x) = x^\top\theta_t$ and $f_s(x) = x^\top\theta_s$, where $x \in \mathbb{R}^{300}$. The true parameters were generated as $\theta_t = \alpha\theta_s + (1-\alpha)\theta_w$, where $\theta_s \sim \mathcal{N}(0, \text{I})$ and $\theta_w \sim \mathcal{N}(0, \text{I})$. The output variable was assumed to follow $y = f_t(x) + \epsilon$, where $x \sim \mathcal{N}(0, \text{I})$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. With the given θ_w and θ_s, we generated $\{(x_i, y_i)\}_{i=1}^n$ with the sample size set to $n = 50$ by randomly sampling x and ϵ. With given θ_w and θ_s, we generated $\{(x_i, y_i)\}_{i=1}^n$ with the

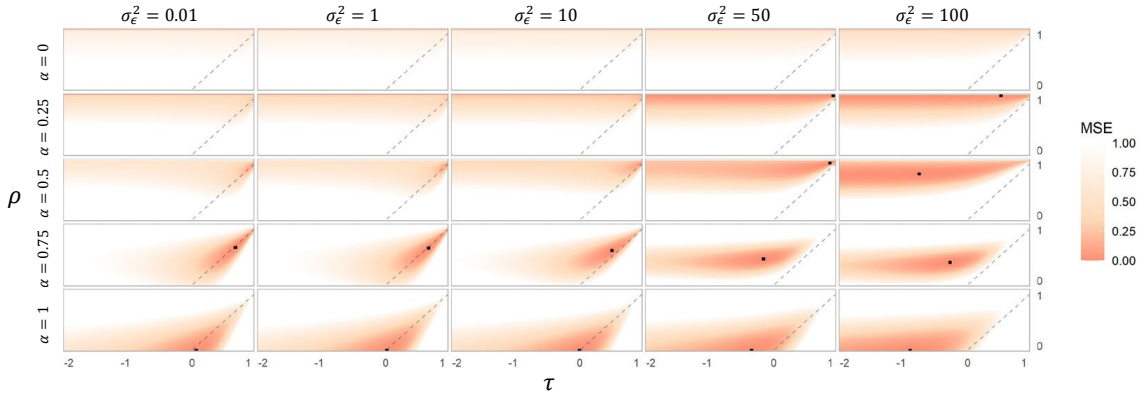


Figure 3.2: Heatmap display of the MSE landscape on the hyperparameter space (τ, ρ), which changes as a function of the bias (α) and variance (σ). With the given τ and ρ, linear ridge regression was used to train $f_w(x; \theta_w)$ on the artificial data. The black dot denotes the lowest MSE.

sample size set to $n = 50$ by randomly sampling x and ϵ. The discrepancy between the source and target models is controlled by the mixing rate $\alpha \in [0, 1]$ for any given θ_w. In particular, if α is set to zero, the source and target models are the same, i.e., $\forall x$: $\mathrm{D}(x) = 0$ in Eq. (3.12). The variance σ^2 of the observational noises affects the magnitude of the variance $\mathbb{E}[\mathrm{V}]$ in the model estimation.

Linear case

For the simplest case, we used linear ridge regression to estimate f_w with the hyperparameter on the ℓ_2-regularization that was fixed at $\lambda = 0.0001$. This case is consistent with the case in Section 3.4. The true source model was used as $f_s(x)$. We then investigated the change in the MSE landscape as a function of the bias α and variance σ, which are summarized in Figure 3.2. For any given values of τ and ρ, the MSE was approximately evaluated by averaging the squared loss over additionally generated 1,000 samples on (x, y) and rescaled to the range in $[0, 1]$. For $\alpha = 0$ at which the source and target models are the same, the MSE became small in the region along $\rho = 1$, which corresponds to the use of the pre-trained source model as the target model with no modification. As α increased while restricting σ to smaller values, the region where the MSE became small was concentrated around $\tau = \rho$, indicating the dominant performance of the density-ratio TL. In contrast, as both α and σ became larger, the region with $\tau < 0$ and $\rho = 0$ tended to be more dominant. This region corresponds to TL with cross-domain similarity regularization. It was confirmed that the pattern of the MSE landscape varies continuously with respect to the bias and variance components.

Nonlinear case

We conduct the same analysis as above for the cases where nonlinear models are assumed for $f_w(x;\theta_w)$, $f_t(x)$, and $f_s(x)$. The analysis in Section 3.4 is not applicable to this case, but it exhibits the same trends as in the linear case. To be specific, we considered three different cases as follows: (a) a random forest is assigned to $f_w(x;\theta_w)$, where the true models of $f_t(x)$ and $f_s(x)$ are assumed to be linear, (b) a linear model is assigned to $f_w(x;\theta_w)$, where the true models are assumed to be nonlinear, and (c) a random forest is assigned to $f_w(x;\theta_w)$, where the true models are assumed to be nonlinear.

To generate artificial data with nonlinearity, we assumed single hidden layer neural networks for the source and target models as

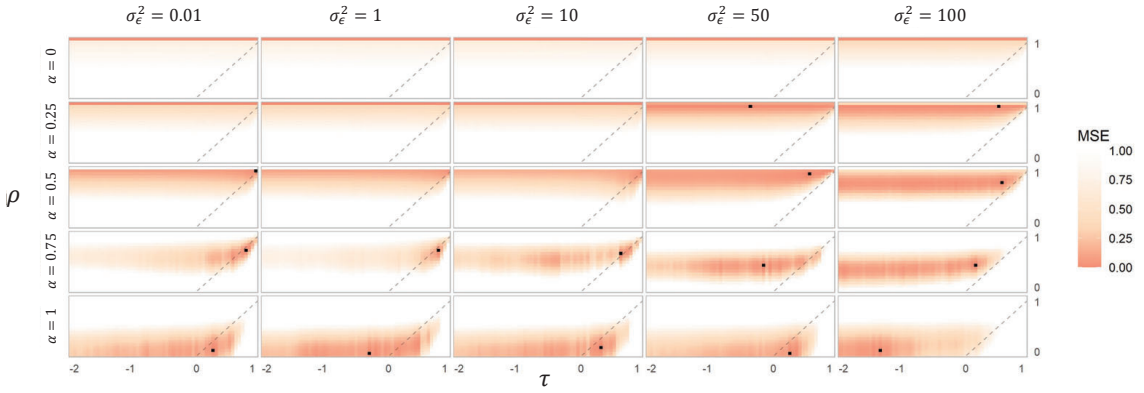
$$\begin{aligned} f_s(x) &= \mathrm{B}_s\varphi(\mathrm{A}_s x), \\ f_t(x) &= \mathrm{B}_t\varphi(\mathrm{A}_t x), \\ \varphi(x) &= \max\{0, x\}. \end{aligned}$$

The weight parameters were generated as $\mathrm{A}_t = \alpha\mathrm{A}_w + (1-\alpha)\mathrm{A}_s$, $\mathrm{B}_t = \alpha\mathrm{B}_w + (1-\alpha)\mathrm{B}_s$, where $\mathrm{A}_s, \mathrm{A}_w \in \mathbb{R}^{50\times 300}$, and $\mathrm{B}_s, \mathrm{B}_w \in \mathbb{R}^{1\times 50}$, and each element of $\mathrm{A}_s, \mathrm{A}_w, \mathrm{B}_s, \mathrm{B}_w$ was drawn from $\mathcal{N}(0, 0.5)$ independently. The output variable was assumed to follow $y = f_t(x) + \epsilon$, where $x \sim \mathcal{N}(0, \mathrm{I})$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We generated 50 samples for the training of $f_w(x;\theta_w)$ and 1,000 samples for the evaluation of the MSE.

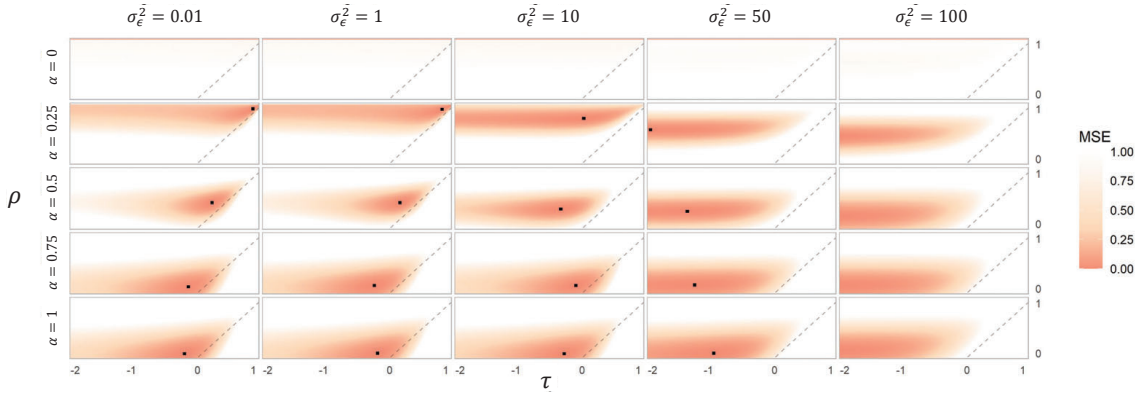
We used linear ridge regression and random forest regression to train $f_w(x;\theta_w)$ with the fixed hyperparameters $\lambda = 0.0001$, $n_{\text{tree}} = 200$ (number of trees), and $n_{\text{variable}} = 100$ (number of randomly selected variables at each split). Figure 3.3 shows the changes in the MSE landscape for various α and σ for each case.

(a) f_t and f_s are linear, f_w is nonlinear When assuming the nonlinear model for $f_w(x;\theta_w)$, a similar trend of the relationship between the hyperparameter preference and magnitudes of the bias and variance components (α and σ) was observed as in the linear case. As α (i.e., $\mathbb{E}_x[\mathrm{D}(x)^2]$) was increased while keeping σ (i.e., $\mathbb{E}_x[\mathrm{V}(x)]$) small, the regions with smaller MSEs were concentrated near $\tau = \rho$. In contrast, as both α and σ were increased, the regions with $\tau < 0$ and $\rho = 0$ became preferable.

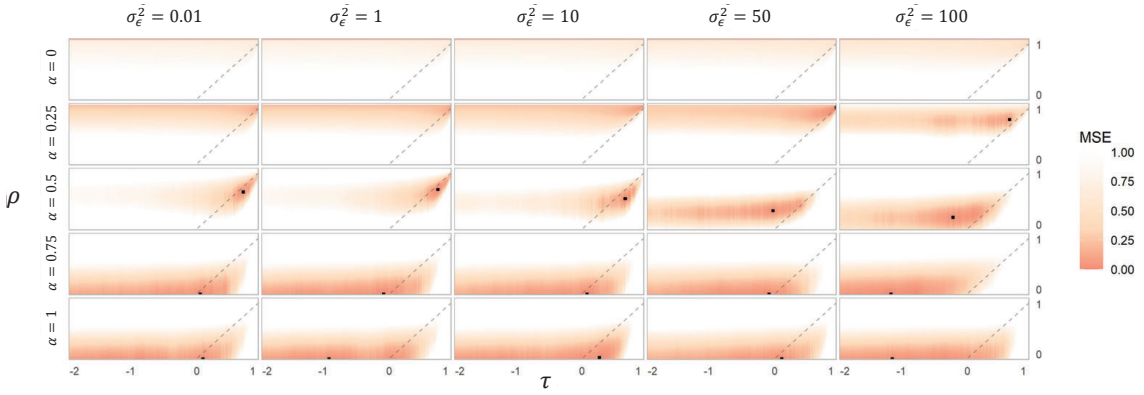
(b) f_t and f_s are nonlinear, f_w is linear In this case, the same argument as Section 3.4 holds because the analysis shown in Section 3.4 does not place any specific assumption on the mathematical forms of $f_t(x)$ and $f_s(x)$. However, in the lower left panel of Figure 3.3 (the case where α is large and σ is small), the best hyperparameters are located slightly off the diagonal. This is because the linear model $f_w(x;\theta_w)$ cannot capture the nonlinearity of $f_t(x)$ and $f_s(x)$; thus, $\mathbb{E}_x[\mathrm{B}_1(x)^2]$ and $\mathbb{E}_x[\mathrm{B}_2(x)^2]$ do not become smaller.



(a) $f_t(x)$, $f_s(x)$: linear models; $f_{\theta_w}(x)$: random forest regression



(b) $f_t(x)$, $f_s(x)$: neural networks; $f_{\theta_w}(x)$: linear ridge regression



(c) $f_t(x)$, $f_s(x)$: neural networks; $f_{\theta_w}(x)$: random forest regression

Figure 3.3: Heatmap display of the MSE landscape on the hyperparameter space (τ, ρ) in under three different settings, where different models were assumed for $f_t(x)$, $f_s(x)$, and $f_w(x;\theta_w)$. The black dot denotes the lowest MSE.

(c) f_t, f_s, and f_w are nonlinear As in (a), the pattern of change in the MSE with respect to α and σ was similar to that in the linear case. Assuming the nonlinear model for $f_w(x;\theta_w)$, we can reduce $\mathbb{E}_x[\mathrm{B}_1(x)^2]$ and $\mathbb{E}_x[\mathrm{B}_2(x)^2]$ to a greater extent than in the case where the linear model is assumed for $f_w(x;\theta_w)$. Hence, the region near the density-ratio TL becomes more favorable when α is larger and σ is smaller.

3.5.2 Real data applications

Task and data

The proposed method was applied to five real data analyses in materials science and robotics applications: (i) the prediction of multiple properties of organic polymers and inorganic compounds (Yamada et al., 2019), (ii) the prediction of multiple properties of polymers (Kim et al., 2018) and low-molecular-weight compounds (monomers, unpublished data), (iii) the prediction of properties of donor molecules in organic solar cells (Paul et al., 2019), obtained from experiments (Lopez et al., 2016) and quantum chemical calculations (Pyzer-Knapp et al., 2015), (iv) the prediction of formation energies of various inorganic compounds and crystal polymorphisms of SiO_2 and CdI_2 (Jain et al., 2013), and (v) the prediction of the feed-forward torques required to follow the desired trajectory at seven joints of a SARCOS anthropomorphic robot arm (Williams & Rasmussen, 2006). The model transfers were conducted exhaustively between all task pairs within each application, which resulted in a total of 185 pairs of the source and target tasks with 9 different combinations of $f_s(x)$ and $f_w(x;\theta_w)$ (a total of 1,665 cases).

Polymers and inorganic compounds The task was to predict five properties (band gap, dielectric constant, refractive index, density, and volume) of inorganic compounds and six properties (band gap, dielectric constant, refractive index, density, volume, and atomization energy) of polymers. The number of pairs for the source and target tasks to be transferred was $110 = 11 \times 10$. The overall datasets represent the structure–property relationships for 1,056 inorganic compounds and 1,070 polymers. See Yamada et al. (2019) for more details on the datasets. The structural information for all the materials was ignored. Only the compositional features were encoded into the 290-dimensional input descriptors by using XenonPy, an open-source platform of materials informatics for Python (Liu et al., 2021)[1].

Polymers and small molecules The task was to predict three properties (band gap, dielectric constant, and refractive index) of polymers and three properties (HOMO-LUMO gap, dielectric constant, and refractive index) of small organic molecules. The

[1] https://github.com/yoshida-lab/XenonPy

number of paired tasks was $30 = 6 \times 5$. The polymeric data consisted of 854 polymers. By performing quantum chemistry calculations based on density functional theory using the Gaussian 09 suite of program codes (Frisch et al., 2016), we produced a dataset on the three properties of 854 small organic molecules that corresponded to the constitutional repeating units of the 854 polymers. In the density functional theory (DFT) calculation, the molecular geometries were optimized at the B3LYP/6-31+G(d) theoretical level of theory. The chemical structure of each monomer was encoded into a descriptor vector of 1,905 binary digits by using two molecular fingerprinting algorithms that refer to PubChem and circular fingerprints implemented in the rcdk package on R (Guha, 2007).

CEP and HOPV The task was to predict the highest occupied molecular orbital (HOMO) energy for donor molecules in an organic solar cell devise. We used two datasets for the HOMO energy levels of 2,322,649, and 351 molecules. The former dataset was obtained from the high-throughput quantum chemistry calculations conducted by the Harvard clean energy project (CEP) (Pyzer-Knapp et al., 2015) and the latter was a collation of the experimental photovoltaic data from the literature, referred to as the Harvard Organic Photovoltaic Dataset (HOPV15) (Lopez et al., 2016). We used the fingerprints of the second task to represent the input chemical structures.

Formation energy of SiO_2 and other compounds We used a dataset of the Materials Project (Jain et al., 2013), which contains the DFT formation energies of 69,641 inorganic compounds. The input crystal structures were translated by the 441-dimensional descriptors obtained by concatenating the 290-dimensional compositional descriptors and 151-dimensional radial distribution function descriptors in XenonPy. We first derived a pre-trained source model by using 80% of the 69,358 training instances after removing 283 instances corresponding to SiO_2. Such a global model originating from the large dataset was transferred to a localized target model on SiO_2 by using the remaining small dataset.

SARCOS robot arm The task was to predict the feed-forward torques required to follow the desired trajectory at seven joints of a SARCOS anthropomorphic robot arm (Williams & Rasmussen, 2006). The number of paired tasks was 42. The dataset contained a total of 44,484 and 4,449 instances for training and testing. The 21 input features described the position, velocity, and acceleration at the seven joints.

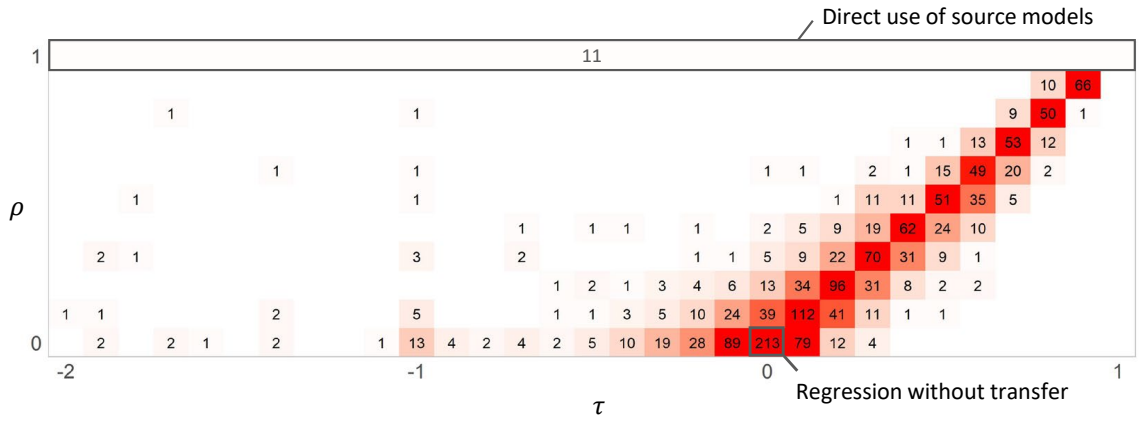


Figure 3.4: Distribution of (τ, ρ) that delivered the lowest MSE in 1,665 cases (185 task pairs and 3^2 combinations of models for $f_s(x)$ and $f_w(x; \theta_w)$). The number in each pixel denotes the count of cases.

Analytical procedure

For each task pair, we used three machine learning algorithms—ridge regression using a linear model (LN), random forest (RF), and neural network (NN)—to estimate $f_s(x)$ and $f_w(x; \theta_w)$. In the source task, the entire dataset was used to train $f_s(x)$ under the default settings of the software packages without adjusting the hyperparameters. In all cases, 50 randomly selected samples were used to train $f_w(x; \theta_w)$. We selected the best model based on five-fold cross-validation. The resulting model was used to predict the remaining data, and the MSE was evaluated.

Results

For each of the 1,665 cases, we investigated the distribution of the hyperparameters selected by the cross-validation (Figure 3.4). In many cases, the distribution of the selected hyperparameters was concentrated in the neighboring areas of density-ratio TL ($\tau = \rho$) and cross-domain similarity regularization ($\tau < 0, \rho = 0$). Density-ratio TL was selected for 609 cases (36.6%) and cross-domain similarity regularization was selected for 176 cases (10.6%). In particular, there was a significant bias toward the neighbors of $\tau = \rho$.

The selected hyperparameters and MSEs for the 1,665 cases are presented in Tables 3.2–3.6. As an illustrative example, Table 3.1 shows the results of the TL from one source task (prediction of dielectric property of small molecules) to five target tasks (prediction of two properties of small molecules and three properties of polymers). This result also indicates the presence of bias toward τ and ρ. It was also observed that in some cases, the choice of the density-ratio model significantly affected the

Source task	Target task	$f_s(x)$	$f_w(x;\theta_w)$			Selected hyperparameters		
			LN	RF	NN	LN	RF	NN
Monomer - Dielectric constant	Monomer - HOMO-LUMO gap	LN	0.8292	0.7435	0.8823	(-0.1, 0.1)	(0.6, 0.4)	(0.1, 0.3)
		RF	0.8302	**0.7139**	0.7421	(-0.1, 0.2)	(0.5, 0.3)	(0.8, 0.8)
		NN	0.8250	0.7372	0.7644	(-0.2, 0.2)	(0.2, 0.3)	(0.4, 0.4)
	Monomer - Refractive index	LN	0.0436	0.0424	0.0439	(0.8, 0.9)	(0.8, 0.9)	(0.8, 0.9)
		RF	0.0463	0.0415	0.0415	(0.9, 0.9)	(- , 1.0)	(- , 1.0)
		NN	0.0365	**0.0355**	0.0505	(0.8, 0.9)	(0.8, 0.9)	(0.4, 0.7)
	Polymer - Band gap	LN	1.0881	0.7862	0.8936	(0.3, 0.1)	(0.0, 0.1)	(0.6, 0.6)
		RF	0.8594	0.7477	**0.7130**	(-0.2, 0.4)	(0.4, 0.3)	(0.8, 0.8)
		NN	0.8654	0.8598	0.8908	(-0.5, 0.1)	(0.3, 0.5)	(0.6, 0.5)
	Polymer - Dielectric constant	LN	0.6031	**0.5358**	0.6376	(-0.4, 0.2)	(0.3, 0.2)	(-0.5, 0.0)
		RF	0.5988	0.5786	0.6678	(-0.2, 0.2)	(0.3, 0.2)	(0.0, 0.4)
		NN	0.6143	0.5478	0.7563	(-0.1, 0.2)	(0.2, 0.3)	(-0.2, 0.1)
	Polymer - Refractive index	LN	**0.3269**	0.3906	0.3442	(0.0, 0.0)	(-0.4, 0.0)	(0.2, 0.4)
		RF	**0.3269**	0.3574	0.3312	(0.0, 0.0)	(0.1, 0.1)	(0.1, 0.2)
		NN	**0.3269**	0.3845	0.4254	(0.0, 0.0)	(-0.1, 0.1)	(-1.7, 0.0)

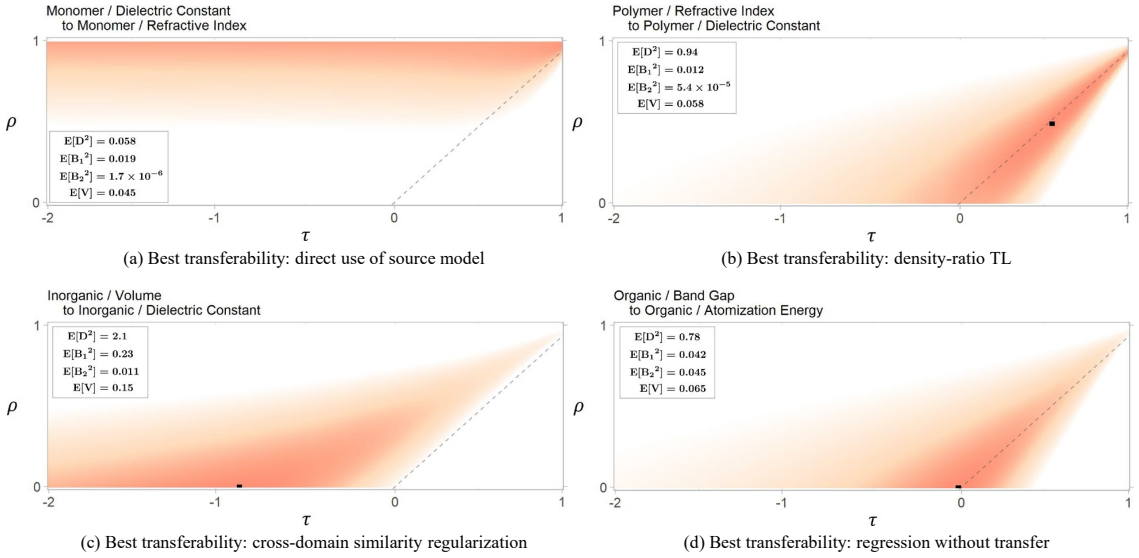
Table 3.1: Selected hyperparameters (the last three columns representing the hyperparameters τ and ρ) and their corresponding MSEs (columns 4–6) for TL from one source task to five target tasks. Three different models (LN: linear, RF: random forest, and NN: neural network) were applied to $f_s(x)$ and $f_w(x;\theta_w)$. Tables 3.2-3.6 provide the full results for all the 1,665 cases.

prediction performance.

We speculate that the four quantities $\mathbb{E}_x[\mathrm{D}^2]$, $\mathbb{E}_x[\mathrm{B}_1^2]$, $\mathbb{E}_x[\mathrm{B}_2^2]$, and $\mathbb{E}_x[\mathrm{V}]$ or their counterparts in general regression, determine the preference of τ and ρ. Figure 3.5 shows the MSE mapped on the hyperparameter space and the four quantities for the four task pairs. They were selected as the typical cases where the four different learning schemes are preferred. The proposed method exhibited a preference to directly use the source models when the difference between the source and target domains ($\mathbb{E}_x[\mathrm{D}^2]$) was small. When $\mathbb{E}_x[\mathrm{D}^2]$ was large, the relative magnitude of $\mathbb{E}_x[\mathrm{D}^2]$ and the other three quantities $\mathbb{E}_x[{\mathrm{B}_1}^2]$, $\mathbb{E}_x[{\mathrm{B}_2}^2]$, and $\mathbb{E}_x[\mathrm{V}]$ determined the selection of the source model; if $\mathbb{E}_x[\mathrm{V}]$ was small, density-ratio TL was preferred, and if $\mathbb{E}_x[\mathrm{V}]$ was large, cross-domain similarity regularization was preferred. Furthermore, when both $\mathbb{E}_x[{\mathrm{B}_1}^2]$ and $\mathbb{E}_x[\mathrm{V}]$ were small, training without transfer was preferred. Such relationships were often observed in other cases as well. However, these inferences were derived from partial observations, and there would be more complex factors to work in the learning mechanism.

Remarks: preference of hyperparameters

In the real data applications, we investigated the relationship between the selected hyperparameters and the bias and variance inherent in the data for the 555 ($= 3 \times 185$)



(a) Best transferability: direct use of source model

(b) Best transferability: density-ratio TL

(c) Best transferability: cross-domain similarity regularization

(d) Best transferability: regression without transfer

Figure 3.5: MSE landscapes of the hyperparameter space for four different cases that exhibited the best transferability in different hyperparameter sets. Sample estimates on three bias-related quantities ($\mathbb{E}_x[\mathrm{D}^2]$, $\mathbb{E}_x[\mathrm{B}_1{}^2]$, and $\mathbb{E}_x[\mathrm{B}_2{}^2]$), and the mean variance ($\mathbb{E}_x[\mathrm{V}]$) are shown on each plot.

cases out of a total of 1,665 cases, where the linear model was assumed for the density-ratio model. If we assume linearity, as described in Section 3.4, the MSE can be expressed as Eq.(3.12). Here, we focused on the relative magnitudes of $\mathbb{E}_x[\mathrm{D}(x)^2]$ and $\mathbb{E}_x[\mathrm{V}(x)]$. The expected value of $\mathbb{E}_x[\mathrm{D}(x)]^2$ was approximated by the mean of 500 samples randomly selected from the test data. For $\mathbb{E}_x[\mathrm{V}(x)]$, the variance in the linear predictor function was calculated using 100 bootstrap sets extracted from the training data. We divided the 555 cases into 16 ($= 4 \times 4$) groups according to the quartiles of $\mathbb{E}_x[\mathrm{D}(x)^2]$ and $\mathbb{E}_x[\mathrm{V}(x)^2]$. The thresholds for each interval and distribution of the selected τ and ρ for each group are shown in Figure 3.6. A striking trend was observed, in which the hyperparameters were significantly concentrated in the domain of density-ratio TL as $\mathbb{E}_x[\mathrm{D}(x)^2]$ increased relative to $\mathbb{E}_x[\mathrm{V}(x)]$ ($\mathbb{E}_x[\mathrm{D}(x)^2]/\mathbb{E}_x[\mathrm{V}(x)] \to \infty$). In contrast, as $\mathbb{E}_x[\mathrm{V}(x)]$ increased, some of the selected hyperparameters appeared in the domain of cross-domain similarity regularization. Nevertheless, many several hyperparameters were still distributed in the region of density-ratio TL. The trend remained unclear when compared with the case of $\mathbb{E}_x[\mathrm{D}(x)^2]/\mathbb{E}_x[\mathrm{V}(x)] \to \infty$.

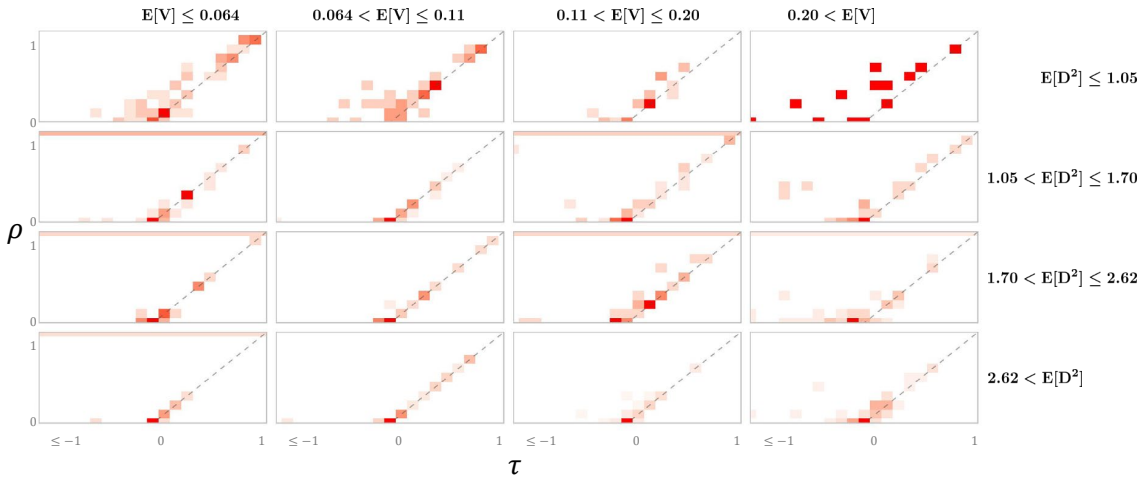


Figure 3.6: Distribution of the selected hyperparameters in 555 cases where the linear model was assumed for $f_w(x;\theta_w)$. The cases were grouped according to the four intervals of $\mathbb{E}[\mathrm{D}^2]$ or $\mathbb{E}[\mathrm{V}]$, which was approximately evaluated by calculating the sample average on the test data. The intervals were determined based on the quantile values of the two quantities. The resulting 16 panels are separately shown. The colors refer to the relative frequency of each cell.

3.6 Summary and future perspectives

In this chapter, we proposed a new class of TL characterized by two hyperparameters, which in turn control the training and prediction procedure. This new class of TL unifies two different types of existing methods that are based on cross-domain similarity regularization and density-ratio estimation. If we use neural networks on the source and target models, the class represents the fine-tuning of the neural networks. In addition, some specific selection of hyperparameters offers the choice of ordinary regression without transfer or direct use of a pre-trained source model as the target. According to the selection of hyperparameters and models, we can derive various learning methods in which these two methods are hybridized.

Cross-domain similarity regularization and density-ratio TL follow opposite learning objectives. In the former case, the target model is regularized to be closer to the source model. In the latter case, the target model is estimated to differ significantly from the source model. Most of the widely used techniques have adopted the former approach that leverages the proximity of the target model to the source model. Interestingly, in many cases, cross-domain similarity regularization rarely exhibited the best transferability according to our empirical study; moreover, density-ratio estimation or its neighboring areas in the hyperparameter space often showed better performance. Although the idea of cross-domain similarity regularization is more widely adopted,

our results indicate that we should further explore the direction based on a different concept, such as density-ratio estimation.

This chapter focused on regression setting. In addition, in the Bayesian framework, we assumed a specific type of likelihood and prior distribution. The empirical risk derived from this assumption takes the sum of the squared loss. With this formulation, we could perform the model training simply by using an existing library for regression. This allows us to keep the implementation cost to practically zero. However, there are also limitations to using the squared loss. We should consider a wide range of loss functions and learning tasks. The treatment of more generic loss functions and discriminant problems will be considered in future research.

Table 3.2: Transfer between various properties of organic polymers and inorganic solid-state materials

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Inorganic - Band gap	Inorganic - Density	LN	8.3248	0.8502	1.0358	**0.8048**	0.8502	0.9970	1.0827	(0.0, 0.0)	(0.1, 0.0)	(0.7, 0.7)
		RF	8.5122				0.8167	1.0142	0.9452	(-0.1, 0.0)	(0.7, 0.5)	(0.2, 0.3)
		NN	8.7239				0.8248	1.0358	1.6049	(0.0, 0.1)	(0.0, 0.0)	(0.0, 0.3)
	Inorganic - Dielectric constant	LN	29.2315	19.8347	19.7574	19.788	20.1473	19.6531	19.6041	(0.3, 0.2)	(0.6, 0.5)	(0.1, 0.2)
		RF	39.1505				19.7176	19.7501	20.0451	(-0.7, 0.3)	(-1.9, 0.3)	(-0.9, 0.0)
		NN	41.2346				**19.4657**	19.8699	19.9731	(0.1, 0.3)	(-1.0, 0.0)	(0.1, 0.4)
	Inorganic - Refractive index	LN	5.2043	1.1319	0.9526	1.2084	1.1109	0.8972	0.7504	(0.2, 0.2)	(0.3, 0.4)	(-1.1, 0.0)
		RF	5.2723				1.1189	0.9347	0.8061	(0.2, 0.2)	(0.5, 0.5)	(0.6, 0.5)
		NN	5.5560				1.1324	0.8571	**0.7091**	(0.2, 0.2)	(0.3, 0.4)	(0.2, 0.4)
	Inorganic - Volume	LN	338.9054	40.292	44.7381	**37.0925**	40.292	45.3988	38.4877	(0.0, 0.0)	(0.3, 0.1)	(0.1, 0.1)
		RF	333.5537				40.292	48.3426	44.4460	(0.0, 0.0)	(0.3, 0.1)	(0.1, 0.0)
		NN	353.4228				40.292	44.7381	92.3290	(0.0, 0.0)	(0.0, 0.0)	(0.1, 0.0)
	Organic - Atomization energy	LN	0.7324	0.1250	0.1837	0.1363	0.7324	0.1912	0.2171	(-2.0, 1.0)	(0.1, 0.0)	(0.1, 0.0)
		RF	0.4507				0.4507	0.1872	0.1639	(-2.0, 1.0)	(0.3, 0.2)	(0.6, 0.6)
		NN	11.5813				0.1519	0.1901	0.2338	(0.7, 0.7)	(0.4, 0.3)	(0.4, 0.5)
	Organic - Band gap	LN	1.8591	0.8071	0.7708	0.9308	0.9027	0.7999	0.9218	(-0.1, 0.0)	(0.3, 0.3)	(0.3, 0.4)
		RF	1.6421				1.0192	**0.7649**	0.9449	(0.8, 0.8)	(0.2, 0.2)	(0.7, 0.7)
		NN	1.4172				0.8882	0.9094	0.9335	(0.2, 0.2)	(0.7, 0.6)	(0.4, 0.5)
	Organic - Density	LN	0.5968	0.1201	0.1212	0.1240	0.1201	0.1212	0.1318	(0.0, 0.0)	(0.0, 0.0)	(0.5, 0.5)
		RF	0.2423				0.1659	**0.1060**	0.1340	(0.3, 0.3)	(0.6, 0.5)	(0.5, 0.4)
		NN	0.1990				0.1559	0.1075	0.1588	(0.3, 0.3)	(0.3, 0.1)	(0.3, 0.1)
	Organic - Dielectric constant	LN	10.5562	**2.9359**	2.9667	3.3149	2.9637	3.0768	2.9597	(0.4, 0.4)	(0.1, 0.2)	(-0.2, 0.0)
		RF	13.1800				3.0465	2.9949	2.9624	(-0.2, 0.1)	(0.5, 0.6)	(0.6, 0.5)
		NN	12.0098				3.0092	3.0176	3.0861	(0.2, 0.2)	(-0.2, 0.0)	(0.3, 0.4)
	Organic - Refractive index	LN	3.5074	0.1783	0.1614	0.1626	**0.1562**	0.1617	0.1728	(0.9, 0.9)	(0.0, 0.1)	(0.1, 0.1)
		RF	3.7685				0.1759	0.1761	0.1723	(-0.3, 0.0)	(0.6, 0.7)	(0.7, 0.7)
		NN	3.6524				0.1649	0.1654	0.2195	(0.1, 0.1)	(0.0, 0.2)	(0.0, 0.3)
	Organic - Volume	LN	524.0981	79.3299	84.7309	78.499	79.3299	84.7309	71.6098	(0.0, 0.0)	(0.0, 0.0)	(0.5, 0.5)
		RF	165.6267				50.6680	84.7309	70.4401	(0.4, 0.4)	(0.0, 0.0)	(0.7, 0.7)
		NN	592.0768				**48.9811**	84.7309	146.4638	(0.1, 0.1)	(0.0, 0.0)	(0.1, 0.0)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Inorganic - Density	Inorganic - Band gap	LN	4.1057	1.3953	1.2503	1.2386	1.3361	1.2897	1.2386	(-0.1, 0.1)	(-0.4, 0.0)	(0.0, 0.0)
		RF	4.3261				1.4270	1.2712	1.4680	(0.9, 0.9)	(-0.1, 0.1)	(0.6, 0.6)
		NN	4.3858				1.2149	**1.2077**	1.4934	(-0.2, 0.0)	(0.7, 0.6)	(0.0, 0.2)
	Inorganic - Dielectric constant	LN	22.1911	21.8313	21.7312	21.9769	19.7227	19.3058	20.0383	(-0.4, 0.1)	(0.4, 0.3)	(-0.3, 0.2)
		RF	21.2934				19.8611	19.6515	20.0582	(-1.4, 0.1)	(-0.1, 0.1)	(0.0, 0.2)
		NN	21.4118				19.9323	**19.2853**	19.4203	(-0.6, 0.0)	(-0.5, 0.2)	(0.5, 0.4)
	Inorganic - Refractive index	LN	1.1543	0.9263	0.9573	0.9562	0.9186	0.8816	0.9159	(0.3, 0.3)	(0.5, 0.4)	(0.2, 0.3)
		RF	1.1465				0.9313	0.9008	0.8904	(-0.1, 0.1)	(0.5, 0.5)	(0.5, 0.5)
		NN	1.1454				0.8952	**0.8629**	0.9264	(0.2, 0.2)	(0.6, 0.5)	(0.1, 0.1)
	Inorganic - Volume	LN	341.5280	41.4555	41.5915	35.3336	**34.4274**	38.3624	40.3459	(0.5, 0.5)	(0.6, 0.5)	(0.4, 0.4)
		RF	331.0445				34.6462	36.7618	56.7208	(0.3, 0.3)	(0.5, 0.4)	(0.3, 0.2)
		NN	336.0994				36.5000	35.3802	47.0810	(0.1, 0.1)	(0.4, 0.3)	(0.2, 0.1)
	Organic - Atomization energy	LN	11.7117	**0.1226**	0.2043	0.1364	0.1557	0.2066	0.1697	(0.8, 0.8)	(0.2, 0.2)	(0.1, 0.0)
		RF	11.6678				0.2311	0.2043	0.1630	(-0.1, 0.0)	(0.0, 0.0)	(0.5, 0.5)
		NN	11.4813				0.1249	0.2032	0.2882	(0.2, 0.2)	(0.3, 0.2)	(0.2, 0.2)
	Organic - Band gap	LN	5.3678	**0.7949**	0.8134	0.9348	0.8232	0.8119	0.9380	(0.0, 0.1)	(0.4, 0.4)	(0.7, 0.8)
		RF	5.5888				0.8012	0.8070	0.8992	(-0.1, 0.0)	(0.1, 0.0)	(0.4, 0.5)
		NN	5.6123				0.8500	0.8147	0.8755	(-0.1, 0.0)	(0.3, 0.3)	(0.4, 0.4)
	Organic - Density	LN	0.2537	0.0807	0.1077	0.0861	**0.0724**	0.0799	0.0871	(0.4, 0.4)	(0.1, 0.1)	(0.7, 0.7)
		RF	0.2384				0.0789	0.1087	0.0978	(0.4, 0.4)	(0.7, 0.6)	(0.7, 0.7)
		NN	0.2548				0.0780	0.0900	0.0861	(0.1, 0.0)	(0.3, 0.2)	(0.0, 0.0)
	Organic - Dielectric constant	LN	3.6153	3.4843	3.0114	3.4526	2.9672	2.9775	2.9401	(0.1, 0.3)	(-0.3, 0.0)	(0.6, 0.6)
		RF	3.3884				2.9186	2.9675	2.9576	(-0.5, 0.0)	(-0.4, 0.0)	(0.9, 0.9)
		NN	3.1747				**2.8104**	2.8984	3.0289	(-0.3, 0.0)	(-1.9, 0.0)	(0.7, 0.5)
	Organic - Refractive index	LN	0.2915	0.1372	**0.1267**	0.1273	0.1592	0.1429	0.1607	(-0.1, 0.0)	(0.8, 0.8)	(0.2, 0.2)
		RF	0.2760				0.1581	0.1325	0.1469	(-0.1, 0.0)	(0.8, 0.8)	(0.6, 0.5)
		NN	0.2383				0.1584	0.1419	0.1759	(-0.3, 0.0)	(-0.2, 0.0)	(0.5, 0.5)
	Organic - Volume	LN	503.2750	90.1019	34.337	**23.8202**	90.1019	31.9216	60.4470	(0.0, 0.0)	(0.1, 0.0)	(0.5, 0.5)
		RF	515.8328				40.9502	32.0241	51.2754	(0.8, 0.8)	(0.2, 0.1)	(0.4, 0.4)
		NN	233.8719				26.7953	34.1709	86.8750	(0.3, 0.3)	(0.1, 0.1)	(0.1, 0.1)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Inorganic - Dielectric constant	Inorganic - Band gap	LN	4.3650	1.1939	1.2648	**1.1813**	1.3808	1.2426	1.2490	(0.8, 0.8)	(0.0, 0.1)	(0.2, 0.2)
		RF	4.6827				1.2466	1.2092	1.2163	(0.3, 0.5)	(0.9, 0.9)	(0.3, 0.4)
		NN	4.5770				1.3438	1.3265	1.3236	(0.5, 0.6)	(0.5, 0.4)	(0.6, 0.6)
	Inorganic - Density	LN	1.4903	0.8372	0.9838	0.8387	0.8297	**0.8183**	0.8568	(0.4, 0.4)	(0.7, 0.6)	(0.5, 0.6)
		RF	1.4070				0.9654	0.8339	1.1390	(-0.5, 0.0)	(0.7, 0.6)	(0.8, 0.8)
		NN	1.7254				0.9368	0.9836	1.0594	(-0.4, 0.0)	(0.6, 0.4)	(0.4, 0.5)
	Inorganic - Refractive index	LN	1.1117	1.0631	1.1264	1.0667	1.1117	1.0731	1.0155	(-2.0, 1.0)	(0.4, 0.4)	(0.6, 0.7)
		RF	1.0244				0.9892	1.0409	1.0480	(0.6, 0.7)	(0.5, 0.4)	(0.6, 0.6)
		NN	0.9051				0.9051	**0.7884**	0.9845	(-2.0, 1.0)	(0.8, 0.8)	(0.2, 0.2)
	Inorganic - Volume	LN	347.9021	36.8553	35.494	38.4926	70.6305	**31.7758**	41.1168	(-1.4, 0.0)	(0.2, 0.0)	(0.6, 0.6)
		RF	383.6335				40.0470	33.4685	62.3273	(0.4, 0.4)	(0.2, 0.1)	(0.7, 0.6)
		NN	368.0522				37.1534	47.9474	89.9738	(-0.1, 0.0)	(0.5, 0.4)	(-0.1, 0.0)
	Organic - Atomization energy	LN	11.6433	0.1507	0.1757	0.1541	0.1379	0.1748	0.1559	(0.3, 0.3)	(0.1, 0.1)	(0.7, 0.7)
		RF	11.3799				**0.1359**	0.1918	0.1671	(0.1, 0.1)	(0.7, 0.7)	(0.5, 0.5)
		NN	0.4436				0.1472	0.1824	0.3845	(0.1, 0.0)	(0.7, 0.7)	(0.1, 0.1)
	Organic - Band gap	LN	5.5536	1.0116	0.8305	1.0162	0.8597	**0.8235**	1.0263	(0.4, 0.5)	(0.9, 0.9)	(0.7, 0.7)
		RF	6.2796				0.8949	0.8596	1.1001	(-0.2, 0.0)	(0.5, 0.4)	(0.1, 0.0)
		NN	1.4527				0.9588	0.8485	1.0897	(0.6, 0.7)	(0.7, 0.6)	(0.0, 0.2)
	Organic - Density	LN	3.4577	0.1250	0.1379	**0.1125**	0.1250	0.1316	0.1179	(0.0, 0.0)	(0.2, 0.1)	(0.3, 0.2)
		RF	0.3574				0.1250	0.1394	0.1288	(0.0, 0.0)	(0.1, 0.0)	(0.6, 0.6)
		NN	3.4316				0.1250	0.1416	0.1508	(0.0, 0.0)	(0.7, 0.7)	(0.3, 0.3)
	Organic - Dielectric constant	LN	3.3257	2.9218	**2.7007**	2.7689	2.9222	2.8424	2.7925	(0.3, 0.6)	(0.7, 0.8)	(0.8, 0.8)
		RF	3.1719				2.7644	2.7255	2.8668	(0.5, 0.6)	(0.9, 0.9)	(0.8, 0.9)
		NN	3.2899				2.8067	2.8131	3.1630	(0.6, 0.7)	(0.3, 0.5)	(0.4, 0.6)
	Organic - Refractive index	LN	0.2068	0.1575	0.1284	0.1482	0.1465	**0.1255**	0.1457	(0.1, 0.1)	(0.7, 0.7)	(-0.1, 0.0)
		RF	0.1986				0.1575	0.1263	0.1458	(0.0, 0.0)	(0.7, 0.7)	(0.2, 0.3)
		NN	0.2164				0.1500	0.1270	0.1982	(-0.2, 0.0)	(0.6, 0.5)	(0.1, 0.2)
	Organic - Volume	LN	163.5935	46.8784	35.5770	**22.9643**	35.2985	37.7310	86.4402	(0.1, 0.1)	(0.4, 0.4)	(0.8, 0.8)
		RF	142.7930				34.0989	35.3355	46.9399	(0.6, 0.6)	(0.2, 0.2)	(-0.1, 0.1)
		NN	573.0292				46.8784	36.4149	55.3258	(0.0, 0.0)	(0.1, 0.1)	(0.1, 0.2)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Inorganic - Refractive index	Inorganic - Band gap	LN	3.6632	1.2790	1.2445	1.1716	1.2554	1.1748	1.2987	(0.1, 0.4)	(0.9, 0.9)	(0.9, 0.9)
		RF	4.6929				**1.1459**	1.1862	1.2803	(0.2, 0.4)	(0.2, 0.5)	(0.6, 0.4)
		NN	4.3252				1.2255	1.2344	1.4055	(-0.7, 0.4)	(0.3, 0.5)	(0.3, 0.5)
	Inorganic - Density	LN	1.6106	0.8502	1.0216	0.8716	0.8502	0.8713	0.9576	(0.0, 0.0)	(0.4, 0.3)	(0.4, 0.3)
		RF	1.6957				**0.7596**	1.0490	0.9408	(-0.1, 0.0)	(0.3, 0.0)	(0.2, 0.1)
		NN	1.9426				0.8502	1.0001	1.6358	(0.0, 0.0)	(0.1, 0.0)	(0.3, 0.3)
	Inorganic - Dielectric constant	LN	20.1115	19.9541	19.7055	22.5833	19.4964	19.0310	19.4271	(-0.5, 0.4)	(0.3, 0.5)	(0.6, 0.6)
		RF	19.4174				18.7022	18.1843	19.0652	(0.5, 0.6)	(0.9, 0.9)	(0.7, 0.7)
		NN	18.9044				17.9844	**17.8379**	18.1182	(0.6, 0.7)	(0.9, 0.9)	(-1.4, 0.6)
	Inorganic - Volume	LN	127.2113	42.1724	42.0894	53.2378	**36.7848**	46.0041	42.0380	(0.1, 0.0)	(0.4, 0.3)	(0.1, 0.0)
		RF	122.1695				42.1724	41.3024	40.5058	(0.0, 0.0)	(0.2, 0.0)	(0.1, 0.1)
		NN	136.8073				40.9691	41.6813	96.4219	(-0.1, 0.0)	(0.1, 0.0)	(0.1, 0.1)
	Organic - Atomization energy	LN	11.8494	**0.1224**	0.1653	0.1231	**0.1224**	0.1653	0.1516	(0.0, 0.0)	(0.0, 0.0)	(0.6, 0.6)
		RF	11.3736				0.1325	0.1708	0.1831	(0.2, 0.1)	(0.6, 0.5)	(0.7, 0.7)
		NN	11.2709				0.1319	0.1872	0.4072	(0.2, 0.2)	(0.3, 0.1)	(0.0, 0.1)
	Organic - Band gap	LN	5.4320	0.7621	0.7922	0.8777	0.7972	**0.7615**	0.8358	(0.2, 0.3)	(0.9, 0.9)	(0.1, 0.4)
		RF	6.6960				0.8060	0.7849	0.9059	(-0.1, 0.0)	(0.2, 0.2)	(0.1, 0.1)
		NN	7.0476				0.8810	0.7922	1.0677	(-0.3, 0.0)	(0.0, 0.0)	(0.5, 0.5)
	Organic - Density	LN	0.2462	0.0828	0.0897	0.0855	0.0778	0.0930	0.1253	(0.1, 0.1)	(0.2, 0.0)	(0.4, 0.4)
		RF	3.5491				0.0828	0.0926	0.1002	(0.0, 0.0)	(0.1, 0.1)	(0.7, 0.7)
		NN	3.4506				**0.0759**	0.0952	0.2158	(0.1, 0.1)	(0.5, 0.4)	(0.3, 0.2)
	Organic - Dielectric constant	LN	4.8174	3.7526	3.1639	3.9243	3.8983	3.3400	3.5859	(0.5, 0.6)	(0.9, 0.9)	(0.2, 0.3)
		RF	13.1831				3.7526	**2.9391**	3.9545	(0.0, 0.0)	(-0.3, 0.0)	(0.6, 0.4)
		NN	14.4241				3.9125	2.9799	3.2049	(-0.1, 0.0)	(-0.2, 0.0)	(-0.1, 0.0)
	Organic - Refractive index	LN	0.1721	0.1511	0.1487	0.1509	0.1469	**0.1376**	0.1612	(0.5, 0.6)	(0.8, 0.8)	(0.8, 0.8)
		RF	0.2163				0.1490	0.1420	0.1428	(0.1, 0.2)	(0.6, 0.6)	(0.7, 0.7)
		NN	0.2152				0.1500	0.1466	0.1832	(0.2, 0.2)	(0.1, 0.1)	(0.6, 0.5)
	Organic - Volume	LN	181.5128	76.2862	77.0331	65.4699	**43.9842**	71.0149	86.6729	(0.5, 0.5)	(0.3, 0.3)	(0.5, 0.6)
		RF	150.6477				76.2862	66.6849	71.3818	(0.0, 0.0)	(0.5, 0.5)	(0.3, 0.3)
		NN	152.4214				76.2862	66.3073	165.6133	(0.0, 0.0)	(0.2, 0.1)	(0.1, 0.0)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Inorganic - Volume	Inorganic - Band gap	LN	2.9424	1.1848	**1.1465**	1.2318	1.3327	**1.1465**	1.3580	(0.1, 0.2)	(0.0, 0.0)	(-0.2, 0.0)
		RF	4.1691				1.3116	1.2382	1.3712	(-0.1, 0.0)	(0.4, 0.3)	(0.0, 0.2)
		NN	2.4963				1.4462	1.3972	1.7492	(0.4, 0.5)	(0.0, 0.2)	(0.2, 0.4)
	Inorganic - Density	LN	9.1403	**0.7424**	0.9233	0.8950	0.8091	0.9130	1.0477	(0.3, 0.3)	(0.5, 0.4)	(0.4, 0.3)
		RF	8.5855				2.2622	0.9960	0.8983	(-2.0, 1.0)	(0.6, 0.5)	(0.4, 0.3)
		NN	8.8716				2.3083	0.8758	1.1343	(-2.0, 1.0)	(0.5, 0.4)	(0.4, 0.3)
	Inorganic - Dielectric constant	LN	29.3191	18.5998	18.7099	18.9905	18.5998	16.1081	18.5181	(0.0, 0.0)	(0.2, 0.2)	(0.6, 0.5)
		RF	20.0348				18.5998	16.5782	16.3408	(0.0, 0.0)	(0.9, 0.9)	(0.5, 0.6)
		NN	22.0991				17.7790	**15.5996**	15.7704	(-0.1, 0.0)	(-0.1, 0.1)	(0.4, 0.4)
	Inorganic - Refractive index	LN	1.9887	**0.8636**	0.9130	1.3798	**0.8636**	0.9130	1.0828	(0.0, 0.0)	(0.0, 0.0)	(0.1, 0.0)
		RF	1.6396				1.0460	1.0360	1.0880	(-0.1, 0.0)	(-0.1, 0.0)	(-0.3, 0.0)
		NN	1.7546				**0.8636**	1.0476	1.1245	(0.0, 0.0)	(-0.2, 0.0)	(-0.1, 0.0)
	Organic - Atomization energy	LN	11.8292	0.1288	0.2192	0.1412	0.1539	0.1903	0.1722	(0.1, 0.1)	(0.4, 0.4)	(0.7, 0.7)
		RF	11.7929				0.1469	0.2192	0.2325	(0.6, 0.6)	(0.0, 0.0)	(0.7, 0.7)
		NN	11.6036				**0.1269**	0.2192	0.2459	(0.5, 0.5)	(0.0, 0.0)	(0.5, 0.5)
	Organic - Band gap	LN	1.4364	0.9339	0.9268	0.8712	**0.8450**	0.8453	1.0204	(0.3, 0.3)	(0.4, 0.4)	(0.1, 0.1)
		RF	1.7842				0.9064	0.8897	0.8712	(0.6, 0.6)	(0.1, 0.1)	(0.0, 0.0)
		NN	1.7793				0.8950	0.9146	1.1818	(0.3, 0.3)	(0.4, 0.4)	(-0.3, 0.0)
	Organic - Density	LN	3.5156	0.0938	0.0957	0.0897	**0.0836**	0.0957	0.1478	(0.1, 0.1)	(0.0, 0.0)	(0.4, 0.2)
		RF	3.3951				0.0938	0.0957	0.1144	(0.0, 0.0)	(0.0, 0.0)	(0.4, 0.3)
		NN	3.4398				0.0938	0.1159	0.1945	(0.0, 0.0)	(0.2, 0.1)	(-0.1, 0.1)
	Organic - Dielectric constant	LN	11.4081	2.8321	2.9153	3.1623	2.9277	2.8334	2.8161	(-0.7, 0.3)	(-1.0, 0.1)	(-1.9, 0.1)
		RF	9.2448				2.9977	2.8176	2.8510	(-0.1, 0.0)	(0.5, 0.5)	(-0.3, 0.0)
		NN	9.6526				2.9591	**2.7632**	3.1282	(0.2, 0.0)	(0.2, 0.1)	(-1.0, 0.0)
	Organic - Refractive index	LN	3.6732	0.1507	**0.1407**	0.1433	0.1546	0.1505	0.1567	(-0.1, 0.0)	(0.3, 0.2)	(0.0, 0.1)
		RF	0.2757				0.1575	**0.1407**	0.1433	(0.4, 0.4)	(0.0, 0.0)	(0.0, 0.0)
		NN	0.2686				0.1580	0.1530	0.1716	(0.9, 0.9)	(0.1, 0.1)	(0.1, 0.1)
	Organic - Volume	LN	139.0114	54.9273	46.3835	46.8442	**29.1443**	49.6727	53.7153	(0.7, 0.7)	(0.1, 0.0)	(0.6, 0.6)
		RF	111.4204				32.8741	36.0432	32.3178	(0.7, 0.7)	(0.9, 0.9)	(0.8, 0.8)
		NN	125.2075				35.9744	47.0725	49.0530	(0.8, 0.8)	(0.9, 0.9)	(0.6, 0.6)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Organic - Atomization energy	Inorganic - Band gap	LN	6.7618				**1.1939**	1.2420	1.2875	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
		RF	2.0048	**1.1939**	1.2420	1.2875	8.6841	1.2415	1.2875	(-0.3, 0.0)	(-0.1, 0.0)	(0.0, 0.0)
		NN	2.4251				2.4251	1.2420	1.5938	(-2.0, 1.0)	(0.0, 0.0)	(0.1, 0.1)
	Inorganic - Density	LN	12.5215				0.8502	1.0194	1.1242	(0.0, 0.0)	(0.1, 0.1)	(0.2, 0.1)
		RF	7.9684	0.8502	1.0216	**0.7935**	1.1871	0.9960	1.2816	(-0.2, 0.0)	(0.4, 0.4)	(0.3, 0.1)
		NN	7.0652				1.1301	1.0323	1.6520	(-0.1, 0.0)	(0.4, 0.3)	(0.4, 0.4)
	Inorganic - Dielectric constant	LN	57.2453				18.7991	17.0726	18.3023	(-0.1, 0.0)	(-0.1, 0.0)	(0.0, 0.0)
		RF	31.0288	18.3758	16.8075	18.3023	18.1313	**16.7010**	17.7502	(-0.4, 0.4)	(0.6, 0.6)	(-0.4, 0.1)
		NN	28.6823				18.1296	16.7870	18.1288	(0.1, 0.2)	(0.2, 0.3)	(-0.3, 0.2)
	Inorganic - Refractive index	LN	6.4829				3.1219	0.8008	0.7889	(0.1, 0.1)	(0.1, 0.1)	(-0.1, 0.0)
		RF	4.4666	0.7157	0.7580	**0.7018**	4.9126	0.7449	0.7155	(-1.6, 0.0)	(0.3, 0.2)	(0.2, 0.3)
		NN	4.1711				5.3363	0.7559	0.8211	(-0.2, 0.0)	(0.1, 0.1)	(0.5, 0.5)
	Inorganic - Volume	LN	362.6684				55.3176	70.9819	58.8229	(0.1, 0.1)	(0.1, 0.1)	(0.0, 0.0)
		RF	111.0220	54.3771	58.3551	58.8229	63.6384	57.8758	**49.6017**	(0.1, 0.0)	(0.5, 0.5)	(0.7, 0.7)
		NN	130.6437				54.3771	55.2097	100.9147	(0.0, 0.0)	(0.3, 0.2)	(0.1, 0.2)
	Organic - Band gap	LN	1.9234				0.9024	0.8359	1.0104	(-0.4, 0.0)	(0.0, 0.0)	(0.3, 0.1)
		RF	1.3037	0.8835	0.8359	1.0136	0.9084	**0.8207**	0.8893	(-1.0, 0.0)	(-0.2, 0.0)	(-0.4, 0.0)
		NN	1.4896				0.8928	0.8359	1.2863	(-0.4, 0.0)	(0.0, 0.0)	(-0.2, 0.1)
	Organic - Density	LN	2.9398				**0.1331**	0.1420	0.1830	(0.0, 0.0)	(0.1, 0.1)	(0.7, 0.7)
		RF	3.3638	**0.1331**	0.1359	0.1356	0.1820	0.1545	0.1496	(0.9, 0.9)	(0.5, 0.5)	(0.5, 0.5)
		NN	3.3331				**0.1331**	0.1683	0.2405	(0.0, 0.0)	(0.5, 0.5)	(0.4, 0.5)
	Organic - Dielectric constant	LN	4.1201				2.1667	2.1462	2.2037	(0.0, 0.0)	(0.0, 0.1)	(0.2, 0.0)
		RF	3.3522	2.1667	**2.0903**	2.2804	2.2590	2.1377	2.2022	(0.5, 0.4)	(0.1, 0.2)	(0.0, 0.1)
		NN	3.6396				2.6567	2.1436	2.6324	(-0.7, 0.0)	(-0.2, 0.0)	(-0.1, 0.0)
	Organic - Refractive index	LN	3.6209				0.1358	**0.1291**	0.1324	(-0.1, 0.0)	(0.0, 0.0)	(0.0, 0.0)
		RF	3.7070	0.1411	**0.1291**	0.1324	0.1526	0.1308	0.1906	(0.3, 0.5)	(-0.1, 0.0)	(0.5, 0.5)
		NN	3.7288				0.1445	**0.1291**	0.1901	(0.4, 0.4)	(0.0, 0.0)	(0.3, 0.4)
	Organic - Volume	LN	568.1656				71.7697	52.8113	121.1668	(0.3, 0.3)	(0.2, 0.2)	(-0.2, 0.0)
		RF	615.4604	50.2788	42.8737	**34.0347**	62.1807	42.9842	73.9654	(0.0, 0.1)	(0.1, 0.1)	(0.1, 0.2)
		NN	633.4821				63.2865	42.8737	156.4083	(0.0, 0.1)	(0.0, 0.0)	(0.0, 0.1)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Organic - Band gap	Inorganic - Band gap	LN	3.2840	1.2759	1.2446	**1.1864**	1.2759	1.2365	1.4975	(0.0, 0.0)	(0.1, 0.0)	(-0.2, 0.1)
		RF	1.5222				1.4018	1.2530	1.3354	(0.1, 0.6)	(0.9, 0.9)	(-0.3, 0.0)
		NN	2.5763				1.4035	1.2247	1.4142	(0.5, 0.5)	(0.3, 0.4)	(0.5, 0.6)
	Inorganic - Density	LN	9.0583	0.8107	1.0313	**0.7387**	1.2600	1.0271	1.6038	(0.1, 0.2)	(-0.1, 0.0)	(0.3, 0.3)
		RF	7.7621				0.8107	0.9507	0.9055	(0.0, 0.0)	(0.5, 0.3)	(0.4, 0.4)
		NN	6.1267				1.0126	1.0133	1.5513	(-0.2, 0.0)	(0.2, 0.2)	(0.1, 0.3)
	Inorganic - Dielectric constant	LN	59.8500	12.5147	12.5708	14.5908	13.6250	12.6773	16.4276	(-0.6, 0.1)	(-1.0, 0.0)	(-1.4, 0.1)
		RF	26.0598				12.7879	**12.2533**	13.0230	(-0.2, 0.3)	(0.7, 0.8)	(0.8, 0.6)
		NN	20.8108				12.6698	12.5708	12.7627	(0.2, 0.2)	(0.0, 0.0)	(0.3, 0.3)
	Inorganic - Refractive index	LN	1.9629	0.9519	0.9521	**0.9149**	2.6326	0.9505	0.9413	(0.0, 0.1)	(0.1, 0.0)	(0.1, 0.1)
		RF	4.6086				1.2439	0.9543	0.9565	(0.6, 0.6)	(0.6, 0.5)	(0.3, 0.3)
		NN	3.9389				0.9720	0.9382	0.9745	(0.3, 0.3)	(0.4, 0.4)	(0.1, 0.2)
	Inorganic - Volume	LN	265.8471	45.7783	52.2646	**35.6658**	45.7783	49.9397	68.7418	(0.0, 0.0)	(0.1, 0.0)	(0.2, 0.2)
		RF	311.8425				57.9366	49.0805	58.2921	(0.0, 0.2)	(0.1, 0.0)	(0.1, 0.2)
		NN	234.1839				45.7783	52.2646	96.6478	(0.0, 0.0)	(0.0, 0.0)	(-0.1, 0.1)
	Organic - Atomization energy	LN	0.6294	**0.1224**	0.1652	0.1278	0.1398	0.1892	0.2064	(0.1, 0.1)	(0.3, 0.3)	(0.5, 0.5)
		RF	0.3871				0.1279	0.1774	0.1590	(0.1, 0.1)	(-0.1, 0.1)	(0.2, 0.1)
		NN	0.4638				0.1334	0.2521	0.2788	(-0.1, 0.0)	(-1.4, 0.0)	(-0.7, 0.0)
	Organic - Density	LN	0.5035	**0.0828**	0.0921	0.0910	0.0911	0.0905	0.0910	(0.5, 0.5)	(0.1, 0.1)	(0.0, 0.0)
		RF	3.3451				**0.0828**	0.0913	0.0935	(0.0, 0.0)	(0.2, 0.1)	(0.3, 0.2)
		NN	3.2977				**0.0828**	0.0929	0.0910	(0.0, 0.0)	(0.1, 0.0)	(0.0, 0.0)
	Organic - Dielectric constant	LN	5.0594	3.0568	**2.9362**	3.0647	3.0899	3.0213	3.0405	(0.1, 0.0)	(0.2, 0.2)	(-0.2, 0.0)
		RF	11.9898				3.1742	2.9820	3.1827	(-0.2, 0.1)	(0.0, 0.2)	(0.2, 0.4)
		NN	10.7901				3.0596	2.9949	3.1632	(0.2, 0.3)	(-0.3, 0.0)	(0.4, 0.4)
	Organic - Refractive index	LN	3.8879	0.1575	0.1302	0.1549	0.1486	0.1219	0.1544	(-0.1, 0.0)	(0.3, 0.4)	(0.0, 0.1)
		RF	3.7425				0.1477	0.1226	0.1419	(-0.3, 0.1)	(0.5, 0.5)	(-1.8, 0.5)
		NN	3.7237				0.1424	**0.1212**	0.1704	(-0.6, 0.2)	(0.2, 0.3)	(-1.8, 0.3)
	Organic - Volume	LN	765.8390	46.8784	36.6309	**23.3203**	46.8784	33.6603	57.9079	(0.0, 0.0)	(0.1, 0.1)	(-0.2, 0.0)
		RF	635.0630				60.8741	31.6817	59.5051	(-0.1, 0.0)	(0.2, 0.2)	(0.7, 0.7)
		NN	627.9848				46.8784	35.7645	69.5504	(0.0, 0.0)	(0.1, 0.1)	(0.1, 0.1)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Organic - Density	Inorganic - Band gap	LN	7.1286	1.2759	1.2530	1.2383	1.3926	1.2412	1.3545	(0.2, 0.2)	(-0.1, 0.0)	(0.1, 0.0)
		RF	4.2054				1.3812	**1.1932**	1.2383	(0.1, 0.1)	(0.4, 0.4)	(0.0, 0.0)
		NN	5.0290				1.3386	1.2481	1.5174	(0.2, 0.2)	(0.3, 0.3)	(0.2, 0.2)
	Inorganic - Density	LN	3.6720	**0.8119**	1.0462	0.8490	**0.8119**	1.0359	0.9342	(0.0, 0.0)	(0.3, 0.3)	(0.2, 0.2)
		RF	4.8314				0.8190	1.0810	0.9240	(0.1, 0.1)	(0.8, 0.8)	(0.7, 0.7)
		NN	5.5866				0.8269	1.0462	1.1842	(0.1, 0.1)	(0.0, 0.0)	(0.4, 0.4)
	Inorganic - Dielectric constant	LN	26.5206	**12.5147**	12.9761	14.6243	14.9834	14.8942	14.6026	(0.2, 0.1)	(0.1, 0.0)	(0.3, 0.4)
		RF	56.2063				**12.5147**	14.8685	15.3146	(0.0, 0.0)	(0.0, 0.1)	(0.8, 0.8)
		NN	38.4862				14.9711	14.9001	14.6243	(0.2, 0.2)	(0.3, 0.3)	(0.0, 0.0)
	Inorganic - Refractive index	LN	2.0198	1.0844	1.1136	1.0864	24.8515	1.1179	**1.0693**	(0.6, 0.6)	(0.2, 0.1)	(0.2, 0.1)
		RF	7.1881				1.0844	1.1136	1.0864	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
		NN	7.6493				1.0844	1.1051	1.2385	(0.0, 0.0)	(0.3, 0.3)	(0.3, 0.2)
	Inorganic - Volume	LN	183.3062	45.8936	46.3379	**42.9493**	45.8936	48.5924	**42.9493**	(0.0, 0.0)	(0.2, 0.2)	(0.0, 0.0)
		RF	559.2701				51.0095	52.3725	44.0897	(0.1, 0.1)	(0.1, 0.1)	(0.6, 0.6)
		NN	593.2866				50.2059	46.3379	76.7744	(0.1, 0.1)	(0.0, 0.0)	(0.4, 0.4)
	Organic - Atomization energy	LN	11.4251	0.1225	0.1950	**0.1184**	0.1225	0.1887	0.1667	(0.0, 0.0)	(0.6, 0.6)	(0.6, 0.6)
		RF	11.1939				0.2544	0.1950	0.1883	(-0.4, 0.0)	(0.0, 0.0)	(0.7, 0.7)
		NN	11.0382				0.3227	0.1853	**0.1184**	(-0.5, 0.0)	(0.3, 0.3)	(0.0, 0.0)
	Organic - Band gap	LN	5.6434	0.8120	0.7749	**0.7745**	1.1492	0.7749	0.8873	(-0.3, 0.0)	(0.0, 0.0)	(0.1, 0.1)
		RF	6.9589				0.8120	0.8431	0.9148	(0.0, 0.0)	(0.1, 0.0)	(0.1, 0.1)
		NN	7.3460				1.0153	0.8244	1.3501	(0.2, 0.2)	(0.1, 0.1)	(0.4, 0.4)
	Organic - Dielectric constant	LN	3.9072	3.0538	**2.8343**	3.0895	2.9771	2.8895	3.0364	(-0.1, 0.0)	(0.1, 0.0)	(0.4, 0.3)
		RF	3.8985				3.1880	2.8822	3.0437	(-0.1, 0.0)	(0.1, 0.0)	(0.2, 0.1)
		NN	4.5906				3.0538	2.9265	3.0895	(0.0, 0.0)	(0.1, 0.0)	(0.0, 0.0)
	Organic - Refractive index	LN	0.3063	0.1634	0.1592	**0.1463**	0.1803	0.1592	0.2235	(-0.1, 0.0)	(0.0, 0.0)	(0.1, 0.1)
		RF	0.2572				0.1997	0.1562	0.2335	(-0.2, 0.0)	(0.1, 0.0)	(0.1, 0.2)
		NN	0.2781				0.2018	0.1568	0.2223	(-0.9, 0.0)	(0.2, 0.1)	(0.1, 0.1)
	Organic - Volume	LN	614.9902	59.2883	52.755	44.4093	47.6037	40.0011	93.6506	(0.3, 0.3)	(0.2, 0.1)	(0.3, 0.2)
		RF	705.4746				59.2883	52.7550	62.8471	(0.0, 0.0)	(0.0, 0.0)	(0.3, 0.0)
		NN	761.2213				**38.208**	42.2101	128.9882	(0.1, 0.1)	(0.1, 0.1)	(0.2, 0.2)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Organic - Dielectric constant	Inorganic - Band gap	LN	5.1508	1.1483	1.1050	1.1131	1.2983	1.1747	1.1375	(-0.5, 0.2)	(0.4, 0.3)	(-0.1, 0.1)
		RF	6.6419				1.3186	**1.0989**	1.2309	(0.3, 0.3)	(0.9, 0.9)	(0.1, 0.0)
		NN	8.7843				1.3605	1.1404	1.5927	(-0.2, 0.0)	(0.5, 0.5)	(0.2, 0.2)
	Inorganic - Density	LN	1.8508	**0.7288**	0.9141	0.8300	**0.7288**	0.9350	1.0492	(0.0, 0.0)	(0.1, 0.0)	(-0.1, 0.0)
		RF	2.3837				2.3837	0.9460	1.0010	(-2.0, 1.0)	(0.9, 0.9)	(0.7, 0.7)
		NN	12.9579				0.9526	1.0487	0.8300	(-0.1, 0.0)	(0.9, 0.9)	(0.0, 0.0)
	Inorganic - Dielectric constant	LN	95.8310	**15.3917**	17.6348	24.9405	18.2690	16.5195	38.0583	(-2.0, 0.1)	(-1.7, 0.0)	(0.3, 0.4)
		RF	28.0255				**15.3917**	16.6785	15.6916	(0.0, 0.0)	(0.2, 0.2)	(0.2, 0.2)
		NN	53.0942				15.9667	17.2676	16.6247	(0.1, 0.0)	(0.2, 0.2)	(0.6, 0.6)
	Inorganic - Refractive index	LN	2.2892	0.8862	0.9110	**0.8301**	1.1526	1.0048	1.3551	(0.0, 0.3)	(0.1, 0.2)	(0.5, 0.3)
		RF	1.5945				1.0292	0.9202	0.9601	(0.7, 0.7)	(0.2, 0.1)	(0.3, 0.3)
		NN	2.8136				0.9963	0.9110	1.0973	(0.2, 0.3)	(0.0, 0.0)	(0.0, 0.1)
	Inorganic - Volume	LN	360.7518	**38.7508**	50.2108	47.451	51.6332	50.2108	47.4510	(-0.1, 0.0)	(0.0, 0.0)	(0.0, 0.0)
		RF	441.5927				144.0797	52.1309	53.3079	(-2.0, 1.0)	(0.2, 0.2)	(0.3, 0.2)
		NN	225.0226				**38.7508**	49.9905	93.4896	(0.0, 0.0)	(0.1, 0.1)	(0.2, 0.2)
	Organic - Atomization energy	LN	0.5885	0.1312	0.1821	0.1348	**0.1183**	0.1953	0.1779	(0.1, 0.1)	(0.2, 0.1)	(0.3, 0.3)
		RF	0.4501				0.1338	0.1819	0.1618	(0.1, 0.0)	(0.1, 0.0)	(0.7, 0.7)
		NN	0.5070				0.1329	0.1889	0.4287	(0.1, 0.0)	(0.1, 0.0)	(0.1, 0.0)
	Organic - Band gap	LN	1.8875	**0.7723**	0.8821	0.8162	0.7845	0.8821	0.9141	(0.2, 0.2)	(0.0, 0.0)	(-0.2, 0.0)
		RF	6.5413				0.7928	0.9062	0.9051	(0.0, 0.2)	(0.0, 0.2)	(0.8, 0.8)
		NN	6.4852				0.7734	0.9110	1.2327	(0.3, 0.4)	(0.0, 0.1)	(0.4, 0.4)
	Organic - Density	LN	0.2138	0.0797	0.0993	0.0758	0.1085	0.0984	0.0858	(0.2, 0.2)	(0.4, 0.3)	(0.5, 0.5)
		RF	3.6059				**0.0703**	0.0996	0.0815	(0.1, 0.0)	(0.1, 0.0)	(0.2, 0.2)
		NN	0.2284				0.0797	0.1246	0.1449	(0.0, 0.0)	(0.3, 0.0)	(0.4, 0.2)
	Organic - Refractive index	LN	3.5631	0.1418	0.1495	0.1442	0.1451	0.1495	0.1442	(-0.1, 0.0)	(0.0, 0.0)	(0.0, 0.0)
		RF	0.2072				0.1382	0.1458	0.1426	(0.7, 0.7)	(0.3, 0.3)	(0.7, 0.6)
		NN	0.2100				0.1322	**0.1307**	0.1524	(0.3, 0.4)	(0.8, 0.8)	(0.5, 0.5)
	Organic - Volume	LN	509.9534	68.7765	56.094	**45.3029**	63.4836	54.8127	74.9539	(0.1, 0.0)	(0.1, 0.0)	(-0.1, 0.0)
		RF	629.4022				55.3764	52.5968	65.1766	(0.1, 0.1)	(0.1, 0.0)	(0.3, 0.2)
		NN	620.5116				68.7765	49.8665	156.4831	(0.0, 0.0)	(0.2, 0.0)	(0.2, 0.0)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Organic - Refractive index	Inorganic - Band gap	LN	5.7446	1.2495	1.2779	1.4768	1.6684	1.3252	**1.2241**	(0.3, 0.3)	(0.1, 0.1)	(0.0, 0.1)
		RF	7.4585				1.3032	1.3005	54.9561	(0.1, 0.2)	(0.2, 0.2)	(0.3, 0.4)
		NN	135.3329				1.2495	1.2779	13.5054	(0.0, 0.0)	(0.0, 0.0)	(-0.1, 0.1)
	Inorganic - Density	LN	4.1329	**0.8784**	1.2265	1.1409	**0.8784**	1.2517	1.3034	(0.0, 0.0)	(-0.1, 0.0)	(-0.1, 0.0)
		RF	3.9346				**0.8784**	1.2265	1.2652	(0.0, 0.0)	(0.0, 0.0)	(0.2, 0.1)
		NN	4.4810				**0.8784**	1.2265	1.7821	(0.0, 0.0)	(0.0, 0.0)	(-0.1, 0.1)
	Inorganic - Dielectric constant	LN	68.8329	22.3194	21.9908	**21.706**	22.8981	25.5512	24.2963	(0.0, 0.1)	(0.2, 0.2)	(0.2, 0.1)
		RF	29.7257				22.3194	21.7199	22.3623	(0.0, 0.0)	(0.6, 0.6)	(-0.1, 0.0)
		NN	35.5461				21.9499	21.7897	22.5455	(0.1, 0.1)	(0.3, 0.2)	(0.4, 0.3)
	Inorganic - Refractive index	LN	2.7019	0.9519	0.9546	1.0119	1.7636	1.0674	0.9632	(0.0, 0.1)	(0.2, 0.2)	(0.0, 0.1)
		RF	1.8395				1.5027	0.9823	0.9651	(0.6, 0.6)	(0.6, 0.6)	(0.3, 0.3)
		NN	1.9231				1.0733	**0.9487**	1.0793	(0.1, 0.1)	(0.1, 0.1)	(0.1, 0.2)
	Inorganic - Volume	LN	215.8081	**45.7783**	52.3133	51.5236	**45.7783**	52.3133	48.5966	(0.0, 0.0)	(0.0, 0.0)	(0.2, 0.1)
		RF	200.0686				**45.7783**	51.6726	58.7672	(0.0, 0.0)	(0.1, 0.1)	(0.3, 0.4)
		NN	250.3513				84.2957	55.3328	51.5236	(0.4, 0.4)	(0.1, 0.0)	(0.0, 0.0)
	Organic - Atomization energy	LN	12.6832	**0.1284**	0.1610	0.1699	**0.1284**	0.1840	0.1994	(0.0, 0.0)	(0.2, 0.2)	(0.1, 0.1)
		RF	11.3184				**0.1284**	0.1637	0.1710	(0.0, 0.0)	(0.2, 0.1)	(0.3, 0.3)
		NN	11.5324				**0.1284**	0.1863	0.4062	(0.0, 0.0)	(0.2, 0.1)	(-0.2, 0.0)
	Organic - Band gap	LN	4.9112	0.7723	0.8889	0.7957	0.7513	0.7974	0.8941	(0.2, 0.2)	(0.4, 0.4)	(0.7, 0.7)
		RF	6.6374				0.7413	0.8632	0.8244	(-0.4, 0.0)	(-0.2, 0.1)	(-0.1, 0.2)
		NN	6.5410				**0.7353**	0.8721	1.2487	(-0.1, 0.0)	(-0.1, 0.1)	(0.2, 0.1)
	Organic - Density	LN	0.5035	0.0840	0.1248	0.0816	0.1054	0.1278	0.1205	(0.1, 0.0)	(0.2, 0.2)	(0.2, 0.1)
		RF	3.7041				0.0840	0.1165	0.0946	(0.0, 0.0)	(0.1, 0.1)	(0.4, 0.3)
		NN	0.4102				**0.0784**	0.1205	0.2202	(-0.1, 0.0)	(0.1, 0.0)	(0.1, 0.0)
	Organic - Dielectric constant	LN	12.5665	2.7939	2.9305	**2.6879**	2.7939	2.9176	2.8350	(0.0, 0.0)	(0.0, 0.1)	(0.2, 0.2)
		RF	6.9307				2.9687	3.3352	2.8176	(-0.1, 0.0)	(-0.3, 0.0)	(-0.1, 0.0)
		NN	6.2910				3.4599	3.2881	3.2870	(-0.6, 0.0)	(-0.3, 0.0)	(-0.2, 0.0)
	Organic - Volume	LN	225.1316	61.1672	67.4030	**47.6749**	61.1672	67.4030	60.8003	(0.0, 0.0)	(0.0, 0.0)	(0.1, 0.1)
		RF	194.1770				61.1672	57.1853	**47.6749**	(0.0, 0.0)	(0.1, 0.1)	(0.0, 0.0)
		NN	228.7282				60.4089	67.4030	143.9603	(-0.1, 0.0)	(0.0, 0.0)	(0.2, 0.1)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Organic - Volume	Inorganic - Band gap	LN	9.5499				1.5874	1.4348	1.2324	(0.2, 0.1)	(0.1, 0.1)	(0.0, 0.0)
		RF	4.2297	**1.1649**	1.1857	1.2324	1.7560	1.2277	2.3249	(0.4, 0.4)	(0.6, 0.5)	(-0.1, 0.2)
		NN	2.4348				1.3455	1.2160	1.5677	(-0.1, 0.0)	(0.0, 0.1)	(-0.1, 0.1)
	Inorganic - Density	LN	7.4853				1.0201	1.3843	1.1315	(-0.1, 0.0)	(0.2, 0.2)	(0.0, 0.0)
		RF	2.0599	**0.6683**	1.0300	1.1315	0.9371	1.0300	1.1362	(0.1, 0.0)	(0.5, 0.3)	(0.1, 0.2)
		NN	2.4071				**0.6683**	1.0084	1.5625	(0.0, 0.0)	(0.3, 0.0)	(0.1, 0.0)
	Inorganic - Dielectric constant	LN	44.5907				**20.8123**	20.9754	21.1843	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
		RF	33.9121	**20.8123**	20.9754	21.1843	53.5124	21.3943	22.0687	(0.1, 0.1)	(0.0, 0.1)	(0.7, 0.7)
		NN	36.9467				59.6354	21.4615	22.9216	(0.1, 0.2)	(0.5, 0.3)	(0.4, 0.2)
	Inorganic - Refractive index	LN	5.4910				0.9169	1.0842	0.9290	(0.1, 0.0)	(0.0, 0.0)	(-0.1, 0.0)
		RF	4.8106	1.0460	1.0842	1.0509	1.3083	1.0842	**0.8951**	(0.0, 0.1)	(0.0, 0.0)	(0.2, 0.0)
		NN	5.5073				1.0460	1.0842	0.9586	(0.0, 0.0)	(0.0, 0.0)	(0.1, 0.1)
	Inorganic - Volume	LN	419.0662				50.5716	45.5493	51.6926	(-0.1, 0.0)	(0.0, 0.0)	(0.1, 0.1)
		RF	58.5194	43.6981	45.5493	52.6707	40.2486	41.4555	91.1113	(0.5, 0.6)	(0.7, 0.6)	(0.9, 0.9)
		NN	106.2649				46.9360	**34.1469**	69.0090	(0.1, 0.2)	(0.3, 0.2)	(-0.1, 0.0)
	Organic - Atomization energy	LN	11.3895				**0.1444**	0.2151	0.1800	(0.0, 0.0)	(0.1, 0.0)	(0.0, 0.0)
		RF	11.7049	**0.1444**	0.2235	0.1800	0.3113	0.2235	0.1777	(-0.2, 0.0)	(0.0, 0.0)	(-0.1, 0.0)
		NN	11.2121				**0.1444**	0.2037	0.2217	(0.0, 0.0)	(0.1, 0.0)	(0.1, 0.0)
	Organic - Band gap	LN	6.9645				0.9716	**0.8740**	1.0811	(0.3, 0.2)	(0.2, 0.1)	(0.2, 0.4)
		RF	5.6516	0.9547	0.9685	0.9625	0.9547	0.8997	1.3958	(0.0, 0.0)	(0.2, 0.2)	(-0.1, 0.0)
		NN	6.1992				0.9221	0.8921	1.3816	(-0.1, 0.0)	(0.2, 0.2)	(0.6, 0.4)
	Organic - Density	LN	3.1746				0.0923	0.1436	0.0896	(0.9, 0.9)	(0.0, 0.0)	(0.3, 0.2)
		RF	3.2909	0.1442	0.1436	0.1397	0.1442	0.0917	0.0978	(0.0, 0.0)	(0.2, 0.0)	(0.2, 0.2)
		NN	3.3894				**0.0733**	0.0863	0.1323	(-0.1, 0.0)	(0.2, 0.1)	(0.4, 0.3)
	Organic - Dielectric constant	LN	12.5501				**1.9382**	2.3842	2.4527	(0.0, 0.0)	(-0.1, 0.1)	(0.3, 0.3)
		RF	10.4751	**1.9382**	1.9473	2.2704	2.3103	1.9473	2.3716	(0.1, 0.0)	(0.0, 0.0)	(-0.1, 0.0)
		NN	11.2851				**1.9382**	2.3736	2.5188	(0.0, 0.0)	(-0.1, 0.0)	(-0.1, 0.1)
	Organic - Refractive index	LN	0.3394				0.1459	0.1415	0.1608	(0.0, 0.0)	(0.0, 0.0)	(0.2, 0.2)
		RF	0.2843	0.1459	0.1415	**0.1412**	0.2547	0.1589	0.1581	(0.3, 0.3)	(0.1, 0.0)	(0.1, 0.1)
		NN	0.3586				0.2351	0.1579	0.2070	(0.0, 0.1)	(0.1, 0.0)	(-0.2, 0.1)

Table 3.3: Transfer between monomeric and polymeric properties

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Monomer - Dielectric constant	Monomer - HOMO-LUMO gap	LN	11.1837				0.8292	0.7435	0.8823	(-0.1, 0.1)	(0.6, 0.4)	(0.1, 0.3)
		RF	11.1503	0.8724	0.7956	0.9173	0.8302	**0.7139**	0.7421	(-0.1, 0.2)	(0.5, 0.3)	(0.8, 0.8)
		NN	11.1456				0.8250	0.7372	0.7644	(-0.2, 0.2)	(0.2, 0.3)	(0.4, 0.4)
	Monomer - Refractive index	LN	0.0429				0.0436	0.0424	0.0439	(0.8, 0.9)	(0.8, 0.9)	(0.8, 0.9)
		RF	0.0415	0.0912	0.0894	0.0947	0.0463	0.0415	0.0415	(0.9, 0.9)	(-2.0, 1.0)	(-2.0, 1.0)
		NN	0.0373				0.0365	**0.0355**	0.0505	(0.8, 0.9)	(0.8, 0.9)	(0.4, 0.7)
	Polymer - Band gap	LN	8.5656				1.0881	0.7862	0.8936	(0.3, 0.1)	(0.0, 0.1)	(0.6, 0.6)
		RF	8.5395	0.8190	0.8131	0.9768	0.8594	0.7477	**0.7130**	(-0.2, 0.4)	(0.4, 0.3)	(0.8, 0.8)
		NN	8.6998				0.8654	0.8598	0.8908	(-0.5, 0.1)	(0.3, 0.5)	(0.6, 0.5)
	Polymer - Dielectric constant	LN	0.7157				0.6031	**0.5358**	0.6376	(-0.4, 0.2)	(0.3, 0.2)	(-0.5, 0.0)
		RF	0.7018	0.5421	0.5884	0.6059	0.5988	0.5786	0.6678	(-0.2, 0.2)	(0.3, 0.2)	(0.0, 0.4)
		NN	0.7418				0.6143	0.5478	0.7563	(-0.1, 0.2)	(0.2, 0.3)	(-0.2, 0.1)
	Polymer - Refractive index	LN	1.5434				0.3269	0.3906	0.3442	(0.0, 0.0)	(-0.4, 0.0)	(0.2, 0.4)
		RF	1.5523	0.3269	0.3609	**0.3199**	0.3269	0.3574	0.3312	(0.0, 0.0)	(0.1, 0.1)	(0.1, 0.2)
		NN	1.5956				0.3269	0.3845	0.4254	(0.0, 0.0)	(-0.1, 0.1)	(-1.7, 0.0)
Monomer - HOMO-LUMO gap	Monomer - Dielectric constant	LN	5.3697				0.2978	0.2906	0.3225	(0.9, 0.9)	(0.6, 0.5)	(0.0, 0.6)
		RF	5.3768	0.2688	0.3198	0.3197	**0.2672**	0.2889	0.3189	(-0.3, 0.0)	(0.4, 0.4)	(-0.3, 0.1)
		NN	5.3148				0.2713	0.2936	0.3518	(-0.2, 0.0)	(0.4, 0.3)	(0.6, 0.6)
	Monomer - Refractive index	LN	3.2709				0.0726	**0.0724**	0.0774	(0.6, 0.5)	(0.4, 0.4)	(0.5, 0.4)
		RF	3.2716	0.0760	0.0873	0.0999	0.0822	0.0794	0.0912	(-0.1, 0.1)	(0.2, 0.3)	(0.4, 0.2)
		NN	3.2502				0.0825	0.0820	0.0858	(0.0, 0.1)	(0.1, 0.3)	(0.6, 0.6)
	Polymer - Band gap	LN	0.8929				0.7679	0.7431	0.7771	(0.8, 0.9)	(0.9, 0.9)	(0.7, 0.6)
		RF	0.7728	0.9659	0.8653	1.0593	**0.6330**	0.6593	0.6366	(0.9, 0.9)	(0.8, 0.8)	(0.9, 0.9)
		NN	0.7933				0.7416	0.8077	0.7816	(0.3, 0.5)	(0.8, 0.7)	(0.8, 0.7)
	Polymer - Dielectric constant	LN	7.8515				0.5053	0.4960	0.4746	(0.1, 0.0)	(0.6, 0.5)	(0.5, 0.6)
		RF	7.8494	**0.4411**	0.5601	0.4746	0.4981	0.4972	0.4982	(0.1, 0.1)	(0.8, 0.7)	(0.6, 0.5)
		NN	7.6928				0.4900	0.5233	0.5251	(0.2, 0.3)	(0.9, 0.9)	(0.7, 0.8)
	Polymer - Refractive index	LN	0.5480				0.3093	0.3320	0.3943	(0.0, 0.0)	(0.2, 0.0)	(-1.0, 0.0)
		RF	0.5736	0.3093	0.3425	**0.2990**	0.3212	0.3305	0.4305	(0.3, 0.3)	(0.1, 0.0)	(0.5, 0.4)
		NN	0.6225				0.3200	0.3308	0.3393	(-0.1, 0.0)	(0.1, 0.0)	(0.6, 0.6)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Monomer - Refractive index	Monomer - Dielectric constant	LN	0.2477	0.2636	0.3548	0.3787	0.1986	0.1793	0.1951	(0.7, 0.8)	(0.9, 0.9)	(0.9, 0.9)
		RF	0.2475				0.1970	0.1739	0.1449	(0.3, 0.6)	(0.7, 0.8)	(0.9, 0.9)
		NN	0.3363				0.1482	**0.1208**	0.1588	(0.6, 0.7)	(0.7, 0.7)	(0.5, 0.6)
	Monomer - HOMO-LUMO gap	LN	11.2381	0.7675	0.8207	0.8733	0.7613	0.6881	0.7807	(0.1, 0.3)	(0.7, 0.6)	(0.1, 0.4)
		RF	11.2907				0.7645	**0.6574**	0.8061	(0.0, 0.3)	(0.6, 0.4)	(0.3, 0.4)
		NN	11.4734				0.7470	0.6939	0.9362	(0.1, 0.2)	(0.5, 0.4)	(0.8, 0.8)
	Polymer - Band gap	LN	8.9156	0.8212	0.9035	0.9738	1.0144	0.8676	0.8260	(0.3, 0.1)	(0.4, 0.3)	(0.8, 0.8)
		RF	8.9567				0.9307	**0.7581**	1.3794	(-0.2, 0.1)	(0.7, 0.5)	(0.9, 0.8)
		NN	9.3052				0.8978	0.8347	0.9124	(-0.1, 0.2)	(0.6, 0.5)	(0.8, 0.8)
	Polymer - Dielectric constant	LN	0.7611	0.5588	0.5989	0.6408	0.5930	**0.5395**	0.6011	(0.8, 0.8)	(0.4, 0.4)	(0.5, 0.7)
		RF	0.7697				0.5814	0.5816	0.5457	(0.2, 0.2)	(0.7, 0.6)	(0.8, 0.8)
		NN	0.8533				0.5767	0.5405	0.5841	(0.3, 0.3)	(0.5, 0.4)	(0.2, 0.2)
	Polymer - Refractive index	LN	1.6566	**0.3363**	0.3407	0.3452	0.3552	0.3368	0.5089	(0.8, 0.8)	(0.5, 0.3)	(0.5, 0.3)
		RF	1.6948				**0.3363**	0.3475	0.3542	(0.0, 0.0)	(0.5, 0.2)	(0.6, 0.5)
		NN	1.7986				0.3452	**0.3363**	0.3745	(0.4, 0.4)	(0.4, 0.2)	(0.4, 0.2)
Polymer - BandGap	Monomer - Dielectric constant	LN	5.3747	0.3469	0.3797	0.3572	0.3016	0.3284	0.3677	(0.3, 0.4)	(0.6, 0.6)	(0.8, 0.7)
		RF	5.3067				**0.2971**	0.3578	0.3016	(0.1, 0.2)	(0.3, 0.3)	(0.1, 0.3)
		NN	5.3711				0.3059	0.3282	0.3276	(0.1, 0.2)	(0.6, 0.6)	(0.6, 0.5)
	Monomer - HOMO-LUMO gap	LN	0.9625	0.7985	0.7576	0.8616	0.7779	0.7101	0.7579	(0.6, 0.7)	(0.8, 0.8)	(0.9, 0.9)
		RF	0.7377				0.6711	**0.6118**	0.6255	(0.8, 0.9)	(0.9, 0.9)	(0.7, 0.8)
		NN	0.9139				0.7652	0.6664	0.7692	(0.9, 0.9)	(0.8, 0.8)	(0.4, 0.4)
	Monomer - Refractive index	LN	3.2637	0.0766	0.0931	0.0879	0.0768	0.0833	0.0871	(0.0, 0.2)	(0.6, 0.6)	(0.6, 0.7)
		RF	3.2461				0.0784	0.0815	0.0817	(0.0, 0.1)	(0.5, 0.3)	(-0.2, 0.2)
		NN	3.2656				0.0768	**0.0761**	0.1053	(0.0, 0.3)	(0.6, 0.4)	(0.6, 0.7)
	Polymer - Dielectric constant	LN	7.9660	0.5855	0.5891	0.6108	**0.4789**	0.4898	0.5220	(0.6, 0.6)	(0.6, 0.5)	(0.6, 0.6)
		RF	7.9748				0.5349	0.5018	0.7531	(0.2, 0.4)	(0.6, 0.6)	(0.6, 0.5)
		NN	7.9956				0.4988	0.5053	0.5082	(0.2, 0.4)	(0.6, 0.4)	(0.8, 0.8)
	Polymer - Refractive index	LN	0.6564	0.3497	0.3598	**0.3423**	0.4311	0.4263	0.4248	(-0.9, 0.0)	(-0.8, 0.0)	(-0.5, 0.0)
		RF	0.6502				0.3735	0.3935	0.3720	(-0.1, 0.1)	(-0.3, 0.0)	(0.1, 0.3)
		NN	0.6748				0.3789	0.4322	0.3916	(0.6, 0.6)	(-0.9, 0.0)	(0.1, 0.0)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
Polymer - Dielectric constant	Monomer - Dielectric constant	LN	0.4840				0.2924	0.3133	0.3245	(0.5, 0.5)	(0.9, 0.9)	(0.7, 0.7)
		RF	0.4474	0.3099	0.3591	0.3292	0.3005	0.3179	**0.2706**	(0.6, 0.6)	(0.9, 0.9)	(0.7, 0.6)
		NN	0.5387				0.2853	0.3113	0.3136	(0.2, 0.1)	(0.5, 0.4)	(0.3, 0.2)
	Monomer - HOMO-LUMO gap	LN	10.8008				0.9456	0.8603	0.9252	(0.2, 0.2)	(0.8, 0.8)	(0.5, 0.5)
		RF	11.0308	0.9321	1.0368	1.0293	**0.8252**	0.8266	1.0141	(0.9, 0.9)	(0.9, 0.9)	(0.7, 0.7)
		NN	10.6479				0.8960	0.9528	0.9730	(0.1, 0.1)	(0.5, 0.5)	(0.4, 0.3)
	Monomer - Refractive index	LN	0.1199				0.0912	0.0859	**0.0842**	(-0.3, 0.0)	(0.5, 0.4)	(0.2, 0.2)
		RF	0.1090	0.0878	0.0980	0.0923	0.0879	0.0869	0.0846	(0.0, 0.1)	(0.4, 0.4)	(0.1, 0.4)
		NN	0.1379				0.0843	0.0874	0.0890	(0.5, 0.5)	(0.6, 0.6)	(0.4, 0.4)
	Polymer - BandGap	LN	9.1635				0.7835	0.7310	0.7243	(0.3, 0.5)	(0.8, 0.7)	(0.5, 0.5)
		RF	9.1746	0.8382	0.8423	0.8265	0.8140	0.7811	**0.7150**	(0.6, 0.7)	(0.8, 0.7)	(0.8, 0.7)
		NN	9.1379				0.8146	0.7304	1.4287	(0.7, 0.7)	(0.7, 0.6)	(0.8, 0.6)
	Polymer - Refractive index	LN	0.6268				0.3037	0.3035	0.3295	(0.3, 0.3)	(0.5, 0.5)	(0.2, 0.2)
		RF	0.6242	0.3080	0.3213	0.3094	0.3051	0.3091	0.3160	(0.8, 0.8)	(0.2, 0.3)	(0.4, 0.5)
		NN	0.6232				0.3289	**0.3021**	0.3094	(0.0, 0.2)	(0.2, 0.2)	(0.0, 0.0)
Polymer - Refractive index	Monomer - Dielectric constant	LN	5.1312				0.2649	0.2996	0.2831	(0.4, 0.3)	(0.1, 0.0)	(0.1, 0.2)
		RF	5.1764	**0.2561**	0.3263	0.3140	0.2745	0.3235	0.2972	(0.2, 0.2)	(0.1, 0.0)	(0.1, 0.0)
		NN	5.0972				0.2621	0.3071	0.4090	(0.1, 0.0)	(0.1, 0.1)	(0.1, 0.0)
	Monomer - HOMO-LUMO gap	LN	10.9818				0.8649	0.8294	0.9158	(0.3, 0.3)	(0.5, 0.4)	(0.7, 0.7)
		RF	11.1228	0.8501	**0.7822**	0.9278	1.0292	0.8434	0.9853	(0.5, 0.4)	(0.6, 0.5)	(0.7, 0.7)
		NN	10.9098				0.9574	0.8242	0.9232	(0.2, 0.1)	(0.4, 0.4)	(0.5, 0.5)
	Monomer - Refractive index	LN	3.1793				0.0745	0.0836	0.0992	(0.3, 0.3)	(0.1, 0.1)	(0.1, 0.2)
		RF	3.1934	**0.0722**	0.0829	0.0862	0.0795	0.0810	0.1097	(0.0, 0.1)	(0.2, 0.1)	(0.8, 0.7)
		NN	3.1711				0.0758	0.0822	0.1628	(0.3, 0.3)	(0.2, 0.1)	(0.6, 0.5)
	Polymer - BandGap	LN	8.9800				0.9409	1.0573	0.9767	(0.1, 0.2)	(0.2, 0.2)	(0.1, 0.1)
		RF	9.0337	**0.8795**	1.0560	0.9561	0.9511	1.0695	2.7043	(0.3, 0.3)	(0.4, 0.4)	(0.8, 0.7)
		NN	8.9310				1.0031	1.0513	1.3597	(-0.1, 0.1)	(0.2, 0.2)	(0.5, 0.5)
	Polymer - Dielectric constant	LN	0.9368				0.5058	0.4930	0.7578	(0.8, 0.8)	(0.7, 0.7)	(0.4, 0.2)
		RF	0.9468	0.4857	0.5632	0.5538	**0.4338**	0.4669	0.5758	(0.3, 0.3)	(0.6, 0.5)	(0.6, 0.4)
		NN	1.0179				0.4782	0.5008	0.5538	(0.3, 0.2)	(0.5, 0.4)	(0.0, 0.0)

Table 3.4: Transfer between theoretical and experimental energy levels of HOMO for the OPV molecules (CEP and HOPV15)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
CEP	HOPV	LN	10.4675	0.2186	0.2232	0.2196	**0.2158**	0.2236	0.2362	(0.1, 0.1)	(0.2, 0.0)	(-0.7, 0.0)
		RF	10.5259				0.2220	0.2249	0.2548	(-0.8, 0.0)	(0.1, 0.0)	(0.0, 0.4)
		NN	0.2936				0.2255	0.2262	0.2220	(-0.4, 0.1)	(0.0, 0.1)	(0.1, 0.3)

Table 3.5: Formation energy of SiO_2/CdI2 and all other inorganic compounds in the Materials Project database

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
All (without SiO2)	SiO2	LN	0.4892	0.2086	0.1928	0.2120	0.2086	**0.1825**	0.1938	(0.0, 0.0)	(0.3, 0.3)	(0.3, 0.3)
		RF	0.4764				0.2055	0.1831	0.2181	(0.1, 0.0)	(0.7, 0.7)	(0.3, 0.3)
		NN	0.3438				0.2006	0.1951	0.2986	(0.8, 0.8)	(0.4, 0.4)	(0.2, 0.1)
All (without CdI2)	CdI2	LN	0.0198	0.0013	0.0020	0.0024	0.0183	0.0026	0.0020	(-1.7, 0.8)	(0.3, 0.3)	(0.6, 0.5)
		RF	0.0214				**0.0012**	0.0016	0.0023	(0.2, 0.2)	(0.6, 0.6)	(0.6, 0.5)
		NN	0.0213				0.0013	0.0021	0.0013	(0.5, 0.5)	(0.4, 0.4)	(0.4, 0.3)

Table 3.6: Transfer on the predictoin of torques at the seven joints in a SARCOS robot arm

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
1st - torque	2nd - torque	LN	71.5928	5.8283	9.3569	11.489	5.8283	9.3593	13.2922	(0.0, 0.0)	(0.1, 0.1)	(0.1, 0.0)
		RF	76.8984				5.8283	10.5782	10.5534	(0.0, 0.0)	(-0.1, 0.0)	(0.9, 0.9)
		NN	75.2280				**5.7442**	9.3569	16.4064	(0.2, 0.2)	(0.0, 0.0)	(0.4, 0.4)
	3rd - torque	LN	13.5088	4.7085	6.1707	7.5097	3.9237	6.3168	8.4576	(-0.1, 0.0)	(-0.2, 0.0)	(0.3, 0.2)
		RF	15.5849				**3.8288**	6.3713	8.5799	(0.2, 0.2)	(0.1, 0.1)	(0.2, 0.2)
		NN	14.1086				3.8624	6.4329	7.5097	(0.4, 0.4)	(0.1, 0.1)	(0.0, 0.0)
	4th - torque	LN	35.8539	5.7022	6.9906	8.0802	4.4868	5.3282	10.4277	(0.2, 0.2)	(0.2, 0.2)	(0.5, 0.5)
		RF	30.2013				5.7353	6.1729	6.3311	(0.1, 0.1)	(0.1, 0.1)	(0.8, 0.8)
		NN	36.2782				**3.9939**	4.9606	8.0802	(0.3, 0.3)	(0.2, 0.2)	(0.0, 0.0)
	5th - torque	LN	4.7382	0.4914	0.6161	0.7309	**0.4675**	0.5851	0.6641	(0.4, 0.4)	(0.2, 0.2)	(0.6, 0.6)
		RF	2.5995				0.5052	0.6554	0.8211	(0.1, 0.1)	(-0.1, 0.0)	(-0.1, 0.0)
		NN	4.5034				0.5200	0.6164	1.0487	(0.7, 0.7)	(0.4, 0.4)	(0.1, 0.2)
	6th - torque	LN	4.0872	1.0296	**1.0269**	1.4113	1.0296	1.1813	1.6859	(0.0, 0.0)	(0.2, 0.2)	(0.2, 0.2)
		RF	3.5763				1.0296	1.0476	1.5204	(0.0, 0.0)	(0.1, 0.1)	(0.7, 0.7)
		NN	3.9801				1.0296	1.0672	1.7652	(0.0, 0.0)	(0.1, 0.1)	(0.2, 0.2)
	7th - torque	LN	7.3774	0.9841	1.3202	1.8180	0.8669	0.9591	1.9543	(0.7, 0.7)	(0.2, 0.2)	(0.9, 0.9)
		RF	6.2008				1.1220	1.1436	1.8786	(0.6, 0.6)	(0.3, 0.3)	(0.9, 0.9)
		NN	7.4243				**0.7752**	1.1278	1.8180	(0.2, 0.2)	(0.1, 0.1)	(0.0, 0.0)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
2nd - torque	1st - torque	LN	41.5468	9.0900	14.9963	18.247	9.3064	16.2701	16.8397	(-0.1, 0.0)	(0.2, 0.3)	(0.6, 0.5)
		RF	49.6228				8.3061	15.3679	20.9052	(0.2, 0.2)	(0.0, 0.1)	(-0.1, 0.1)
		NN	46.7523				**7.8290**	16.2357	24.5267	(0.1, 0.1)	(0.1, 0.1)	(0.4, 0.4)
	3rd - torque	LN	18.9478	5.0582	5.6266	7.4251	4.1746	3.9728	4.2545	(0.3, 0.3)	(0.5, 0.5)	(0.7, 0.7)
		RF	25.9562				3.8081	4.7614	4.6055	(-0.2, 0.0)	(0.6, 0.7)	(0.7, 0.6)
		NN	23.3294				**3.3664**	4.0497	5.4699	(0.4, 0.4)	(0.6, 0.6)	(0.7, 0.7)
	4th - torque	LN	33.9689	5.0732	7.2780	6.8826	4.9299	6.8884	10.7021	(0.1, 0.1)	(0.1, 0.1)	(0.5, 0.5)
		RF	50.8812				6.2204	8.1020	9.0565	(0.2, 0.2)	(-0.2, 0.2)	(0.6, 0.6)
		NN	44.0185				**4.7459**	7.1065	12.8886	(0.1, 0.1)	(0.1, 0.1)	(0.0, 0.1)
	5th - torque	LN	3.4367	**0.4847**	0.7342	0.7805	**0.4847**	0.6474	1.0074	(0.0, 0.0)	(0.3, 0.3)	(0.6, 0.5)
		RF	2.3409				**0.4847**	0.6952	0.8257	(0.0, 0.0)	(0.1, 0.1)	(-0.2, 0.1)
		NN	2.9967				**0.4847**	0.6568	1.0325	(0.0, 0.0)	(-0.1, 0.0)	(0.3, 0.4)
	6th - torque	LN	3.5789	1.1441	1.1173	1.4337	1.1858	1.1562	1.5120	(0.1, 0.2)	(0.2, 0.2)	(0.5, 0.5)
		RF	2.4966				1.1903	1.2049	1.4965	(-0.1, 0.0)	(0.1, 0.0)	(0.3, 0.1)
		NN	3.2119				**1.0033**	1.2771	2.1275	(0.6, 0.6)	(0.3, 0.3)	(0.5, 0.5)
	7th - torque	LN	7.5389	0.9432	1.3170	1.9618	0.9432	1.3111	1.5989	(0.0, 0.0)	(-0.3, 0.2)	(-1.9, 0.3)
		RF	8.8288				1.0461	1.3100	1.5960	(0.4, 0.3)	(0.3, 0.4)	(0.6, 0.7)
		NN	8.8907				**0.7609**	1.3276	1.9390	(0.1, 0.1)	(-0.3, 0.1)	(-0.1, 0.3)
3rd - torque	1st - torque	LN	36.9880	9.2101	14.0481	19.0287	**7.2746**	15.5467	21.0889	(0.7, 0.7)	(-0.1, 0.0)	(-1.9, 0.0)
		RF	32.5157				8.8891	13.3262	14.9954	(0.4, 0.4)	(0.2, 0.1)	(0.9, 0.9)
		NN	37.4099				7.4104	14.0481	20.7171	(0.2, 0.2)	(0.0, 0.0)	(-0.2, 0.0)
	2nd - torque	LN	52.2074	7.1514	9.6002	12.544	5.9891	6.4451	7.8072	(0.0, 0.1)	(0.9, 0.9)	(0.9, 0.9)
		RF	59.2989				6.8353	8.8055	9.9978	(0.1, 0.2)	(0.9, 0.9)	(0.8, 0.8)
		NN	53.1016				**5.6217**	7.0240	8.8396	(0.2, 0.3)	(0.9, 0.9)	(0.5, 0.6)
	4th - torque	LN	14.2936	5.7022	6.3006	8.2317	5.7022	6.6110	8.5316	(0.0, 0.0)	(0.2, 0.1)	(0.6, 0.6)
		RF	12.2491				5.7022	6.9619	9.9118	(0.0, 0.0)	(0.0, 0.1)	(0.9, 0.9)
		NN	18.3896				**4.8195**	6.5787	11.0696	(0.1, 0.1)	(0.1, 0.2)	(0.2, 0.2)
	5th - torque	LN	1.3317	0.4914	0.6113	0.7600	0.4851	0.7878	0.7354	(0.2, 0.2)	(0.8, 0.7)	(0.8, 0.8)
		RF	1.3493				0.5023	0.6150	0.7001	(0.1, 0.1)	(0.8, 0.8)	(0.9, 0.9)
		NN	1.2180				**0.4758**	0.7211	0.9044	(-0.1, 0.0)	(0.7, 0.7)	(0.2, 0.1)
	6th - torque	LN	3.2589	1.0296	1.0342	1.4117	1.1177	1.0614	1.0588	(0.2, 0.2)	(0.3, 0.3)	(0.7, 0.7)
		RF	2.2578				1.1677	1.0440	1.3502	(-0.1, 0.0)	(0.3, 0.2)	(0.7, 0.7)
		NN	3.3803				1.0026	**0.9700**	1.7429	(0.2, 0.3)	(0.3, 0.3)	(0.2, 0.2)
	7th - torque	LN	2.8752	0.9841	1.2593	1.7290	0.8945	1.2976	1.6259	(0.1, 0.1)	(0.2, 0.3)	(0.5, 0.5)
		RF	2.6090				1.0918	1.2990	1.4541	(0.4, 0.5)	(0.5, 0.6)	(0.8, 0.8)
		NN	3.3272				**0.8828**	1.3315	2.5987	(0.2, 0.2)	(0.3, 0.4)	(-0.3, 0.0)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
4th - torque	1st - torque	LN	34.4023	9.0900	14.9821	17.6851	8.0995	12.9936	19.9786	(-0.1, 0.0)	(0.4, 0.4)	(0.9, 0.9)
		RF	29.7653				8.5322	13.1431	14.5646	(0.1, 0.1)	(0.3, 0.3)	(0.9, 0.9)
		NN	29.3330				**6.5680**	11.8912	19.9880	(0.4, 0.4)	(0.4, 0.4)	(0.2, 0.1)
	2nd - torque	LN	61.6112	5.9021	8.9564	11.2634	**5.6931**	9.2281	10.4496	(-0.1, 0.0)	(0.4, 0.4)	(0.6, 0.5)
		RF	53.2669				5.7954	9.1785	9.9473	(0.1, 0.2)	(-0.1, 0.2)	(0.7, 0.6)
		NN	60.5081				5.9021	8.9926	15.0512	(0.0, 0.0)	(0.2, 0.3)	(0.1, 0.0)
	3rd - torque	LN	9.5071	4.8041	6.8103	7.3951	**3.7396**	6.8087	7.3903	(0.2, 0.2)	(-0.1, 0.1)	(0.3, 0.5)
		RF	7.0593				3.9574	6.5700	6.7297	(0.4, 0.4)	(0.0, 0.1)	(0.7, 0.6)
		NN	9.2799				4.1697	6.9722	9.6155	(0.6, 0.6)	(0.5, 0.5)	(-0.2, 0.1)
	5th - torque	LN	1.6568	**0.4847**	0.6529	0.7575	**0.4847**	0.7286	0.8032	(0.0, 0.0)	(-0.4, 0.0)	(0.6, 0.5)
		RF	1.3709				**0.4847**	0.6988	0.8216	(0.0, 0.0)	(0.4, 0.3)	(-0.7, 0.0)
		NN	1.5629				0.5083	0.7514	0.9661	(0.2, 0.2)	(0.3, 0.3)	(0.1, 0.1)
	6th - torque	LN	3.7349	1.1902	1.1738	1.4646	1.3766	1.1738	1.3505	(0.1, 0.1)	(0.0, 0.0)	(0.6, 0.5)
		RF	3.0673				1.2000	**1.1021**	1.3935	(0.0, 0.1)	(-0.1, 0.0)	(0.6, 0.4)
		NN	3.4658				1.2626	1.1738	1.7482	(0.9, 0.9)	(0.0, 0.0)	(-0.3, 0.1)
	7th - torque	LN	1.9605	1.1503	1.4696	1.6791	0.8748	0.7716	0.7966	(0.7, 0.7)	(0.8, 0.8)	(0.8, 0.8)
		RF	1.2223				0.8717	0.9576	0.9453	(0.7, 0.8)	(0.8, 0.9)	(0.7, 0.8)
		NN	1.1993				0.6443	**0.6389**	0.7314	(0.7, 0.7)	(0.9, 0.9)	(0.8, 0.8)
5th - torque	1st - torque	LN	31.5277	9.2101	14.0866	18.1637	7.7211	13.9029	16.2014	(0.3, 0.3)	(0.4, 0.4)	(0.8, 0.8)
		RF	22.2098				8.9821	13.7433	13.9902	(0.1, 0.1)	(0.4, 0.3)	(0.9, 0.9)
		NN	36.9024				**6.8177**	15.6776	22.3460	(0.1, 0.1)	(0.3, 0.2)	(0.2, 0.4)
	2nd - torque	LN	57.1984	**7.1514**	9.9036	12.4656	**7.1514**	10.5736	11.4729	(0.0, 0.0)	(0.5, 0.3)	(0.5, 0.2)
		RF	20.8640				7.2125	10.3424	14.0582	(0.3, 0.3)	(-0.1, 0.0)	(0.9, 0.9)
		NN	50.6720				**7.1514**	11.3694	14.3235	(0.0, 0.0)	(-0.3, 0.1)	(0.0, 0.2)
	3rd - torque	LN	13.1073	4.0836	5.8472	7.6634	**3.5957**	5.7964	6.7290	(0.4, 0.4)	(0.6, 0.6)	(0.5, 0.5)
		RF	12.0133				4.0692	5.4758	5.5414	(-0.1, 0.0)	(0.2, 0.2)	(0.8, 0.8)
		NN	10.1603				3.8504	5.6525	9.9498	(-0.1, 0.0)	(0.2, 0.3)	(-0.1, 0.0)
	4th - torque	LN	18.5145	**4.9229**	7.3865	9.3259	4.9658	7.2390	12.0601	(0.1, 0.0)	(0.1, 0.0)	(0.5, 0.1)
		RF	51.1187				5.6206	7.0200	7.8489	(0.2, 0.2)	(0.4, 0.3)	(0.6, 0.5)
		NN	20.4569				**4.9229**	7.2771	9.3259	(0.0, 0.0)	(0.1, 0.0)	(0.0, 0.0)
	6th - torque	LN	3.3549	1.0296	1.0305	1.4273	1.1063	1.1505	1.7214	(0.1, 0.2)	(0.1, 0.1)	(0.8, 0.8)
		RF	3.3608				1.1890	1.0305	1.5606	(0.2, 0.2)	(0.0, 0.0)	(0.0, 0.1)
		NN	2.6057				1.1182	**1.0052**	1.6663	(0.0, 0.1)	(-0.1, 0.0)	(-0.1, 0.1)
	7th - torque	LN	3.5702	**0.9841**	1.1762	1.8464	**0.9841**	1.7796	2.1937	(0.0, 0.0)	(0.4, 0.1)	(0.6, 0.2)
		RF	6.2471				1.1093	1.4049	1.7934	(0.3, 0.3)	(0.3, 0.2)	(0.9, 0.9)
		NN	3.5524				**0.9841**	1.2099	1.8464	(0.0, 0.0)	(0.3, 0.1)	(0.0, 0.0)

Source task	Target task	$f_s(x)$	Direct	Learning without transfer			$f_w(x;\theta_w)$			Hyperparameter		
				LN	RF	NN	LN	RF	NN	LN	RF	NN
6th - torque	1st - torque	LN	44.5223				9.2817	16.4639	20.4946	(-0.1, 0.0)	(0.8, 0.7)	(0.7, 0.6)
		RF	23.4212	**9.0900**	17.1192	18.0173	**9.0900**	16.4584	19.4356	(0.0, 0.0)	(0.7, 0.6)	(0.8, 0.9)
		NN	35.5639				**9.0900**	16.7334	20.8783	(0.0, 0.0)	(0.2, 0.2)	(-0.1, 0.0)
	2nd - torque	LN	55.8246				5.5422	8.5438	6.9889	(0.3, 0.3)	(0.4, 0.3)	(0.8, 0.8)
		RF	57.9289	5.9021	8.8853	10.6535	5.9021	9.5813	8.9513	(0.0, 0.0)	(0.1, 0.1)	(0.7, 0.5)
		NN	65.0809				**4.8473**	8.2101	15.0102	(0.2, 0.2)	(0.4, 0.3)	(0.5, 0.4)
	3rd - torque	LN	11.8226				4.8041	6.5447	4.7942	(0.0, 0.0)	(0.3, 0.3)	(0.9, 0.9)
		RF	9.9701	4.8041	7.0090	7.5097	4.1922	7.1206	6.1710	(-0.1, 0.0)	(0.9, 0.9)	(0.8, 0.8)
		NN	11.7579				**3.1749**	5.7504	9.4471	(0.3, 0.3)	(0.4, 0.3)	(0.4, 0.4)
	4th - torque	LN	18.6317				6.6960	6.7851	14.4824	(0.0, 0.0)	(0.0, 0.0)	(0.6, 0.2)
		RF	14.1091	6.6960	6.7851	8.0746	7.8263	6.4084	7.0956	(0.2, 0.2)	(0.5, 0.5)	(0.9, 0.9)
		NN	20.9403				**5.5404**	6.6138	8.0746	(-0.1, 0.0)	(0.1, 0.1)	(0.0, 0.0)
	5th - torque	LN	2.4805				**0.4767**	0.7044	0.7184	(0.5, 0.5)	(0.3, 0.2)	(0.3, 0.2)
		RF	2.1727	0.5290	0.7253	0.6816	0.5290	0.6808	0.6977	(0.0, 0.0)	(0.1, 0.0)	(0.3, 0.3)
		NN	3.1706				0.5290	0.7253	0.6816	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
	7th - torque	LN	4.1357				**0.9432**	1.3832	3.6334	(0.0, 0.0)	(0.0, 0.0)	(0.9, 0.9)
		RF	2.7571	**0.9432**	1.3832	1.8069	1.1371	1.2501	1.2843	(0.9, 0.9)	(0.3, 0.2)	(0.9, 0.9)
		NN	5.0599				**0.9432**	1.3832	2.6140	(0.0, 0.0)	(0.0, 0.0)	(0.2, 0.0)
7th - torque	1st - torque	LN	26.1455				8.2293	13.3603	17.8432	(0.5, 0.5)	(0.5, 0.4)	(0.4, 0.4)
		RF	23.1883	9.7971	15.2251	17.863	8.6954	14.7708	15.3444	(0.2, 0.2)	(0.4, 0.5)	(0.9, 0.9)
		NN	25.5984				**6.9347**	13.6454	17.8630	(0.6, 0.6)	(0.6, 0.6)	(0.0, 0.0)
	2nd - torque	LN	60.4728				**6.3065**	9.7745	9.9898	(0.3, 0.3)	(0.4, 0.4)	(0.8, 0.8)
		RF	51.9486	7.1587	9.8779	12.4959	6.4229	9.5041	10.7557	(0.2, 0.1)	(0.3, 0.2)	(0.8, 0.7)
		NN	59.2143				6.6774	10.3166	11.9388	(0.1, 0.1)	(0.1, 0.0)	(0.4, 0.5)
	3rd - torque	LN	10.2416				4.1459	5.7212	5.8396	(0.1, 0.1)	(0.5, 0.5)	(0.5, 0.5)
		RF	7.3155	5.2217	6.7050	7.3494	**3.9404**	6.6709	6.2541	(0.2, 0.2)	(0.5, 0.4)	(0.7, 0.5)
		NN	8.7142				5.2217	6.3961	9.9788	(0.0, 0.0)	(0.3, 0.3)	(-0.1, 0.0)
	4th - torque	LN	12.8400				3.8986	3.7911	4.1658	(0.3, 0.3)	(0.8, 0.8)	(0.9, 0.9)
		RF	7.7968	4.8628	7.9861	8.4601	5.1348	5.6002	5.8848	(0.3, 0.5)	(0.8, 0.8)	(0.9, 0.9)
		NN	10.5561				**3.7655**	3.9798	10.0019	(0.1, 0.1)	(0.7, 0.7)	(0.2, 0.3)
	5th - torque	LN	1.8687				0.5287	0.6895	0.8496	(0.0, 0.0)	(0.2, 0.2)	(0.3, 0.3)
		RF	1.3216	0.5287	0.6874	0.7900	0.5350	0.7038	0.8149	(0.2, 0.2)	(0.3, 0.3)	(0.7, 0.6)
		NN	1.6022				**0.5170**	0.7280	1.0479	(-0.1, 0.0)	(-0.1, 0.0)	(-0.2, 0.0)
	6th - torque	LN	4.3088				1.1733	1.1762	2.1451	(0.0, 0.0)	(0.0, 0.0)	(0.9, 0.9)
		RF	2.7986	1.1733	1.1762	1.3857	1.1798	1.2039	1.5165	(-0.1, 0.0)	(0.4, 0.2)	(0.6, 0.3)
		NN	3.8235				**1.1419**	1.1762	1.6955	(0.1, 0.2)	(0.0, 0.0)	(0.4, 0.4)

Chapter 4

Affine model transfer

In conventional supervised TL, cross-domain differences are modeled and estimated by using a given set of source models and samples of a target domain. For example, if there is a functional relationship between the source and target domains, only domain-specific factors are additionally learned using the target samples to shift the source models to the target. However, the general methodology for modeling and estimating such cross-domain shifts has been less studied. This chapter presents a TL framework that simultaneously and separately estimates the domain shifts and domain-specific factors using the given target samples. Assuming consistency and invertibility of the domain transformation functions, we derive an optimal family of functions to represent the cross-domain shift. The newly derived class of transformation functions takes the same form as that of invertible neural networks by using affine coupling layers, which are widely used in generative deep learning. We show that the proposed method encompasses a wide range of existing methods, including the most common TL procedure based on feature extraction using neural networks. We also clarify the theoretical properties of the proposed method, such as the convergence rate of the generalization error, and demonstrate the practical benefits of separately modeling and estimating the domain-specific factors through several case studies.

4.1 Transfer learning via transformation function

4.1.1 Affine model transfer

This chapter considers regression problems with squared loss. Recall that we assume that the output of the target domain $y \in \mathcal{Y} \subset \mathbb{R}$ follows $y = f_t(x) + \epsilon$, where $f_t : \mathcal{X} \to \mathbb{R}$ is the true model on the target domain, and the observation noise ϵ has mean zero and variance σ^2. We are given n samples $\{(x_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$ from the target domain and the feature representation $f_s(x) \in \mathcal{F}_s$ from one or more source

domains. Typically, f_s is given as a vector $f_s(x) = [f_1(x), f_2(x), \ldots, f_M(x)]^\top$, which includes the output of the source models, observed data in the source domains, or learned features in a pre-trained source neural network, etc., but it can also be a non-vector feature such as a tensor, graph, or text. Hereinafter, f_s is referred to as the source features.

As described in Section 2.1.2, Du et al. (2017) provided a TL framework using the transformation function. The following is a slight generalization of its procedure:

1. With the source features, perform a variable transformation of the observed outputs as $z_i = \phi(y_i, f_s(x_i))$ using the data transformation function $\phi : \mathcal{Y} \times \mathcal{F}_s \to \mathbb{R}$.

2. Train an intermediate model $\hat{g}(x)$ using the transformed sample set $\{(x_i, z_i)\}_{i=1}^n$ to predict the transformed output z for any given x.

3. Obtain a target model $\hat{f}_t(x) = \psi(\hat{g}(x), f_s(x))$ using the model transformation function $\psi : \mathbb{R} \times \mathcal{F}_s \to \mathcal{Y}$ that combines $\hat{g}$ and f_s to define a predictor.

The difference from Du et al. (2017) is that we use two transformation functions: data transformation function and model transformation function. Du et al. (2017) considers the case where the model transformation function ψ is equal to the inverse of the data transformation function ϕ^{-1}, but we consider a more generical case that eliminates this constraint.

This class of TL includes several approaches proposed in previous studies. For example, Kuzborskij & Orabona (2013; 2017) proposed a learning algorithm consisting of linear data transformation and linear model transformation, $\phi = y - \langle \theta, f_s(x) \rangle$ and $\psi = g(x) + \langle \theta, f_s(x) \rangle$, with a pre-defined coefficient θ. In this case, the factors unexplained by the linear combination of source features are learned with g, and the target output is predicted additively with the common factor $\langle \theta, f_s(x) \rangle$ and the additionally learned g. Recall that, in Section 3, we developed a Bayesian TL with density-ratio modeling (Minami et al., 2021), which is equivalent to the following transformation functions: for $\mathcal{F}_s \subset \mathbb{R}$, $\phi = (y - \tau f_s(x))/(1 - \tau)$ and $\psi = \rho g(x) + (1 - \rho) f_s(x)$ with two varying hyperparameters $\tau < 1$ and $0 \leq \rho \leq 1$. This includes TL using density-ratio estimation (Liu & Fukumizu, 2016) and neural network-based fine-tuning as special cases when the two hyperparameters belong to specific regions.

The performance of this TL procedure strongly depends on the design of the two transformation functions ϕ and ψ. In other words, if unfavorable transformation functions are used, the TL will fail. For example, multiplicative transformation functions do not accurately capture linear domain shifts. In this section, we mathematically derive the properties that the transformation functions must satisfy.

For simplicity, we denote the transformation functions as $\phi_{f_s}(\cdot) = \phi(\cdot, f_s(x))$ and $\psi_{f_s}(\cdot) = \psi(\cdot, f_s(x))$. To derive the optimal class of ϕ and ψ, we make the following assumptions:

Assumption 4.1 (Differentiability)**.** *The data transformation function ϕ is differentiable with respect to the first argument.*

Assumption 4.2 (Invertibility)**.** *The model transformation function ψ is invertible with respect to the first argument, i.e., its inverse $\psi_{f_s}^{-1}$ exists.*

Assumption 4.3 (Consistency)**.** *For any distribution on the target domain $p_t(x, y)$, and for all $x \in \mathcal{X}$,*

$$\psi_{f_s}(g^*(x)) = \mathbb{E}_{p_t}[Y|X = x],$$

where $g^*(x) = \mathbb{E}_{p_t}[\phi_{f_s}(Y)|X = x]$.

The regression function that minimizes the mean squared error is given by the conditional mean. In Assumption 4.3, g^* is defined to be the best predictor function for the transformed variable $z = \phi_{f_s}(y)$ in terms of the mean squared error. Assumption 4.3 states that composing the optimal g^* with the model transformation function ψ_{f_s} leads to the best predictor $\mathbb{E}_{p_t}[Y|X = x]$ for the target domain. This assumption corresponds to the unbiased condition of Du et al. (2017).

Under these assumptions, we derive the optimal class of the transformation functions, which minimizes the mean squared error.

Theorem 4.1. *Let $g_1, g_2 : \mathcal{F}_s \to \mathbb{R}$ denote arbitrary functions. If Assumptions 4.1–4.3 hold, then the transformation functions ϕ and ψ satisfy the following two properties:*

(i) $\psi_{f_s}^{-1} = \phi_{f_s}$.

(ii) $\psi_{f_s}(g) = g_1(f_s) + g_2(f_s) \cdot g$.

Proof. According to Assumption 4.3, it holds that for any $p_t(y|x)$,

$$\psi_{f_s}\left(\int \phi_{f_s}(y)p_t(y|x)\mathrm{d}y\right) = \int yp_t(y|x)\mathrm{d}y. \tag{4.1}$$

(i) Let δ_{y_0} be the Dirac delta function supported on y_0. By substituting $p_t(y|x) = \delta_{y_0}$ into Eq. (4.1), we have

$$\psi_{f_s}(\phi_{f_s}(y_0)) = y_0 \ (\forall y_0 \in \mathcal{Y}).$$

Under Assumption 4.2, this implies the property (i).

(ii) For simplicity, we assume the input x is fixed and $p_t(y|x) > 0$. Applying the property (i) to Eq. (4.1) yields

$$\int \phi_{f_s}(y)p_t(y|x)\mathrm{d}y = \phi_{f_s}\left(\int yp_t(y|x)\mathrm{d}y\right).$$

We consider a two-component mixture $p_t(y|x) = (1-\epsilon)q(y|x) + \epsilon h(y|x)$ with a mixing rate $\epsilon \in [0,1]$, where q and h denote arbitrary probability density functions. Then, we have

$$\int \phi_{f_s}(y)\big\{(1-\epsilon)q(y|x) + \epsilon h(y|x)\big\}\mathrm{d}y = \phi_{f_s}\bigg(\int y\big\{(1-\epsilon)q(y|x) + \epsilon h(y|x)\big\}\mathrm{d}y\bigg).$$

Taking the derivative at $\epsilon = 0$, we have

$$\int \phi_{f_s}(y)\big\{h(y|x) - q(y|x)\big\}\mathrm{d}y = \phi'_{f_s}\bigg(\int yq(y|x)\mathrm{d}y\bigg)\bigg(\int y\big\{h(y|x) - q(y|x)\big\}\mathrm{d}y\bigg),$$

which yields

$$\int \big\{h(y|x) - q(y|x)\big\}\big\{\phi_{f_s}(y) - \phi'_{f_s}\big(\mathbb{E}_q[Y|X=x]\big)y\big\}\mathrm{d}y = 0. \tag{4.2}$$

For Eq. (4.2) to hold for any q and h, $\phi_{f_s}(y) - \phi'_{f_s}\big(\mathbb{E}_q[Y|X=x]\big)y$ must be independent of y. Thus, the function ϕ_{f_s} and its inverse $\psi_{f_s} = \phi_{f_s}^{-1}$ are limited to affine transformations. □

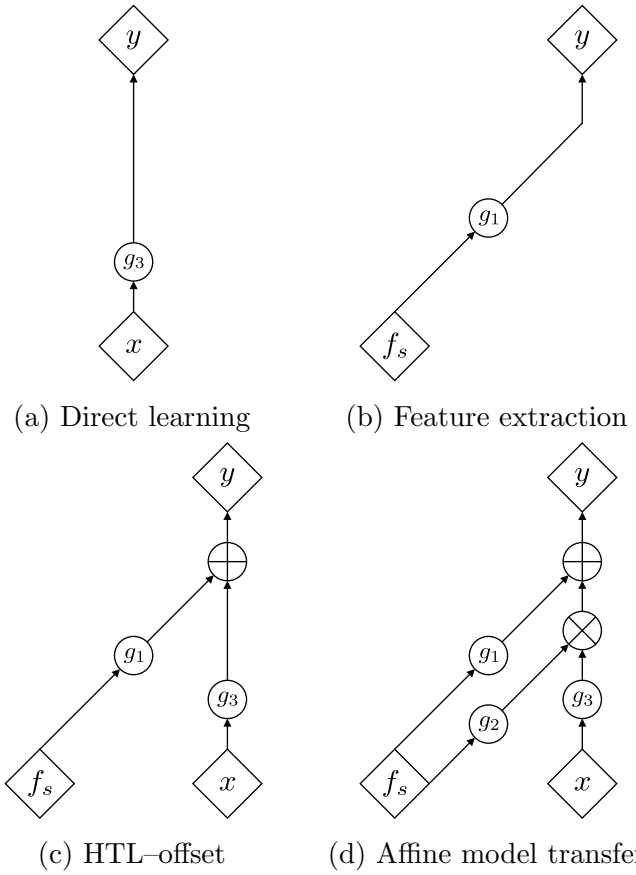
Theorem 4.1 implies that the mean squared error is minimized when the data and model transformation functions are given by an affine transformation and its inverse, respectively. In summary, the optimal class for HTL is expressed as follows:

$$\mathcal{H} = \big\{x \mapsto g_1(f_s(x)) + g_2(f_s(x)) \cdot g_3(x) \mid g_1 \in \mathcal{G}_1, g_2 \in \mathcal{G}_2, g_3 \in \mathcal{G}_3\big\},$$

where $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_3$ are arbitrarily function classes. Here, g_1 and g_2 are modeled as functions of f_s that represent the common factors between the source and target domains. g_3 is modeled as a function of x to capture the domain-specific factors unexplainable by the source features.

We have derived the optimal form of the transformation functions when the squared loss is employed. Even for general convex loss functions, (i) of Theorem 4.1 still holds. However, (ii) of Theorem 4.1 does not generally hold because the optimal transformation function depends on the employed loss. The optimal models for various convex loss functions are discussed in Section 4.1.3.

Here, the affine transformation is found to be optimal in terms of minimizing the mean squared error. We can also derive the same function class by minimizing the upper bound of the estimation error in the HTL procedure. The details are discussed in Section 4.1.3.



(a) Direct learning (b) Feature extraction

(c) HTL–offset (d) Affine model transfer

Figure 4.1: Model architectures for the affine model transfer and related procedures. (a) Direct learning predicts the output using only the original input x, whereas (b) feature extraction-based neural transfer predicts the output using only the source features f_s. (c) The HTL procedure proposed in Kuzborskij & Orabona (2013) (HTL-offset) constructs the predictor as the sum of $g_1(f_s)$ and $g_3(x)$. (d) The affine model transfer encompasses the above procedures, computing g_1 and g_2 as functions of the source features and constructing the predictor as an affine combination with g_3.

4.1.2 Relation to existing methods

The affine model transfer encompasses some existing TL procedures, for example, a learning scheme without transfer. By setting $g_1(f_s) = 0$ and $g_2(f_s) = 1$, the prediction model is estimated without using the source features, which corresponds to an ordinary direct learning procedure.

In a prior study, Kuzborskij & Orabona (2013) employed a two-step procedure, where the source features were combined with pre-defined weights, and an auxiliary model was additionally learned for the residuals unexplainable by the source features. The affine model transfer can represent this HTL as a special case by setting $g_2(f_s) = 1$, but differs in the following two aspects. First, the existing method models only the difference (residuals) from the source domains, whereas our model also considers the cross-domain ratio relationship, i.e., we also consider the function $g_2(f_s)$. Another distinctive feature of affine model transfer is the learning procedure, which can estimate the data and model transformation functions simultaneously, as described in Section 4.2.

The affine model transfer can be naturally expressed as an architecture of neural networks. This architecture, called affine coupling layers, is widely used for invertible neural networks in flow-based generative modeling (Dinh et al., 2014; 2017). Neural networks based on affine coupling layers have been proven to have universal approximation ability (Teshima et al., 2020). This implies that the affine transfer model has the potential to represent a wide range of function classes, despite its simple architecture based on the affine coupling of three functions.

As mentioned in Section 2.1.2, when a pre-trained source model is provided as a neural network, TL is usually performed with the intermediate layer as an input to the model in the target domain. This is called a feature extractor or frozen featurizer, and has been experimentally and theoretically proven to have strong transfer capability, making it the de facto standard for TL (Yosinski et al., 2014; Tripuraneni et al., 2020). The affine model transfer encompasses the neural feature extractor as a special subclass, which is equivalent to setting $g_2(f_s)g_3(x) = 0$. A performance comparison of the affine model transfer with the neural feature extractor is presented in Section 4.4.2.

The affine model transfer can also be interpreted as generalizing the feature extractor by adding a product term $g_2(f_s)g_3(x)$. This additional term allows for the inclusion of unknown factors in the transferred model that are unexplainable by source features alone. Furthermore, this assists in avoiding negative transfer. The usual TL based only on $g_1(f_s)$ attempts to explain and predict the data generation process in the target domain by using features from only the source domain. However, in the presence of domain-specific factors, a negative transfer can occur owing to a lack of descriptive power. The additional term compensates for this shortcoming. The

comparison with the behavior in the case with non-relative source features is described in Section 4.4.1.

4.1.3 Other perspectives on affine model transfer

In Section 4.1.1, we derived the optimal form of the transformation functions when the squared loss is employed. This section discusses other perspectives on the transformation functions.

Transformation functions for general loss functions

Here, we discuss the optimal transformation function for general loss functions.

Let $\ell(y, y') \geq 0$ be a convex loss function that returns zero if and only if $y = y'$, and let $g^*(x)$ be the optimal predictor that minimizes the expectation of ℓ with respect to the distribution p_t, followed by x and y transformed by ϕ:

$$g^*(x) = \arg\min_g \mathbb{E}_{p_t}\big[\ell(g(x), \phi_{f_s}(y))\big].$$

The function g that minimizes the expected loss

$$\mathbb{E}_{p_t}\big[\ell(g(x), \phi_{f_s}(y))\big] = \iint \ell(g(x), \phi_{f_s}(y)) p_t(x, y) \mathrm{d}x \mathrm{d}y$$

should be a solution to the Euler–Lagrange equation

$$\frac{\partial}{\partial g(x)} \int \ell(g(x), \phi_{f_s}(y)) p_t(x, y) \mathrm{d}y = \int \frac{\partial}{\partial g(x)} \ell(g(x), \phi_{f_s}(y)) p_t(y|x) \mathrm{d}y \, p_t(x) = 0. \tag{4.3}$$

Denote the solution of Eq. (4.3) by $G(x; \phi_{f_s})$. Although G depends on the loss ℓ and distribution p_t, we omit these considerations from the argument for notational simplicity. Using this notation, the minimizer of the expected loss $\mathbb{E}_{x,y}[\ell(g(x), y)]$ can be expressed as $G(x; \mathrm{id})$, where id denotes an identity function.

Here, we consider the following assumption that generalizes Assumption 4.3:

Assumption 4.4. *For any distribution on the target domain $p_t(x, y)$ and all $x \in \mathcal{X}$, the following holds:*

$$\psi_{f_s}(g^*(x)) = \arg\min_g \mathbb{E}_{x,y}[\ell(g(x), y)].$$

Equivalently, the transformation functions ϕ_{f_s} and ψ_{f_s} satisfy

$$\psi_{f_s}\big(G(x; \phi_{f_s})\big) = G(x; \mathrm{id}). \tag{4.4}$$

Assumption 4.4 states that if the best predictor $G(x;\phi_{f_s})$ for the data transformed by ϕ is provided to the model transformation function ψ, it is consistent with the overall best predictor $G(x;\mathrm{id})$ in the target region in terms of the loss function ℓ. We consider all pairs of ψ and ϕ that satisfy this consistency condition.

Here, let us prove the following proposition:

Proposition 4.2. *Under Assumption 4.1, 4.2, and 4.4, it holds that* $\psi_{f_s}^{-1} = \phi_{f_s}$.

Proof. The proof is analogous to that of Theorem 4.1. For any $y_0 \in \mathcal{Y}$, let $p_t(y|x) = \delta_{y_0}$. Combining this with Eq. (4.3) leads to

$$\frac{\partial}{\partial g(x)} \ell(g(x), \phi_{f_s}(y_0)) = 0 \ (\forall y_0 \in \mathcal{Y}).$$

Because $\ell(y, y')$ returns the minimum value zero if and only if $y = y'$, we obtain $G(x;\phi_{f_s}) = \phi_{f_s}(y_0)$. Similarly, we have $G(x;\mathrm{id}) = y_0$. From these two facts and Assumption 4.4, we have $\psi_{f_s}(\phi_{f_s}(y_0)) = y_0$, proving that the proposition is true. □

Proposition 4.2 indicates that the first statement of Theorem 4.1 holds for general loss functions. However, the second assertion of Theorem 4.1 generally depends on the type of loss function. Through the following examples, we describe the optimal class of transformation functions for several loss functions.

Example 1 (Squared loss)**.** Let $\ell(y, y') = |y - y'|^2$. As a solution of Eq. (4.3), we can see that the optimal predictor is the conditional expectation $\mathbb{E}_{p_t}[\phi_{f_s}(Y)|X = x]$. As discussed in Section 4.1, the transformation functions ϕ_{f_s} and ψ_{f_s} should be affine transformations.

Example 2 (Absolute loss)**.** Let $\ell(y, y') = |y - y'|$. Substituting this into Eq. (4.3), we have

$$\begin{aligned} 0 &= \int \frac{\partial}{\partial g(x)} \big|g(x) - \phi_{f_s}(y)\big| p_t(y|x) \mathrm{d}y \\ &= \int \mathrm{sign}\big(g(x) - \phi_{f_s}(y)\big) p_t(y|x) \mathrm{d}y \\ &= \int_{\phi_{f_s}(y) \geq g(x)} p_t(y|x) \mathrm{d}y - \int_{\phi_{f_s}(y) < g(x)} p_t(y|x) \mathrm{d}y. \end{aligned}$$

Assuming that ϕ_{f_s} is monotonically increasing, we have

$$0 = \int_{y \geq \phi_{f_s}^{-1}(g(x))} p_t(y|x) \mathrm{d}y - \int_{y < \phi_{f_s}^{-1}(g(x))} p_t(y|x) \mathrm{d}y.$$

This yields

$$\int_{\phi_{f_s}^{-1}(g(x))}^{\infty} p_t(y|x)\mathrm{d}y = \int_{-\infty}^{\phi_{f_s}^{-1}(g(x))} p_t(y|x)\mathrm{d}y.$$

The same result is obtained even if ϕ_{f_s} is monotonically decreasing. Consequently,

$$\phi_{f_s}^{-1}(g(x)) = \mathrm{Median}[Y|X=x],$$

which results in

$$G(x;\phi_{f_s}) = \phi_{f_s}\big(\mathrm{Median}[Y|X=x]\big).$$

This implies that Eq. (4.4) holds for any ϕ_{f_s} including an affine transformation, and the function form cannot be identified from this analysis.

Example 3 (Exponential-squared loss)**.** As an example of a case where the affine transformation is not optimal, consider the loss function $\ell(y,y') = |e^y - e^{y'}|^2$. Substituting this into Eq. (4.3), we have

$$\begin{aligned} 0 &= \int \frac{\partial}{\partial g(x)} \big|\exp(g(x)) - \exp(\phi_{f_s}(y))\big|^2 p_t(y|x)\mathrm{d}y \\ &= 2\exp(g(x)) \int \big(\exp(g(x)) - \exp(\phi_{f_s}(y))\big) p_t(y|x)\mathrm{d}y. \end{aligned}$$

Therefore,

$$G(x;\phi_{f_s}) = \log \mathbb{E}\big[\exp(\phi_{f_s}(Y))|X=x\big].$$

Consequently, Eq. (4.4) becomes

$$\log \mathbb{E}\big[\exp(\phi_{f_s}(Y))\big] = \phi_{f_s}\big(\log \mathbb{E}\big[\exp(Y)\big]\big).$$

Even when ϕ_{f_s} is an affine transformation, this equation does not generally hold.

Example 4 (0-1 loss)**.** For the classification settings, the 0-1 loss $\ell(y,y') = \mathbb{1}_{y\neq y'}$ is commonly used, where $y, y' \in \{-1, 1\}$ and $\mathbb{1}$ is an indicator function, i.e., $\mathbb{1}_{y\neq y'} = 1$ if $y \neq y'$, and $\mathbb{1}_{y\neq y'} = 0$ otherwise. In this case, the optimal predictor g^* is the solution of the following minimization problem:

$$\arg\min_g \int \mathbb{1}_{g(x)\neq\phi_{f_s}(y)} p_t(y|x)\mathrm{d}y = \int \mathbb{1}_{\phi_{f_s}^{-1}(g)\neq y} p_t(y|x)\mathrm{d}y. \tag{4.5}$$

Because the indicator function takes the value 0 if and only if $\phi_{f_s}^{-1}(g) = y$, to minimize the objective function of Eq.(4.5), we should set $\phi_{f_s}^{-1}(g)$ on the maximizer of $p_t(y|x)$. Consequently, we have

$$g^*(x) = \phi_{f_s}\big(\arg\max p_t(y|x)\big).$$

As in Example 2, the function form cannot be identified from this analysis.

Example 5 (logistic loss)**.** Another choice for the loss function in classification problems is the logistic loss $\ell(y, y') = \log(1 + \exp(-yy'))$ for $y, y' \in \mathbb{R}$. This loss function does not return zero even if $y = y'$. Therefore, Proposition 4.2 does not hold, and the transformation functions need not be inverse functions of each other.

Analysis of the optimal function class based on the upper bound of the estimation error

Here, we discuss the optimal class for the transformation function based on the upper bound of the estimation error.

In addition to Assumptions 4.1 and 4.2, we assume the following in place of Assumption 4.3:

Assumption 4.5. *The transformation functions ϕ and ψ are Lipschitz continuous with respect to the first argument, i.e., there exist constants μ_ϕ and μ_ψ such that*

$$\phi(a, c) - \phi(a', c) \leq \mu_\phi \|a - a'\|_2, \quad \psi(b, c) - \psi(b', c) \leq \mu_\psi \|b - b'\|_2$$

for any $a, a' \in \mathcal{Y}$ and $b, b' \in \mathbb{R}$, and for any given $c \in \mathcal{F}_s$.

Note that each Lipschitz constant is a function of the second argument f_s, i.e., $\mu_\phi = \mu_\phi(f_s)$ and $\mu_\psi = \mu_\psi(f_s)$.

Under Assumptions 4.1, 4.2, and 4.5, the estimation error is upper bounded as

follows:

$$\begin{aligned}
\mathop{\mathbb{E}}_{x,y}\Big[\big|f_t(x)-\hat{f}_t(x)\big|^2\Big] &= \mathop{\mathbb{E}}_{x,y}\Big[\big|\psi(g(x),f_s(x))-\psi(\hat{g}(x),f_s(x))\big|^2\Big] \\
&\le \mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\big|g(x)-\hat{g}(x)\big|^2\Big] \\
&\le 3\mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\big(\big|g(x)-\phi(f_t(x),f_s(x))\big|^2 \\
&\qquad\qquad + \big|\phi(f_t(x),f_s(x))-\phi(y,f_s(x))\big|^2 \\
&\qquad\qquad + \big|\phi(y,f_s(x))-\hat{g}(x)\big|^2\big)\Big] \\
&\le 3\mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\big|\psi^{-1}(f_t(x),f_s(x))-\phi(f_t(x),f_s(x))\big|^2\Big] \\
&\qquad\qquad + 3\mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\mu_\phi(f_s(x))^2\big|f_t(x)-y\big|^2\Big] \\
&\qquad\qquad + 3\mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\big|z-\hat{g}(x)\big|^2\Big] \\
&= 3\mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\big|\psi^{-1}(f_t(x),f_s(x))-\phi(f_t(x),f_s(x))\big|^2\Big] \\
&\qquad\qquad + 3\sigma^2\mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\mu_\phi(f_s(x))^2\Big] \\
&\qquad\qquad + 3\mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^2\big|z-\hat{g}(x)\big|^2\Big].
\end{aligned} \tag{4.6}$$

The derivation of this inequality is based on Du et al. (2017). We use the Lipschitz property of ψ and ϕ for the first and third inequalities, and the second inequality comes from the numeric inequality $(a-d)^2 \le 3(a-b)^2+3(b-c)^2+3(c-d)^2$ for $a,b,c,d\in\mathbb{R}$.

According to this inequality, the upper bound of the estimation error is decomposed into three terms: discrepancy between the two transformation functions, variance of the noise, and estimation error for the transformed data. Although it is intractable to find an optimal solution of ϕ,ψ, and $\hat{g}$ that minimizes all these terms together, it is possible to find a solution that minimizes the first and second terms expressed as the functions of only ϕ and ψ. Obviously, the first term, which represents the discrepancy between the two transformation functions, reaches its minimum (zero) when $\phi_{f_s}=\psi_{f_s}^{-1}$. The second term, which is related to the variance of the noise, is minimized when the differential coefficient $\frac{\partial}{\partial u}\psi_{f_s}(u)$ is a constant, i.e., when ψ_{f_s} is a linear function. This is verified as follows. From $\phi_{f_s}=\psi_{f_s}^{-1}$ and the continuity of ψ_{f_s}, it follows that

$$\mu_\psi=\max\Big|\frac{\partial}{\partial u}\psi_{f_s}(u)\Big|,\quad \mu_\phi=\max\Big|\frac{\partial}{\partial u}\psi_{f_s}^{-1}(u)\Big|=\frac{1}{\min\big|\frac{\partial}{\partial u}\psi_{f_s}(u)\big|}.$$

Thus, the product $\mu_\phi \mu_\psi$ takes the minimum value (one) when the maximum and minimum of the differential coefficient are the same. Therefore, we can write

$$\phi(y, f_s(x)) = \frac{y - g_1(f_s(x))}{g_2(f_s(x))}, \quad \psi(g(x), f_s(x)) = g_1(f_s(x)) + g_2(f_s(x))g(x),$$

where $g_1, g_2 : \mathcal{F}_s \to \mathbb{R}$ are arbitrary functions. Thus, the minimization of the third term in the upper bound of the estimation error can be expressed as

$$\min_{g_1, g_2, g} \mathop{\mathbb{E}}_{x,y} |y - g_1(f_s(x)) + g_2(f_s(x))g(x)|^2.$$

As a result, the suboptimal function class for the upper bound of the estimated function is given as

$$\mathcal{H} = \big\{ x \mapsto g_1(f_s(x)) + g_2(f_s(x)) \cdot g_3(x) \mid g_1 \in \mathcal{G}_1, g_2 \in \mathcal{G}_2, g_3 \in \mathcal{G}_3 \big\}.$$

This is the same function class derived in Section 4.1.

It is notable that the same result can be derived for the loss function $\ell(y, y') = |y - y'|^p$, $0 < p \leq 1$. Let $m_p = \mathbb{E}|\epsilon|^p$. As in the case of Eq. (4.6), we have

$$\begin{aligned} \mathbb{E}_{x,y}\big[|f_t(x) - \hat{f}_t(x)|^p\big] \leq \mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^p \big|\psi^{-1}(f_t(x), f_s(x)) - \phi(f_t(x), f_s(x))\big|^p\Big] \\ + m_p \mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^p \mu_\phi(f_s(x))^p\Big] \\ + \mathop{\mathbb{E}}_{x,y}\Big[\mu_\psi(f_s(x))^p \big|z - \hat{g}(x)\big|^p\Big]. \end{aligned} \tag{4.7}$$

because for $0 < p \leq 1$, the inequality $|a - d|^p \leq |a - b|^p + |b - c|^p + |c - d|^p$ holds for $a, b, c, d \in \mathbb{R}$.

Transformation functions for classification problems Here, we discuss the case of classification settings, i.e., when we use the 0-1 loss or logistic loss instead of the squared loss.

For $y, y' \in \{-1, 1\}$, we have $\mathbb{1}_{y \neq y'} = \frac{1}{2}|y - y'|$. Eq. (4.6) with $p = 1$ suggests that we obtain the same conclusion as in the case of the squared loss when the 0-1 loss is adopted. However, for the case of the logistic loss $\ell(y, y') = \log(1 + \exp(-yy'))$, $y, y' \in \mathbb{R}$, the derivation used in Eq. (4.6) or Eq. (4.7) cannot be applied and it is impossible to conclude that ϕ and ψ are linear, or even that they are in an inverse function relationship.

Algorithm 4.1 Block relaxation algorithm (Zhou et al., 2013).

Initialize
$a_0, \quad b_0 \neq 0, \quad c_0 \neq 0$
repeat
$a_{t+1} = \arg\min_a F(a,\ b_t,\ c_t)$
$b_{t+1} = \arg\min_b F(a_{t+1},\ b,\ c_t)$
$c_{t+1} = \arg\min_c F(a_{t+1},\ b_{t+1},\ c)$
until convergence

4.2 Modeling and estimation

In this section, we focus on using kernel methods for the affine transfer model. Let $\mathcal{H}_1, \mathcal{H}_2$, and $\mathcal{H}_3$ be reproducing kernel Hilbert spaces (RKHSs) with positive-definite kernels k_1, k_2, and k_3, which define the feature mappings $\Phi_1 : \mathcal{F}_s \to \mathcal{H}_1, \Phi_2 : \mathcal{F}_s \to \mathcal{H}_2$, and $\Phi_3 : \mathcal{X} \to \mathcal{H}_3$, respectively. For the proposed model class, the ℓ_2-regularized empirical risk with the squared loss is given as follows:

$$F_{\alpha,\beta,\gamma} = \frac{1}{n}\sum_{i=1}^{n}\left\{y_i - \langle\alpha, \Phi_1(f_s(x_i))\rangle_{\mathcal{H}_1} - \langle\beta, \Phi_2(f_s(x_i))\rangle_{\mathcal{H}_2}\langle\gamma, \Phi_3(x_i)\rangle_{\mathcal{H}_3}\right\}^2 + \lambda_1\|\alpha\|^2_{\mathcal{H}_1} + \lambda_2\|\beta\|^2_{\mathcal{H}_2} + \lambda_3\|\gamma\|^2_{\mathcal{H}_3}, \tag{4.8}$$

where $\lambda_1, \lambda_2, \lambda_3 > 0$ are hyperparameters for the regularization. According to the representer theorem, the minimizer of $F_{\alpha,\beta,\gamma}$ with respect to the parameters $\alpha \in \mathcal{H}_1$, $\beta \in \mathcal{H}_2$, and $\gamma \in \mathcal{H}_3$ reduces to

$$\alpha = \sum_{i=1}^{n} a_i\Phi_1(f_s(x_i)), \quad \beta = \sum_{i=1}^{n} b_i\Phi_2(f_s(x_i)), \quad \gamma = \sum_{i=1}^{n} c_i\Phi_3(x_i)$$

with the n-dimensional unknown parameter vectors $a, b, c \in \mathbb{R}^n$. Substituting this expression into Eq. (4.8), we can obtain the objective function as

$$\begin{aligned} F_{\alpha,\beta,\gamma} &= \frac{1}{n}\|y - K_1a - (K_2b)\circ(K_3c)\|_2^2 + \lambda_1 a^\top K_1 a + \lambda_2 b^\top K_2 b + \lambda_3 c^\top K_3 c \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - k_1^{(i)\top}a - b^\top M^{(i)}c\right)^2 + \lambda_1 a^\top K_1 a + \lambda_2 b^\top K_2 b + \lambda_3 c^\top K_3 c \\ &\coloneqq F(a, b, c). \end{aligned} \tag{4.9}$$

Here, the symbol $\circ$ denotes the Hadamard product. K_I is the Gram matrix associated with the kernel k_I for $I \in \{1, 2, 3\}$. $k_I^{(i)} = [k_I(x_i, x_1) \cdots k_I(x_i, x_n)]^\top$ denotes the i-th column of the Gram matrix. The $n \times n$ matrix $M^{(i)}$ is given by the tensor product $M^{(i)} = k_2^{(i)} \otimes k_3^{(i)}$ of $k_2^{(i)}$ and $k_3^{(i)}$.

Because the model is linear with respect to the parameter a and bilinear for b and c, the optimization of Eq. (4.9) can be solved using well-established techniques for low-rank tensor regression, such as CP-decomposition (Harshman, 1970), Tucker decomposition (Tucker, 1966), and Tensor-Train decomposition (Oseledets, 2011). In this chapter, we use the block relaxation algorithm (Zhou et al., 2013) described in Algorithm 4.1. It updates a, b, and c by repeatedly fixing two of the three parameters and minimizing the objective function for the remaining parameter. By fixing two parameters, the resulting subproblem can be solved analytically, because the objective function is expressed in a quadratic form for the remaining parameter. Starting from arbitrary initial values, the algorithm iteratively updates the parameters (a_t, b_t, c_t) at iteration t to $(a_{t+1}, b_{t+1}, c_{t+1})$ as follows:

$$
\begin{aligned}
a_{t+1} &= (K_1 + n\lambda_1 I_n)^{-1}(y - (K_2 b_t) \circ (K_3 c_t)), \\
b_{t+1} &= (\mathrm{diag}(K_3 c_t)^2 K_2 + n\lambda_2 I_n)^{-1}\mathrm{diag}(K_3 c_t)(y - K_1 a_{t+1}), \\
c_{t+1} &= (\mathrm{diag}(K_2 b_{t+1})^2 K_3 + n\lambda_3 I_n)^{-1}\mathrm{diag}(K_2 b_{t+1})(y - K_1 a_{t+1}),
\end{aligned}
$$

where y is a vector of n observed outputs, I_n denotes the identity matrix of size n, and $\mathrm{diag}(v)$ is a diagonal matrix whose diagonal element is given by the vector v.

Algorithm 4.1 alternately estimates the parameters (a, b) of the transformation function and the parameter c in the predictive model of the transformed output with the given transformed dataset $\{(x_i, z_i)\}_{i=1}^n$. The consistency and asymptotic normality of this estimator have been proven in Zhou et al. (2013).

4.3 Theoretical results

In this section, we present two theoretical properties—generalization bound and excess risk bound.

To begin with, recall the notations introduced in Section 2.2. Let $(\mathcal{Z}, P)$ be an arbitrary probability space, and set $\{z_i\}_{i=1}^n$ to be independent random variables distributed according to P. For a function $f : \mathcal{Z} \to \mathbb{R}$, define the expectation of f with respect to P and its empirical counterpart as

$$
Pf = \mathbb{E}_P f(z), \qquad P_n f = \frac{1}{n}\sum_{i=1}^{n} f(z_i).
$$

Let $\ell(y, y')$ be a non-negative loss bounded from above by $L > 0$ such that for any fixed $y' \in \mathcal{Y}$, $y \mapsto \ell(y, y')$ is μ_ℓ-Lipschitz for some $\mu_\ell > 0$.

Recall that the function class proposed in this chapter is

$$\mathcal{H} = \big\{x \mapsto g_1(f_s(x)) + g_2(f_s(x)) \cdot g_3(x) \mid g_1 \in \mathcal{G}_1, g_2 \in \mathcal{G}_2, g_3 \in \mathcal{G}_3\big\}.$$

In particular, the following discussion in this section assumes that g_1, g_2, and g_3 are represented by linear functions on the RKHSs.

4.3.1 Generalization bound

The optimization problem is expressed as follows:

$$\min_{\alpha,\beta,\gamma} P_n \ell\big(y,\ \langle \alpha, \Phi_1\rangle_{\mathcal{H}_1} + \langle \beta, \Phi_2\rangle_{\mathcal{H}_2}\langle \gamma, \Phi_3\rangle_{\mathcal{H}_3}\big) + \lambda_\alpha \|\alpha\|^2_{\mathcal{H}_1} + \lambda_\beta \|\beta\|^2_{\mathcal{H}_2} + \lambda_\gamma \|\gamma\|^2_{\mathcal{H}_3}, \quad (4.10)$$

where $\Phi_1 = \Phi_1(f_s(x)), \Phi_2 = \Phi_2(f_s(x))$, and $\Phi_3 = \Phi_3(x)$ denote the feature maps, and $\lambda_\alpha, \lambda_\beta, \lambda_\gamma > 0$ are the regularization parameters. Without loss of generality, it is assumed that $\|\Phi_1\|^2_{\mathcal{H}_1} \leq 1, \|\Phi_2\|^2_{\mathcal{H}_2} \leq 1$, and $\|\Phi_3\|^2_{\mathcal{H}_3} \leq 1$. Hereinafter, we will omit the suffixes $\mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$ in the norms if there is no ambiguity.

Let $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ be a solution of Eq. (4.10). For any α, we have

$$\begin{aligned} \lambda_\alpha \|\hat{\alpha}\|^2 &\leq P_n \ell(y, \langle \hat{\alpha}, \Phi_1\rangle + \langle \hat{\beta}, \Phi_2\rangle\langle \hat{\gamma}, \Phi_3\rangle) + \lambda_\alpha \|\hat{\alpha}\|^2 + \lambda_\beta \|\hat{\beta}\|^2 + \lambda_\gamma \|\hat{\gamma}\|^2 \\ &\leq P_n \ell(y, \langle \alpha, \Phi_1\rangle) + \lambda_\alpha \|\alpha\|^2. \end{aligned} \quad (4.11)$$

The first inequality holds because $\ell(\cdot,\cdot)$ and $\|\cdot\|$ are non-negative. For the second inequality, we use the fact that the parameter set $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ is the minimizer of Eq. (4.10). Denoting $\hat{R}_s = \inf_\alpha\{P_n \ell(y, \langle \alpha, \Phi_1\rangle) + \lambda_\alpha \|\alpha\|^2\}$, we obtain $\|\hat{\alpha}\|^2 \leq \lambda_\alpha^{-1} \hat{R}_s$. Because the same inequality as Eq. (4.11) holds for $\lambda_\beta \|\hat{\beta}\|^2, \lambda_\gamma \|\hat{\gamma}\|^2$ and $P_n \ell(y, \hat{h})$, we have $\|\hat{\beta}\|^2 \leq \lambda_\beta^{-1} \hat{R}_s, \|\hat{\gamma}\|^2 \leq \lambda_\gamma^{-1} \hat{R}_s$, and $P_n \ell(y, \hat{h}) \leq \hat{R}_s$. Moreover, we obtain $P\ell(y, \hat{h}) = \mathbb{E}[P_n \ell(y, \hat{h})] \leq \mathbb{E}[\hat{R}_s]$. Therefore, it is sufficient to consider the following hypothesis class $\mathcal{H}$ and loss class $\mathcal{L}$:

$$\begin{aligned} \mathcal{H} = \big\{&x \mapsto \langle \alpha, \Phi_1\rangle + \langle \beta, \Phi_2\rangle\langle \gamma, \Phi_3\rangle \\ &\mid \|\alpha\|^2 \leq \lambda_\alpha^{-1} \hat{R}_s,\ \|\beta\|^2 \leq \lambda_\beta^{-1} \hat{R}_s,\ \|\gamma\|^2 \leq \lambda_\gamma^{-1} \hat{R}_s, P\ell(y,h) \leq \mathbb{E}[\hat{R}_s]\big\}, \\ \mathcal{L} = \big\{&(x,y) \mapsto \ell(y,h) \mid h \in \mathcal{H}\big\}. \end{aligned}$$

Here, we show the generalization bound of the proposed model class. The following theorem is based on Kuzborskij & Orabona (2017), showing that the difference between the generalization error and empirical error of this hypothesis class can be bounded using the magnitude of the relevance of the source and target domains.

Theorem 4.3 (Generalization bound)**.** *There exists a constant C depending only on $\lambda_\alpha, \lambda_\beta, \lambda_\gamma$, and L such that for any $\eta > 0$ and $h \in \mathcal{H}$, with probability at least $1 - e^{-\eta}$,*

$$P\ell(y, h) - P_n\ell(y, h) = \tilde{O}\left(\left(\sqrt{\frac{R_s}{n}} + \frac{\mu_\ell^2 C^2 + \sqrt{\eta}}{n}\right)\left(\sqrt{L}C + \sqrt{L\eta}\right) + \frac{C^2 L + L\eta}{n}\right),$$

where $R_s = \inf_\alpha\{P\ell(y, \langle \alpha, \Phi_1 \rangle) + \lambda_\alpha \|\alpha\|^2\}$.

The proof is given in Section 4.5.1.

Because Φ_1 is the feature map from the source feature space $\mathcal{F}_s$ onto the RKHS $\mathcal{H}_1$, R_s corresponds to the true risk of training in the target domain using only the source features f_s. If R_s is sufficiently small, e.g., $R_s = \tilde{O}(n^{-1/2})$, the convergence rate indicated by Theorem 4.3 becomes n^{-1}, which is an improvement over the naive convergence rate $n^{-1/2}$. This means that if training in the source domain yields feature representations strongly related to the target domain, the convergence of training in the target domain is accelerated. Theorem 4.3 measures this cross-domain relation using the metric R_s.

Theorem 4.3 is based on Theorem 11 of Kuzborskij & Orabona (2017) in which the function class $g_1 + g_3$ is considered. Our work differs in the following two aspects: the source features are modeled not only additively but also multiplicatively, i.e., we consider the function class $g_1 + g_2 \cdot g_3$ as well as the estimation of the parameters for the source feature combination, i.e., the parameters of the functions g_1 and g_2. In particular, the latter affects the resulting rate in Theorem 4.3. Without estimating the source combination parameters, the rate indicated by Theorem 4.3 improves only up to $n^{-3/4}$. The details are discussed in Section 4.5.1.

4.3.2 Excess risk bound

In this section, we analyze the excess risk, which is the difference between the risk of the estimated function and the smallest possible risk within the function class.

Recall that we consider the functions g_1, g_2, and g_3 to be the elements of the RKHSs $\mathcal{H}_1, \mathcal{H}_2$, and $\mathcal{H}_3$ with kernels k_1, k_2, and k_3, respectively. Define the kernel $k^{(1)} = k_1$, $k^{(2)} = k_2 \cdot k_3$, and $k = k^{(1)} + k^{(2)}$. Let $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}$, and $\mathcal{H}$ be the RKHSs with $k^{(1)}, k^{(2)}$, and k, respectively. For $m = 1, 2$, consider the normalized Gram matrix $K^{(m)} = \frac{1}{n}(k^{(m)}(x_i, x_j))_{i,j=1,\dots,n}$ and its eigenvalues $(\hat{\lambda}_i^{(m)})_{i=1}^n$, arranged in a non-increasing order.

We prepare the following additional assumptions:

Assumption 4.6. *There exists an $h^* \in \mathcal{H}$ satisfying $P(y - h^*(x))^2 = \inf_{h \in \mathcal{H}} P(y - h(x))^2$. Similarly, there exists an $h^{(m)*} \in \mathcal{H}^{(m)}$ satisfying $P(y - h^{(m)*}(x))^2 = \inf_{h \in \mathcal{H}^{(m)}} P(y - h(x))^2$ for $m = 1, 2$.*

Assumption 4.7. *For $m = 1, 2$, there exist positive real numbers $a_m > 0$ and $s_m \in (0, 1)$ such that $\hat{\lambda}_j^{(m)} \leq a_m j^{-1/s_m}$.*

Assumption 4.6 is used in Bartlett et al. (2005) and is not overly restrictive, as it holds for many regularization algorithms and convex, uniformly bounded function classes.

In the analysis of kernel methods, Assumption 4.7 is standard (Steinwart & Christmann, 2008), and is known to be equivalent to the classical covering or entropy number assumption (Steinwart et al., 2009). s_m measures the complexity of the RKHS, with larger values corresponding to more complex function spaces.

Under Assumption 4.6, we obtain the following excess risk bound for the proposed model class. The proof is based on Bartlett et al. (2005) and is presented in Section 4.5.2.

Theorem 4.4. *Let $\hat{h}$ be any element of $\mathcal{H}$ satisfying $P_n \ell(y, \hat{h}(x)) = \inf_{h \in \mathcal{H}} P_n \ell(y, h(x))$. Suppose that Assumption 4.6 is satisfied. Then, there exists a constant c depending only on μ_ℓ such that for any $\eta > 0$, with probability at least $1 - 5e^{-\eta}$,*

$$P(y - \hat{h}(x))^2 - P(y - h^*(x))^2 \leq c \left(\min_{0 \leq \kappa_1, \kappa_2 \leq n} \left\{ \frac{\kappa_1 + \kappa_2}{n} + \left(\frac{1}{n} \sum_{j=\kappa_1+1}^{n} \hat{\lambda}_j^{(1)} + \sum_{j=\kappa_2+1}^{n} \hat{\lambda}_j^{(2)} \right)^{\frac{1}{2}} \right\} + \frac{\eta}{n} \right).$$

Theorem 4.4 is a multiple-kernel version of Corollary 6.7 of Bartlett et al. (2005) and a data-dependent version of Theorem 2 of Kloft & Blanchard (2011), which considers the eigenvalues of the Hilbert-Schmidt operators on $\mathcal{H}$ and $\mathcal{H}^{(m)}$. Theorem 4.4 concerns the eigenvalues of the Gram matrices $K^{(m)}$ computed from the data.

The following corollary follows from Theorem 4.4 and Assumption 4.7.

Corollary 4.5. *Let $\hat{h}$ be any element of $\mathcal{H}$ satisfying $P_n(y - \hat{h}(x))^2 = \inf_{h \in \mathcal{H}} P_n(y - h(x))^2$. Suppose that Assumption 4.6 and 4.7 are satisfied. Then, for any $\eta > 0$ with probability at least $1 - 5e^{-\eta}$,*

$$P(y - \hat{h}(x))^2 - P(y - h^*(x))^2 = O\left(n^{-\frac{1}{1 + \max\{s_1, s_2\}}} \right).$$

Corollary 4.5 suggests that the convergence rate of the excess risk depends on the decay rates of the eigenvalues of two Gram matrices $K^{(1)}$ and $K^{(2)}$. $1/s_1$ is the decay rate of the eigenvalues of $K^{(1)} = \frac{1}{n}(k_1(f_s(x_i), f_s(x_j)))_{i,j=1,\ldots,n}$, representing the learning efficiency using only the source features. s_2 is the decay rate of the eigenvalues of the Hadamard product of the Gram matrices $K_2 = \frac{1}{n}(k_2(f_s(x_i), f_s(x_j))_{i,j=1,\ldots,n}$ and $K_3 = \frac{1}{n}(k_3(x_i, x_j))_{i,j=1,\ldots,n}$. The effect of combining the source features and the original

inputs appears here. In general, it is difficult to discuss the relationship between the spectra of two Gram matrices K_2, K_3 and their Hadamard product $K_2 \circ K_3$. Intuitively, the smaller the overlap between the space spanned by the source features f_s and by the original input x, the smaller the overlap between $\mathcal{H}_2$ and $\mathcal{H}_3$, because $\mathcal{H}_2$ is defined by the kernel $k_2(f_s(x), f_s(x'))$ and $\mathcal{H}_3$ is defined by the kernel $k_3(x, x')$. In other words, with increasing difference between the information contained in the source features f_s and the original input x, the tensor product $\mathcal{H}_2 \otimes \mathcal{H}_3$ will become more complex, and the inverse of the decay rate s_2 is expected to be larger.

We experimentally investigated how the inverse decay rate s_2 in Corollary 4.5 is related to the degree of overlap in the spaces spanned by the original input x and source features f_s.

For the original input $x \in \mathbb{R}^{100}$, we randomly constructed a set of 10 orthonormal bases, and then generated 100 samples from their spanning space. For the source features $f_s \in \mathbb{R}^{100}$, we selected d bases randomly from the 10 orthonormal bases selected for x and the remaining $10 - d$ bases from their orthogonal complement space. We then generated 100 samples of f_s from the space spanned by these 10 bases. The overlap number d can be regarded as the degree of overlap between two spaces spanned by the samples of x and f_s. We generated 100 different sample sets of x and f_s.

We calculated the Hadamard product of the Gram matrices K_2 and K_3 using the samples of x and f_s, respectively. For the computation of K_2 and K_3, all combinations of the following five kernels were tested:

Linear kernel $k(x, x') = \dfrac{x^\top x}{2\gamma^2} + 1,$

Matérn kernel $k(x, x') = \dfrac{2^{1-\nu}}{\Gamma(\nu)} \left(\dfrac{\sqrt{2\nu}\|x - x'\|_2}{\gamma} \right)^\nu K_\nu \left(\dfrac{\sqrt{2\nu}\|x - x'\|_2}{\gamma} \right)$ for $\nu = \dfrac{1}{2}, \dfrac{3}{2}, \dfrac{5}{2}, \infty,$

where $K_\nu(\cdot)$ is a modified Bessel function and $\Gamma(\cdot)$ is a gamma function. Note that for $\nu = \infty$, the Matérn kernel is equivalent to the Gaussian radial basis function (RBF) kernel. The scale parameter γ of both kernels was set to $\gamma = \sqrt{\dim(x)} = \sqrt{10}$. For a given matrix K, the decay rate of the eigenvalues was estimated as the smallest value of s that satisfies $\lambda_i \leq \|K\|_F^2 \cdot i^{-\frac{1}{s}}$, where $\|\cdot\|_F$ denotes the Frobenius norm. Note that this inequality holds for any matrix K with $s = 1$ (Vershynin, 2018).

Figure 4.2 shows the change in the inverse decay rates with variation in d for various combinations of the kernels. In all cases, the decay rate of $K_2 \circ K_3$ showed a clear trend of monotonically decreasing as the degree of overlap d increased. In other words, the greater the overlap between the spaces spanned by x and f_s, the smaller the decay rate and the complexity of the RKHS $\mathcal{H}_2 \otimes \mathcal{H}_3$.

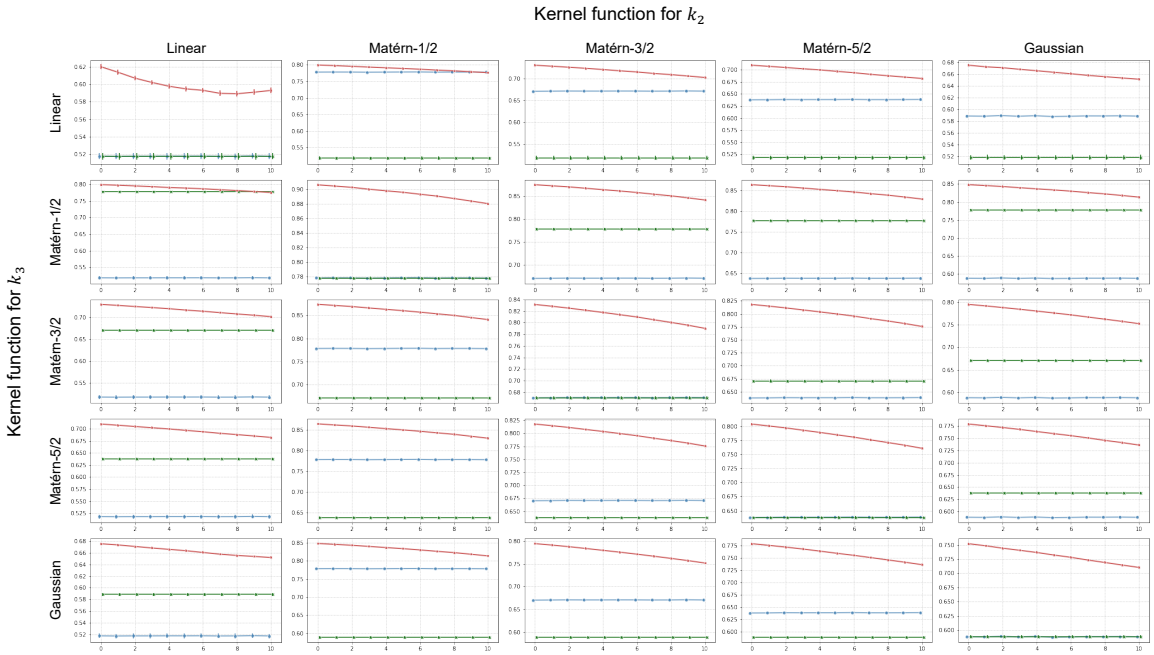


Figure 4.2: Decay rates of eigenvalues of K_2 (blue lines), K_3 (green lines), and $K_2 \circ K_3$ (red lines) for all combinations of five different kernels. The vertical axis represents the decay rate, and the horizontal axis represents the overlap dimension d in the space where x and f_s are distributed.

4.4 Experimental results

We demonstrate the potential of the affine model transfer through three different case studies: (i) prediction of feed-forward torque at seven joints of the SARCOS anthropomorphic robot arm (Williams & Rasmussen, 2006), (ii) prediction of lattice thermal conductivity of inorganic crystalline materials (Yamada et al., 2019), (iii) TL for bridging the gap between experimental and theoretical values of specific heat capacity for organic polymers (Hayashi et al., 2022). The Python code is available at https://github.com/mshunya/AffineTL.

4.4.1 Kinematics of robot arms

We experimentally investigated the learning performance of the affine model transfer and compared it with those of several naive methods. The objective was to predict the feed-forward torques required to follow the desired trajectory at seven different joints of the SARCOS anthropomorphic robot arm (Williams & Rasmussen, 2006). Twenty-one features representing the joint position, velocity, and acceleration were used as the input variable $x \in \mathbb{R}^d (d = 21)$. The target task was to predict the torque value at one joint, and the source task was defined as the prediction of torque at the other six joints. The experiments were conducted with seven different tasks (denoted as Torques 1–7) corresponding to the seven different joints. We first trained neural networks to predict the six source torques. The source knowledge obtained was reused to build a prediction model for the target torque. The dataset included 44,484 training samples and 4,449 test samples. We selected $\{5, 10, 15, 20, 30, 40, 50\}$ samples randomly from the training set. The prediction performances of the trained models were evaluated using the 4,449 test samples. Experiments were repeated 20 times with independently sampled different datasets. The experiment was designed for TL with a fairly small sample size.

Model definition and hyperparameter search

Source model For each target task, a multi-task neural network was trained to predict the torque values of the remaining six source tasks. Figure 4.3 shows the details of the network structure. The source model shares four layers up to the final layer, and only the output layer is task-specific. We denote the output of the trained source neural network as f_s and the output from the shared layer preceding the output layer as f_{ext}.

Target model For comparison, the following ten procedures were tested, including two existing HTLs (Kuzborskij & Orabona, 2013; Du et al., 2017) and three standard

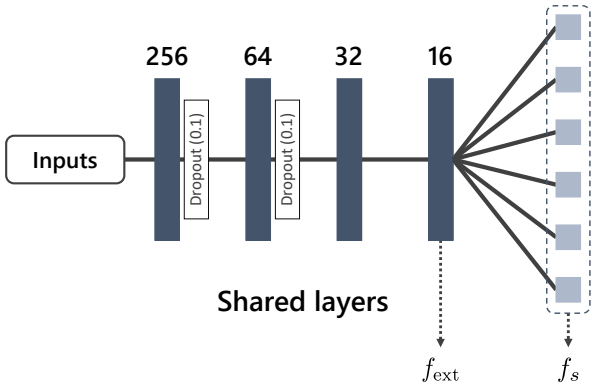


Figure 4.3: Architecture of source models. We used a multi-task neural network with four shared layers with widths of 256-64-32-16, and one task-specific layer. For TL, we used the 6-dimensional output as the source features f_s and the final shared layer with 16 dimensions as f_{ext}.

neural network-based TLs (Yosinski et al., 2014; Finn et al., 2017):

No transfer
: Train a model with input x with no transfer.

Only source
: Train a model $g_1(f_s)$ using only the source features f_s as input.

Input augmentation
: Perform an ordinary regression with the augmented input vector concatenating x and f_s.

HTL-offset (Kuzborskij & Orabona, 2013)
: Calculate the transformed output $z_i = y_i - g_1(f_s)$, where $g_1(f_s)$ is the model pre-trained in **Only source**, and train an additional model with input x_i to predict z_i.

HTL-scale (Du et al., 2017)
: Calculate the transformed output $z_i = y_i / g_1(f_s)$, where $g_1(f_s)$ is the model pre-trained in **Only source**, and train an additional model with input x_i to predict z_i.

Affine transfer (full)
: Train the model $g_1(f_s) + g_2(f_s) \cdot g_3(x)$.

Affine transfer (constrained)
: Train the model $g_1(f_s) + g_3(x)$.

Feature extraction
: Train a model using the extracted feature f_{ext} as input.

Algorithm 4.2 Block relaxation algorithm for **Affine transfer (full)**.

Initialize

$a_0 \leftarrow (K_1 + \lambda_1 I_n)^{-1} y, \quad b_0 \sim \mathcal{N}(\mathbf{0}, I_n), \quad c_0 \sim \mathcal{N}(\mathbf{0}, I_n), \quad d_0 \leftarrow 0.5$

repeat

$a \leftarrow (K_1 + \lambda_1 I_n)^{-1}(y - (K_2 b + \mathbf{1}) \circ (K_3 c) - d)$

$b \leftarrow (\mathrm{Diag}(K_3 c)^2 K_2 + \lambda_2 I_n)^{-1}((K_3 c) \circ (y - K_1 a - K_3 c - d))$

$c \leftarrow (\mathrm{Diag}(K_2 b + \mathbf{1})^2 K_3 + \lambda_3 I_n)^{-1}((K_2 b + \mathbf{1}) \circ (y - K_1 a - d))$

$d \leftarrow \langle y - K_1 a - (K_2 b + \mathbf{1}) \circ (K_3 c), \mathbf{1} \rangle / n$

until convergence

Fine-tuning
: Tuning the weights of the shared layers in the source model by using the target samples.

MAML (Finn et al., 2017)
: Initializing the target neural network using the model-agnostic meta-learning algorithm (MAML), and updating its weights using the target samples.

Note that the first seven procedures use the output vector f_s of the source model as the source features, whereas **Feature extraction** uses the output from the intermediate layer of the source model. In addition, **Fine-tuning** and **MAML** differ from the other methods in that they reuse the parameters (not features) from the source model.

We used kernel ridge regression with the RBF kernel $\exp(-\|x - x'\|^2 / 2\ell^2)$ to train the model for each procedure. For **No transfer**, **Only source**, **Augmented**, **HTL-offset**, **HTL-scale**, and **Feature extraction**, the scale parameter ℓ was set to the square root of the input dimension: $\ell = \sqrt{21}$ for **No transfer**, **HTL-offset**, and **HTL-scale**, $\ell = \sqrt{6}$ for **Only source**, $\ell = \sqrt{27}$ for **With source**, and $\ell = \sqrt{16}$ for **Feature extraction**. For **Affine transfer (full)** and **Affine transfer (constrained)**, we considered the following kernels:

$$k_1(f_s(x), f_s(x')) = \exp\Big(-\frac{1}{2\ell^2}\|f_s(x) - f_s(x')\|_2^2\Big) \ (\ell = \sqrt{6}),$$

$$k_2(f_s(x), f_s(x')) = \exp\Big(-\frac{1}{2\ell^2}\|f_s(x) - f_s(x')\|_2^2\Big) \ (\ell = \sqrt{6}),$$

$$k_3(x, x') = \exp\Big(-\frac{1}{2\ell^2}\|x - x'\|_2^2\Big) \ (\ell = \sqrt{27})$$

for g_1, g_2, and g_3 in the affine transfer model, respectively.

For **No transfer**, **Only source**, **Augmented**, **HTL–offset**, **HTL–scale**, and **Feature extraction**, the regularization parameter λ was selected in five-fold cross-validation in which grid search was performed over 50 grid points in the interval

$[10^{-4}, 10^{2}]$. For **Affine transfer (full)** and **Affine transfer (constrained)**, the hyperparameters to be optimized were the three regularization parameters λ_1, λ_2, and λ_3. We performed five-fold cross-validation to identify the best hyperparameter set from the candidate points; $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ for λ_1 and $\{10^{-2}, 10^{-1}, 1, 10\}$ for λ_2 and λ_3.

To learn **Affine transfer (full)** and **Affine transfer (constrained)**, we used the following objective functions:

Affine transfer (full)

$$\|y - (K_1 a + (K_2 b + \mathbf{1}) \circ (K_3 c) + d)\|_2^2 + \lambda_1 a^\top K_1 a + \lambda_2 b^\top K_2 b + \lambda_3 c^\top K_3 c,$$

Affine transfer (constrained)

$$\frac{1}{n}\|y - (K_1 a + K_3 c + d)\|_2^2 + \lambda_1 a^\top K_1 a + \lambda_3 c^\top K_3 c.$$

Algorithm 4.2 summarizes the block relaxation algorithm for **Affine transfer (full)**. For **Affine transfer (constrained)**, we found the optimal parameters as follows:

$$\begin{bmatrix} \hat{a} \\ \hat{c} \\ \hat{d} \end{bmatrix} = \left(\begin{bmatrix} K_1 \\ K_3 \\ \mathbf{1}^\top \end{bmatrix} \begin{bmatrix} K_1 & K_3 & \mathbf{1} \end{bmatrix} + \begin{bmatrix} \lambda_1 K_1 & & \\ & \lambda_3 K_3 & \\ & & 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} K_1 \\ K_3 \\ \mathbf{1}^\top \end{bmatrix} y$$

The stopping criterion of the algorithm was set as

$$\max_{\theta \in \{a,b,c\}} \frac{\max_i \left| \theta_i^{(\mathrm{new})} - \theta_i^{(\mathrm{old})} \right|}{\max_i \left| \theta_i^{(\mathrm{old})} \right|} < 10^{-4}, \tag{4.12}$$

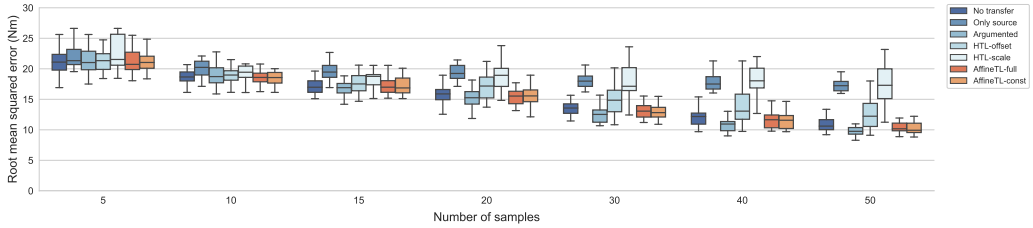
where θ_i denotes the i-th element of the parameter θ. This convergence criterion is employed in several existing machine learning libraries, e.g., scikit-learn (Pedregosa et al., 2011)[1].

As a reference, we also performed the TL method using the source parameters.

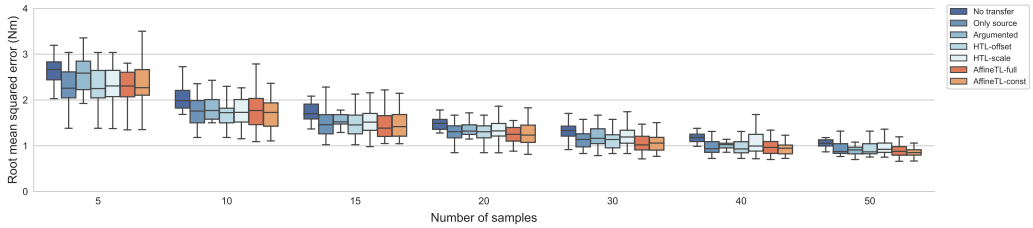
In **Fine-tuning**, the target network was constructed by adding a one-dimensional output layer to the shared layers of the source network. As initial values for the training, we used the weights of the source neural network for the shared layer and the average of the multidimensional output layer of the source network for the output layer. Adam (Kingma & Ba, 2015) was used for the optimization. The learning rate was fixed at 0.01 and the number of training epochs was selected from $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100\}$ through five-fold cross-validation.

MAML (Finn et al., 2017) is one of the commonly used algorithms for meta-learning (Li et al., 2017; Hospedales et al., 2020) or learning-to-learn (Thrun & Pratt,

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html



(a) Target domain: Torque 1



(b) Target domain: Torque 7

Figure 4.4: Box plots showing the distribution of RMSE for the seven analysis procedures at (a) Torque 1 and (b) Torque 7. Each boxplot shows the median and the first and third quartiles.

2012), which explores the initial values through training on the source domains so that the subsequent training would be accelerated. In this experiment, a fully connected neural network with 256-64-32-16-1 layer width was built, and the initial values were searched through MAML using the six source tasks. The obtained base model was tuned with the target samples. As in **Fine-tuining**, Adam with a fixed learning rate of 0.01 was used for the optimization. The number of training epochs was selected from $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100\}$ through five-fold cross-validation.

For more details on the experimental conditions and procedure, refer to the Python code that we provided.

Results

Table 4.1 summarizes the prediction performance of the ten different procedures for various numbers of training samples in the seven tasks. In most cases, the affine transfer model achieved the best prediction performance in terms of the root mean squared error (RMSE). In several other cases, direct learning without transfer produced the best results; in almost all cases, ordinary TL using only the source features and the two existing HTL models showed no advantage over the affine transfer model. In this experiment, the performance was evaluated for the cases where the number of

training samples was extremely small. The affine transfer model tended to be more advantageous in these cases, such as with a sample size of $n = 5$ or 10.

Figure 4.4 highlights the RMSE values for Torque 1 and Torque 7. The joint of Torque 1 is located closest to the root of the arm. Therefore, the learning task for Torque 1 is less relevant to those for the other joints, and the transfer from Torques 2–6 to Torque 1 would not be effective. In fact, as shown in Figure 4.4a and Table 4.1, relying only on the source features (**Only source**) failed to acquire predictive ability. In addition, **HTL-offset** and **HTL-scale** likewise showed poor prediction performance owing to the negative effect of failure in the variable transformation using the values of the other torques. In particular, the two HTL models achieved lower predictive performance than direct learning (**No transfer**), resulting in the occurrence of negative transfer.

Torque 7 was measured at the joint closest to the end of the arm. Therefore, Torque 7 strongly depended on the torques at the other six joint positions, and the procedures based on the source features, including **Only source**, were more effective than in the other tasks. In particular, the affine model transfer achieved the best performance among the other methods. This is consistent with the theoretical result that the transfer capability of the affine model transfer can be further improved when the risk of learning using only the source features is sufficiently small.

As mentioned above, the standard neural TL procedures **Feature extraction**, **Fine-tuning**, and **MAML** were also conducted for reference. It may be noted that these procedures differ from the affine model transfer in that they use the intermediate layers or parameters of the source neural network. Nevertheless, the affine transfer model showed the best performance for the four settings (Torques 1, 2, 5, 6). However, for Torques 3, 4, and 7, **Fine-tuning** and **MAML** showed the best performance. Even in this case, the affine transfer model showed comparable performance, especially for very small sample settings ($n = 5$ or 10). Considering that the affine model transfer uses only the output of the source model as the source knowledge and does not have access to the parameters and internal representations, the above results indicate the immense potential of the affine model transfer.

Table 4.1: Performance on predicting the torque values at seven different joints of the SARCOS anthropomorphic robot arm. The mean and standard deviation of the root mean square error with respect to 20 test sets are reported for varying numbers of training samples and the seven different tasks. Ten different methods were tested: **No transfer** (Direct), **Only source** (Only), **Input augmentation** (Augmented), **HTL-offset** (Offset), **HTL-scale** (Scale), **Affine transfer (full)** (AffineTL-full), **Affine transfer (constrained)** (AffineTL-const), **Feature extraction** (FE), **Fine-tuning** (FT) and **MAML**. Bold means the best score among **No transfer** and the six TL methods based on f_s; wavy line means the best score among all ten methods.

Target	Model	Number of training samples						
		$n < d$			$n \approx d$	$n > d$		
		5	10	15	20	30	40	50
Torque 1	Direct	**21.2 ± 2.05**	19.0 ± 2.04	17.3 ± 1.66	15.8 ± 1.57	13.7 ± 1.42	12.2 ± 1.56	10.8 ± 1.11
	Only	24.9 ± 11.4	20.3 ± 1.77	19.5 ± 1.43	19.3 ± 1.35	18.3 ± 1.92	18.0 ± 1.76	17.5 ± 1.63
	Augmented	21.3 ± 2.35	18.9 ± 1.71	**16.8 ± 1.27**	**15.3 ± 1.97**	**12.7 ± 1.56**	**11.0 ± 1.58**	**9.93 ± 1.08**
	Offset	24.6 ± 11.5	19.1 ± 2.02	17.7 ± 1.58	17.1 ± 2.15	15.2 ± 3.37	14.3 ± 3.82	12.8 ± 2.97
	Scale	24.9 ± 8.53	20.3 ± 3.32	20.0 ± 7.54	19.1 ± 3.19	17.8 ± 2.83	18.5 ± 3.25	18.1 ± 4.44
	AffineTL-Full	21.3 ± 2.10	18.9 ± 2.09	17.4 ± 1.93	15.6 ± 1.68	13.4 ± 1.95	11.6 ± 1.48	10.4 ± 0.925
	AffineTL-Const	21.4 ± 1.91	**18.7 ± 1.91**	17.2 ± 1.38	15.7 ± 1.85	13.0 ± 1.43	11.5 ± 1.47	10.3 ± 0.994
	FE	25.2 ± 11.6	21.3 ± 3.85	20.3 ± 1.89	20.1 ± 2.38	18.8 ± 1.66	18.0 ± 1.60	17.7 ± 1.90
	FT	24.5 ± 5.35	21.0 ± 2.81	19.6 ± 1.37	19.3 ± 2.30	16.1 ± 1.87	14.4 ± 1.42	13.1 ± 1.10
	MAML	28.6 ± 8.85	22.4 ± 3.17	20.8 ± 2.09	20.4 ± 3.11	16.7 ± 2.97	14.4 ± 1.83	13.6 ± 1.10
Torque 2	Direct	15.8 ± 2.38	13.0 ± 1.42	11.5 ± 0.966	10.4 ± 0.821	9.39 ± 0.978	8.38 ± 0.767	7.72 ± 0.753
	Only	15.3 ± 2.31	13.0 ± 1.51	12.4 ± 2.22	11.9 ± 3.02	12.1 ± 7.43	10.2 ± 1.82	9.45 ± 1.99
	Augmented	15.7 ± 2.48	12.7 ± 1.47	**11.1 ± 1.33**	**9.96 ± 1.41**	**8.65 ± 1.05**	**7.65 ± 0.929**	**6.99 ± 0.615**
	Offset	15.2 ± 2.28	12.8 ± 1.62	12 ± 2.45	11.8 ± 3.11	11.9 ± 7.51	9.89 ± 1.87	9.12 ± 2.08
	Scale	15.2 ± 2.29	**12.6 ± 1.59**	12.1 ± 2.32	11.7 ± 3.12	11.9 ± 7.52	9.95 ± 1.86	9.15 ± 2.05
	AffineTL-Full	**14.4 ± 1.60**	12.7 ± 1.82	11.5 ± 1.93	10.8 ± 1.68	9.58 ± 1.97	7.96 ± 1.04	7.52 ± 0.674
	AffineTL-Const	14.6 ± 1.86	12.8 ± 1.45	11.4 ± 1.48	10.5 ± 1.64	9.39 ± 1.79	8.17 ± 1.06	7.58 ± 0.87
	FE	15.6 ± 2.09	15.1 ± 7.63	12.5 ± 1.90	11.4 ± 1.16	11.1 ± 1.75	10.1 ± 1.19	9.45 ± 1.46
	FT	26.2 ± 5.24	20.3 ± 3.05	17.3 ± 2.92	15.5 ± 2.95	12.6 ± 2.26	10.9 ± 1.52	9.36 ± 1.19
	MAML	22.2 ± 7.22	14.8 ± 4.51	13.0 ± 2.51	11.5 ± 2.19	9.86 ± 1.27	8.90 ± 1.12	7.89 ± 0.729
Torque 3	Direct	9.93 ± 1.65	8.17 ± 0.996	7.84 ± 2.60	6.97 ± 1.10	5.97 ± 0.917	5.33 ± 0.942	4.56 ± 0.401
	Only	8.99 ± 2.98	7.62 ± 2.35	6.91 ± 1.65	6.45 ± 1.20	5.66 ± 0.908	5.31 ± 0.968	4.95 ± 0.964
	Augmented	9.66 ± 1.72	7.78 ± 0.978	6.74 ± 1.01	6.25 ± 1.15	5.29 ± 1.27	**4.68 ± 1.24**	**4.03 ± 0.652**
	Offset	8.96 ± 2.98	7.48 ± 2.31	6.87 ± 1.65	6.42 ± 1.21	5.60 ± 0.844	5.23 ± 0.993	4.85 ± 1.01
	Scale	9.06 ± 2.94	7.59 ± 2.29	6.91 ± 1.42	6.69 ± 1.41	5.65 ± 0.964	5.41 ± 1.22	4.98 ± 0.888
	AffineTL-Full	**8.64 ± 1.33**	**7.22 ± 1.41**	6.67 ± 1.27	6.07 ± 1.00	5.25 ± 1.17	4.89 ± 1.15	4.28 ± 0.829
	AffineTL-Const	8.94 ± 1.40	7.33 ± 1.20	**6.50 ± 1.14**	**5.97 ± 0.918**	**5.18 ± 1.00**	4.76 ± 0.958	4.22 ± 0.745
	FE	8.67 ± 1.36	8.10 ± 2.20	6.69 ± 1.06	6.49 ± 0.804	5.70 ± 0.966	5.31 ± 0.836	5.08 ± 0.797
	FT	9.08 ± 2.08	7.50 ± 1.13	6.72 ± 1.04	5.91 ± 0.893	5.06 ± 0.610	4.65 ± 0.513	4.27 ± 0.364
	MAML	9.51 ± 4.96	7.10 ± 0.976	6.45 ± 1.05	5.91 ± 0.794	5.16 ± 0.504	4.87 ± 0.533	4.79 ± 0.525
Torque 4	Direct	14.2 ± 2.30	11.1 ± 2.39	9.49 ± 2.18	7.80 ± 0.978	6.91 ± 0.778	6.06 ± 0.630	5.47 ± 0.653
	Only	12.3 ± 3.60	9.23 ± 2.43	7.81 ± 1.74	6.83 ± 1.45	6.21 ± 1.02	6.19 ± 1.37	5.22 ± 0.629
	Augmented	13.8 ± 2.83	9.69 ± 1.64	8.52 ± 1.80	7.06 ± 1.03	5.97 ± 0.905	**5.16 ± 0.740**	**4.69 ± 0.698**
	Offset	12.3 ± 3.62	9.08 ± 2.35	**7.67 ± 1.68**	6.73 ± 1.40	6.14 ± 1.01	6.16 ± 1.38	5.12 ± 0.582
	Scale	12.3 ± 3.62	9.10 ± 2.36	7.72 ± 1.62	6.74 ± 1.37	6.12 ± 1.04	6.16 ± 1.38	5.14 ± 0.524
	AffineTL-Full	**12.0 ± 3.11**	8.95 ± 2.05	7.89 ± 1.92	6.83 ± 1.75	**5.66 ± 1.07**	5.27 ± 1.12	4.87 ± 1.02
	AffineTL-Const	12.2 ± 3.40	**8.64 ± 1.95**	7.87 ± 1.87	**6.44 ± 1.09**	5.67 ± 1.12	5.25 ± 1.06	4.78 ± 0.665
	FE	13.2 ± 5.01	9.40 ± 3.15	8.05 ± 2.10	6.69 ± 1.36	5.85 ± 1.10	5.71 ± 1.08	5.53 ± 1.11
	FT	12.1 ± 3.71	8.71 ± 1.69	6.86 ± 1.24	5.90 ± 1.06	5.02 ± 0.736	4.43 ± 0.518	4.04 ± 0.392
	MAML	14.0 ± 7.08	10.8 ± 3.48	9.57 ± 1.98	9.37 ± 2.37	7.99 ± 2.35	6.68 ± 1.24	6.08 ± 1.33

Target	Model	Number of training samples						
		$n < d$			$n \approx d$	$n > d$		
		5	10	15	20	30	40	50
Torque 5	Direct	1.08 ± 0.169	0.986 ± 0.0901	0.932 ± 0.165	0.860 ± 0.127	0.737 ± 0.123	0.686 ± 0.0937	**0.608 ± 0.0705**
	Only	1.11 ± 0.155	1.01 ± 0.0894	1.02 ± 0.146	0.964 ± 0.148	0.846 ± 0.125	0.797 ± 0.111	0.739 ± 0.103
	Augmented	1.08 ± 0.160	0.985 ± 0.0898	0.895 ± 0.125	**0.849 ± 0.135**	**0.737 ± 0.129**	**0.679 ± 0.102**	0.623 ± 0.110
	Offset	1.11 ± 0.173	0.998 ± 0.0949	0.982 ± 0.163	0.944 ± 0.152	0.806 ± 0.113	0.738 ± 0.11	0.693 ± 0.0987
	Scale	1.15 ± 0.248	0.993 ± 0.0933	0.970 ± 0.151	0.939 ± 0.124	0.806 ± 0.0966	0.754 ± 0.0842	0.776 ± 0.211
	AffineTL-Full	**1.03 ± 0.121**	**0.935 ± 0.126**	**0.878 ± 0.129**	0.862 ± 0.129	0.762 ± 0.144	0.726 ± 0.117	0.635 ± 0.068
	AffineTL-Const	1.04 ± 0.114	0.971 ± 0.0999	0.897 ± 0.113	0.888 ± 0.129	0.739 ± 0.122	0.702 ± 0.0920	0.629 ± 0.0672
	FE	1.08 ± 0.0931	1.01 ± 0.0819	0.990 ± 0.0782	0.946 ± 0.123	0.857 ± 0.120	0.825 ± 0.0977	0.767 ± 0.0924
	FT	1.22 ± 0.342	1.08 ± 0.0945	1.03 ± 0.0698	0.976 ± 0.118	0.858 ± 0.113	0.765 ± 0.119	0.714 ± 0.139
	MAML	1.45 ± 0.478	1.18 ± 0.184	1.07 ± 0.207	0.999 ± 0.194	0.801 ± 0.193	0.703 ± 0.125	0.612 ± 0.0615
Torque 6	Direct	1.86 ± 0.246	1.67 ± 0.194	1.50 ± 0.167	1.36 ± 0.156	1.21 ± 0.143	1.11 ± 0.088	1.07 ± 0.0969
	Only	1.95 ± 0.25	1.88 ± 0.407	1.79 ± 0.206	1.80 ± 0.378	1.61 ± 0.216	1.58 ± 0.173	1.55 ± 0.200
	Augmented	1.84 ± 0.171	1.65 ± 0.200	**1.48 ± 0.183**	**1.33 ± 0.207**	**1.17 ± 0.200**	**1.03 ± 0.117**	**0.964 ± 0.115**
	Offset	1.92 ± 0.257	1.84 ± 0.426	1.72 ± 0.262	1.71 ± 0.421	1.44 ± 0.271	1.39 ± 0.245	1.39 ± 0.289
	Scale	1.91 ± 0.256	1.89 ± 0.425	1.81 ± 0.326	1.84 ± 0.398	1.68 ± 0.300	1.59 ± 0.248	1.59 ± 0.242
	AffineTL-Full	**1.82 ± 0.229**	**1.64 ± 0.191**	1.58 ± 0.224	1.41 ± 0.248	1.24 ± 0.212	1.13 ± 0.307	0.996 ± 0.0963
	AffineTL-Const	1.86 ± 0.202	1.70 ± 0.179	1.55 ± 0.275	1.45 ± 0.276	1.23 ± 0.209	1.09 ± 0.141	1.02 ± 0.0923
	FE	1.88 ± 0.231	1.70 ± 0.138	1.81 ± 0.479	1.64 ± 0.215	1.61 ± 0.261	1.48 ± 0.149	1.45 ± 0.193
	FT	2.48 ± 0.446	2.14 ± 0.315	2.08 ± 0.359	1.80 ± 0.280	1.47 ± 0.299	1.29 ± 0.185	1.17 ± 0.125
	MAML	2.69 ± 0.675	2.20 ± 0.529	1.96 ± 0.524	1.69 ± 0.371	1.42 ± 0.373	1.22 ± 0.114	1.15 ± 0.0839
Torque 7	Direct	2.67 ± 0.321	2.12 ± 0.42	1.84 ± 0.421	1.53 ± 0.305	1.34 ± 0.203	1.17 ± 0.126	1.05 ± 0.096
	Only	2.29 ± 0.583	1.76 ± 0.441	1.55 ± 0.407	1.42 ± 0.585	1.16 ± 0.243	0.999 ± 0.231	0.942 ± 0.164
	Augmented	2.55 ± 0.408	1.90 ± 0.433	1.68 ± 0.417	1.39 ± 0.366	1.20 ± 0.236	1.01 ± 0.142	0.901 ± 0.112
	Offset	2.29 ± 0.588	1.71 ± 0.405	1.55 ± 0.408	1.41 ± 0.586	1.15 ± 0.249	0.995 ± 0.233	0.935 ± 0.167
	Scale	2.32 ± 0.580	1.75 ± 0.428	1.59 ± 0.395	1.42 ± 0.569	1.21 ± 0.249	1.06 ± 0.249	0.967 ± 0.161
	AffineTL-Full	**2.29 ± 0.533**	1.75 ± 0.447	**1.49 ± 0.380**	1.30 ± 0.327	1.07 ± 0.250	0.975 ± 0.180	0.889 ± 0.145
	AffineTL-Const	2.32 ± 0.552	**1.71 ± 0.419**	1.49 ± 0.373	**1.26 ± 0.257**	**1.06 ± 0.220**	**0.950 ± 0.163**	**0.885 ± 0.156**
	FE	2.30 ± 0.471	1.73 ± 0.392	1.57 ± 0.445	1.31 ± 0.236	1.35 ± 0.507	1.10 ± 0.180	1.06 ± 0.166
	FT	2.34 ± 0.740	1.62 ± 0.380	1.32 ± 0.301	1.08 ± 0.198	0.934 ± 0.108	0.816 ± 0.0718	0.777 ± 0.0574
	MAML	2.60 ± 1.41	1.89 ± 0.459	1.66 ± 0.311	1.59 ± 0.311	1.28 ± 0.288	1.19 ± 0.210	1.07 ± 0.105

4.4.2 Lattice thermal conductivity of inorganic crystals

Here, we describe the relationship between the qualitative differences in the source features and learning behavior of the affine model transfer, in comparison with those of ordinary feature extractors using neural networks. The target task was to predict the lattice thermal conductivity (LTC) of inorganic crystalline materials, where the LTC is the amount of vibrational energy propagated by phonons in a crystal. In general, LTC can be calculated ab initio by performing many-body electronic structure calculations based on quantum mechanics. However, it is very time-consuming to perform the first-principles calculations for thousands of crystals, which will be used as a training sample set for a surrogate statistical model. Therefore, we performed TL for the source task of predicting an alternative, computationally tractable physical property called scattering phase space (SPS), which is known to be physically related to the LTC.

Data

We used the dataset from Ju et al. (2021), which contains SPS and LTC data for 320 and 45 inorganic compounds, respectively. The input compounds were translated to 290-dimensional compositional descriptors using XenonPy[2] (Liu et al., 2021).

Model definition and hyperparameter search

Fully connected neural networks with a LeakyReLu activation function with $\alpha = 0.01$ were used for both the source and target models. The model training was conducted using the Adam optimizer. Hyperparameters such as the width of the hidden layer, learning rate, number of epochs, and regularization parameters were adjusted using five-fold cross-validation. For more details on the experimental conditions and procedure, refer to the Python code that we provided.

Source model In the preliminary step, neural networks with three hidden layers for predicting the SPS were trained using 80% of the 320 samples. The hidden layer width L was randomly selected from the range $[50, 100]$, and we trained a neural network with a structure of (input)-L-L-L-1. 100 models with different numbers of neurons were randomly generated and the top 10 source models that showed the highest generalization performance in the source domain were selected. Then, in the target task, an intermediate layer of a source model was used as the feature extractor.

[2] https://github.com/yoshida-lab/XenonPy

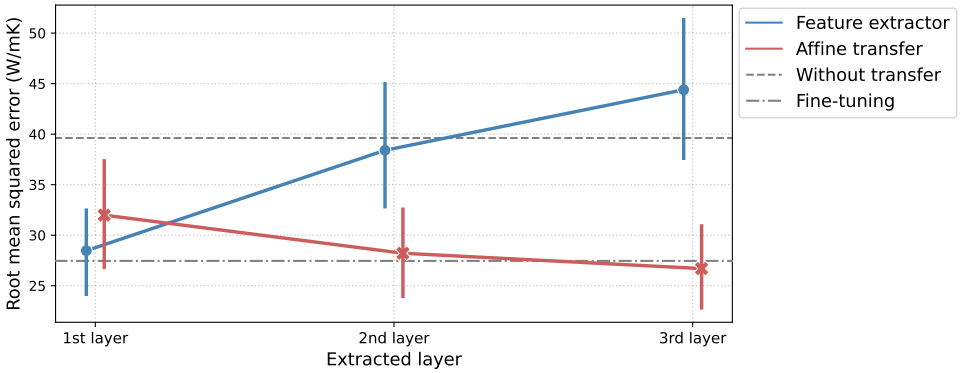


Figure 4.5: Change in RMSEs between the affine transfer model and ordinary feature extraction when using different levels of intermediate layers as the source features. The line plot shows the mean and 95% confidence interval. As the baselines, the RMSEs for direct learning without transfer and fine-tuned neural networks are depicted as dotted and dashed lines, respectively.

Target model The functions g_1, g_2, and g_3 in the affine transfer model were modeled using neural networks. We used a neural network with one hidden layer for g_1, g_2, and g_3. A model was trained using 40 randomly selected samples of the LTC, and its performance was evaluated with the remaining 5 samples. For each of the 10 source models, we performed the training and testing 10 times with different sample partitions and compared the mean values of the RMSE among four different methods: (i) the affine model transfer using neural networks to model the three functions g_1, g_2 and g_3, (ii) a neural network that uses the XenonPy compositional descriptors as input without transfer, (iii) a neural network that uses the source features as input, and (iv) fine-tuning of the pre-trained neural networks. The width of the layers of each neural network, number of training epochs, and dropout rate were optimized during five-fold cross-validation looped within each training set.

Results

Figure 4.5 shows the change in prediction performance of the TL models using the source features obtained from different intermediate layers selected from the first to the third layers. The affine transfer model and ordinary feature extraction showed opposite patterns. The performance of feature extraction improved when the first intermediate layer closest to the input layer was used as the source feature and it gradually degraded when the layers closer to the output were used. When the third intermediate layer was used, a negative transfer occurred in feature extraction, as its performance became poorer than that of direct learning. In contrast, the affine

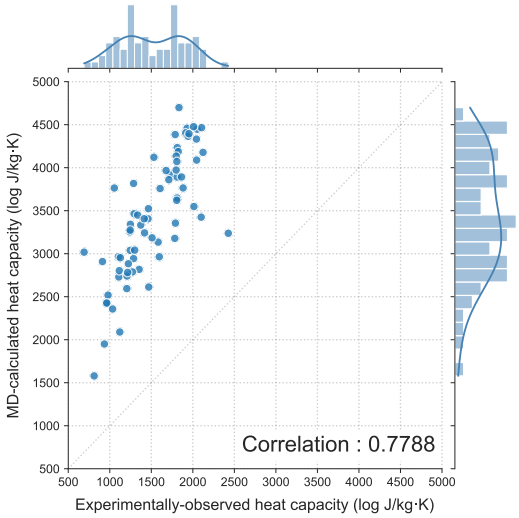


Figure 4.6: MD-calculated (vertical axis) and experimental values (horizontal axis) of the specific heat capacity at a constant pressure for various amorphous polymers.

transfer model performed better when the second and third intermediate layers closer to the output were used. The affine transfer model using the third intermediate layer reached a level of accuracy slightly better than that of fine-tuning, which intuitively uses more information to transfer than the extracted features.

In general, the features encoded in an intermediate layer of a neural network are increasingly task-independent for layers closer to the input, and the features become more task-specific for layers closer to the output (Yosinski et al., 2014). Because the first layer does not differ much from the original input, using both x and f_s in the affine model transfer does not contribute much to performance improvement. However, when using each of the second and third layers as the feature extractor, the use of both x and f_s contributes to improving the expressive power of the model, because the feature extractor acquires different representational capabilities from the original input. In contrast, a model based only on f_s from a source task-specific feature extractor cannot account for the data in the target domain; hence its performance would be poorer than that of direct learning without transfer, i.e., a negative transfer would occur.

4.4.3 Heat capacity of organic polymers

We highlight the benefits of separately modeling and estimating the domain-specific factors through a case study in polymer chemistry. The objective was to predict the specific heat capacity at a constant pressure C_{P} for any given organic polymer

Table 4.2: Force-field parameters that form the General AMBER force field (Wang et al., 2004) version 2 (GAFF2), and their detailed descriptions.

Pparameter	Description
mass	Atomic mass
σ	Equilibrium radius of van der Waals (vdW) interactions
ϵ	Depth of the potential well of vdW interactions
charge	Atomic charge of Gasteiger model
r_0	Equilibrium length of chemical bonds
K_{bond}	Force constant of bond stretching
polar	Bond polarization defined by the absolute value of charge difference between atoms in a bond
θ_0	Equilibrium angle of bond angles
K_{angle}	Force constant of bond bending
K_{dih}	Rotation barrier height of dihedral angles

with its chemical structure in the repeating unit. Specifically, we conducted TL to bridge the gap between the experimental values and physical properties calculated from molecular dynamics (MD) simulations.

As shown in Figure 4.6, there was a large systematic bias between the experimental and calculated values; the MD-calculated properties $C_{\mathrm{P}}^{\mathrm{MD}}$ exhibited an evident overestimation with respect to their experimental values. As discussed in Hayashi et al. (2022), this observation is inevitable because classical MD calculations do not reflect the presence of quantum effects in the real system: the vibrational energy of the classical harmonic oscillator is significantly larger than that of the quantum harmonic oscillator at the same frequency. Hence, the upward bias was observed in $C_{\mathrm{P}}^{\mathrm{MD}}$, which mainly originated from the lack of quantum effects. According to Einstein's theory for specific heat in physical chemistry, the logarithmic ratio between $C_{\mathrm{P}}^{\mathrm{exp}}$ and $C_{\mathrm{P}}^{\mathrm{MD}}$ can be calibrated using the following equation (Hayashi et al., 2022):

$$\log C_{\mathrm{P}}^{\mathrm{exp}} = \log C_{\mathrm{P}}^{\mathrm{MD}} + 2\log\left(\frac{\hbar\omega}{k_B T}\right) + \log \frac{\exp\left(\frac{\hbar\omega}{k_B T}\right)}{\left[\exp\left(\frac{\hbar\omega}{k_B T}\right) - 1\right]^2}, \tag{4.13}$$

where k_B is the Boltzmann constant, $\hbar$ is the Planck constant, ω is the frequency of molecular vibrations, and T is the temperature. The bias is a monotonically decreasing function of frequency ω, which is described as a black-box function of polymers with their molecular features. Hereinafter, we consider the calibration of this systematic bias using the affine transfer model.

Data

The experimental values of the specific heat capacity of the 70 polymers were collected from PoLyInfo (Otsuka et al., 2011). The MD simulation was also applied to calculate their heat capacities. To enable the models to predict the log-transformed heat capacity,

Table 4.3: Comparison of three prediction models for experimental values of specific heat capacity with and without using the MD-calculated properties as source features (mean and standard deviation of RMSE).

Model	RMSE (log J/kg · K)
$y = \alpha_0 + \alpha_1 f_s + \epsilon$	0.1403 ± 0.04610
$y = f_s + x^\top \gamma + \epsilon$	0.1368 ± 0.04265
$y = \alpha_0 + \alpha_1 f_s - (\beta f_s + 1) x^\top \gamma + \epsilon$	**0.1357 ± 0.04173**

a given polymer with its chemical structure was translated into the 190-dimensional force-field descriptors using RadonPy[3] (Hayashi et al., 2022). We randomly sampled 60 training polymers and tested the prediction performance of a trained model on the remaining 10 polymers 20 times. The PoLyInfo sample identifiers for the selected polymers are listed in the code.

The descriptor x represents the distribution of the ten different force-field parameters ($t \in T = \{\text{mass}, \sigma, \epsilon, \text{charge}, r_0, K_{\text{bond}}, \text{polar}, \theta_0, K_{\text{angle}}, K_{\text{dih}}\}$ that constitute the empirical potential (i.e., the General AMBER force field (Wang et al., 2004) version 2 (GAFF2)) of the classical MD simulation. The detailed descriptions of the parameters are listed in Table 4.2. For each t, pre-defined values are assigned to the constituent elements in a polymer, such as individual atoms (mass, charge, σ, and ϵ), bonds (r_0, K_{bond}, and polar), angles (θ_0 and K_{angle}), or dihedral angles (K_{dih}). The probability density function of the assigned values of t is then estimated and discretized into 10 points corresponding to 10 different element species such as hydrogen and carbon for mass, and 20 equally spaced grid points for the other parameters. Thus, the 190-dimensional descriptor vector x consists of nine blocks with 10 or 20 elements corresponding to the different types of force-field parameters.

The source feature f_s was given by the log-transformed value of C_{P}^{MD}. Therefore, f_s was no longer a function of x; this experiment was intended for calibrating the MD-calculated properties rather than for conventional TL.

Model definition and hyperparameter search

In addition to the affine model transfer, ordinary linear regression and the log-difference model were used for comparison in this experiment. The details of each model are as follows:

Ordinary linear regression Simply, the experimental heat capacity $y = \log C_{\text{P}}^{\text{exp}}$ was regressed on the MD-calculated property, without regularization as $\hat{y} = \alpha_0 + \alpha_1 f_s$,

[3] https://github.com/RadonPy/RadonPy

Algorithm 4.3 Block relaxation algorithm for the model in Eq. (4.15).

Initialize
$\alpha_0 \leftarrow \hat{\alpha}_{0,\mathrm{olr}}, \quad \alpha_1 \leftarrow \hat{\alpha}_{1,\mathrm{olr}}, \quad \beta \leftarrow 0, \ \ \gamma \leftarrow \hat{\gamma}_{\mathrm{diff}}$
repeat
$\alpha \leftarrow \arg\min_\alpha F_{\alpha,\beta,\gamma}$
$\beta \leftarrow \arg\min_\beta F_{\alpha,\beta,\gamma}$
$\gamma \leftarrow \arg\min_\gamma F_{\alpha,\beta,\gamma}$
until convegence

where $\hat{y}$ denotes the conditional expectation and $f_s = \log C_{\mathrm{P}}^{\mathrm{MD}}$.

Learning the log-difference We calculated the log-difference $\log C_{\mathrm{P}}^{\mathrm{exp}} - \log C_{\mathrm{P}}^{\mathrm{MD}}$ and trained the linear model, i.e., we trained the model $\hat{y} = f_s + x^\top \gamma$. Note that this model corresponds to the theoretical model of Eq. (4.13).

As mentioned above, the input x was divided into 10 blocks describing the discretized density function of a force-field parameter. For each block, the corresponding parameters in γ should be estimated smoothly. To this end, fused regularization was introduced in the objective function to be minimized:

$$\lambda_1 \|\gamma\|_2^2 + \lambda_2 \sum_{t \in T} \sum_{j=2}^{J_t} \left(\gamma_{t,j} - \gamma_{t,j-1}\right)^2, \tag{4.14}$$

where $J_t = 10$ for $t = \mathrm{mass}$ and $J_t = 20$ otherwise. $\gamma_{t,j}$ denotes the coefficient for the j-th grid point of the force-field parameter t.

The hyperparameters λ_1 and λ_2 for the scale- and smoothness-regularizers were determined based on five-fold cross-validation across 25 equally spaced grids in the interval $[10^{-2}, 10^2]$ for λ_1 and in the candidate set $\{50, 100, 150\}$ for λ_2.

Affine transfer The log-transformed value of $C_{\mathrm{p}}^{\mathrm{exp}}$ was modeled as

$$y := \log C_{\mathrm{P}}^{\mathrm{exp}} = \underbrace{\alpha_0 + \alpha_1 f_s}_{g_1} - \underbrace{(\beta f_s + 1)}_{g_2} \cdot \underbrace{x^\top \gamma}_{g_3} + \epsilon, \tag{4.15}$$

where ϵ denotes an observation noise, and $\alpha_0, \alpha_1, \beta$, and γ are the unknown parameters to be estimated. For $\alpha_1 = 1$ and $\beta = 0$, Eq. (4.15) is consistent with the theoretical equation in Eq. (4.13) in which the quantum effect is linearly modeled as $\alpha_0 + x^\top \gamma$.

The fused regularization expressed as Eq. (4.14) was also introduced for the affine model transfer. Consequently, in the model training, the objective function was given as follows:

$$F_{\alpha,\beta,\gamma} = \frac{1}{n}\sum_{i=1}^{n}\left\{y_i - (\alpha_0 + \alpha_1 f_s(x_i) - (\beta f_s(x_i) + 1)x^\top\gamma)\right\}^2 + \lambda_\beta\beta^2 + \lambda_{\gamma,1}\|\gamma\|_2^2 + \lambda_{\gamma,2}\sum_{t\in T}\sum_{j=2}^{J_t}\left(\gamma_{t,j} - \gamma_{t,j-1}\right)^2,$$

where $\alpha = [\alpha_0\ \alpha_1]^\top$. With a fixed $\lambda_\beta = 1$, the remaining hyperparameters $\lambda_{\gamma,1}$ and $\lambda_{\gamma,2}$ were optimized through five-fold cross-validation over 25 equally spaced grids in the interval $[10^{-2}, 10^2]$ for $\lambda_{\gamma,1}$ and the candidate set $\{50, 100, 150\}$ for $\lambda_{\gamma,2}$.

The algorithm to estimate the parameters α, β, and γ is described in Algorithm 4.3, where $\alpha_{0,\text{olr}}$ and $\alpha_{1,\text{olr}}$ are the estimated parameters of the ordinary linear regression model, and $\hat{\gamma}_{\text{diff}}$ is the estimated parameter of the log-difference model. For each step, the full conditional minimization of $F_{\alpha,\beta,\gamma}$ with respect to each parameter can be made analytically as

$$\begin{aligned}
&\arg\min_{\alpha} F_{\alpha,\beta,\gamma} \\
&\quad = (F_s^\top F_s)^{-1} y_s^\top (y + (\beta f_s(X) + 1) \circ (X\gamma)), \\
&\arg\min_{\beta} F_{\alpha,\beta,\gamma} \\
&\quad = -(f_s(X)^\top \mathrm{diag}(X\gamma)^2 f_s(X) + n\lambda_2)^{-1} f_s(X)^\top \mathrm{diag}(X\gamma)(y - F_s\alpha + X\gamma), \\
&\arg\min_{\gamma} F_{\alpha,\beta,\gamma} \\
&\quad = -(X^\top \mathrm{diag}(f_s(X)\beta + 1)^2 X + \Lambda)^{-1} X^\top \mathrm{diag}(f_s(X)\beta + 1)(y - F_s\alpha),
\end{aligned}$$

where X denotes the matrix in which the i-th row is x_i, $y = [y_1 \cdots y_n]^\top$, $f_s(X) = [f_s(x_1) \cdots f_s(x_n)]^\top$, $F_s = [f_s(X)\ \ \mathbf{1}]$, and $d = 190$. Λ is a matrix containing the two regularization parameters $\lambda_{\gamma,1}$ and $\lambda_{\gamma,2}$ as

$$\Lambda = D^\top D,$$

where

$$D = \begin{bmatrix} \lambda_{\gamma,1} I_d \\ \lambda_{\gamma,2} M \end{bmatrix},\ M = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \leftarrow m\text{-th rows},$$

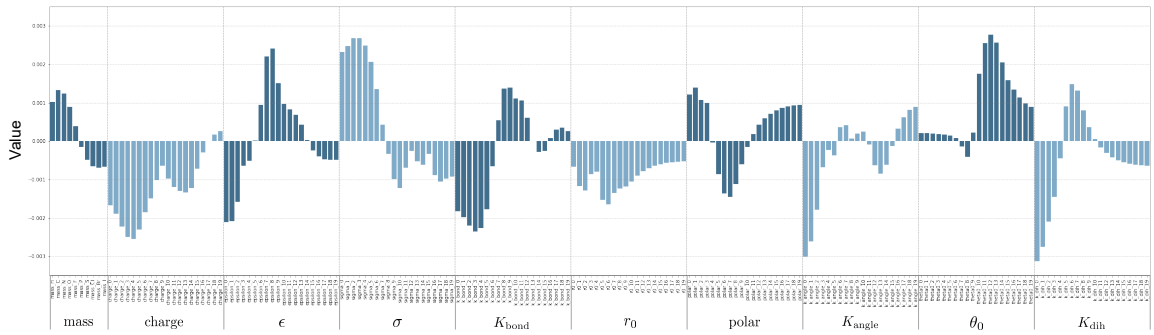


Figure 4.7: Bar plot of the regression coefficients γ of the linear calibrator for addressing the discrepancy between the experimental and MD-calculated specific heat capacities of amorphous polymers.

where $m \in \{10, 30, 50, 70, 90, 110, 130, 150, 170\}$. Note that the matrix M is the same as the matrix $[\mathbf{0} \;\; I_{189}] - [I_{189} \;\; \mathbf{0}]$ except that the m-th row is all zeros. Note also that $M \in \mathbb{R}^{189\times 190}$, and therefore $D \in \mathbb{R}^{279\times 190}$ and $\Lambda \in \mathbb{R}^{190\times 190}$.

The stopping criterion for the algorithm was the same as Eq. (4.12).

Results

Table 4.3 summarizes the prediction performance (RMSE) of the three models. The ordinary linear model $y = \alpha_0 + \alpha_1 f_s + \epsilon$, which ignored the force-field descriptors, exhibited the lowest prediction performance. The other two calibration models, $y = f_s + x^\top \gamma + \epsilon$ and the full model in Eq. (4.15), reached almost the same accuracy, but the latter achieved slightly better prediction accuracy. The estimated parameters of the full model were $\alpha_1 \approx 0.889$ and $\beta \approx -0.004$. The model form is highly consistent with the theoretical equation in Eq. (4.13) as well as the restricted model ($\alpha_1 = 1, \beta = 0$). This supports the validity of the theoretical model in Hayashi et al. (2022), which explains the discrepancy between the experimental and calculated values owing to the presence or absence of quantum effects.

It is expected that physicochemical insights can be obtained by examining the estimated coefficient parameter γ in the term of $x^\top \gamma$, which would capture the contribution of the force-field parameters to the quantum effects yielding the systematic bias in the MD calculations. Figure 4.7 shows the mean values of the estimated parameter γ for the full calibration model. The magnitude of the quantum effect is a monotonically increasing function of the frequency of the harmonic oscillator ω, and is known to be highly related to the depth of the potential well in van der Waals interaction (ϵ) and in bond rotation (K_{dih}), the force constants of bond-stretching (K_{bond}) and -bending (K_{angle}), and the mass of the atoms (mass). According to

physicochemical intuition, it is considered that as ϵ, K_{bond}, K_{angle}, and K_{dih} decrease, their potential energy surfaces become shallow. This decreases the frequency ω, resulting in decreased quantum effects for C_{P}. Further, theoretically, because the molecular vibration of light-weight atoms is faster than that of heavy atoms, ω and quantum effects for C_{P} should increase with decreasing mass. These physical relationships could be captured consistently with the estimated coefficients. The coefficients in the lower regions of ϵ, K_{bond}, K_{angle}, and K_{dih} showed large negative values, indicating that the polymers containing a greater number of atoms, bonds, angles, and dihedral angles with lower values of these force-field parameters will have smaller quantum effects. Conversely, the coefficients in the lower regions of mass showed positive large values, meaning that the polymers containing a greater number of atoms with smaller masses had larger quantum effects. By separately including the domain-common and domain-specific factors in the transfer model, we could infer the features relevant to the cross-domain differences.

4.5 Proofs

In this section, we provide the proofs for Theorem 4.3, Theorem 4.4, and Corollary 4.5.

4.5.1 Proof of Theorem 4.3

To bound the generalization error, we use the empirical and population Rademacher complexities $\hat{\mathfrak{R}}_S(\mathcal{F})$ and $\mathfrak{R}(\mathcal{F})$ of the hypothesis class $\mathcal{F}$, defined as:

$$\hat{\mathfrak{R}}_S(\mathcal{F}) = \mathbb{E}_\sigma \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(x_i), \qquad \mathfrak{R}(\mathcal{F}) = \mathbb{E}_S \hat{\mathfrak{R}}_S(\mathcal{F}),$$

where $\{\sigma_i\}_{i=1}^n$ is a set of Rademacher variables that are independently distributed and each takes one of the values in $\{-1, +1\}$ with equal probability, and S denotes a set of samples. The following proof is based on the proof of Theorem 11 in Kuzborskij & Orabona (2017).

Proof of Theorem 4.3. For any hypothesis class $\mathcal{F}$ with feature map Φ, where $\|\Phi\|^2 \leq 1$, the following inequality holds as mentioned in Section 2.2:

$$\mathbb{E}_\sigma \sup_{\|\theta\|^2 \leq \Lambda} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \langle \theta, \Phi(x_i) \rangle \leq \sqrt{\frac{\Lambda}{n}}.$$

The proof is given, for example, in Theorem 6.12 of Mohri et al. (2018). Thus, the empirical Rademacher complexity of $\mathcal{H}$ is bounded as

$$\begin{aligned}
\hat{\mathfrak{R}}_S(\mathcal{H}) &= \mathbb{E}_\sigma \sup_{\substack{\|\alpha\|_{\mathcal{H}_1}^2 \le \lambda_\alpha^{-1}\hat{R}_s, \\ \|\beta\|_{\mathcal{H}_2}^2 \le \lambda_\beta^{-1}\hat{R}_s, \\ \|\gamma\|_{\mathcal{H}_3}^2 \le \lambda_\gamma^{-1}\hat{R}_s}} \frac{1}{n}\sum_{i=1}^n \sigma_i \Big\{ \langle \alpha, \Phi_1(f_s(x_i))\rangle_{\mathcal{H}_1} \\
&\qquad\qquad + \langle \beta, \Phi_2(f_s(x_i))\rangle_{\mathcal{H}_2} \langle \gamma, \Phi(x_i)\rangle_{\mathcal{H}_3} \Big\} \qquad (4.16) \\
&\le \mathbb{E}_\sigma \sup_{\|\alpha\|_{\mathcal{H}_1}^2 \le \lambda_\alpha^{-1}\hat{R}_s} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle \alpha, \Phi_1(f_s(x_i))\rangle_{\mathcal{H}_1} \\
&\qquad + \sup_{\substack{\|\beta\|_{\mathcal{H}_2}^2 \le \lambda_\beta^{-1}\hat{R}_s, \\ \|\gamma\|_{\mathcal{H}_3}^2 \le \lambda_\gamma^{-1}\hat{R}_s}} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle \beta\otimes\gamma, \Phi_2(f_s(x_i))\otimes\Phi(x_i)\rangle_{\mathcal{H}_2\otimes\mathcal{H}_3} \\
&\le \mathbb{E}_\sigma \sup_{\|\alpha\|_{\mathcal{H}_1}^2 \le \lambda_\alpha^{-1}\hat{R}_s} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle \alpha, \Phi_1(f_s(x_i))\rangle_{\mathcal{H}_1} \\
&\qquad + \sup_{\|\beta\otimes\gamma\|_{\mathcal{H}_2\otimes\mathcal{H}_3}^2 \le \lambda_\beta^{-1}\lambda_\gamma^{-1}\hat{R}_s^2} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle \beta\otimes\gamma, \Phi_2(f_s(x_i))\otimes\Phi(x_i)\rangle_{\mathcal{H}_2\otimes\mathcal{H}_3} \\
&\le \sqrt{\frac{\hat{R}_s}{\lambda_\alpha n}} + \sqrt{\frac{\hat{R}_s^2}{\lambda_\beta\lambda_\gamma n}} \qquad (4.17) \\
&\le \sqrt{\frac{\hat{R}_s}{n}}\left\{\sqrt{\frac{1}{\lambda_\alpha}} + \sqrt{\frac{L}{\lambda_\beta\lambda_\gamma}}\right\}.
\end{aligned}$$

The first inequality follows from the subadditivity of the supremum. The last inequality follows from the fact that $\hat{R}_s \le P_n\ell(y, \langle 0, \Phi_1\rangle) + \lambda_\alpha\|0\|^2 \le L$.

Let $C = \sqrt{\frac{1}{\lambda_\alpha}} + \sqrt{\frac{L}{\lambda_\beta\lambda_\gamma}}$. By applying Talagrand's lemma (Mohri et al., 2018) and Jensen's inequality, we can obtain

$$\mathfrak{R}(\mathcal{L}) = \mathbb{E}\hat{\mathfrak{R}}_S(\mathcal{L}) \le \mu_\ell \mathbb{E}\hat{\mathfrak{R}}_S(\mathcal{H}) \le C\mu_\ell \mathbb{E}\sqrt{\frac{\hat{R}_s}{n}} \le C\mu_\ell\sqrt{\frac{\mathbb{E}\hat{R}_s}{n}}.$$

To apply Corollary 3.5 of Bartlett et al. (2005), we should solve the equation

$$r = C\mu_\ell\sqrt{\frac{r}{n}}, \qquad (4.18)$$

and obtain $r^* = \frac{\mu_\ell^2 C^2}{n}$. Thus, for any $\eta > 0$ with probability at least $1 - e^{-\eta}$, there exists a constant $C' > 0$ that satisfies

$$P_n \ell(y, h) \leq C' \left(\mathbb{E}\hat{R}_s + \frac{\mu_\ell^2 C^2}{n} + \frac{\eta}{n} \right) \leq C' \left(R_s + \frac{\mu_\ell^2 C^2}{n} + \frac{\eta}{n} \right). \tag{4.19}$$

Note that, for the last inequality, because $\hat{R}_s \leq P_n \ell(y, \langle \alpha, \Phi_1 \rangle) + \lambda_\alpha \|\alpha\|^2$ for any α, taking the expectation of both sides yields $\mathbb{E}\hat{R}_s \leq P\ell(y, \langle \alpha, \Phi_1 \rangle) + \lambda_\alpha \|\alpha\|^2$, and which gives $\mathbb{E}\hat{R}_s \leq \inf_\alpha \{ P\ell(y, \langle \alpha, \Phi_1 \rangle) + \lambda_\alpha \|\alpha\|^2 \} = R_s$. Consequently, by applying Theorem 1 of Srebro et al. (2010), we have

$$\begin{aligned} P\ell(y, h(x)) &\leq P_n \ell(y, h(x)) \\ &+ \tilde{O}\left(\left(\sqrt{\frac{R_s}{n}} + \frac{\mu_\ell C + \sqrt{\eta}}{n} \right) \left(\sqrt{L} C + \sqrt{L\eta} \right) + \frac{C^2 L + L\eta}{n} \right). \end{aligned}$$

Here, we use $\hat{\mathfrak{R}}_S(\mathcal{H}) \leq C \sqrt{\frac{\hat{R}_s}{n}} \leq C \sqrt{\frac{L}{n}}$. □

Remark 1. As in Kuzborskij & Orabona (2013), without the estimation of the parameters α and β, the right-hand side of Eq. (4.16) becomes $\frac{1}{\sqrt{n}} \left(c_1 + c_2 \sqrt{\hat{R}_s} \right)$ with the constants $c_1 > 0$ and $c_2 > 0$, and Eq. (4.18) becomes

$$r = \frac{1}{\sqrt{n}} (c_1 + c_2 \sqrt{r}).$$

This yields the solution

$$r^* = \left(\frac{c_2}{2\sqrt{n}} + \sqrt{\left(\frac{c_2}{2\sqrt{n}} \right)^2 + \frac{c_1}{\sqrt{n}}} \right)^2 \leq \frac{c_2^2}{n} + \frac{c_1}{\sqrt{n}},$$

where we use the inequality $\sqrt{x} + \sqrt{x + y} \leq \sqrt{4x + 2y}$. Thus, Eq. (4.19) becomes

$$P_n \ell(y, h) \leq C' \left(R_s + \frac{c_2^2}{n} + \frac{c_1}{\sqrt{n}} + \frac{\eta}{n} \right).$$

Consequently, we have the following result:

$$\begin{aligned} P\ell(y, h(x)) &\leq P_n \ell(y, h(x)) \\ &+ \tilde{O}\left(\left(\sqrt{\frac{R_s}{n}} + \frac{\sqrt{c_1}}{n^{3/4}} + \frac{c_2 + \sqrt{\eta}}{n} \right) \left(c_1 + c_2 \sqrt{L} + \sqrt{L\eta} \right) + \frac{(c_1 + c_2 \sqrt{L})^2 + L\eta}{n} \right). \end{aligned}$$

This means that even if $R_s = \tilde{O}(n^{-1/2})$, the resulting rate only improves to $\tilde{O}(n^{-3/4})$.

4.5.2 Proof of Theorem 4.4

Recall that the loss function $\ell(\cdot, \cdot)$ is assumed to be μ_ℓ-Lipschitz for the first argument. In addition, we impose the following assumption.

Assumption 4.8. *There exists a constant $B \geq 1$ such that for every $h \in \mathcal{H}$, $P(h - h^*) \leq BP(\ell(y, h) - \ell(y, h^*))$.*

Because we consider $\ell(y, y') = (y - y')^2$ in Theorem 4.4, Assumption 4.8 holds, as stated in Bartlett et al. (2005).

First, we show the following corollary, which is a slight modification of Theorem 5.4 of Bartlett et al. (2005).

Corollary 4.6. *Let $\hat{h}$ be any element of $\mathcal{H}$ satisfying $P_n\ell(y, \hat{h}) = \inf_{h \in \mathcal{H}} P_n\ell(y, h)$, and let $\hat{h}^{(m)}$ be any element of $\mathcal{H}^{(m)}$ satisfying $P_n\ell(y, \hat{h}^{(m)}) = \inf_{h \in \mathcal{H}^{(m)}} P_n\ell(y, h)$. Define*

$$\hat{\psi}(r) = c_1\hat{\mathfrak{R}}\{h \in \mathcal{H} : \max_{m \in \{1,2\}} P_n(h^{(m)} - \hat{h}^{(m)})^2 \leq c_3 r\} + \frac{c_2\eta}{n},$$

where c_1, c_2, and c_3 are some constants depending only on B and μ_ℓ. Then, for any $\eta > 0$ with probability at least $1 - 5e^{-\eta}$,

$$P\ell(y, \hat{h}) - P\ell(y, h^*) \leq \frac{705}{B}\hat{r}^* + \frac{(11\mu_\ell + 27B)\eta}{n},$$

where $\hat{r}^*$ is the fixed point of $\hat{\psi}$.

Proof. Define the function ψ as

$$\psi(r) = \frac{c_1}{2}\mathfrak{R}\{h \in \mathcal{H} : \mu_\ell^2 \max P(h^{(m)} - h^{(m)*})^2 \leq r\} + \frac{(c_2 - c_1)\eta}{n}.$$

Notice that because $\mathcal{H}, \mathcal{H}^{(1)}$, and $\mathcal{H}^{(2)}$ are all convex and thus star-shaped around each of its points, Lemma 3.4 of Bartlett et al. (2005) implies that ψ is a sub-root. In addition, define the sub-root function ψ_m as

$$\psi_m(r) = \frac{c_1^{(m)}}{2}\mathfrak{R}\{h^{(m)} \in \mathcal{H}^{(m)} : \mu_\ell^2 P(h^{(m)} - h^{(m)*})^2 \leq r\} + \frac{(c_2 - c_1)\eta}{n}.$$

Let r_m^* be the fixed point of $\psi_m(r_m)$. Now, for $r_m \geq \psi_m(r_m)$, Corollary 5.3 of Bartlett et al. (2005) and the condition of the loss function imply that with probability at least $1 - e^{-\eta}$,

$$\begin{aligned}\mu_\ell^2 P(\hat{h}^{(m)} - h^{(m)*})^2 &\leq B\mu_\ell^2 P(\ell(y, \hat{h}^{(m)}) - \ell(y, \hat{h}^{(m)*})) \\ &\leq 705\mu_\ell^2 r_m + \frac{(11\mu_\ell + 27B)B\mu_\ell^2\eta}{n}.\end{aligned}$$

Denote the right-hand side by s_m, and define $r = \max r_m$ and $s = \max s_m$. Because $s \geq s_m \geq r_m \geq r_m^*$, we obtain $s \geq \psi_m(s)$ according to Lemma 3.2 of Bartlett et al. (2005). Thus,

$$s \geq 10\mu_\ell^2 \mathbb{E}\hat{\mathfrak{R}}\{h^{(m)} \in \mathcal{H}^{(m)} : \mu_\ell^2 P(h^{(m)} - h^{(m)*})^2 \leq s\} + \frac{11\mu_\ell^2 \eta}{n}.$$

Therefore, by applying Corollary 2.2 of Bartlett et al. (2005) to the class $\mu_\ell \mathcal{H}^{(m)}$, it follows that with probability at least $1 - e^{-\eta}$,

$$\{h^{(m)} \in \mathcal{H}^{(m)} : \mu_\ell^2 P(h^{(m)} - h^{(m)*})^2 \leq s\} \subseteq \{h^{(m)} \in \mathcal{H}^{(m)} : \mu_\ell^2 P_n(h^{(m)} - h^{(m)*})^2 \leq 2s\}.$$

This implies that with probability at least $1 - 2e^{-\eta}$,

$$\begin{aligned} P_n(\hat{h}^{(m)} - h^{(m)*})^2 &\leq 2\left(705r + \frac{(11\mu_\ell + 27B)B\eta}{n}\right) \\ &\leq 2\left(705 + \frac{(11\mu_\ell + 27B)B}{n}\right) r, \end{aligned}$$

where the second inequality follows from $r \geq \psi(r) \geq \frac{c_2\eta}{n}$. Define $2\left(705 + \frac{(11\mu_\ell+27B)B}{n}\right) = c'$. According to the triangle inequality in $L_2(P_n)$, it holds that

$$\begin{aligned} P_n(h^{(m)} - \hat{h}^{(m)})^2 &\leq \left(\sqrt{P_n(h^{(m)} - h^{(m)*})^2} + \sqrt{P_n(h^{(m)*} - \hat{h}^{(m)})^2}\right)^2 \\ &\leq \left(\sqrt{P_n(h^{(m)} - h^{(m)*})^2} + \sqrt{c'r}\right)^2. \end{aligned}$$

By applying Corollary 2.2 of Bartlett et al. (2005) to $\mu_\ell \mathcal{H}^{(m)}$ for $r \geq \psi_m(r)$, it follows that with probability at least $1 - 4e^{-\eta}$,

$$\begin{aligned} &\{h \in \mathcal{H} : \mu_\ell^2 \max P(h^{(m)} - h^{(m)*})^2 \leq r\} \\ &\quad = \bigcap_{m=1}^{2} \{h^{(m)} \in \mathcal{H}^{(m)} : \mu_\ell^2 P(h^{(m)} - h^{(m)*})^2 \leq r\} \\ &\quad \subseteq \bigcap_{m=1}^{2} \{h^{(m)} \in \mathcal{H}^{(m)} : \mu_\ell^2 P_n(h^{(m)} - h^{(m)*})^2 \leq 2r\} \\ &\quad \subseteq \bigcap_{m=1}^{2} \{h^{(m)} \in \mathcal{H}^{(m)} : \mu_\ell^2 P_n(h^{(m)} - \hat{h}^{(m)})^2 \leq (\sqrt{2r} + \sqrt{c'r})^2\} \\ &\quad = \{h \in \mathcal{H} : \mu_\ell^2 \max P_n(h^{(m)} - \hat{h}^{(m)})^2 \leq c_3 r\}, \end{aligned}$$

where $c_3 = (\sqrt{2} + \sqrt{c'})^2$. Combining this with Lemma A.4 of Bartlett et al. (2005) leads to the following inequality: with probability at least $1 - 5e^{-x}$,

$$\begin{aligned}
\psi(r) &= \frac{c_1}{2}\mathfrak{R}\{h \in \mathcal{H} : \mu_\ell^2 \max P(h^{(m)} - h^{(m)*})^2 \le r\} + \frac{(c_2 - c_1)\eta}{n} \\
&\le c_1 \hat{\mathfrak{R}}_S\{h \in \mathcal{H} : \mu_\ell^2 \max P(h^{(m)} - h^{(m)*})^2 \le r\} + \frac{c_2\eta}{n} \\
&\le c_1 \hat{\mathfrak{R}}_S\{h \in \mathcal{H} : \mu_\ell^2 \max P_n(h^{(m)} - \hat{h}^{(m)})^2 \le c_3 r\} + \frac{c_2\eta}{n} \\
&= \hat{\psi}(r).
\end{aligned}$$

Letting $r = r^*$ and using Lemma 4.3 of Bartlett et al. (2005), we obtain $r^* \le \hat{r}^*$, thus proving the statement. □

Proof of Theorem 4.4. Define $R = \max_m \sup_{h \in \mathcal{H}^{(m)}} P_n(y - h(x))^2$. For any $h \in \mathcal{H}^{(m)}$, we obtain

$$\begin{aligned}
P_n(h^{(m)}(x) - \hat{h}^{(m)}(x))^2 &\le 2P_n(y - h^{(m)}(x))^2 + 2P_n(y - \hat{h}^{(m)}(x))^2 \\
&\le 4 \sup_{h \in \mathcal{H}^{(m)}} P_n(y - h(x))^2 \\
&\le 4R.
\end{aligned}$$

From the symmetry of σ_i and the fact that $\mathcal{H}^{(m)}$ is convex and symmetric, we obtain

the following:

$$
\begin{aligned}
&\hat{\mathfrak{R}}\{h \in \mathcal{H} : \max P_n(h^{(m)} - \hat{h}^{(m)})^2 \leq 4R\} \\
&\qquad = \mathbb{E}_\sigma \sup_{\substack{h^{(m)} \in \mathcal{H}^{(m)} \\ P_n(h^{(m)} - \hat{h}^{(m)})^2 \leq 4R}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \sum_{m=1}^{2} h^{(m)}(x_i) \\
&\qquad = \mathbb{E}_\sigma \sup_{\substack{h^{(m)} \in \mathcal{H}^{(m)} \\ P_n(h^{(m)} - \hat{h}^{(m)})^2 \leq 4R}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \sum_{m=1}^{2} (h^{(m)}(x_i) - \hat{h}^{(m)}(x_i)) \\
&\qquad \leq \mathbb{E}_\sigma \sup_{\substack{h^{(m)}, g^{(m)} \in \mathcal{H}^{(m)} \\ P_n(h^{(m)} - g^{(m)})^2 \leq 4R}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \sum_{m=1}^{2} (h^{(m)}(x_i) - g^{(m)}(x_i)) \\
&\qquad = 2\mathbb{E}_\sigma \sup_{\substack{h^{(m)} \in \mathcal{H}^{(m)} \\ P_n h^{(m)2} \leq R}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \sum_{m=1}^{2} h^{(m)}(x_i) \\
&\qquad \leq 2 \sum_{m=1}^{2} \mathbb{E}_\sigma \sup_{\substack{h^{(m)} \in \mathcal{H}^{(m)} \\ P_n h^{(m)2} \leq R}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h^{(m)}(x_i) \\
&\qquad \leq 2 \sum_{m=1}^{2} \left\{ \frac{2}{n} \sum_{j=1}^{n} \min\{R, \hat{\lambda}_j^{(m)}\} \right\}^{\frac{1}{2}} \\
&\qquad \leq \left\{ \frac{16}{n} \sum_{m=1}^{2} \sum_{j=1}^{n} \min\{R, \hat{\lambda}_j^{(m)}\} \right\}^{\frac{1}{2}}.
\end{aligned}
$$

The second inequality arises from the sub-additivity of the supremum and the third inequality follows from Theorem 6.6 of Bartlett et al. (2005). To obtain the last inequality, we use $\sqrt{x} + \sqrt{y} \leq \sqrt{2(x+y)}$. Thus, we have

$$
\begin{aligned}
&2c_1 \hat{\mathfrak{R}}\{h \in \mathcal{H} : \max P_n(h^{(m)} - \hat{h}^{(m)})^2 \leq 4R\} + \frac{(c_2+2)\eta}{n} \\
&\qquad \leq 4c_1 \left\{ \frac{16}{n} \sum_{m=1}^{2} \sum_{j=1}^{n} \min\left\{R, \hat{\lambda}_j^{(m)}\right\} \right\}^{\frac{1}{2}} + \frac{(c_2+2)\eta}{n}
\end{aligned}
$$

for the constants c_1 and c_2. To apply Corollary 4.6, we should solve the following inequality for r

$$
r \leq 4c_1 \left\{ \frac{16}{n} \sum_{m=1}^{2} \sum_{j=1}^{n} \min\left\{r, \hat{\lambda}_j^{(m)}\right\} \right\}^{\frac{1}{2}}.
$$

For any integer $\kappa_m \in [0, n]$, the right-hand side is bounded as

$$4c_1 \left\{ \frac{16}{n} \sum_{m=1}^{2} \sum_{j=1}^{n} \min \left\{ r, \hat{\lambda}_j^{(m)} \right\} \right\}^{\frac{1}{2}} \leq 4c_1 \left\{ \frac{16}{n} \sum_{m=1}^{2} \left(\sum_{j=1}^{\kappa_m} r + \sum_{j=\kappa_m+1}^{n} \hat{\lambda}_j^{(m)} \right) \right\}^{\frac{1}{2}}$$
$$= \left\{ \left(\frac{256c_1^2}{n} \sum_{m=1}^{2} \kappa_m \right) r + \frac{256c_1^2}{n} \sum_{m=1}^{2} \sum_{j=\kappa_m+1}^{n} \hat{\lambda}_j^{(m)} \right\}^{\frac{1}{2}},$$

and we obtain the solution r^* as

$$r^* \leq \frac{128c_1^2}{n} \sum_{m=1}^{2} \kappa_m + \left(\left\{ \frac{128c_1^2}{n} \sum_{m=1}^{2} \kappa_m \right\}^2 + \frac{256c_1^2}{n} \sum_{m=1}^{2} \sum_{j=\kappa_m+1}^{n} \hat{\lambda}_j^{(m)} \right)^{\frac{1}{2}}$$
$$\leq \frac{256c_1^2}{n} \sum_{m=1}^{2} \kappa_m + \left(\frac{256c_1^2}{n} \sum_{m=1}^{2} \sum_{j=\kappa_m+1}^{n} \hat{\lambda}_j^{(m)} \right)^{\frac{1}{2}}.$$

By optimizing the right-hand side with respect to κ_1 and κ_2, we obtain the solution as

$$r^* \leq \min_{0 \leq \kappa_1, \kappa_2 \leq n} \left\{ \frac{256c_1^2}{n} \sum_{m=1}^{2} \kappa_m + \left(\frac{256c_1^2}{n} \sum_{m=1}^{2} \sum_{j=\kappa_m+1}^{n} \hat{\lambda}_j^{(m)} \right)^{\frac{1}{2}} \right\}.$$

Furthermore, according to Corollary 4.6, there exists a constant c such that with probability at least $1 - 5e^{-\eta}$,

$$P(y - \hat{h}(x))^2 - P(y - h^*(x))^2$$
$$\leq c \left(\min_{0 \leq \kappa_1, \kappa_2 \leq n} \left\{ \frac{1}{n} \sum_{m=1}^{2} \kappa_m + \left(\frac{1}{n} \sum_{m=1}^{2} \sum_{j=\kappa_m+1}^{n} \hat{\lambda}_j^{(m)} \right)^{\frac{1}{2}} \right\} + \frac{\eta}{n} \right).$$

□

Proof of Corollary 4.5. By using the inequality $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ for $x \geq 0, y \geq 0$,

we have

$$
\begin{aligned}
&P(y-\hat{h}(x))^2 - P(y-h^*(x))^2 \\
&\quad = O\left(\min_{0\le\kappa_1,\kappa_2\le n} \left\{ \frac{\kappa_1+\kappa_2}{n} + \left(\frac{1}{n}\sum_{j=\kappa_1+1}^{n} \hat{\lambda}_j^{(1)} + \frac{1}{n}\sum_{j=\kappa_2+1}^{n} \hat{\lambda}_j^{(2)} \right)^{\frac{1}{2}} \right\} + \frac{\eta}{n} \right) \\
&\quad \le O\left(\min_{0\le\kappa_1,\kappa_2\le n} \left\{ \frac{\kappa_1+\kappa_2}{n} + \left(\frac{1}{n}\sum_{j=\kappa_1+1}^{n} j^{-\frac{1}{s_1}} + \frac{1}{n}\sum_{j=\kappa_2+1}^{n} j^{-\frac{1}{s_2}} \right)^{\frac{1}{2}} \right\} + \frac{\eta}{n} \right) \\
&\quad \le O\left(\min_{0\le\kappa_1,\kappa_2\le n} \left\{ \frac{\kappa_1+\kappa_2}{n} + \left(\frac{1}{n}\sum_{j=\kappa_1+1}^{n} j^{-\frac{1}{s_1}} \right)^{\frac{1}{2}} + \left(\frac{1}{n}\sum_{j=\kappa_2+1}^{n} j^{-\frac{1}{s_2}} \right)^{\frac{1}{2}} \right\} + \frac{\eta}{n} \right).
\end{aligned}
$$

Because it holds that

$$
\sum_{j=\kappa_m+1}^{n} j^{-\frac{1}{s_m}} < \int_{\kappa_m}^{\infty} x^{-\frac{1}{s_m}} \mathrm{d}x < \left[\frac{1}{1-\frac{1}{s_m}} x^{1-\frac{1}{s_m}} \right]_{\kappa_m}^{\infty} = \frac{s_m}{1-s_m} \kappa_m^{1-\frac{1}{s_m}},
$$

for $m = 1, 2$, we should solve the following minimization problem:

$$
\min_{0\le\kappa_1,\kappa_2\le n} \left\{ \frac{\kappa_1+\kappa_2}{n} + \left(\frac{1}{n}\frac{s_1}{1-s_1}\kappa_1^{1-\frac{1}{s_1}} \right)^{\frac{1}{2}} + \left(\frac{1}{n}\frac{s_1}{1-s_1}\kappa_2^{1-\frac{1}{s_1}} \right)^{\frac{1}{2}} \right\} \equiv g(\kappa).
$$

By taking the derivative, we have

$$
\frac{\partial g(\kappa)}{\partial \kappa_1} = \frac{1}{n} + \frac{1}{2}\left(\frac{1}{n}\frac{s_1}{1-s_1}\kappa_1^{1-\frac{1}{s_1}} \right)^{-\frac{1}{2}} \left(-\frac{\kappa_1^{-\frac{1}{s_1}}}{n} \right).
$$

By setting this to zero, we find the optimal κ_1 as

$$
\kappa_1 = \left(\frac{s_1}{1-s_1}\frac{4}{n} \right)^{\frac{s_1}{1+s_1}}.
$$

Similarly, we have

$$
\kappa_2 = \left(\frac{s_2}{1-s_2}\frac{4}{n} \right)^{\frac{s_2}{1+s_2}},
$$

and

$$
\begin{aligned}
&P(y-\hat{h}(x))^2 - P(y-h^*(x))^2 \\
&\quad \le O\left(\frac{1}{n}\left(\frac{s_1}{1-s_1}\frac{4}{n}\right)^{\frac{s_1}{1+s_1}} + \frac{1}{n}\left(\frac{s_2}{1-s_2}\frac{4}{n}\right)^{\frac{s_2}{1+s_2}}\right. \\
&\qquad\qquad \left. + 2^{\frac{1-s_1}{1+s_1}}\left(\frac{s_1}{1-s_1}\frac{1}{n}\right)^{\frac{1}{1+s_1}} + 2^{\frac{1-s_2}{1+s_2}}\left(\frac{s_2}{1-s_2}\frac{1}{n}\right)^{\frac{1}{1+s_2}} + \frac{\eta}{n}\right) \\
&\quad = O\left(n^{-\frac{1}{1+s_1}} + n^{-\frac{1}{1+s_2}}\right) \\
&\quad = O\left(n^{-\frac{1}{1+\max\{s_1,s_2\}}}\right).
\end{aligned}
$$

□

4.6 Summary and future perspectives

In this chapter, we defined a general class of TL based on affine model transformations and clarified their learning capability and applicability. We considered a procedure consisting of two stages: first, the source features and target samples were transformed, and then the domain transfer model was estimated using the transformed data. The affine model transformation was shown to minimize the expected squared loss in the class of two-stage TL. The affine transfer model is structurally common to a low-rank tensor regression model and an invertible neural network model with affine coupling layers. In the context of TL, the model can be used to represent and estimate the cross-domain shift and domain-specific factors simultaneously and separately.

Currently, the most widely applied methods of TL reuse features acquired by pre-trained neural networks in the source domain. Although such procedural approaches, including feature extraction and fine-tuning, appear plausible, they lack a theoretical foundation. In addition, the existing methods are designed to describe the target domain using only the features acquired in the source domain, and thus cannot adequately deal with the cases where domain-specific factors are present. Our affine model transfer is a principled methodology based on the minimum expected square loss. It also has the ability to simultaneously and separately handle common and unique factors of the domains.

The present methodology provides a general framework that can handle any model including neural networks and pre-defined features. As described, we can represent an ordinary TL based on feature extraction by using the intermediate layers of a pre-trained neural network as the source features in the affine transfer model. Furthermore, as shown in the case study, the affine transfer model can be used as a calibrator between computational models and real-world systems by defining the

predicted values from the physics simulators as the source features. Lastly, we expect to be able to formulate various kinds of TL by designing the source features and the three coupling functions that make up the optimal form of the transfer models.

Chapter 5

Conclusions and future works

5.1 Concluding remarks

In this thesis, we developed the general frameworks for supervised TL in regression settings. From a probabilistic perspective, Chapter 3 proposed a new class of TL that combines cross-domain similarity regularization and density-ratio estimation. This class is characterized by two hyperparameters that control the training and prediction. From a functional perspective, in Chapter 4, we derived the optimal function class of TL, which minimizes the expected squared loss. These methods are widely applicable because they can be handled with any machine learning model, including neural networks, and because the estimation algorithms can be easily implemented without the need to store the source samples. Furthermore, theoretical analyses were performed for each method to clarify some characteristic properties. These methods are expected to play an important role in many real-world applications.

The relationship between these two proposed methods is shown in Figure 5.1. The Bayesian TL with density-ratio modeling (Bayesian density-ratio transfer), proposed in Chapter 3, bridges similarity regularization and dissimilarity regularization, i.e., it bridges TL based on Bayesian inference and TL based on density-ratio estimation. Because the Bayesian density-ratio transfer is a framework that uses the model outputs, it is notable that only a limited of methods can be described. For example, it does not include regularization using the source parameters introduced in Section 2.1.1, or the sample importance weighting methods based on density-ratio estimation introduced in Section 2.1.3. In contrast, as described in Chapter 3, Bayesian TL based on the model outputs and TL based on density-ratio estimation of the conditional probability distributions are encompassed.

The affine model transfer proposed in Chapter 4 includes the offset and scale HTL and feature extraction of neural networks as well as the dissimilarity regularization approach. Among the approaches bridged by the Bayesian density-ratio transfer, only

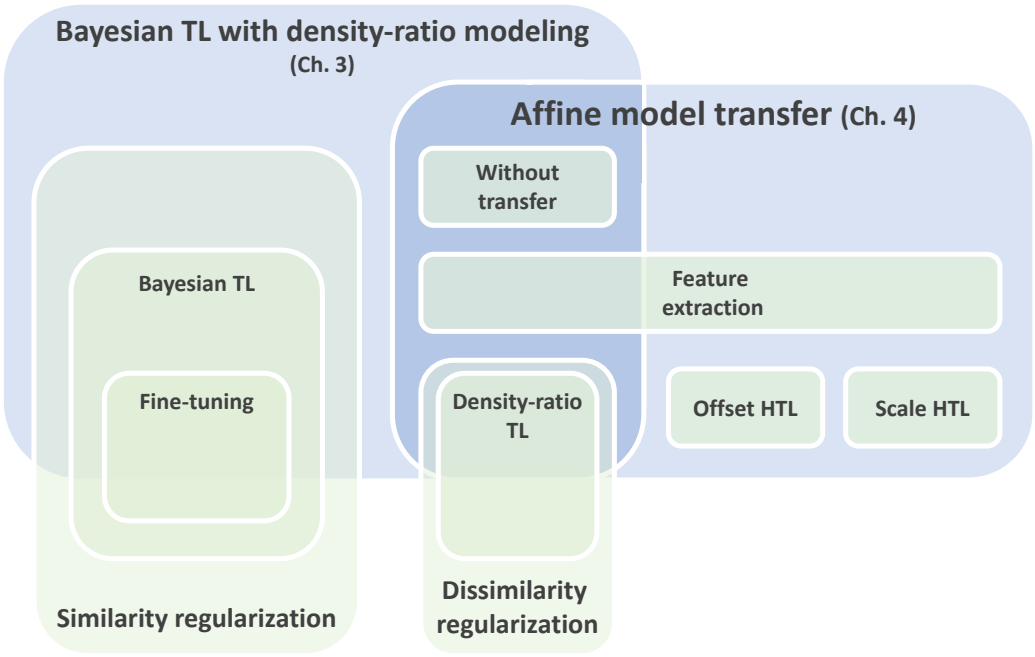


Figure 5.1: Relationship diagram of the proposed methods and related approaches. We proposed two methods: Bayesian transfer learning (TL) with density-ratio modeling in Chapter 3 and the affine model transfer in Chapter 4. These are strongly related to several existing TL approaches, including cross-domain similarity regularization, which is the most natural approach, and feature extraction, which is a widely used procedure. In addition, TL based on density-ratio estimation and hypothesis transfer learning (HTL) are bridged.

the case $\tau = \rho$ satisfies the consistency assumption (Assumption 4.3) used in Chapter 4. In other words, the affine model transfer can represent TL based on density-ratio estimation.

It is worth mentioning that both of these methods encompass direct learning without transfer, which cannot be represented by many standard TL procedures, e.g., fine-tuning and feature extraction. This is owing to the differences in the assumptions related to the cross-domain relationship. In neural transfer, the source models are trained with a large amount of data covering a wide range of domains. Thus, it is supposed that the acquired knowledge is useful for training in the target domain. However, in practical applications, it is difficult to obtain source models that are rich enough to describe the entire target domain. With inappropriate source domains, the features or weights obtained through pre-training do not have much information for the training in the target domain, resulting in failure of TL.

In contrast, both the methods proposed in this thesis do not assume the domain

inclusion relationship and focus on the cross-domain discrepancy. In other words, they allow for the existence of both the domain-common and domain-specific factors. In particular, in the affine model transfer, these two factors are modeled separately, and the contribution of the source knowledge is determined through the selection of the hyperparameters and models. Even when the source knowledge is completely unnecessary for the training in the target domain, direct learning without transfer is selected, and no negative transfer occurs. This procedure of explicitly modeling the domain-specific factors has not received much attention to date but will be an important approach in real-world applications.

5.2 Future perspectives

The following are important challenges to be addressed in the future.

Extension to other loss functions and learning problems This thesis focused mainly on regression problems with the squared loss. For practical applications, we also need to consider various learning tasks, for example, classification problems, unsupervised learning, and reinforcement learning. For other loss functions, different function classes or learning approaches are needed, as discussed in Chapter 4. A more rigorous examination of the general loss function and learning tasks is one of the topics for future research.

Theories for small sample regime In Chapter 4, the convergence rates of the generalization error and excess risk were derived. These results were evaluated according to the order by ignoring the constant terms, and showed asymptotic behavior when the number of samples was sufficiently large. However, as discussed in Chapter 1, TL is typically applied to the case where the number of samples is extremely small. In such cases, the constant term in the theoretical equation cannot be ignored, and evaluation in terms of the order does not make sense. A new theoretical approach for TL is needed, which can be used in real-world data analysis with limited samples.

Source feature selection As mentioned in Chapter 1, the selection of the source domain is important for the effective use of TL. In Chapter 4, the theoretical analysis led to the conclusion that we need to use source features with which learning in the target domain can be performed efficiently. However, as noted above, this theoretical analysis may be effective for a small number of samples. A method to accurately calculate the transferability with a limited number of samples is needed.

Extrapolative prediction Statistics has been traditionally formulated on the concept of interpretation. Methods such as TL aim to develop high-performing models from extremely limited datasets. This resembles extrapolation, in which predictions are made for regions with no data. For example, in Section 4.4.3, the affine transfer model was used to calibrate the observed values from the simulated values of heat capacity, and we successfully constructed a model that is consistent with the theoretical equation. The theoretical formula is the ultimately extrapolative model. This implies that TL has the potential for extrapolative prediction. In other words, it is expected that methodologies that adaptively select and use knowledge from other domains to predict the extrapolated domains can be realized through TL.

References

Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. *European conference on computer vision*, pp. 329–344, 2014.

Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. *Proceedings of the IEEE international conference on computer vision*, pp. 769–776, 2013.

Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, 2004.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. *Annals of Statistics*, 33:1497–1537, 2005.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando C Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2009.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. *Advances in neural information processing systems*, 29, 2016.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20, 2006.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv*, abs/1512.01274, 2015.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1853–1865, 2014.

Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv*, abs/1702.05374, 2017.

Hal Daumé. Frustratingly easy domain adaptation. *arXiv*, abs/0907.1815, 2007.

Hal Daumé and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv*, abs/1410.8516, 2014.

Laurent Dinh, Jascha Narain Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *International Conference on Learning Representations*, 2017.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. *International conference on machine learning*, pp. 647–655, 2014.

Simon S Du, Jayanth Koushik, Aarti Singh, and Barnabás Póczos. Hypothesis transfer learning via transformation functions. *Advances in Neural Information Processing Systems*, 30, 2017.

Simon Shaolei Du, Wei Hu, Sham M. Kakade, J. Lee, and Qi Lei. Few-shot learning via learning the representation, provably. *arXiv*, abs/2002.09434, 2021.

Valerii Fedorov, W. J. Studden, and Eugene M. Klimko. Theory of optimal experiments. *Biometrika*, 59:697, 1972.

Jenny Rose Finkel and Christopher D. Manning. Hierarchical bayesian domain adaptation. *Proceedings of the 2009 annual conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

Chelsea Finn, P. Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 2017.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135, 1999.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33 (1):1–22, 2010.

M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, Jr. Montgomery, J. A., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, O. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J Fox. Gaussian 09, 2016. Gaussian, Inc., Wallingford CT.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1): 2096–2030, 2016.

Muhammad Ghifary, W Bastiaan Kleijn, and Mengjie Zhang. Domain adaptive neural networks for object recognition. *Pacific Rim international conference on artificial intelligence*, pp. 898–904, 2014.

Mehmet Gönen and Adam Margolin. Kernelized bayesian transfer learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28, 2014.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2014.

Rajarshi Guha. Chemical informatics functionality in R. *Journal of Statistical Software*, 18(5):1–16, 2007.

Richard A. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-model factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

Yoshihiro Hayashi, Junichiro Shiomi, Junko Morikawa, and Ryo Yoshida. RadonPy: Automated physical property calculation using all-atom classical molecular dynamics simulations for polymer informatics. *npj Computational Materials*, 8(222), 2022.

James J Heckman. Sample selection bias as a specification error. *Econometrica: Journal of the econometric society*, pp. 153–161, 1979.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv*, abs/1503.02531, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:5149–5169, 2020.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

Randy Jalem, Kenta Kanamori, Ichiro Takeuchi, Masanobu Nakayama, Hisatsugu Yamasaki, and Toshiya Saito. Bayesian-driven first-principles calculations for accelerating exploration of fast ion conductors for rechargeable battery application. *Scientific Reports*, 8(1):1–10, 2018.

Sheng Ju, Ryo Yoshida, Chang Liu, Kenta Hongo, Terumasa Tadano, and Junichiro Shiomi. Exploring diamond-like lattice thermal conductivity crystals via feature-based transfer learning. *Physical Review Materials*, 5(5):053801, 2021.

Chiho Kim, Anand Chandrasekaran, Tran Doan Huan, Deya Das, and Rampi Ramprasad. Polymer genome: A data-powered polymer informatics platform for property predictions. *The Journal of Physical Chemistry C*, 122(31):17575–17585, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2015.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, 31, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Hassabis Demis, Clopath Claudia, Kumaran Dharshan, and

Hadsell Raia. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Marius Kloft and Gilles Blanchard. The local rademacher complexity of lp-norm multiple kernel learning. *Advances in Neural Information Processing Systems*, 2011.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47:1902–1914, 2001.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. *International Conference on Machine Learning*, pp. 5468–5479, 2020.

Ilja Kuzborskij and Francesco Orabona. Stability and hypothesis transfer learning. *International Conference on Machine Learning*, pp. 942–950, 2013.

Ilja Kuzborskij and Francesco Orabona. Fast rates by transferring from auxiliary hypotheses. *Machine Learning*, 106(2):171–195, 2017.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Workshop on challenges in representation learning, ICML*, 3:896, 2013.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. *AAAI Conference on Artificial Intelligence*, 2017.

Chang Liu, Erina Fujita, Yukari Katsura, Yuki Inada, Asuka Ishikawa, Ryuji Tamura, Kaoru Kimura, and Ryo Yoshida. Machine learning to predict quasicrystals from chemical compositions. *Advanced Materials*, 33(36):2102507, 2021.

Song Liu and Kenji Fukumizu. Estimating posterior ratio for classification: transfer learning from probabilistic perspective. *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 747–755, 2016.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. *International conference on machine learning*, pp. 97–105, 2015.

Steven A Lopez, Edward O Pyzer-Knapp, Gregor N Simm, Trevor Lutzow, Kewei Li, Laszlo R Seress, Johannes Hachmann, and Alán Aspuru-Guzik. The Harvard organic photovoltaic dataset. *Scientific Data*, 3(1):1–7, 2016.

Nan Lu, Tianyi Zhang, Tongtong Fang, Takeshi Teshima, and Masashi Sugiyama. Rethinking importance weighting for transfer learning. *arXiv*, abs/2112.10157, 2021.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

Zvika Marx, Michael T Rosenstein, Leslie Pack Kaelbling, and Thomas G Dietterich. Transfer learning with an ensemble of background tasks. *NIPS Workshop on Inductive Transfer*, 2005.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Shahar Mendelson. Geometric parameters of kernel machines. *Proceedings of the 15th Annual Conference on Learning Theory*, 2002.

Shunya Minami, Song Liu, Stephen Wu, Kenji Fukumizu, and Ryo Yoshida. A general class of transfer learning regression without implementation cost. *Proceedings of AAAI Conference on Artificial Intelligence*, 35:8992–8999, 2021.

Shunya Minami, Kenji Fukumizu, Yoshihiro Hayashi, and Ryo Yoshida. Transfer learning with affine model transformation. *arXiv*, abs/2210.09745, 2022.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet S. Talwalkar. *Foundations of Machine Learning*. MIT press, 2018.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Shingo Otsuka, Isao Kuwajima, Junko Hosoya, Yibin Xu, and Masayoshi Yamazaki. PoLyInfo: Polymer database for polymeric materials design. *2011 International Conference on Emerging Intelligent Data and Web Technologies*, pp. 22–29, 2011.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22:57:1–57:64, 2021.

Arindam Paul, Dipendra Jha, Reda Al-Bahrani, Wei-keng Liao, Alok Choudhary, and Ankit Agrawal. Transfer learning using ensemble neural networks for organic solar cell screening. *2019 International Joint Conference on Neural Networks*, pp. 1–8, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

Edward O Pyzer-Knapp, Kewei Li, and Alan Aspuru-Guzik. Learning from the Harvard clean energy project: the use of neural networks to accelerate materials discovery. *Advanced Functional Materials*, 25(41):6495–6502, 2015.

Rajat Raina, Andrew Y Ng, and Daphne Koller. Constructing informative priors using transfer learning. *Proceedings of the 23rd International Conference on Machine Learning*, pp. 713–720, 2006.

Rajat Raina, Alexis Battle, Honglak Lee, Ben Packer, and A. Ng. Self-taught learning: transfer learning from unlabeled data. *International Conference on Machine Learning*, 2007.

Ievgen Redko, Amaury Habrard, and Marc Sebban. Theoretical analysis of domain adaptation with optimal transport. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 737–753, 2017.

Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. A survey on domain adaptation theory: learning bounds and theoretical guarantees. *arXiv*, abs/2004.11829, 2020.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10674–10685, 2022.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, 2019.

Burr Settles. Active learning. *Synthesis lectures on artificial intelligence and machine learning*, 6(1):1–114, 2012.

Rahul Kumar Sevakula, Vikas Singh, Nishchal Kumar Verma, Chandan Kumar, and Yan Cui. Transfer learning for molecular cancer classification using deep neural networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16:2089–2100, 2019.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90: 227–244, 2000.

Ravid Shwartz-Ziv, Micah Goldblum, Hossein Souri, Sanyam Kapoor, Chen Zhu, Yann LeCun, and Andrew Gordon Wilson. Pre-train your loss: Easy bayesian transfer learning with informative priors. *arXiv*, abs/2205.10279, 2022.

Danny Silver, Goekhan Bakir, Kristin Bennett, Rich Caruana, Massimiliano Pontil, Stuart Russell, and Prasad Tadepalli. Inductive transfer : 10 years later, 2005. NIPS workshop.

Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Smoothness, low noise and fast rates. *Advances in Neural Information Processing Systems*, 23, 2010.

Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

Ingo Steinwart, Don R. Hush, and Clint Scovel. Optimal rates for regularized least squares regression. *Proceedings of the 22nd Annual Conference on Learning Theory*, pp. 79—-93, 2009.

Masashi Sugiyama and Muller Klaus-Robert. Input-dependent estimation of generalization error under covariate shift. *Statistics and Risk Modeling*, 23:249–279, 2005.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007a.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems*, 2007b.

Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746, 2008.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. *European conference on computer vision*, pp. 443–450, 2016.

Masaaki Takada and Hironori Fujisawa. Transfer learning via ℓ_1 regularization. *Advances in Neural Information Processing Systems*, 33:14266–14277, 2020.

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. *Advances in Neural Information Processing Systems*, 33: 3362–3373, 2020.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Nilesh Tripuraneni, Michael Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. *Advances in Neural Information Processing Systems*, 33:7852–7862, 2020.

Nilesh Tripuraneni, Chi Jin, and Michael Jordan. Provable meta-learning of linear representations. *International Conference on Machine Learning*, pp. 10434–10443, 2021.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Junmei Wang, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. Development and testing of a general amber force field. *Journal of Computational Chemistry*, 25, 2004.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Marvin N. Wright and Andreas Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77 (1):1–17, 2017.

Stephen Wu, Yukiko Kondo, Masa-aki Kakimoto, Bin Yang, Hironao Yamada, Isao Kuwajima, Guillaume Lambard, Kenta Hongo, Yibin Xu, Junichiro Shiomi, et al. Machine-learning-assisted discovery of polymers with high thermal conductivity using a molecular design algorithm. *npj Computational Materials*, 5(1):1–11, 2019.

Hironao Yamada, Chang Liu, Stephen Wu, Yukinori Koyama, Sheng Ju, Junichiro Shiomi, Junko Morikawa, and Ryo Yoshida. Predicting materials properties with little data using shotgun transfer learning. *ACS Central Science*, 5:1717–1730, 2019.

Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Neural Computation*, 25:1324–1370, 2013.

Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer Learning*. Cambridge University Press, 2020.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, pp. 3320–3328, 2014.

Chao Zhang, Lei Zhang, and Jieping Ye. Generalization bounds for domain adaptation. *Advances in neural information processing systems*, 4:3320–3328, 2012.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Hua Zhou, Lexin Li, and Hongtu Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502): 540–552, 2013.