Adaptive Transfer Kernel Learning for Transfer Gaussian Process Regression

Pengfei Wei[1], Yiping Ke[2], Yew Soon Ong[2,3], Zejun Ma[1]
Speech & Audio Team, Bytedance AI Lab, Singapore[1]
School of Computer Science and Engineering, Nanyang Technological University[2]
Center for Frontier Artificial Intelligence Research, A*STAR Singapore[3]

Abstract—Transfer regression is a practical and challenging problem with important applications in various domains, such as engineering design and localization. Capturing the relatedness of different domains is the key of adaptive knowledge transfer. In this paper, we investigate an effective way of explicitly modelling domain relatedness through transfer kernel, a transfer-specified kernel that considers domain information in the covariance calculation. Specifically, we first give the formal definition of transfer kernel, and introduce three basic general forms that well cover existing related works. To cope with the limitations of the basic forms in handling complex real-world data, we further propose two advanced forms. Corresponding instantiations of the two forms are developed, namely $Trk_{\alpha\beta}$ and Trk_{ω} based on multiple kernel learning and neural networks, respectively. For each instantiation, we present a condition with which the positive semi-definiteness is guaranteed and a semantic meaning is interpreted to the learned domain relatedness. Moreover, the condition can be easily used in the learning of $TrGP_{\alpha\beta}$ and $TrGP_{\omega}$ that are the Gaussian process models with the transfer kernels $Trk_{\alpha\beta}$ and Trk_{ω} respectively. Extensive empirical studies show the effectiveness of $TrGP_{\alpha\beta}$ and $TrGP_{\omega}$ on domain relatedness modelling and transfer adaptiveness.

Index Terms—Transfer regression, domain relatedness, transfer kernel.

1 Introduction

Transfer learning is a learning strategy leveraging knowledge across domains. It attracts increasing research attention in the last decade, and has achieved a great success in various real-world tasks [1], including sentiment analysis, digital classification, object recognition, etc. With research efforts largely confined to classification problems, the problem of transfer regression has been less studied despite its popularity in practical applications such as localization [2], time-series extrapolation [3], engineering design [4], motion detection [5], to name just a few.

The major difference between transfer regression and transfer classification lies in the predictive function where the former predicts continuous values while the latter outputs discrete ones. This makes some assumptions widely adopted in transfer classification invalid in transfer regression. For instance, the covariate shift assumption [6], which assumes that the discrepancy of two domains only lies in marginal distributions but not in conditional ones, is widely adopted in transfer classification. However, this assumption does not hold in transfer regression as the continuous predictive functions are unlikely to be equal across domains. This necessitates the consideration of the prediction function discrepancy in transfer regression.

The scarcity of the target data in transfer regression problems, even for the unlabeled one, brings further challenges. This is due to either expensive generation or uninformative collection of unlabelled data. For instance, in engine design [4], designs with arbitrary parameters are meaningless while collecting feasible parameters needs to run extremely time-consuming and expensive physical simulations. In this paper, we focus on this challenging transfer regression setting where the target domain only has limited labelled data.

Similar to transfer classification, a brute-force transfer that assumes all domains are mutually relevant is inadvisable in transfer regression due to negative transfer. The crucial step in achieving adaptive transfer regression is to precisely capture the relatedness of domains, based on which to adaptively transfer useful source knowledge to the target task. Clearly, the notion of *relatedness* is the key. Despite that numerous transfer methods have been proposed, there is no unified definition on this notion.

Existing studies represent *relatedness* either implicitly or explicitly. Implicit representation of *relatedness* is through the domain shared structure, e.g., latent feature embedding [7], corresponding instance [8], or the shared hypothesis space [9]. The notion of *relatedness* involved is qualitatively expressed by specifying a form of shared structure. This naturally poses a challenge: it is difficult to interpret how a specified shared structure quantitatively associates with domain relatedness. Another representation of *relatedness* is to model domain relatedness parametrically. In this case, relatedness is quantitatively modelled based on the data from domains. Compared with the implicit representation, such an explicit representation of *relatedness* is much easier to interpret (as it is quantitatively measurable) and employ (as it is data-dependent). One effective explicit representation is through learnable similarity coefficients [2], [10]–[12].

In this paper, we aim to develop novel transfer regression methods that are capable of explicitly modeling domain relatedness. We use Gaussian process (*GP*) as the base regression model. It is well known that *GP* is capable of handling prior beliefs on the characteristics of underlying

functions [13]. We specially take advantage of such a capability to handle the discrepancy of the source and target predictive functions by modeling their joint distribution in a transfer-specified manner. As *GP* is also a kernel-based model, it can be achieved by developing transfer-specified kernel. Instead of treating instance pairs equally as standard kernels do, a transfer kernel calculates the covariance of two instances considering the domain(s) where they come from.

A natural way of differentiating covariance calculation with respect to domains is to exploit distinct functions for the intra-domain and inter-domain instance pairs. For instance, existing works [10], [14]–[17] study a transfer kernel, called Trk_λ, by using a standard kernel k for intra-domain instance pairs and using λk for inter-domain ones where λ is a learnable coefficient. This Trk_λ is parametrically concise and enables computationally cheap *GP* modelling. However, the conciseness of Trk_λ also leads to deficiencies on the expressiveness and adaptiveness. The single coefficient λ in Trk_λ only captures a compromised relationship of domains on a specific data characteristic given by k. Considering real-world data are complex (diversified data characteristics) and heterogenous (coexistence of consistency and divergence across domains), Trk_λ has difficulty in precisely modelling rich data characteristics and accurately capturing heterogenous domain relatedness.

Another critical requirement of designing a transfer kernel is to ensure that it satisfies the Mercer's Theorem [18], i.e., a transfer kernel must be positive semi-definite (PSD). For Trk_λ, different works exploit different ways to guarantee its positive semi-definiteness. For instance, in [14] and [15], the authors employ the Cholesky decomposition or an EM strategy to enforce a PSD but "free-form" relatedness Gram matrix. In contrast, the authors in [10] propose a sufficient condition regularizing the value range of λ to handle the PSD issue. With this condition, Trk_λ is semantically interpreted by different transfer schemes. Following works [16], [17] also adopt this succinct but effective condition.

In this paper, we aim to design flexible and powerful transfer kernels. Considering the lack of a clear and unified definition on transfer kernel in the literature, we give a formal definition of transfer kernel that takes into account domain information in covariance calculation. We then propose three basic general forms of transfer kernel, namely *concatenation form transfer kernel* $Trk_{\mathbf{C}}$, *summation form transfer kernel* $Trk_{\mathbf{S}}$ and *product form transfer kernel* $Trk_{\mathbf{P}}$, using three basic operations to handle instance-domain tuples. The widely studied Trk_λ is one of the instantiations of $Trk_{\mathbf{P}}$. To cope with complex real-world data, we further propose two advanced general forms of transfer kernel, namely *linear product form transfer kernel* $Trk_{\mathbf{LP}}$ and *polynomial product form transfer kernel* $Trk_{\mathbf{PP}}$. Transfer kernels $Trk_{\alpha\beta}$ and Trk_ω are then instantiated for $Trk_{\mathbf{LP}}$ and $Trk_{\mathbf{PP}}$, respectively. Each of $Trk_{\alpha\beta}$ and Trk_ω is with a succinct theorem to ensure its positive semi-definiteness. The theorem also encodes semantic interpretations of the transfer kernel, and can be readily incorporated to the learning of the corresponding transfer *GP* model, which simplifies the optimization.

The rest of the paper is organized as follows. Section 2 summarizes related works. Section 3 presents the problem setting and gives the formal definition of transfer kernel with three basic forms. Section 4 and 5 elaborate the proposed $Trk_{\alpha\beta}$ and Trk_ω, respectively. Section 6 discusses empirical evaluations. Finally, section 7 concludes the paper.

2 Related Work and Preliminaries

Herein, we conduct a comprehensive review on transfer kernel, and then introduce some preliminaries.

Transfer Kernel. One popular form of transfer kernel is Trk_λ and its variants, which have been widely used in a variety of practical fields, such as geostatistics [19], computing engineering [20], industrial dynamics [15], human physiology [21], sensor networks [22] and natural language processing [23]. To be complete, we try to review related works from different fields as much as possible, but utilize the unified name "transfer kernel" instead of the specific name in different fields. This form of transfer kernel defines function covariance as the product of two terms, one for domains and given by a parametric similarity coefficient, and another for input data and represented by a standard kernel. Works [2], [10], [14]–[16], [21]–[23] fall within the scope of this form. More specifically, in [14], [15], [21]–[23], the authors focus on multi-task problems, and propose a free-form similarity matrix where each element encodes the relatedness of two tasks. Cao et al. [10] directly uses a single parametric coefficient to model the relatedness of two domains in the typical two-domains transfer problem. In [16], Wagle et al. follow [10] but improve by further distinguishing the source-to-source covariance from the target-to-target covariance. Wei et al. [2] extend the transfer kernel of [10] to a multiple source transfer problem by assigning each source-target domain pair a similarity coefficient.

Another form of transfer kernel is based on the linear model of coregionalization (LMC) that is widely used in geostatistics [24], multi-output learning [25], and multi-class learning [26]. It exploits multiple similarity coefficients to enable a more flexible relatedness modeling. For instance, in [27], the authors exploit a similar transfer kernel for soil properties using simulated annealing. Moreover, in [28], an LMC-based transfer kernel is proposed to model the relatedness of multiple time sequence data. A compositional kernel structure is used to model the data covariance, and a non-parametric indicator matrix is used to model the task covariance. However, the indicator matrix only allows binary values, which highly constrains the modeling of domain relatedness. Our $Trk_{\alpha\beta}$ also falls within the scope of this form of transfer kernel.

Positive Semi-definiteness. A major challenge in the design of a transfer kernel is to ensure its positive semi-definiteness. In some cases, the PSD issue can be easily solved by exploiting available problem information [22] or confining the value of relatedness [23], [28]. For instance, in [22], a transfer kernel similar to Trk_λ is used for a setting with multiple environmental domains, and a spherical parameterization is allotted to the modeling of relatedness. By using binary values for the similarity matrix, the transfer kernel in [28] is naturally PSD. Herein, we cover scenarios where no assumptions can be made a-priori, nor enough domain information is available to determine a PSD transfer kernel.

Another way of ensuring PSD transfer kernel is to follow a constraint handling strategy explicitly applied during the optimization. For instance, the authors in [27] formulate a

constrained optimization problem which is addressed with a meta-heuristic algorithm. It conducts PSD test on the candidate solution in each iteration of the optimization, which is cumbersome and time-consuming. Moreover, works [14], [15], [21] exploit Cholesky decomposition [29] to decompose the relatedness matrix as the product of two low-triangular matrices, and then utilize this decomposition in the search by means of an Expectation Maximization algorithm. It ensures PSD kernel matrices, but it lacks of a clear-cut relationship between a given low-triangular matrix and the elements in the original relatedness matrix.

To overcome the above limitations, recent studies explore a more holistic way to ensure the positive semi-definiteness. Cao et al. [10] propose a sufficient condition that constrains the value range of the similarity coefficient between -1 to 1. Such a condition not only guarantees the positive semi-definiteness of the transfer kernel, but also enables a semantic interpretation on the learned similarity coefficient. Following [10], the work [2] further provides a sufficient and necessary condition for the positive semi-definiteness of the transfer kernel used in the multiple-source transfer scenario, and demonstrates a pathology. Wagle et al. [16] directly use the conclusion of [10] and [30] to construct a PSD transfer kernel. In a recent work [17], the authors simply adopts [10] for positive semi-definiteness.

Other Related Methods. Instance-based transfer methods provide another way for domain relatedness modelling. The well-known TrAdaBoost [31] iteratively reweights the contribution of each data by a fine-grained relatedness modeling. However, it suffers from the weight convergence issues [32]. In [11], this was addressed with an error adjustment rule based on experimental approximation. Chu et al. [8] propose a selective SVM (STM) based on sample reweighting for facial expression analysis, a transfer classification problem. Recent methods also employ deep learning for *S-T* similarity modeling [33], [34]. They exploit adversarial learning to aggregate domains.

There are some other methods that can be applied to transfer regression problems although they do not explicitly model the domain relatedness. In [35], the authors present a doubly stochastic variational inference algorithm for deep *GP*. The method can handle transfer regression setting by training with the combination of the source and target labelled data. Deep kernel learning [36] combines kernel methods with deep learning architectures. A set of deep features are learned before the kernel calculation. Based on [36], Patacchiola et al. [37] propose a deep kernel transfer method for few-shot learning, which can be applied to both classification and regression tasks. Recently, a modula *GP* [38] is proposed for multi-output learning, which ensembles a dictionary of *GP*s without revisiting any data.

Unsupervised Domain Adaptation. Unsupervised domain adaptation (UDA) is a popular transfer problem setting where the source domain has sufficient labelled data and the target domain has only unlablled data. Although different from the transfer regression setting studied in this paper, we present a review of UDA methods as they also provide insights in achieving successful knowledge transfer. Generally, UDA methods can be divided into subspace based ones and deep learning based ones. An early subspace-based UDA method is transfer component analysis (TCA) [39] that aims to learn the shared features in a reproducing kernel Hilbert space. Gong et al. [40] propose to generate intermediate subspaces along the geodesic path between source and target subspaces on the Grassmann manifold. Fernando et al. [41] propose to align the source and target subspaces directly. Other methods also consider important data properties in transfer, e.g., second-order statistics [42] and geometric discriminative structure [43]. A recent work, MEDA [44], combines marginal and conditional distribution alignments with geometric structure learning, and yields comparable results with deep learning based methods.

Recently, deep learning based methods have dominated the UDA studies. Long et al. [45] develop a deep adaptation network (DAN) to learn transferable features. In the following work [46], a joint adaptation network (JDN) is proposed to minimize the joint distribution of full-layers features. Instead of explicitly aligning domain distributions, Ganin et al. [47] propose an adversarial-based training network for UDA. In [48], Saito et al. attempt to align the source and target distributions using the task-specific decision boundaries. Xu et al. [49] present an innovative discovery for UDA, that is, task-specific features with larger norms are more transferable. To dynamically weight the learning losses of alignment and discriminability, Xiao et al. [50] propose a Dynamic Weighted Learning (DWL) framework. More recently, the authors in [51] develop a Cross-domain Contrastive Learning (CDCL) that exploits the contrastive self-supervised learning to align features. In contrast, Shen et al. [52] use contrastive learning only for pre-training, and obtains competitive results with strong UDA methods. Note that all these UDA methods are mainly tested on the transfer classification problems. It is not trivial to calibrate these methods to the transfer regression setting.

Unsupervised Domain Adaptation Regression. Unsupervised domain adaptation regression is a subproblem of UDA. This setting is still different from our transfer regression setting, mainly on the availability of the target unlabeled data, although it is for regression tasks. Related works [8], [33], [34] discussed above are proposed for this unsupervised setting. Moreover, in [53], the authors introduce a domain-invariant partial-least-squares regression model to align the source and target distributions in the latent-variable space. The model allows the integration of (partially) labeled data from the source and target domains as well as entirely unlabeled data from the latter, which makes it suitable for both the unsupervised setting and our transfer regression setting. Chen et al. [54] observe that classification is robust to feature scaling but regression is not. They then propose to close the domain gap through orthogonal bases of the representation spaces, which are free from feature scaling, for domain adaptation regression. Recently, Mathelin et al. [55] present a novel adversarial reweighting approach to handle unsupervised regression tasks under the covariate shift assumption.

Primitive Kernels. Kernels encode all assumptions about the form of functions that *GP* is modelling. Some primitive kernels are widely used in *GP* models including:

- constant kernel: $C(\mathbf{x}, \mathbf{x}') = \sigma^2$,
- linear kernel: $LIN(\mathbf{x}, \mathbf{x}') = \sigma^2 \mathbf{x}^\mathsf{T} \mathbf{x}$,
- polynomial kernel: $POLY(\mathbf{x}, \mathbf{x}') = \sigma^2 (c + \mathbf{x}^\mathsf{T} \mathbf{x})^d$,
- radial basis function: $RBF(\mathbf{x}, \mathbf{x}') = \sigma^2 exp(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2l^2})$,

Table 1
Notations

Notation	Description
$\mathbf{X}^{\mathcal{S}}, \mathbf{X}^{\mathcal{T}}$	The source and target data matrix.
$\mathbf{y}^{\mathcal{S}}, \mathbf{y}^{\mathcal{T}}$	The source and target label vector.
$n_{\mathcal{S}}, n_{\mathcal{T}}$	The number of source and target data.
$Trk_{\mathbf{C}}$	The concatenation form transfer kernel shown in Eq. (1).
$Trk_{\mathbf{S}}$	The summation form transfer kernel shown in Eq. (2).
$Trk_{\mathbf{P}}$	The product form transfer kernel shown in Eq. (3).
$Trk_{\mathbf{LP}}$	The linear product form transfer kernel shown in Eq. (9).
$Trk_{\mathbf{PP}}$	The polynomial product form transfer kernel shown in Eq. (11).
Trk_{λ}	The transfer kernel using a single similarity coefficient.
$Trk_{\alpha\beta}$	The transfer kernel using multiple kernels.
Trk_{ω}	The transfer kernel using neural kernel network.

- rational quadratic: $RQ(\mathbf{x},\mathbf{x}') = \sigma^2(1+\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\alpha l^2})^{\frac{1}{\alpha}}$,
- periodic: $PER(\mathbf{x},\mathbf{x}') = \sigma^2 exp(-\frac{2sin^2(\pi||\mathbf{x}-\mathbf{x}'||/p)}{l^2})$.

Neural Kernel Network. A neural kernel network (NKN) [56] is a neural net computing compositional kernel structures. It is based on the well-known composition rules for kernels [57]: for any two kernels k_1 and k_2, (1) $\lambda_1 k_1 + \lambda_2 k_2$ with the compositional weights $\lambda_1, \lambda_2 \geq 0$ is a kernel, and (2) the product $k_1 k_2$ is a kernel. A NKN consists of several linear and product layers. By setting all the compositional weights ***non-negative***, every unit of the network is a valid kernel. As proved by [57], NKN has very powerful expressiveness to approximate arbitrary stationary kernels.

3 Problem Setting and Definition

In this section, we introduce the problem setting, and then propose a formal definition of transfer kernel.

3.1 Problem Setting

We work on the transfer regression scenario with one source domain $\mathcal{S}$ and one target domain $\mathcal{T}$. Traditionally, $\mathcal{S}$ equips with sufficient labelled data for training. However, instead of accessing sufficient unlabelled data of $\mathcal{T}$, we deal with a challenging setting where only scarce target labelled data is available for training. Specifically, we denote the source input matrix as $\mathbf{X}^{\mathcal{S}} \in \mathbb{R}^{n_{\mathcal{S}} \times dim}$ and its function value vector as $\mathbf{y}^{\mathcal{S}} \in \mathbb{R}^{n_{\mathcal{S}}}$. Likewise, we define $\mathbf{X}^{\mathcal{T}} \in \mathbb{R}^{n_{\mathcal{T}} \times dim}$ and $\mathbf{y}^{\mathcal{T}} \in \mathbb{R}^{n_{\mathcal{T}}}$ for $\mathcal{T}$. Note that no prior domain knowledge is known, and $n_{\mathcal{T}} \ll n_{\mathcal{S}}$. Our goal is to utilize $[\mathbf{X}^{\mathcal{S}}, \mathbf{y}^{\mathcal{S}}]$ and $[\mathbf{X}^{\mathcal{T}}, \mathbf{y}^{\mathcal{T}}]$ to train a regression model that generalizes good predictive performance on the new coming unlabeled data of $\mathcal{T}$. The notations are summarized in Table 1.

Based on standard *GP*, we propose the transfer-specified *GP*, denoted as *TrGP*, that defines a joint Gaussian distribution over the function space of $\mathcal{S}$ and $\mathcal{T}$. With the zero-mean assumption, a *TrGP* is completely determined by the kernel used for the modelling of function covariance. To handle the discrepancy of predictive functions across domains, we propose to develop transfer-specified kernels to explicitly capture the relatedness of domains so as to control the knowledge transfer strength.

3.2 Transfer Kernel Definition

Although many studies [10], [14]–[17] have worked with some specific transfer kernels, e.g., Trk_{λ}, there is no clear definition for transfer kernel. In this paper, we give a general formal definition for transfer kernel. Let $\mathcal{D}$ be an arbitrary set containing domain indicators with $\mathbf{d}_i \in \mathcal{D}$ as the domain indicator for domain $\mathcal{D}_i$. Different measures can be used to define $\mathbf{d}_i$, e.g., simply using one-hot vector or using the representative domain feature embedding. For $\mathcal{D} = \{\mathcal{S}, \mathcal{T}\}$ with no extra prior domain knowledge, it can be simply defined by scalars, i.e., $d_S = 0$ and $d_T = 1$. Let $\mathcal{X}$ be an arbitrary set of input samples with $\mathbf{x}$ as an individual sample. For each $\mathbf{x}$, we construct a tuple $(\mathbf{x}, \mathbf{d})$ indicating $\mathbf{x}$ belongs to the domain whose domain indicator is $\mathbf{d}$. Formally, we have the following definition for transfer kernel.

Definition 1. A transfer kernel *Trk* calculates the covariance of an instance pair $(\mathbf{x}, \mathbf{x}')$ with its domain information $(\mathbf{d}, \mathbf{d}')$ into account, $Trk : (\mathcal{X} \times \mathcal{D}) \times (\mathcal{X} \times \mathcal{D}) \to \mathbb{R}$. Let $\mathcal{H}$ denote an arbitrary real Hilbert space with inner product $\langle \cdot, \cdot \rangle$. For $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{D} \to \mathcal{H}$,

$$Trk\Big((\mathbf{x},\mathbf{d}),(\mathbf{x}',\mathbf{d}')\Big) = \Big\langle (\mathbf{x},\mathbf{d}),(\mathbf{x}',\mathbf{d}') \Big\rangle$$

defines a transfer kernel.

Definition 1 gives a general description, but does not elaborate how to compute the inner product. Specifically, we present three ways of handling the pairwise instance-domain input covariance calculation:

$$Trk_{\mathbf{C}}\Big((\mathbf{x},\mathbf{d}),(\mathbf{x}',\mathbf{d}')\Big) = k^{\mathcal{F}}\Big([\mathbf{x},\mathbf{d}],[\mathbf{x}',\mathbf{d}']\Big), \tag{1}$$

$$Trk_{\mathbf{S}}\Big((\mathbf{x},\mathbf{d}),(\mathbf{x}',\mathbf{d}')\Big) = k^{\mathcal{X}}(\mathbf{x},\mathbf{x}') + k^{\mathcal{D}}(\mathbf{d},\mathbf{d}'), \tag{2}$$

$$Trk_{\mathbf{P}}\Big((\mathbf{x},\mathbf{d}),(\mathbf{x}',\mathbf{d}')\Big) = k^{\mathcal{D}}(\mathbf{d},\mathbf{d}')k^{\mathcal{X}}(\mathbf{x},\mathbf{x}'), \tag{3}$$

where $[\cdot,\cdot]$ is the feature concatenation operator. We call Eqs (1-3) *concatenation form transfer kernel*, *summation form transfer kernel* and *product form transfer kernel* respectively. These are three general forms of transfer kernel motivated from three basic mathematical operations. Note that $Trk_{\mathbf{C}}$, $Trk_{\mathbf{S}}$, and $Trk_{\mathbf{P}}$ must be PSD as Mercer kernels. For $Trk_{\mathbf{C}}$, it is PSD using any standard kernel as $k^{\mathcal{F}}$. For $Trk_{\mathbf{S}}$ and $Trk_{\mathbf{P}}$, according to the composition rules of kernel [57], they are PSD as long as $k^{\mathcal{D}}$ and $k^{\mathcal{X}}$ are PSD.

It can be seen that $Trk_{\mathbf{C}}$ computes the covariance by reconstructing a new feature space concatenating the original feature embedding with the domain embedding. This form highly depends on a good domain embedding. For the two-domains scenario where d_S and d_T are scalars, this is not a wise choice. For $Trk_{\mathbf{S}}$ and $Trk_{\mathbf{P}}$, both of them decouple the calculation of instance covariance and domain covariance. One significant advantage of doing so is that we can explicitly utilize domain covariance as the learnable domain relatedness. Precisely, $Trk_{\mathbf{S}}$ takes domain covariance as an offset of instance covariance, while $Trk_{\mathbf{P}}$ uses domain covariance to reweight instance covariance. The widely studied transfer kernel Trk_{λ} is an instantiation of $Trk_{\mathbf{P}}$.

3.3 Transfer Kernel Trk_{λ}

To be complete, we briefly show how Trk_{λ} falls within the scope of $Trk_{\mathbf{P}}$. As shown in [10], Trk_{λ} is defined as:

$$Trk_{\lambda}(\mathbf{x},\mathbf{x}') = \begin{cases} \lambda k(\mathbf{x},\mathbf{x}'), \ \delta(\mathbf{x},\mathbf{x}') = 0, \\ k(\mathbf{x},\mathbf{x}'), \ \delta(\mathbf{x},\mathbf{x}') = 1, \end{cases}$$

where $\delta(\mathbf{x}, \mathbf{x}') = 1$ if $\mathbf{x}$ and $\mathbf{x}'$ are from the same domain, otherwise $\delta(\mathbf{x}, \mathbf{x}') = 0$, λ is a learnable coefficient measuring domain relatedness, and $k(\cdot, \cdot)$ is a standard kernel. It can be seen that *Trk*$_\lambda$ defines $k^{\mathcal{D}}$ in Eq. (3) as follows:

$$k^{\mathcal{D}}(\mathbf{d}, \mathbf{d}') = \begin{cases} \lambda, & \delta(\mathbf{x}, \mathbf{x}') = 0, \\ 1, & \delta(\mathbf{x}, \mathbf{x}') = 1. \end{cases}$$

To further ensure $k^{\mathcal{D}}$ PSD, λ is set to be within -1 to 1. Such a value range exposes more rationales on *Trk*$_\lambda$. For the case $|\lambda| = 1$, *Trk*$_\lambda$ models the extreme transfer scenario where the two domains are consistently related, and so all the source data is fully used in transfer. For the case $|\lambda| = 0$, *Trk*$_\lambda$ models another extreme transfer scenario in which the two domains are completely unrelated, and thus only target data is used to avoid negative transfer. For the case $|\lambda| < 1$, *Trk*$_\lambda$ fine-tunes the impact $\mathcal{S}$ has on $\mathcal{T}$, and the learned λ reflects the corresponding relatedness.

3.4 Learning and Inference with Transfer Kernel

A *TrGP* defines a joint Gaussian distribution over the function space of $\mathcal{S}$ and $\mathcal{T}$. Herein, we show how to learn a *TrGP*. Specifically, we use the *TrGP* with *Trk*$_\lambda$ as an example. As the objective is to boost the target task, we propose to optimize the target conditional distribution given the labelled source and target data, i.e., $p(\mathbf{y}^{\mathcal{T}}|\mathbf{X}^{\mathcal{T}}, \mathbf{X}^{\mathcal{S}}, \mathbf{y}^{\mathcal{S}})$. By denoting $\delta_{\mathcal{S}}^2$ and $\delta_{\mathcal{T}}^2$ as the noise variance for $\mathcal{S}$ and $\mathcal{T}$ respectively, we have the joint distribution:

$$p(\mathbf{y}^{\mathcal{T}}, \mathbf{y}^{\mathcal{S}}|\mathbf{X}^{\mathcal{T}}, \mathbf{X}^{\mathcal{S}}) \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_\lambda),$$

where

$$\mathbf{C}_\lambda = \begin{bmatrix} \mathbf{K}_{\mathcal{S}\mathcal{S}} & \lambda\mathbf{K}_{\mathcal{S}\mathcal{T}} \\ \lambda\mathbf{K}_{\mathcal{T}\mathcal{S}} & \mathbf{K}_{\mathcal{T}\mathcal{T}} \end{bmatrix}$$

is the covariance matrix calculated by *Trk*$_\lambda$ on $\mathbf{X}^{\mathcal{S}}$ and $\mathbf{X}^{\mathcal{T}}$. Then we can obtain the target conditional distribution:

$$p(\mathbf{y}^{\mathcal{T}}|\mathbf{X}^{\mathcal{T}}, \mathbf{X}^{\mathcal{S}}, \mathbf{y}^{\mathcal{S}}) \sim \mathcal{N}(\boldsymbol{\mu}_*, \mathbf{C}_*),$$

where

$$\begin{aligned} \boldsymbol{\mu}_* =& \lambda\mathbf{K}_{\mathcal{T}\mathcal{S}}(\mathbf{K}_{\mathcal{S}\mathcal{S}} + \delta_{\mathcal{S}}^2\mathbf{I})^{-1}\mathbf{y}^{\mathcal{S}}, \\ \mathbf{C}_* =& (\mathbf{K}_{\mathcal{T}\mathcal{T}} + \delta_{\mathcal{T}}^2\mathbf{I}) - \lambda^2\mathbf{K}_{\mathcal{T}\mathcal{S}}(\mathbf{K}_{\mathcal{S}\mathcal{S}} + \delta_{\mathcal{S}}^2\mathbf{I})^{-1}\mathbf{K}_{\mathcal{S}\mathcal{T}}. \end{aligned}$$

The log-likelihood is defined as:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{2}log|\mathbf{C}_*| - \frac{1}{2}(\mathbf{y}^{\mathcal{T}} - \boldsymbol{\mu}_*)^\top\mathbf{C}_*^{-1}(\mathbf{y}^{\mathcal{T}} - \boldsymbol{\mu}_*) - \frac{n}{2}log2\pi \tag{4}$$

where $\boldsymbol{\theta}$ includes all the learnable variables. The optimization can be done by the conjugate gradient descent or any other advanced optimization algorithms. Once we obtain the (sub)optimal $\boldsymbol{\theta}$, we can use them to predict the output value for a new test target point. The inference process is the same as in standard GPs. Given a new target point $\mathbf{x}_{te}^{\mathcal{T}}$, the prediction can be obtained by

$$y_{te}^{\mathcal{T}} = [\mathbf{k}_{\mathbf{x}\mathcal{S}}^{opt}, \mathbf{k}_{\mathbf{x}\mathcal{T}}^{opt}]\mathbf{C}_*^{opt\,-1}[\mathbf{y}^{\mathcal{S}}; \mathbf{y}^{\mathcal{T}}],$$

where $\mathbf{k}_{\mathbf{x}\mathcal{S}}^{opt}$ ($\mathbf{k}_{\mathbf{x}\mathcal{T}}^{opt}$) is the kernel vector calculated using $\mathbf{x}_{te}^{\mathcal{T}}$ and $\mathbf{X}^{\mathcal{S}}$ ($\mathbf{X}^{\mathcal{T}}$) with the optimal $\boldsymbol{\theta}$.

4 Linear Product Transfer Kernel

A major limitation of *Trk*$_\lambda$ is on its expressiveness and adaptiveness. As *Trk*$_\lambda$ uses a single coefficient λ to model the domain relatedness, it can only capture a compromised relationship across domains on a specific data characteristic given by k. Herein, we introduce a more advanced transfer kernel that alleviates the limitation of *Trk*$_\lambda$ in some extents.

4.1 Transfer Kernel *Trk*$_{\alpha\beta}$

To enhance the expressive power, we use multiple kernel learning strategy [58]. Specifically, we assume that a transfer kernel lies in a kernel function space expanded by a set of independent primitive kernels $k_i, i = 1, ..., N$. To enable adaptiveness, we assign different coefficients to different primitive kernels. We then investigate the transfer kernel:

$$Trk_{\alpha\beta}(\mathbf{x}, \mathbf{x}') = \begin{cases} \sum_i \beta_i k_i(\mathbf{x}, \mathbf{x}'), & \delta(\mathbf{x}, \mathbf{x}') = 0, \\ \sum_i \alpha_i k_i(\mathbf{x}, \mathbf{x}'), & \delta(\mathbf{x}, \mathbf{x}') = 1, \end{cases}$$

where $\beta_i, i = 1, ..., N$ ($\alpha_i \geq 0, i = 1, ..., N$) are the combination coefficients of primitive kernels. Note that we use the conical combination coefficients for α_is ($\alpha_i s \geq 0$). We set α_is to be non-negative as they are the similarity coefficients for the case where two domains are the same. Regarding β_is, we derive the condition that they should satisfy later.

The intuition behind *Trk*$_{\alpha\beta}$ is that the two sets of coefficient variables, α_is and β_is, allow diverse similarities on primitive kernels, and thus represent the heterogeneous fine-grained relatedness of $\mathcal{S}$ and $\mathcal{T}$. Specifically, if β_i is close to α_i, $\mathcal{S}$ is closely related to $\mathcal{T}$ on k_i. On the contrary, if β_i is very dissimilar to α_i, it means that the two domains differ considerably on k_i. The similarity of β_i and α_i indicates how much knowledge can be transferred across domains on k_i.

4.2 Positive Semi-definiteness of *Trk*$_{\alpha\beta}$

We now consider to obtain a PSD *Trk*$_{\alpha\beta}$. To make *Trk*$_{\alpha\beta}$ PSD, it is equivalent to guaranteeing that the corresponding Gram matrix is PSD. Next, we provide a necessary and sufficient condition on α_is and β_is to guarantee that the Gram matrix constructed from *Trk*$_{\alpha\beta}$ is PSD for any data.

Theorem 1. A Gram matrix

$$\mathbf{K}_* = \begin{bmatrix} \sum_{i=1}^N \alpha_i\mathbf{K}_{\mathcal{S}\mathcal{S}}^i & \sum_{i=1}^N \beta_i\mathbf{K}_{\mathcal{S}\mathcal{T}}^i \\ \sum_{i=1}^N \beta_i\mathbf{K}_{\mathcal{T}\mathcal{S}}^i & \sum_{i=1}^N \alpha_i\mathbf{K}_{\mathcal{T}\mathcal{T}}^i \end{bmatrix}$$

with $\alpha_i > 0, i = 1, 2, ..., N$ is PSD for any covariance matrices $\mathbf{K}^i, i = 1, 2, ..., N$, in the form

$$\mathbf{K}^i = \begin{bmatrix} \mathbf{K}_{\mathcal{S}\mathcal{S}}^i & \mathbf{K}_{\mathcal{S}\mathcal{T}}^i \\ \mathbf{K}_{\mathcal{T}\mathcal{S}}^i & \mathbf{K}_{\mathcal{T}\mathcal{T}}^i \end{bmatrix},$$

if and only if $|\beta_i| \leq \alpha_i$ for $i = 1, 2, ..., N$.

Proof 1. Necessary condition: We first rewrite $\mathbf{K}_*$ as:

$$\mathbf{K}_* = \sum_{i=1}^N \alpha_i\mathbf{K}_*^i, \tag{5}$$

with

$$\mathbf{K}_*^i = \begin{bmatrix} \mathbf{K}_{\mathcal{S}\mathcal{S}}^i & \beta_i/\alpha_i\mathbf{K}_{\mathcal{S}\mathcal{T}}^i \\ \beta_i/\alpha_i\mathbf{K}_{\mathcal{T}\mathcal{S}}^i & \mathbf{K}_{\mathcal{T}\mathcal{T}}^i \end{bmatrix}. \tag{6}$$

Since $\mathbf{K}_*$ is *PSD* for **any** covariance matrices $\mathbf{K}^i, i = 1, 2, ..., N$ and $\alpha_i s > 0$, we can derive

$$\mathbf{K}_*^i \succeq \mathbf{0},\ i = 1, 2, ..., N.$$

Based on the Schur complement theorem [59], we derive:

$$\mathbf{K}^i_{\mathcal{TT}} - (\beta_i/\alpha_i)^2(\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T}(\mathbf{K}^i_{\mathcal{SS}})^{-1}\mathbf{K}^i_{\mathcal{ST}} \succeq \mathbf{0}. \quad (7)$$

Since $\mathbf{K}^i \succeq \mathbf{0},\ i = 1, 2, ..., N$, we have:

$$\mathbf{K}^i_{\mathcal{TT}} - (\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T}(\mathbf{K}^i_{\mathcal{SS}})^{-1}\mathbf{K}^i_{\mathcal{ST}} \succeq \mathbf{0}.$$

We then rewrite Eq. (7):

$$\begin{aligned}\mathbf{K}^i_{\mathcal{TT}} &- (\beta_i/\alpha_i)^2\mathbf{K}^i_{\mathcal{TT}} + (\beta_i/\alpha_i)^2\mathbf{K}^i_{\mathcal{TT}} \\ &- (\beta_i/\alpha_i)^2(\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T}(\mathbf{K}^i_{\mathcal{SS}})^{-1}\mathbf{K}^i_{\mathcal{ST}} \succeq \mathbf{0}.\end{aligned}$$

By setting $\mathbf{M} = \mathbf{K}^i_{\mathcal{TT}} - (\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T}(\mathbf{K}^i_{\mathcal{SS}})^{-1}\mathbf{K}^i_{\mathcal{ST}}$, we have:

$$\mathbf{M} \succeq \mathbf{0},$$

$$(1 - (\beta_i/\alpha_i)^2)\mathbf{K}^i_{\mathcal{TT}} + (\beta_i/\alpha_i)^2\mathbf{M} \succeq \mathbf{0}, \quad (8)$$

As $\mathbf{K}^i_{\mathcal{TT}} \succeq \mathbf{0}$ and Eq. (8) must hold for all $\mathbf{K}^i$, we induce:

$$(1 - (\beta_i/\alpha_i)^2) \geq 0,\ i = 1, 2, ..., N,$$

and thus we have: $|\beta_i| \leq \alpha_i$ for $i = 1, 2, ..., N$.
Sufficient condition: Similarly, we rewrite $\mathbf{K}_*$ as Eq. (5) with Eq. (6). We then consider:

$$\begin{aligned}\mathbf{N} &= \mathbf{K}^i_{\mathcal{TT}} - (\beta_i/\alpha_i)^2(\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T}(\mathbf{K}^i_{\mathcal{SS}})^{-1}\mathbf{K}^i_{\mathcal{ST}} \\ &= \mathbf{K}^i_{\mathcal{TT}} - (\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T}(\mathbf{K}^i_{\mathcal{SS}})^{-1}\mathbf{K}^i_{\mathcal{ST}} \\ &+ (1 - (\beta_i/\alpha_i)^2)(\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T}(\mathbf{K}^i_{\mathcal{SS}})^{-1}\mathbf{K}^i_{\mathcal{ST}}\end{aligned}$$

Since $\mathbf{M} \succeq \mathbf{0}$ and $|\beta_i| \leq \alpha_i$ for $i = 1, 2, ..., N$, we obtain $\mathbf{N} \succeq \mathbf{0}$. Considering $\mathbf{K}^i_{\mathcal{SS}} \succeq \mathbf{0}$, we obtain $\mathbf{K}^*_i \succeq \mathbf{0}$ based on the Schur complement theorem. As the conical sum of PSD matrices is also *PSD*, we obtain that $\mathbf{K}_*$ is *PSD*.
To sum up, we conclude that, for $i = 1, 2, ..., N$, if $\mathbf{K}_*$ is *PSD* with $\alpha_i > 0$ for any $\mathbf{K}^i$, the coefficients β_is should satisfy $|\beta_i| \leq \alpha_i$.

Note that, when $\alpha_i = 0$, the corresponding primitive kernel k_i does not contribute, and thus no knowledge on k_i can be transferred across domains. In this case, β_i must be 0 too. With Theorem 1, the PSD issue of $Trk_{\alpha\beta}$ is properly handled.

We further explore the semantic significance of $Trk_{\alpha\beta}$ following Theorem 1. We start with β_i and α_i for each primitive kernel k_i. Theorem 1 states that $|\beta_i|$ never goes beyond α_i. It is reasonable as α_i alludes to inter-domain case, and thus implies the maximum transfer capacity. The extreme case $|\beta_i| = \alpha_i$ indicates a full knowledge transfer on the primitive kernel k_i, while the opposite case $\beta_i = 0$ implies no knowledge is transferred. When considering all primitive kernels, we can see that $Trk_{\alpha\beta}$ enables rich fine-grained relatedness between two domains.

4.3 General Form of $Trk_{\alpha\beta}$

Different from Trk_λ that seamlessly fits in $Trk_{\mathbf{P}}$, $Trk_{\alpha\beta}$ is an instantiation of an extended Eq. (3). Herein, we propose another general form of transfer kernel, called *linear product form transfer kernel*, as follows:

$$Trk_{\mathbf{LP}}\Big((\mathbf{x}, \mathbf{d}), (\mathbf{x}', \mathbf{d}')\Big) = \sum\nolimits_i k_i^{\mathcal{D}}(\mathbf{d}, \mathbf{d}')k_i^{\mathcal{X}}(\mathbf{x}, \mathbf{x}'), \quad (9)$$

where $k_i^{\mathcal{D}}$ and $k_i^{\mathcal{X}}$ with $i = 1, ..., N$ are primitive kernels for the domain covariance term and the instance covariance term, respectively. Specifically, $Trk_{\alpha\beta}$ is an instantiation of $Trk_{\mathbf{LP}}$ using standard kernels as $k_i^{\mathcal{X}}$ and

$$k_i^{\mathcal{D}}(\mathbf{d}, \mathbf{d}') = \begin{cases} \beta_i,\ \delta(\mathbf{x}, \mathbf{x}') = 0, \\ \alpha_i,\ \delta(\mathbf{x}, \mathbf{x}') = 1. \end{cases}$$

Compared Eq. (9) with Eq. (3), we can see that $Trk_{\mathbf{LP}}$ is a linear combination of multiple $Trk_{\mathbf{P}}$s, in which each $Trk_{\mathbf{P}}$ plays a role in capturing the domain relatedness on one data characteristic. Thus, $Trk_{\mathbf{LP}}$ is superior to $Trk_{\mathbf{P}}$ on both expressiveness and adaptiveness, and such a superiority also applies to their instantiations. Correspondingly, $Trk_{\alpha\beta}$ is a more general and powerful transfer kernel than Trk_λ, and Trk_λ is a special case of $Trk_{\alpha\beta}$ with $N = 1$.

4.4 Transfer GP Learning with $Trk_{\alpha\beta}$

In this section, we present the learning of *TrGP* with $Trk_{\alpha\beta}$, called $TrGP_{\alpha\beta}$. We also show how Theorem 1 is easily exploited in the learning of $TrGP_{\alpha\beta}$. Following Section 3.4, we have $p(\mathbf{y}^\mathcal{T}|\mathbf{X}^\mathcal{T}, \mathbf{X}^\mathcal{S}, \mathbf{y}^\mathcal{S}) \sim \mathcal{N}(\boldsymbol{\mu}_*, \mathbf{C}_*)$ where

$$\begin{aligned}\boldsymbol{\mu}_* &= (\sum\nolimits_{i=1}^N \beta_i \mathbf{K}^i_{\mathcal{TS}})(\sum\nolimits_{i=1}^N \alpha_i \mathbf{K}^i_{\mathcal{SS}} + \delta^2_\mathcal{S}\mathbf{I})^{-1}\mathbf{y}^\mathcal{S}, \\ \mathbf{C}_* &= (\sum\nolimits_{i=1}^N \alpha_i \mathbf{K}^i_{\mathcal{TT}} + \delta^2_\mathcal{T}\mathbf{I}) - (\sum\nolimits_{i=1}^N \beta_i \mathbf{K}^i_{\mathcal{TS}}) \\ &\quad \times (\sum\nolimits_{i=1}^N \alpha_i \mathbf{K}^i_{\mathcal{SS}} + \delta^2_\mathcal{S}\mathbf{I})^{-1}(\sum\nolimits_{i=1}^N \beta_i \mathbf{K}^i_{\mathcal{ST}}).\end{aligned}$$

The log-likelihood is defined as Eq. (4), where $\boldsymbol{\theta}$ includes all α_is and β_is, the parameters in k_i and the noise variance.

Note that we also need to take the relationship of α_is and β_is into account in the learning. That is, we use $\alpha_i \geq 0, i = 1, ..., N$ and $|\beta_i| \leq \alpha_i, i = 1, ..., N$ as constraints in the optimization. Specifically, for each α_i and β_i, in the *l-th* iteration during learning, we calculate the updated α_i^l and use $max(0, \alpha_i^l)$ to avoid negative values of α_i^l. Next we calculate the updated β_i^l. If the updated β_i^l violates the constraint $|\beta_i| \leq \alpha_i$, we project β_i^l to be a value near the closer boundary:

$$\beta_i^l = \begin{cases} \alpha_i^l - \epsilon_i^l,\ if\ \beta_i^l > \alpha_i^l, \\ -\alpha_i^l + \epsilon_i^l,\ if\ \beta_i^l < -\alpha_i^l, \end{cases}$$

where ϵ_i^l is a variable with very small random value guaranteed to be within $[0, 2\alpha_i^l]$. We use ϵ_i^l to explore more diverse values of β_i^l in the search.

5 Polynomial Product Transfer Kernel

While $Trk_{\alpha\beta}$ enhances the expressiveness and adaptiveness compared with Trk_λ, it is still imperfect. One obvious shortcoming of $Trk_{\alpha\beta}$ is that it only captures the linear relationship across primitive kernels and ignores the nonlinearity. In this section, we introduce a transfer neural kernel network, called Trk_ω, a more flexible and powerful transfer kernel that can model arbitrary stationary kernels and capture domain relatedness in a hierarchical manner.

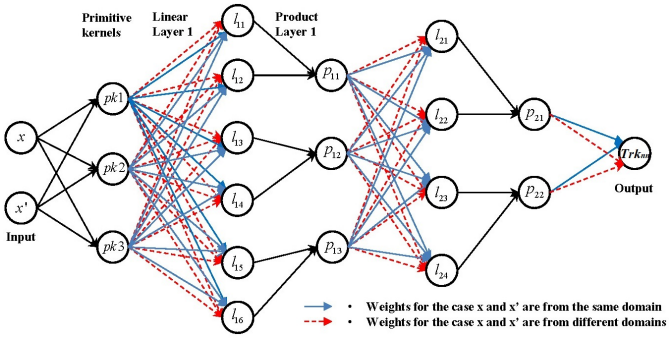


Figure 1. The architecture of a Trk_ω. This Trk_ω contains 3 primitive kernels, 3 linear layers and 2 product layers. The red dash line arrow denotes the weights of the linear layer for k^d_{cks}, and the blue solid line arrow denotes the weights of the linear layer for k^s_{cks}. The black solid line are connections shared by k^d_{cks} and k^s_{cks}.

5.1 Overview of Trk_ω

Basically, the proposed Trk_ω consists of two compositional kernel structures represented by two neural networks, one for the covariance of intra-domain instance pairs and another for that of inter-domain ones. It is defined as:

$$Trk_\omega(\mathbf{x},\mathbf{x}') = \begin{cases} k^d_{cks}(\mathbf{x},\mathbf{x}'), & \delta(\mathbf{x},\mathbf{x}')=0, \\ k^s_{cks}(\mathbf{x},\mathbf{x}'), & \delta(\mathbf{x},\mathbf{x}')=1, \end{cases}$$

For Trk_ω, we have the following remarks. Firstly, each individual k^d_{cks} or k^s_{cks} is a compositional kernel structure. However, either of them is not necessarily to be a NKN [56] since we do not specially constrain the compositional weights to be non-negative. Secondly, k^d_{cks} and k^s_{cks} are coupled to one single network by sharing the same compositional structure but with different compositional weights. The data covariance is given by k^d_{cks} if two instances are from different domains, otherwise it is given by k^s_{cks}. The domain relatedness is modeled by the difference of the compositional weights of k^d_{cks} and k^s_{cks}. Thirdly, the positive semi-definiteness of Trk_ω is not guaranteed yet. To obtain a PSD Trk_ω, we need to carefully design the way of coupling k^d_{cks} and k^s_{cks}, specifically their compositional weights.

5.2 Architecture of Trk_ω

Trk_ω starts with a set of primitive kernels. Afterwards, linear layer and product layer stack alternately. The k^d_{cks} and k^s_{cks} share all the product layers, but differ on the linear layers. Figure 1 gives an example of Trk_ω architecture.

The first layer of Trk_ω consists of several primitive kernels. Every primitive kernel is characterized by some learnable hyper-parameters. Note that we utilize multiple copies of every primitive kernel to enable diverse parametrization. The linear layer of Trk_ω is a fully-connected layer with two sets of compositional weights, one for k^s_{cks} and another for k^d_{cks}. This is different from NKN that only has one set of compositional weights in every linear layer. Moreover, different from NKN that simply uses non-negative weights to ensure the positive semi-definiteness, Trk_ω needs a more complex relationship of the weights between k^s_{cks} and k^d_{cks} to do so. Simply using non-negative weights for k^s_{cks} and k^d_{cks} cannot give a PSD Trk_ω. The product layer introduces multiplications to Trk_ω. The connectivity pattern is fixed and there is no trainable parameters in the product layers. Finally, we also include a nonlinear activation function for every layer. Following [56], we use the exponential function on the output of the linear or product layers.

5.3 Positive Semi-definiteness of Trk_ω

To ensure Trk_ω PSD, we propose a sufficient condition given by Theorem 2.

Theorem 2. A transfer neural kernel network Trk_ω consists of n_k primitive kernels $(k_1,...,k_{n_k})$, n_l linear layers, and n_p product layers. The l-*th* $(l=1,2,...,n_l)$ linear layer includes a non-negative weight matrix $\mathbf{W}^{s_l} \in \mathbb{R}^{n^l_I \times n^l_O}$ for k^s_{cks}, and a weight matrix $\mathbf{W}^{d_l} \in \mathbb{R}^{n^l_I \times n^l_O}$ for k^d_{cks}, where n^l_I and n^l_O are the number of input and output units for the l-th linear layer, respectively. Such a Trk_ω is PSD if for $l=1,2,...,n_l$, $|w^{d_l}_{ij}| \le w^{s_l}_{ij}$ with $i=1,2,...,n^l_I$ and $j=1,2,...,n^l_O$ where $w^{d_l}_{ij}$ $(w^{s_l}_{ij})$ is the element of $\mathbf{W}^{d_l}$ $(\mathbf{W}^{s_l})$ in the i-th row and j-th column.

Proof 2. To prove Trk_ω is a PSD kernel under the above condition, we first prove every unit of the network is a PSD kernel. Then the entire network is PSD according to the composition rules of kernel. We start from the first linear layer, which is a linear combination of the primitive kernels. The number of input units is equal to that of the primitive kernels, i.e., $n^1_I = n_k$. For every output unit $k^1_j, j=1,...,n^1_O$, it has the following form:

$$k^1_j(\mathbf{x},\mathbf{x}') = \begin{cases} \sum_{i=1}^{n^1_I} w^{d_1}_{ij} k_i(\mathbf{x},\mathbf{x}'), & \delta(\mathbf{x},\mathbf{x}')=0, \\ \sum_{i=1}^{n^1_I} w^{s_1}_{ij} k_i(\mathbf{x},\mathbf{x}'), & \delta(\mathbf{x},\mathbf{x}')=1. \end{cases} \tag{10}$$

The Gram matrix of eq. (10) is:

$$\mathbf{K}^1_j = \begin{bmatrix} \sum_{i=1}^{n^1_I} w^{s_1}_{ij}\mathbf{K}^i_{\mathcal{SS}} & \sum_{i=1}^{n^1_I} w^{d_1}_{ij}\mathbf{K}^i_{\mathcal{ST}} \\ \sum_{i=1}^{n^1_I} w^{d_1}_{ij}\mathbf{K}^i_{\mathcal{TS}} & \sum_{i=1}^{n^1_I} w^{s_1}_{ij}\mathbf{K}^i_{\mathcal{TT}} \end{bmatrix},$$

with

$$\mathbf{K}^i = \begin{bmatrix} \mathbf{K}^i_{\mathcal{SS}} & \mathbf{K}^i_{\mathcal{ST}} \\ \mathbf{K}^i_{\mathcal{TS}} & \mathbf{K}^i_{\mathcal{TT}} \end{bmatrix}$$

is the kernel matrix calculated by the primitive kernel k_i over the input source and target data. Since when $w^{s_1}_{ij} = 0$, $w^{d_1}_{ij}$ must be 0 $(|w^{d_l}_{ij}| \le w^{s_l}_{ij})$, we can rewrite:

$$\mathbf{K}^1_j = \sum\nolimits_{i=1}^{n^1_I} w^{s_l}_{ij} \begin{bmatrix} \mathbf{K}^i_{\mathcal{ST}} & w^{d_l}_{ij}/w^{s_l}_{ij}\mathbf{K}^i_{\mathcal{ST}} \\ w^{d_l}_{ij}/w^{s_l}_{ij}\mathbf{K}^i_{\mathcal{TS}} & \mathbf{K}^i_{\mathcal{TT}} \end{bmatrix}.$$

We now consider:

$$\begin{aligned} \mathbf{N} &= \mathbf{K}^i_{\mathcal{TT}} - (w^{d_l}_{ij}/w^{s_l}_{ij})^2 (\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T} (\mathbf{K}^i_{\mathcal{SS}})^{-1} \mathbf{K}^i_{\mathcal{ST}} \\ &= [1-(w^{d_l}_{ij}/w^{s_l}_{ij})^2]\mathbf{K}^i_{\mathcal{TT}} \\ &\quad + (w^{d_l}_{ij}/w^{s_l}_{ij})^2 [\mathbf{K}^i_{\mathcal{TT}} - (\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T} (\mathbf{K}^i_{\mathcal{SS}})^{-1} \mathbf{K}^i_{\mathcal{ST}}]. \end{aligned}$$

Since $\mathbf{K}^i$ is PSD, according to Schur complement theorem [59] we have:

$$\mathbf{K}^i_{\mathcal{TT}} \succeq \mathbf{0} \text{ and } \mathbf{K}^i_{\mathcal{TT}} - (\mathbf{K}^i_{\mathcal{ST}})^\mathsf{T} (\mathbf{K}^i_{\mathcal{SS}})^{-1} \mathbf{K}^i_{\mathcal{ST}} \succeq \mathbf{0}.$$

Furthermore, $[1-(w^{d_l}_{ij}/w^{s_l}_{ij})^2] \ge 0$ since $|w^{d_l}_{ij}| \le w^{s_l}_{ij}$. We then derive: $\mathbf{N} \succeq \mathbf{0}$. Considering $\mathbf{K}^i_{\mathcal{SS}} \succeq \mathbf{0}$ and $w^{s_l}_{ij} \ge 0$, based on Schur complement theorem [59], we can derive: $\mathbf{K}^1_j \succeq \mathbf{0}$. Equivalently, k^1_j is a valid kernel.

Since the above proof is applicable for every $k_j^1, j = 1, ..., n_O^1$, we obtain that all the output units of the first linear layer are PSD. These output units are the input units for the next product layer. According to the composition rules of kernel, we induce that all the output units of the product layer are also PSD. Then these outputs are taken as the inputs for the next linear layer. The similar proof can be applied to the following alternate linear and product layers. To conclude, we can prove that all the units of *Trk*$_\omega$ are PSD when $|w_{ij}^{d_l}| \leq w_{ij}^{s_l}$ for $l = 1, 2, ..., n_l$, $i = 1, 2, ..., n_I^l$ and $j = 1, 2, ..., n_O^l$, and thus *Trk*$_\omega$ is PSD under this condition.

Note that we omit the exponential operation for every unit in the proof since it does not change the positive semi-definiteness of a unit. The proof of PSD k_j^1 is the same as that of sufficient condition in Theorem 1. To be complete, we still present the full proof in Theorem 2.

We now look into the semantic interpretation of Theorem 2. Given a linear layer, an output unit is a linear combination of several input kernels, which is exactly a *Trk*$_{\alpha\beta}$. The conclusion of Theorem 2 on each node of linear layers also conforms to that of Theorem 1, and thus similar interpretation on Theorem 1 can be applied here. More specifically, the weight represents the contribution of an input kernel to the output unit, and thus implies how much knowledge is exploited from this input kernel. Theorem 2 shows that when the weights of k_{cks}^d do not go beyond that of k_{cks}^s, *Trk*$_\omega$ is PSD. This is reasonable as k_{cks}^s is used to calculate the covariance of two instances from the same domain, and thus its weights imply the maximum transfer capacity.

5.4 General Form of *Trk*$_\omega$ and Discussions

The above discussion briefly shows the relationship between *Trk*$_{\alpha\beta}$ and *Trk*$_\omega$. Herein, we investigate the more general connection between the two transfer kernels. Similar as Section 4.3, we are interested in the general form from which *Trk*$_\omega$ is instantiated. To do so, we propose the following general form of transfer kernel, called *polynomial product form transfer kernel*:

$$Trk_{\mathbf{PP}}\Big((\mathbf{x}, \mathbf{d}), (\mathbf{x}', \mathbf{d}')\Big) = \sum\nolimits_t k_t^{\mathcal{D}}(\mathbf{d}, \mathbf{d}') \prod\nolimits_j k_j^{\mathcal{X}}(\mathbf{x}, \mathbf{x}')^{p_{tj}}. \tag{11}$$

We now compare Eqs. (9) and (11). It can be seen that, *Trk*$_{\mathbf{LP}}$ is the linear combination of primitive kernels while *Trk*$_{\mathbf{PP}}$ is the linear combination of the polynomial of primitive kernels. *Trk*$_{\mathbf{PP}}$ can be degenerated to *Trk*$_{\mathbf{LP}}$ if we set $p_{tj} = 1$ if and only if $j = t$ otherwise $p_{tj} = 0$. Different from *Trk*$_{\mathbf{LP}}$ that only models linear relationship of primitive kernels, *Trk*$_{\mathbf{PP}}$ takes the nonlinearity into account.

However, *Trk*$_{\mathbf{PP}}$ can be reformulated as similar as *Trk*$_{\mathbf{LP}}$ if we use another intermediate kernel as follows:

$$\hat{k}_t^{\mathcal{X}}(\mathbf{x}, \mathbf{x}') = \prod\nolimits_{j=1}^{n_k} k_j^{\mathcal{X}}(\mathbf{x}, \mathbf{x}')^{p_{tj}}.$$

In terms of functionality, $k_t^{\mathcal{D}}$ plays the exactly same role in both *Trk*$_{\mathbf{PP}}$ and *Trk*$_{\mathbf{LP}}$, that is, it models the relatedness of domains on either $\hat{k}_t^{\mathcal{X}}$ or $k_t^{\mathcal{X}}$. The major difference between *Trk*$_{\mathbf{PP}}$ and *Trk*$_{\mathbf{LP}}$ is on the expressive power, where the former is much more powerful and flexible than the latter.

We now look into *Trk*$_\omega$. It is an instantiation of *Trk*$_{\mathbf{PT}}$. To be clear, we can rewrite *Trk*$_\omega$ as follows:

$$Trk_\omega(\mathbf{x}, \mathbf{x}') = \begin{cases} \sum_{t=1}^T \omega_t^d \prod_{j=1}^{n_k} k_j(\mathbf{x}, \mathbf{x}')^{p_{tj}}, \delta(\mathbf{x}, \mathbf{x}') = 0, \\ \sum_{t=1}^T \omega_t^s \prod_{j=1}^{n_k} k_j(\mathbf{x}, \mathbf{x}')^{p_{tj}}, \delta(\mathbf{x}, \mathbf{x}') = 1, \end{cases} \tag{12}$$

where each of k_{cks}^s and k_{cks}^d is represented as a weighted polynomial of primitive kernels. The number of polynomials T and the degree p_{tj} are determined by the architecture of *Trk*$_\omega$. The coefficients $\{\omega_t^s, \omega_t^d\}_{t=1}^T$ can be inferred from the weight matrices of the linear layers. This reformulation shows that k_{cks}^s and k_{cks}^d lie in the same kernel function space expanded by the polynomials of the primitive kernels, but they perform differently on these base kernels. The coefficients $\{\omega_t^s\}_{t=1}^T$ and $\{\omega_t^d\}_{t=1}^T$ fully characterize k_{cks}^s and k_{cks}^d, and their element-wise distance indicates the domain relatedness on the corresponding polynomial of kernels.

We further elaborate the relationship of *Trk*$_\omega$ with NKN. Clearly, the intra-domain compositional kernel structure k_{cks}^s is a NKN with $\{\omega_t^s \geq 0\}_{t=1}^T$. Based on the Theorem 3 in [56], k_{cks}^s can approximate arbitrary stationary kernels, which again verifies the expressive power of *Trk*$_\omega$. Theorem 2 also shows us ω_t^d can take negative values. In this case, the individual inter-domain compositional kernel structure k_{cks}^d is not a NKN and not PSD. However, by coupling with k_{cks}^s, the whole *Trk*$_\omega$ becomes a PSD transfer kernel. This shows that *Trk*$_\omega$ is not a simple combination of two NKNs, but a specialized design for the transfer purpose.

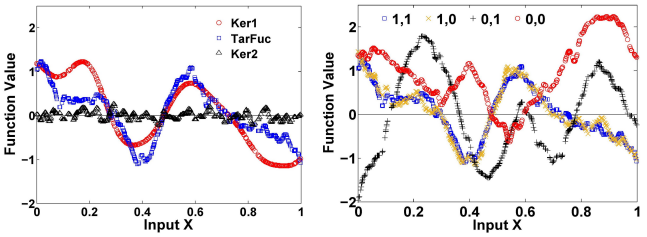
Finally, we discuss more on the PSD issue of *Trk*$_\omega$. With Theorem 2, we only obtain a sufficient condition for PSD *Trk*$_\omega$. A complete (sufficient and necessary) condition makes *Trk*$_\omega$ more theoretically solid. Fortunately, with Eq. (12) and Theorem 1, we can readily obtain such a complete condition, that is, $|\omega_t^d| \leq \omega_t^s$ and $\omega_t^s \geq 0$ with $t = 1, ..., T$. However, as both ω_t^d and ω_t^s involve the complex neural network calculations, it is not friendly to be used in the following *GP* learning. In contrast, although Theorem 2 is only a sufficient condition, it still gives a PSD *Trk*$_\omega$, and it can be readily incorporated in the following *GP* optimization.

5.5 Transfer GP Learning with *Trk*$_\omega$

As the overall architecture of *Trk*$_\omega$ is differentiable, the *TrGP* using *Trk*$_\omega$, i.e., *TrGP*$_\omega$, is trainable using gradient-based optimization. All the parameters include three categories: (1) parameters in the primitive kernels, (2) weights matrices of the linear layers, and (3) the noise variances of $\mathcal{S}$ and $\mathcal{T}$. We group all the parameters together, denoted as $\mathbf{\Theta}$, and learn them jointly. The learning procedure of *TrGP*$_\omega$ is similar as discussed in Section 3.4. Note that we also need to consider Theorem 2 in the learning. Regarding the non-negative weights of k_{nkn}^s, we use $f(x) = log(1 + exp(x))$ to enforce the nonnegativity constraint. Regarding the inequality constraints on the weights of k_{nkn}^s and k_{nkn}^d, we can check the fulfilment of these constraints in each epoch, and map the violated $w_{ij}^{d_l}$ to a value that is close to the feasible boundary, as *TrGP*$_{\alpha\beta}$ does. Alternatively, we can further constrain $w_{ij}^{d_l}$ as non-negative to enforce k_{nkn}^d also a NKN. Then, we satisfy $|w_{ij}^{d_l}| \leq w_{ij}^{s_l}$ by setting $w_{ij}^{s_l} = w_{ij}^{d_l} + {\epsilon_{ij}^l}^2$. In this case, $w_{ij}^{d_l}$ and ϵ_{ij}^l are the optimization variables.

Table 2
Comparison results of $TrGP_{mk}$ with $TrGP_{\lambda}$.

Problem (sim_1,sim_2)	Learned Relatedness $TrGP_{\alpha\beta}$	Learned Relatedness $TrGP_{\lambda}$	RMSE $TrGP_{\alpha\beta}$	RMSE $TrGP_{\lambda}$	Problem (sim_1,sim_2)	Learned Relatedness $TrGP_{\alpha\beta}$	Learned Relatedness $TrGP_{\lambda}$	RMSE $TrGP_{\alpha\beta}$	RMSE $TrGP_{\lambda}$	Problem (sim_1,sim_2)	Learned Relatedness $TrGP_{\alpha\beta}$	Learned Relatedness $TrGP_{\lambda}$	RMSE $TrGP_{\alpha\beta}$	RMSE $TrGP_{\lambda}$
(0,0)	(0.057,0.008)	**0.042**	0.279	0.280	(0.4,0)	(0.375,0.039)	0.261	0.235	0.267	(0.8,0)	(0.793,0.032)	0.468	0.191	0.208
(0,0.2)	(0.087,0.174)	0.114	0.264	0.266	(0.4,0.2)	(0.438,0.172)	0.301	0.220	0.236	(0.8,0.2)	(0.807,0.254)	0.543	0.152	0.206
(0,0.4)	(0.015,0.443)	0.080	0.254	0.280	(0.4,0.4)	(0.420,0.446)	**0.433**	0.175	0.176	(0.8,0.4)	(0.757,0.426)	0.556	0.125	0.156
(0,0.6)	(0.080,0.623)	0.155	0.216	0.300	(0.4,0.6)	(0.383,0.644)	0.539	0.124	0.146	(0.8,0.6)	(0.792,0.611)	0.678	0.093	0.105
(0,0.8)	(0.079,0.805)	0.227	0.171	0.293	(0.4,0.8)	(0.410,0.763)	0.515	0.117	0.151	(0.8,0.8)	(0.798,0.800)	**0.806**	0.060	0.061
(0,1)	(0.015,1)	0.838	0.122	0.152	(0.4,1)	(0.686,1)	0.907	0.108	0.137	(0.8,1)	(0.768,1)	0.951	0.037	0.083
(0.2,0)	(0.247,0.056)	0.136	0.247	0.271	(0.6,0)	(0.644,0.078)	0.498	0.212	0.247	(1,0)	(0.999,0.062)	0.826	0.074	0.120
(0.2,0.2)	(0.190,0.231)	**0.198**	0.237	0.237	(0.6,0.2)	(0.611,0.173)	0.488	0.199	0.230	(1,0.2)	(1,0.094)	0.801	0.076	0.153
(0.2,0.4)	(0.195,0.364)	0.210	0.221	0.225	(0.6,0.4)	(0.580,0.413)	0.537	0.164	0.173	(1,0.4)	(0.999,0.364)	0.984	0.066	0.081
(0.2,0.6)	(0.214,0.616)	0.280	0.181	0.222	(0.6,0.6)	(0.578,0.565)	**0.580**	0.111	0.110	(1,0.6)	(1,0.457)	0.937	0.068	0.075
(0.2,0.8)	(0.2,0.763)	0.516	0.162	0.202	(0.6,0.8)	(0.583,0.760)	0.849	0.109	0.153	(1,0.8)	(1.0.812)	0.981	0.041	0.048
(0.2,1)	(0.184,1)	0.875	0.112	0.188	(0.6,1)	(0.741,1)	0.969	0.025	0.039	(1,1)	(0.995,0.999)	**1.000**	0.001	0.001
GP_{Tar} RMSE	0.296													



(a) (b)

Figure 2. (a) Data points generated from *GP*s with distinct exponential kernels; red '○'s from $k_1 = 0.1 * exp(-\frac{x^2}{2*1^2})$; black '△'s from $k_2 = 0.01 * exp(-\frac{x^2}{2*0.1^2})$; blue '□'s from $k = k_1 + k_2$. (b) Source points from four special cases $(\beta_1^{gt}, \beta_2^{gt}) \in \{(1,1),(1,0),(0,1),(0,0)\}$. Notice that for (1,1) the points precisely follow the target function values in (a).

We also discuss the learning complexity of $TrGP_{\omega}$. The usage of Trk_{ω} actually does not bring extra computation cost to the model learning. Considering a Trk_{ω} with m connections, the computational cost of the forward pass is $O(n^2m)$. Note that the computational cost of the kernel matrix inversions is $O(n^3)$. By adequately designing the architecture of Trk_{ω} (as what we have done in the experiments), the number of parameters, m, is usually much smaller than that of training data, n (in our experimental studies, Trk_{ω} has only tens of parameters, but the training points are in hundreds and thousands). In this case, the computational cost of $TrGP_{\omega}$ is still dominated by the kernel matrix inversions, which is the same as the standard *GP*.

6 Experimental Studies

In this section, we aim to empirically assess the capability of $TrGP_{\alpha\beta}$ and $TrGP_{\omega}$ on domain relatedness capturing and evaluate their predictive effectiveness. Root mean squared error (RMSE) is used as the evaluation metric. We use one synthetic dataset and four real-world datasets.

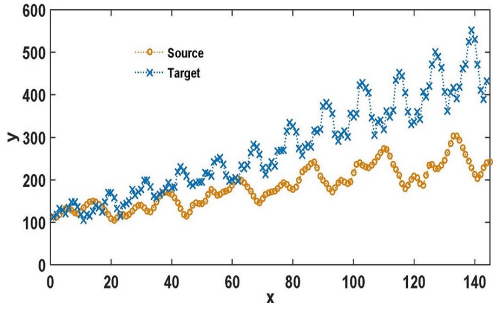
6.1 Domain Relatedness Capture

We use a set of synthetic datasets to verify the explicit domain relateness modelling. Considering that the domain covariance $k_t^{\mathcal{D}}$ plays the exactly same role in both $Trk_{\alpha\beta}$ and Trk_{ω}, we focus on $Trk_{\alpha\beta}$ to avoid neural network calculation. The synthetic dataset is constructed as follows.

• **Synthetic data**. We build on a $Trk_{\alpha\beta}$ with predefined primitive kernels k_is, and ground-truth combination coefficients α_i^{gt}s and β_i^{gt}s. For a set of inputs $\mathbf{X}$, we obtain target function values $\mathbf{f}_{tar}$ from a *GP* with zero mean and covariance matrix given by $\sum_i \alpha_i^{gt} k_i$. We then sample source function values $\mathbf{f}_{src}$ from $p(\mathbf{f}_{src}|\mathbf{f}_{tar}, \mathbf{X})$. By varying α_i^{gt}s and β_i^{gt}s, we can control the ground-truth domain relatedness $sim_i = \beta_i^{gt}/\alpha_i^{gt}$ on each primitive kernel k_i.

We construct a broad spectrum of transfer problems upon the synthetic data in order to analyze the distance between the learned relatedness $\beta_i^{opt}/\alpha_i^{opt}$ and the ground-truth relatedness sim_i for all the primitive kernels. Specifically, we generate 500 inputs $\mathbf{X} \in \mathbb{R}^{500\times 1}$ and predefine two primitive kernels (k_1 and k_2) using two exponential covariance functions with different amplitudes and length scales. As shown in Fig. 2(a), k_1 draws 'big trends' while k_2 depicts 'small perturbations'. Next, we generate target and source function values based on $Trk_{\alpha\beta}$. We fix $\alpha_1^{gt} = 1$ and $\alpha_2^{gt} = 1$, but vary β_1^{gt} and β_2^{gt} in a range set $\mathcal{V} = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Thus, the corresponding sim_is also vary in $\mathcal{V}$. We then generate $\mathbf{f}_{tar}$ from a *GP* using $\alpha_1^{gt} k_1 + \alpha_2^{gt} k_2$. Fig. 2(a) shows that the sampled target points possess the characteristics of both kernels. Given $\mathbf{f}_{tar}$, for each combination $(\beta_1^{gt}, \beta_2^{gt})$ we generate source points using $p(\mathbf{f}_{src}|\mathbf{f}_{tar}, \mathbf{X})$. Fig. 2(b) shows four special cases $(\beta_1^{gt}, \beta_2^{gt}) \in \{(1,1),(1,0),(0,1),(0,0)\}$. In total, we can generate $C_6^1 \times C_6^1 = 36$ different source functions. For each of them, we use 380 points as the source train data, 20 as the target train data, and the rest 100 as the target test data. Note that the function values of source train data vary across the 36 problems function, while the values of the target train and test stay the same.

• **Relatedness analysis**. Table 2 shows the learned relatedness tuple ($\beta_1^{opt}/\alpha_1^{opt}$, $\beta_2^{opt}/\alpha_2^{opt}$) and the transfer performance. The results of two other *GP* models are also included: GP_{Tar}, which uses target train data only, and $TrGP_{\lambda}$ which adopts transfer kernel Trk_{λ}. Columns 1, 6, and 11 show the 36 problems with the ground-truth relatedness (sim_1, sim_2). Columns 2, 7, and 12 show ($\beta_1^{opt}/\alpha_1^{opt}$,$\beta_2^{opt}/\alpha_2^{opt}$) learned by $TrGP_{mk}$, and columns 3, 8, and 13 list out the single relatedness (i.e., λ) learned by $TrGP_{\lambda}$. The rest columns show the transfer performance (RMSE) of $TrGP_{mk}$ and $TrGP_{\lambda}$. Paying attention to the domain relatedness capture, Table 2 reveals that, overall, $TrGP_{mk}$ successfully recovers the sim_is on different problems, compared with ($\beta_1^{opt}/\alpha_1^{opt}$, $\beta_2^{opt}/\alpha_2^{opt}$) and



(a) The source and target data with similar patterns.

(sim_1, sim_2). This outcome indicates the effectiveness of *TrGP*$_{mk}$ in capturing heterogeneous domain relatedness. On the other side, the single coefficient λ learned by *TrGP*$_{\lambda}$ can accurately reflect the similarity value for the problems where $sim_1 = sim_2$ (in bold in Table 2). For the problems $sim_1 \neq sim_2$, however, the λ is usually between sim_1 and sim_2, and so can only describe a compromised relatedness.

• **Transfer performance**. We now turn to the transfer performance. Firstly, from Table 2 we observe that both *TrGP*$_{mk}$ and *TrGP*$_{\lambda}$ can achieve adaptive transfer as they are not worse than GP_{Tar}. Furthermore, we notice that the RMSE of both models tends to decrease as the relatedness of two domains increases, which is aligned with the cases where the models are transferring more knowledge. This further supports the adaptiveness of both methods. Also, we find that both models perform similar in problems where $sim_1 = sim_2$. However, for the other cases *TrGP*$_{mk}$ consistently outperforms *TrGP*$_{\lambda}$, suggesting that the former can better deal with more complex tasks than the latter.

6.2 Time Series Extrapolation

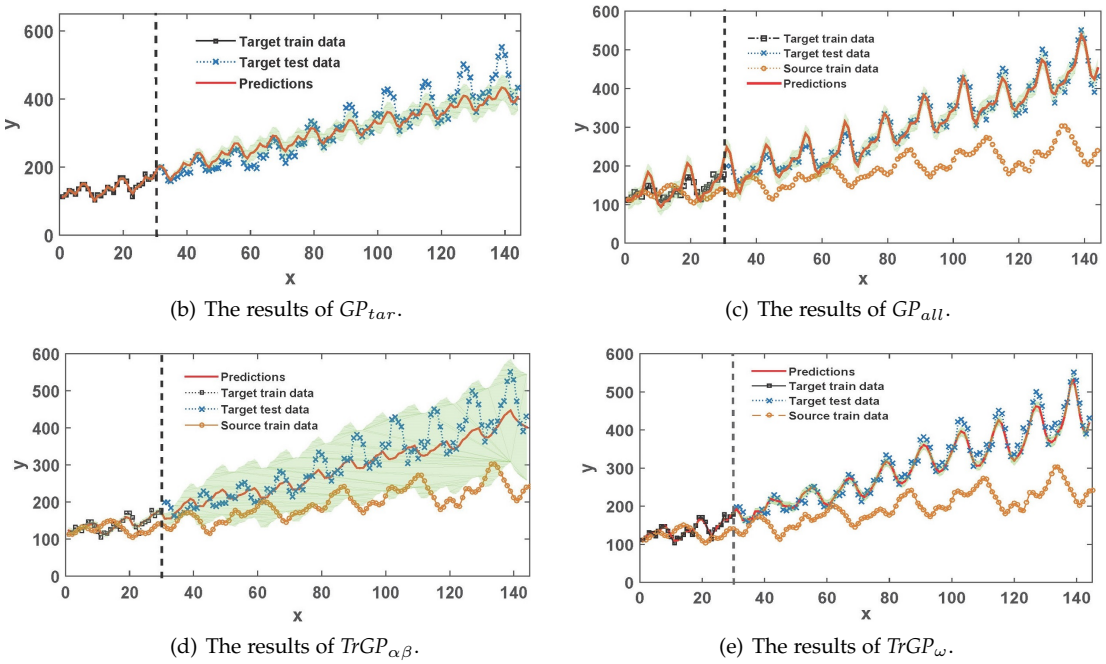
Time series data extrapolation is a popular regression task. In this section, we conduct experiment on a time series dataset [57] to compare the transfer capacity of *TrGP*$_{\alpha\beta}$ and *TrGP*$_{\omega}$ on data extrapolation. The dataset includes the airline passenger volume data, and we construct two sets of data with different patterns. The task is to predict the future data patterns. As shown in Figure 3(a), both the source and target data include the periodic and increasing waves. Note that a wave is a curve with the approximate contour '$\wedge$'. Specifically, the wave appears periodically along a slope. The amplitude of the wave is also growing with the period. However, it is worth noting that the pattern characteristics of the waves are significantly different between the two domains. The source data has longer period, flatter slope, and slower amplitude growth than the the target data. Our objective is to utilize sufficient source data and limited target data to extrapolate the subsequent unseen patterns for the target. In the experiment, we take the first 30 target points and 288 source points as the training data. The rest target data (from 31 to 144) are used as the test points.

• **Configuration**. We design a *Trk*$_{\omega}$ with three linear layers and two product layers for this problem. We use two LIN kernels, two RBF kernels, two PER kernels, and two RQ kernels as primitive kernels. For this *Trk*$_{\omega}$, the maximum degree of the polynomials of primitive kernels is 4, and the number of the polynomials of degree 4 is 330 (C_{8+4-1}^{8-1}). Such a *Trk*$_{\omega}$ can model complex data characteristics by diverse polynomials of primitive kernels, e.g., LIN^4 models the long-term smooth trend, PER^4 models the periodicity, and LIN$\times$PER$\times$RBF$\times$RQ models the periodic variation.

For optimization matters of *TrGP*$_{\omega}$, we use 'Adamoptimizer' with the learning rate as 0.001, and set the epoch number as 20000. We compare with several baselines including GP_{tar} that only uses 30 target data to train a *GP* model, GP_{all} that just simply combines the 30 target training data and 288 source training data, and *TrGP*$_{\alpha\beta}$. For GP_{tar} and GP_{all}, we exploit a NKN with the same network structure as *TrGP*$_{\omega}$ but only 1 set of weights as the kernel. For *TrGP*$_{\alpha\beta}$, we use the same set of primitive kernels with *TrGP*$_{\omega}$.

• **Transfer performance**. We plot the predictions of each method in Figure 3. As can be seen from 3(b), GP_{tar} can well fit target training points, but it fails to fit the subsequent target test ones. Since no information is available except for the 30 target training points, the predictions proceed with the target training pattern. However, the amplitude growth pattern is completely lost. Regarding GP_{all} in Figure 3(c), it also fails to do accurate extrapolation on the target test data, and even worse, it fails to fit the target training data. Interestingly, we observe that GP_{all} generates a new pattern. That is, from the first wave, every three waves form a period. In such a period, the amplitude of the wave slowly grows larger, and achieves the largest at the third wave. Afterwards, a new period starts, but the amplitude of the first wave in the new period decreases compared with that of the last wave in the former period. Such a new pattern is a negative result of the brute-force data integration that completely ignores the divergence between different domains. For *TrGP*$_{\alpha\beta}$, Figure 3(d) shows that it yields almost the same results as GP_{tar}. This implies that *TrGP*$_{\alpha\beta}$ does not transfer any knowledge from the source to the target task. The bad performance is mainly due to the deficiency of *TrGP*$_{\alpha\beta}$ on the expressive power. *TrGP*$_{\alpha\beta}$ uses a linear summation of the primitive kernels, and thus fails to capture and transfer the knowledge that are represented by the product of primitive kernels. Moreover, *TrGP*$_{\alpha\beta}$ obtains much larger standard deviation than GP_{tar} since the source data pull down the prediction confidence. Finally, from Figure 3(e), we can see that *TrGP*$_{\omega}$ not only fits the target training data well, but also generates sensible extrapolation. The predictions capture all the patterns including the periodicity, linear increase, and amplitude growth. Meanwhile, the standard deviations are also small, which implies the source data indeed provide help for the target. These results clearly show the superiority of *TrGP*$_{\omega}$ to *TrGP*$_{\alpha\beta}$.

• **Weights analysis**. We further analyze the similarity of the weights directly learned from *TrGP*$_{\omega}$. There are three linear layers in this *Trk*$_{\omega}$, and we focus on the weights of the first linear layer, i.e., the combination weights of primitive kernels. For each node, we calculate the ratio of the weights of k_{cks}^{d} over the corresponding ones of k_{cks}^{s} as the relatedness of the primitive kernels to construct this node. Figure 4 shows the heat map of the learned relatedness on each primitive kernel for the 8 nodes of the first linear layer. The y-axis denotes node index, and the x-axis denotes primitive kernel. Thus, each row represents the learned combination weights of the 8 primitive kernels for one node. In a horizontal view, it can be observed that different nodes have



(b) The results of GP_{tar}. (c) The results of GP_{all}.

(d) The results of $TrGP_{\alpha\beta}$. (e) The results of $TrGP_{\omega}$.

Figure 3. The black squares are the target training data. The yellow circles are the source training data. The blue 'x's are the ground-truth of the target test points. The red line is the prediction mean and the green region represents the standard deviation.

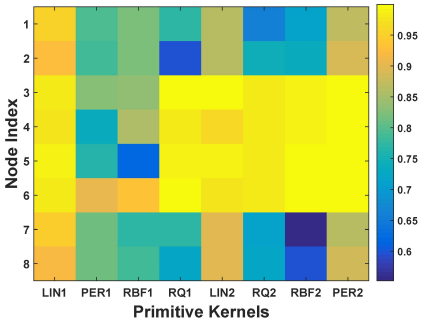


Figure 4. The heat map of the similarity for the first linear layer.

different combinations of primitive kernels. This shows the rich expressive power of Trk_{ω} from a side. In a vertical view, we further find that the two domains are more similar on the linear kernel, but less similar on the periodic kernel. This well corresponds to the source and target patterns as shown in Figure 3(a). Moreover, the relatedness of the same type of primitive kernel can be dramatically different due to different parametrizations, e.g., PER1 vs. PER2.

Except for the analyses on the learned weights of the first layer, we also investigate the weights of the polynomial of primitive kernels, i.e., ω_s and ω_d, which can be inferred from the learned weights of Trk_{ω}. Recall that this Trk_{ω} can express 330 polynomials with degree 4, and we only investigate some representative ones. Specifically, we calculate ω_t^s and ω_t^d for $\text{LIN}_1{}^4$, $\text{PER}_1{}^4$, $\text{RBF}_1{}^4$, $\text{RQ}_1{}^4$, and $\text{LIN}_1{\times}\text{PER}_1{\times}\text{RBF}_1{\times}\text{RQ}_1$. We use the ratio ω_t^d/ω_t^s to represent the relatedness on the polynomial. Table 3 shows the results. The high similarity on $\text{LIN}_1{}^4$ and $\text{RQ}_1{}^4$ correspond to the similar long-term smooth trend and small-scale variations of the source and target functions in Figure 3(a). In contrast, the low similarity on $\text{PER}_1{}^4$ and $\text{RBF}_1{}^4$ reflects

Table 3
Similarity of some polynomial of primitive kernels.

Polynomials	$\text{LIN}_1{}^4$	$\text{PER}_1{}^4$	$\text{RBF}_1{}^4$	$\text{RQ}_1{}^4$	$\text{LIN}_1{\times}\text{PER}_1{\times}\text{RBF}_1{\times}\text{RQ}_1$
Relatedness	0.9130	0.4354	0.4056	0.9453	0.6385

the dissimilar periodicity and variation speed. Regarding the $\text{LIN}_1{\times}\text{PER}_1{\times}\text{RBF}_1{\times}\text{RQ}_1$ that models both the long-time trend and the short-time derivation, the relatedness is an intermediate value. All these results show the capability of Trk_{ω} on capturing the fine-grained domain relatedness.

6.3 Real-world Regression Problems

Finally, we verify the effectiveness of $TrGP_{\alpha\beta}$ and $TrGP_{\omega}$ on four real-world regression applications.

Wine Quality. The wine quality dataset includes red and white wine samples [60]. Physicochemical variables, e.g. volatile acidity, are used as features and the sensory variable is used as label. We use the quality prediction on white wine ($\mathbf{W}$) as the source task and the quality prediction on red wine ($\mathbf{R}$) as the target one. Specifically, $\mathbf{W}$ includes 500 training data points, and $\mathbf{R}$ includes 10 training data points and 500 test data points.

Appliance Energy Prediction. This dataset is used to predict the appliance energy use in different months [61]. The house conditions data, e.g. humidity, monitored by the sensors are used as features, and the amount of appliance energy use is used as label. The data of May (500 for training) is used as the source task, and the data of January (5 for training and 500 for test) is used as the target task.

Air Quality. The air quality dataset [62] contains responses of gas multisensor devices from different seasons. The averaged responses from the metal oxide chemical sensors are features, and the ground-truth provided by a co-located reference certified analyzer is used as label. Data from Summer

and Winter are used for source and target, respectively. The data configuration follows that of wine quality dataset.

UJIIndoorLoc. This dataset covers signals from different buildings of Universitat Jaume I [63]. The signal strength intensity from 520 wireless access points is used as features, and the location is used as label. We use building 1 including 1000 training data as the source, and we use building 2 including 25 training data and 1000 test data as the target.

We design a Trk_ω with two linear layers and one product layer for the regression problems. Two RBF kernels and one linear kernel are used as primitive kernels. We use 'Adamoptimizer' with the learning rate as 0.001 for the training, and set the epoch number as 5000. We run five restarts with different initializations, and take the average root mean squared error (RMSE) as the final result. We compare $TrGP_{\alpha\beta}$ and $TrGP_\omega$ with several baselines including GP_{tar}, GP_{all}, $TrGP_\lambda$ [10], $TrGP_{\lambda_{var}}$ [16], DKL [36], DGP [35], DI-PLS [53], DKT [37], WANN [55], and MGP [38].

The comparison results are shown in Table 4. The best result is highlighted using bold, and the runner-up is highlighted using the bold and italic. From Table 4, we can see that our $TrGP_\omega$ achieves the best on 3 datasets and the second best on 1 dataset, while $TrGP_{\alpha\beta}$ is the runner-up on two datasets. Overall, $TrGP_\omega$ is the clear winner among all the baselines, which shows that $TrGP_\omega$ is a very promising method for transfer regression problems.

We specifically pay attention to the four methods using transfer kernels, that is, $TrGP_\lambda$, $TrGP_{\lambda_{var}}$, $TrGP_{\alpha\beta}$ and $TrGP_\omega$. All the four baselines generally achieve better results than the two naive methods. Among these 4 methods, $TrGP_\omega$ is the best, followed by $TrGP_{\alpha\beta}$, and lastly $TrGP_{\lambda_{var}}$ and $TrGP_\lambda$. The deviation of the transfer performance across these methods is due to the difference of their expressive power and capacity in modeling domain relatedness. $TrGP_\lambda$ uses a single similarity coefficient and a single primitive kernel, which highly constrains its expressive power and transfer capacity. $TrGP_{\lambda_{var}}$ is a variant of $TrGP_\lambda$, and it is only applicable for RBF kernel to ensure the positive semi-definiteness. $TrGP_{\alpha\beta}$ uses a single linear layer, and thus can not express the data characteristics represented by the product of primitive kernels. In contrast, our $TrGP_\omega$ establishes a deep kernel neural network approximating arbitrary stationary kernels.

7 Conclusions and Future Works

In this paper, we work on the challenging transfer regression problems where target domain only has limited labelled data. Our aim is to explicitly model the domain relatedness, based on which adaptively control the knowledge transfer strength across domains. To do so, we focus on the transfer-specified kernels that take domain information into account in covariance calculation. A formal definition of transfer kernel is given, and two advanced general forms of transfer kernel, i.e., $Trk_{\mathbf{LP}}$ and $Trk_{\mathbf{PP}}$, are proposed. We further instantiated two transfer kernels $Trk_{\alpha\beta}$ and Trk_ω based on $Trk_{\mathbf{LP}}$ and $Trk_{\mathbf{PP}}$, respectively. Specifically, $Trk_{\alpha\beta}$ exploits multiple kernel learning strategy to capture the fine-grained domain relatedness on primitive kernels with a linear combination relation. Trk_ω further considers nonlinearity, and is represented by a neural network modelling domain relatedness in a layer-to-layer manner. For each of them, we also propose a condition that not only validates the positive semi-definiteness but also assigns semantic interpretations on the learned domain relatedness. Such conditions can be further used as computationally cheap constraints in the corresponding *TrGP* learning. Empirical evaluations show their effectiveness.

We also remark that all *GP* models studied in this paper need to calculate the inverse of the covariance matrix, which is computationally expensive if the number of input data is extremely large. Although this issue can be alleviated with powerful computing resources, we realize that there are some studies, e.g., [64], [65], working on this issue through computationally cheap approximations of the covariance matrix. However, directly applying these methods to *TrGP* is infeasible as approximations may eliminate the transfer capacity. How to effectively incorporate these works into the transfer framework is one of our future research directions.

Acknowledgments

The authors thank Dr. Ramon Sagarna and Dr. Leong Tze Yun for the comments on this paper. This research/project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative, and the Ministry of Education, Singapore under its MOE Academic Research Fund Tier 2 (STEM RIE2025 Award T2EP20220-0016). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, and the Ministry of Education, Singapore.

References

[1] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[2] P. Wei, R. Sagarna, Y. Ke, Y.-S. Ong, and C.-K. Goh, "Source-target similarity modelings for multi-source transfer gaussian process regression," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3722–3731.

[3] P. Wei, X. Qu, Y. Ke, T.-Y. Leong, and Y. S. Ong, "Adaptive knowledge transfer based on transfer neural kernel network," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1485–1493.

[4] A. Tan Wei Min, R. Sagarna, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Knowledge transfer through machine learning in aircraft design," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 48–60, 2017.

[5] X. Xie, G. Liu, Q. Cai, P. Wei, and H. Qu, "Multi-source sequential knowledge regression by using transfer rnn units," *Neural Networks*, vol. 119, pp. 151–161, 2019.

[6] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning under covariate shift." *Journal of Machine Learning Research*, vol. 10, no. 9, 2009.

[7] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He, "Deep subdomain adaptation network for image classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[8] W.-S. Chu, F. De la Torre, and J. F. Cohn, "Selective transfer machine for personalized facial expression analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 3, pp. 529–545, 2016.

[9] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 770–787, 2009.

Table 4
Comparison results (RMSE) on four real-world regression tasks.

Methods	Wine Quality	Air Quality	Engery Usage	Building Location	Average
GP_{tar}	1.7720±0.0563	0.9329±0.0819	1.8939±0.0154	0.2163±0.0531	1.2038±0.0517
GP_{all}	1.2634±0.0156	1.0134±0.0059	1.9030±0.0089	0.1177±0.0046	1.0744±0.0088
DI-PLS	0.7838±0.0033	0.8632±0.0099	1.1969±0.0265	0.1666±0.0040	0.7526±0.0109
WANN	0.7724±0.0122	1.0545±0.0012	1.3635±0.0329	0.1412±0.0128	0.8329±0.0148
DKL	1.0322±0.0317	1.1491±0.0481	1.3137±0.0293	0.1246±0.0205	0.9049±0.0324
DGP	***0.7689±0.0621***	1.0289±0.0339	1.1892±0.0536	0.1667±0.0203	0.7884±0.0425
DKT	0.8318±0.0154	1.1530±0.0999	**1.1125±0.0167**	0.1404±0.0179	0.8094±0.0375
MGP	0.8733±0.0043	0.9678±0.0025	1.1940±0.0177	0.1244±0.0012	0.7899±0.0064
$TrGP_{\lambda}$	1.0672±0.0236	0.9246±0.0522	1.5120±0.0108	0.1339±0.0073	0.9094±0.0235
$TrGP_{\lambda_{var}}$	1.0635±0.0178	0.9180±0.0437	1.4996±0.0069	0.1296±0.0025	0.9027±0.0177
$TrGP_{\alpha,\beta}$	0.8395±0.0125	***0.8172±0.0268***	1.2389±0.0067	***0.1023±0.0421***	***0.7495±0.0220***
$TrGP_{\omega}$	**0.7233±0.0021**	**0.7915±0.0046**	***1.1660±0.0027***	**0.0963±0.0026**	**0.6943±0.0030**

[10] B. Cao, S. J. Pan, Y. Zhang, D.-Y. Yeung, and Q. Yang, "Adaptive transfer learning," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010, pp. 407–712.

[11] D. Pardoe and P. Stone, "Bo- osting for regression transfer," in *International Conference on Machine Learning*, 2010, pp. 863–870.

[12] B. Liu, Y. Xiao, and Z. Hao, "A selective multiple instance transfer learning method for text categorization problems," *Knowledge-Based Systems*, vol. 141, pp. 178–187, 2018.

[13] C. Williams and E. Rasmussen, "Gaussian processes for regression," in *Advances in neural information processing systems*, 1996, pp. 514–520.

[14] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task gaussian process prediction," in *Annual Conference on Neural Information Processing Systems*, 2008, pp. 153–160.

[15] K. M. Chai, "Generalization errors and learning curves for regression with multi-task gaussian processes," in *Advances in neural information processing systems*, 2009, pp. 279–287.

[16] N. Wagle and E. W. Frew, "Forward adaptive transfer of gaussian process regression," *Journal of Aerospace Information Systems*, vol. 14, no. 4, pp. 214–231, 2017.

[17] B. Da, Y.-S. Ong, A. Gupta, L. Feng, and H. Liu, "Fast transfer gaussian process regression with large-scale sources," *Knowledge-Based Systems*, vol. 165, pp. 208–218, 2019.

[18] J. Mercer, "Xvi. functions of positive and negative type, and their connection the theory of integral equations," *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, vol. 209, no. 441-458, pp. 415–446, 1909.

[19] H. Wackernagel, *Multivariate Geostatistics: An Introduction with Applications*. Springer Science & Business Media, 1998.

[20] J. McFarland, S. Mahadevan, V. Romero, and L. Swiler, "Calibration and uncertainty analysis for computer simulations with multivariate output," *AIAA journal*, vol. 46, no. 5, pp. 1253–1265, 2008.

[21] R. Dürichen, M. A. Pimentel, L. Clifton, A. Schweikard, and D. A. Clifton, "Multitask gaussian processes for multivariate physiological time-series analysis," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 1, pp. 314–322, 2014.

[22] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings, "Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*. IEEE, 2008, pp. 109–120.

[23] T. Cohn and L. Specia, "Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 32–42.

[24] T. C. Haas, "Multivariate geostatistics: An introduction with applications," *Journal of the American Statistical Association*, vol. 91, no. 435, pp. 1375–1377, 1996.

[25] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[26] M. A. Alvarez, L. Rosasco, N. D. Lawrence *et al.*, "Kernels for vector-valued functions: A review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[27] R. Lark and A. Papritz, "Fitting a linear model of coregionalization for soil properties using simulated annealing," *Geoderma*, vol. 115, no. 3-4, pp. 245–260, 2003.

[28] A. Tong and J. Choi, "Discovering latent covariance structures for multiple time series," in *International Conference on Machine Learning*, 2019, pp. 6285–6294.

[29] N. J. Higham, *Analysis of the Cholesky decomposition of a semi-definite matrix*. Oxford University Press, 1990.

[30] A. Melkumyan and F. Ramos, "Multi-kernel gaussian processes," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011, pp. 1408–1403.

[31] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 193–200.

[32] S. Al-Stouhi and C. Reddy, "Adaptive boosting for transfer learning using dynamic updates," *Machine Learning and Knowledge Discovery in Databases*, pp. 60–75, 2011.

[33] H. Zhao, S. Zhang, G. Wu, J. M. F. Moura, J. P. Costeira, and G. J. Gordon, "Adversarial multiple source domain adaptation," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 8559–8570.

[34] S. Zhao, B. Li, X. Yue, Y. Gu, P. Xu, R. Hu, H. Chai, and K. Keutzer, "Multi-source domain adaptation for semantic segmentation," in *Advances in Neural Information Processing Systems*, 2019, pp. 7285–7298.

[35] H. Salimbeni and M. Deisenroth, "Doubly stochastic variational inference for deep gaussian processes," in *Advances in Neural Information Processing Systems*, 2017, pp. 1–12.

[36] A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing, "Stochastic variational deep kernel learning," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2586–2594, 2016.

[37] M. Patacchiola, J. Turner, E. J. Crowley, and A. Storkey, "Bayesian meta-learning for the few-shot setting via deep kernels," in *Advances in Neural Information Processing Systems*, 2020, pp. 16 108–16 118.

[38] P. Moreno-Muñoz, A. Artés, and M. Alvarez, "Modular gaussian processes for transfer learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1–12, 2021.

[39] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *Neural Networks, IEEE Transactions on*, vol. 22, no. 2, pp. 199–210, 2011.

[40] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *CVPR*, 2012, pp. 2066–2073.

[41] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *ICCV*, 2013, pp. 2960–2967.

[42] B. Sun, J. Feng, and K. Sa-enko, "Return of frustratingly easy domain adaptation," in *AAAI*, 2016, pp. 2058–2065.

[43] J. Zhang, W. Li, and P. Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," in *CVPR*, 2017, pp. 692–701.

[44] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang, and P. S. Yu, "Visual domain adaptation with manifold embedded distribution alignment," in *ACM Multimedia Conference (ACM MM)*, 2018, pp. 402–410.

[45] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[46] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *ICML*, 2017, pp. 2208–2217.

[47] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[48] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3723–3732.

[49] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1426–1435.

[50] N. Xiao and L. Zhang, "Dynamic weighted learning for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15 242–15 251.

[51] R. Wang, Z. Wu, Z. Weng, J. Chen, G.-J. Qi, and Y.-G. Jiang, "Cross-domain contrastive learning for unsupervised domain adaptation," *IEEE Transactions on Multimedia*, 2022.

[52] K. Shen, R. M. Jones, A. Kumar, S. M. Xie, J. Z. HaoChen, T. Ma, and P. Liang, "Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 19 847–19 878.

[53] R. Nikzad-Langerodi, W. Zellinger, E. Lughofer, and S. Saminger-Platz, "Domain-invariant partial-least-squares regression," *Analytical chemistry*, vol. 90, no. 11, pp. 6693–6701, 2018.

[54] X. Chen, S. Wang, J. Wang, and M. Long, "Representation subspace distance for domain adaptation regression," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1749–1759.

[55] A. de Mathelin, G. Richard, F. Deheeger, M. Mougeot, and N. Vayatis, "Adversarial weighting for domain adaptation in regression," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2021, pp. 49–56.

[56] S. Sun, G. Zhang, C. Wang, W. Zeng, J. Li, and R. Grosse, "Differentiable compositional kernel learning for gaussian processes," *arXiv preprint arXiv:1806.04326*, 2018.

[57] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin, "Structure discovery in nonparametric regression through compositional kernel search," in *International Conference on Machine Learning*, 2013, pp. 1166–1174.

[58] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, no. 7, pp. 2211–2268, 2011.

[59] F. Zhang, *The Schur Complement and Its Applications*. Springer Science & Business Media, 2005, vol. 4.

[60] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[61] L. M. Candanedo, V. Feldheim, and D. Deramaix, "Data driven prediction models of energy use of appliances in a low-energy house," *Energy and Buildings*, vol. 140, pp. 81–97, 2017.

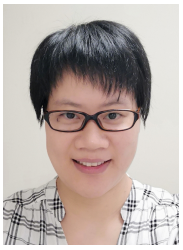
[62] S. De Vito, G. Fattoruso, M. Pardo, F. Tortorella, and G. Di Francia, "Semi-supervised learning techniques in artificial olfaction: A novel approach to classification problems and drift counteraction," *IEEE Sensors Journal*, vol. 12, no. 11, pp. 3215–3224, 2012.

[63] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "UJIIndoorloc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems." in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2014, pp. 261–270.

[64] E. Snelson and Z. Ghahramani, "Local and global sparse gaussian process approximations," in *Artificial Intelligence and Statistics*, 2007, pp. 524–531.

[65] A. Naish-Guzman and S. Holden, "The generalized fitc approximation," in *Advances in Neural Information Processing Systems*, 2008, pp. 1057–1064.

Pengfei Wei received the Ph.D. degree in 2019, School of Computer Science and Engineering, Nanyang Technological University, Singapore. He is currently a research scientist in Bytendance AI lab in Singapore. His research interests include data analysis, transfer learning, domain adaptation, and reinforcement learning.



Yiping Ke is an associate professor at the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore. She obtained her Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology in 2008 and B.Sc. degree in Computer Science from Fudan University in 2003. Her research lies in Data Mining and Artificial Intelligence, with a focus on Graph Learning, Graph Analytics, and Graph Mining. She has successfully rolled out technologies that advance the best practice in a variety of domains including engineering design, neuroscience, biology, etc.

Yew-Soon Ong (M'99-SM'12-F'18) received the Ph.D. degree in artificial intelligence in complex engineering design from the University of Southampton, U.K., in 2003. He is a President Chair Professor in Computer Science at the Nanyang Technological University (NTU) and concurrently the Chief Artificial Intelligence Scientist of the Agency for Science, Technology and Research (A*Star) Singapore. At NTU, he serves as co-Director of the Singtel-NTU Cognitive & Artificial Intelligence Joint Lab. His core research interest is in artificial and computational intelligence where he has received four IEEE outstanding paper awards. He was listed as a Thomson Reuters highly cited researcher and among the World's Most Influential Scientific Minds. He is the inaugural Editor-in-Chief of the IEEE Transactions on Emerging Topics in Computational Intelligence and associate editor of the IEEE on Transactions on Evolutionary Computation, IEEE Transactions on Neural Networks & Learning Systems, IEEE on Transactions on Cybernetics, IEEE Transactions on Artificial Intelligence.

Zejun Ma received his bachelor degree from Shanghai Jiaotong University, Shanghai in 2007. He received both his Master degree and PhD degree from Institue of automation, Chinese academy of Science, Beijing in 2009 and 2012, respectively. Currently he serves as a Director at Bytedance AI Lab. His recent research interests include speech and language topics in diverse machine learning areas, such as speech recognition, speech synthesis, music information retrieval, and natural language understanding. He has many publications on top-tier conferences and journals, such as SIGKDD, Interspeech, ICASSP, ASRU, etc.