International Journal of Production Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

A multisource domain adaptation method for quality prediction in small-batch production systems

Dengyu Li & Kaibo Wang

To cite this article: Dengyu Li & Kaibo Wang (2022) A multisource domain adaptation method for quality prediction in small-batch production systems, International Journal of Production Research, 60:20, 6268-6281, DOI: 10.1080/00207543.2021.1989076

To link to this article: https://doi.org/10.1080/00207543.2021.1989076

Published online: 21 Oct 2021.

INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
2022, VOL. 60, NO. 20, 6268–6281
https://doi.org/10.1080/00207543.2021.1989076

Taylor & Francis
Taylor & Francis Group

A multisource domain adaptation method for quality prediction in small-batch production systems

Dengyu Li and Kaibo Wang

Department of Industrial Engineering, Tsinghua University, Beijing, People's Republic of China

ABSTRACT

Quality prediction for small-batch production processes is a complex problem due to limitations in available training samples. In this study, a multisource domain adaptation joint-Y partial least square (PLS) method is proposed to learn the similarities between domains and use them to construct a quality prediction model. Without constraints on the number of source and target domains, the proposed method can transfer more historical information for the in-operation process than traditional methods. Numerical experiments and a real-world case study of quality prediction in computer wafer production are performed to verify the effectiveness of the proposed method. The results show that the prediction accuracy of the proposed method is high in cases with few training samples in the target domain compared to the accuracies of the joint-Y PLS model and the traditional PLS model.

ARTICLE HISTORY
Received 2 April 2021
Accepted 22 September 2021

KEYWORDS
Quality prediction; small-batch process; domain adaptation; partial least square; multitask learning

1. Introduction

Small-batch production is gradually becoming a mainstream manufacturing mode for various products such as metals, semiconductors, pharmaceuticals and chemicals (De Vin, De Vries, and Streppel 2000; Jia, Zhang, and You 2020; Yin et al. 2012). Two primary characteristics of small-batch production are multiple varieties and process similarity (Zhang et al. 2011). With the increasing demand for high-quality process control, it is essential to make effective quality predictions during production to reduce the costs of defective products and to make quick adjustments. However, in small-batch production processes, the limited number of samples in one batch is typically not sufficient to make precise predictions, while simply combining groups of batches using a shared model would miss the various patterns of different batches. To describe the challenges of quality prediction in small-batch processes, computer wafer production is considered a motivating example. Single crystals of silicon are the primary raw material in wafer production, and only a few wafers can be made from one single crystal of silicon, whose physical and chemical properties are typically different from each other. Therefore, to make wafers from different single crystals of silicon have similar quality, wafers from one single crystal of silicon are considered a small batch in wafer production processes. Additionally, the functional relationship between process parameters and the final quality of wafers from different batches are similar but different, which leads to the problem of modelling multiple small-batch processes with similar properties.

Among data-driven models, many multivariate methods, such as ridge regression (Hoerl and Kennard 1970), partial least square (PLS) (Geladi and Kowalski 1986), principal component regression (PCR) (Linares and Mederos 1981)and support vector machine (SVM) (Boser, Guyon, and Vapnik 1996) approaches, have been shown to be effective in handling quality prediction problems with high-dimensional of feature spaces. However, these methods all require the feature spaces (the number of input variables and the meaning of each variable) of the same size, which is a requirement that is sometimes violated in real-world small-batch quality prediction problems due to the differences in process procedures and the absence of certain variables. Joint-Y PLS (JYPLS) (Garcia-Munoz, MacGregor, and Kourti 2005) and joint-Y kernel PLS (JYKPLS) (Chu et al. 2018) are two variants of PLS that assume a common latent feature space of different batches and aim to model similar processes with different input feature spaces.

These methods require relatively large numbers of training samples to construct an effective model. However, in small-batch processes, it is difficult to obtain sufficient training data for an in-operation batch. In recent

CONTACT Kaibo Wang kbwang@tsinghua.edu.cn Department of Industrial Engineering, Tsinghua University, Beijing 100084, People's Republic of China

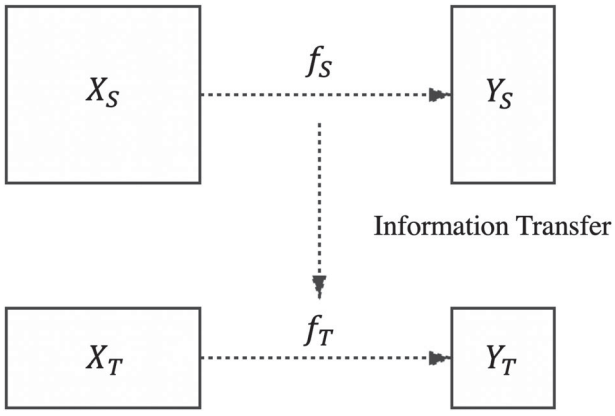

Figure 1. Information transformation from a source (X_S, Y_S) with many samples to a target (X_T, Y_T) with a few samples ($f\cdot$ is the functional relationship for the corresponding domain).

years, domain adaptation (DA), a method of transfer learning, has been extensively used to effectively model similar processes with limited training samples (Duan, Xu, and Chang 2012; Long, Wang, Ding, Sun, et al. 2013; Pan, Tsang, Kwok, & Yang, 2011). To describe DA more completely, the definitions of related concepts are briefly introduced below. A domain $\mathcal{D}$ consists of the feature space $\mathcal{X}$ and the marginal distribution $p(X)$, which is shown by input samples $X = \{x_1, x_2, \cdots, x_n\} \in \mathcal{X}$. A target consists of the label space $\mathcal{Y}$ and the relationship function $f(\cdot)$. A general problem of DA typically consists of two types of domains: source domain $D_S = \{(x_i, y_i)\}_{i=1}^{n_S}$ and target domain $D_T = \{(x_i, y_i)\}_{i=1}^{n_T}$, where the information of the source domain is greater than that of the target domain ($n_S \gg n_T$). The goal of transfer learning is to learn the function of the target task $f_T(\cdot)$ using the information of D_S and D_T, with the assumption that $f_S(\cdot) \neq f_T(\cdot)$ (Weiss, Khoshgoftaar, and Wang 2016). Specifically, DA learns how to map the feature spaces of the source and target domains into a common feature space so that modelling the function between labels and the common latent features has sufficient training samples. The idea of the complete learning process is briefly shown in Figure 1. In the wafer production example, historical batches are considered the source domains, while the in-process batch is the target domain. To construct an effective model to predict the quality of the in-process batch, using limited training samples is insufficient, and the information from historical batches is then required.

DA is a method of feature transfer and is one of the three main categories of transfer learning (Cheng, Tsung, and Wang 2017), which has been widely used in production areas such as quality control and fault diagnosis (K. Wang and Tsung 2020; Z. Wang et al. 2020). The goal of DA is to solve a learning method in the target domain using the information of training data from the source domain, which is suitable for small-batch processes (Long et al. 2014; Long, Wang, Ding, Pan, et al. 2013). Note that the training data (source domain) and the application data (target domain) may have different distributions, which is different from the common assumption of traditional statistical and machine learning models (Muandet, Balduzzi, and Schölkopf 2013). There are many existing methods of DA, such as transfer component analysis (TCA) (Pan, 2011), joint distribution adaptation (JDA) (Long, Wang, Ding, Sun, et al. 2013) and geodesic flow kernel (GFK) (Gopalan, Li, and Chellappa 2011). These methods, however, assume the same feature space $\mathcal{X}_S = \mathcal{X}_T$ and the same label space $\mathcal{Y}_S = \mathcal{Y}_T$, which is categorised as homogeneous DA. More general cases when $\mathcal{X}_S \neq \mathcal{X}_T$ ($\mathcal{Y}_S = \mathcal{Y}_T$ holds) are called heterogeneous DA and use techniques such as manifold alignment (C. Wang and Mahadevan 2011) and discriminative distribution alignment (Yao et al. 2020). Although DA is widely used in neural networks, there have been few attempts to combine DA with statistical models, where DA focuses on learning the mapping from the original feature spaces to the common latent feature space and statistical models focus on learning the functional relation between the common latent feature space and the label space. Domain-invariant PLS (Nikzad et al. 2018) aligns the source and target distributions in the latent feature space using a domain regularisation method. Domain adaptation joint-Y PLS (DA-JYPLS) (Jia, Zhang, and You 2020) adds a regularised term in the objective function to describe the trade-off between maximising the covariances of input and output variables and minimising the differences of the distributions of the source and target domains.

Among the methods listed above, DA-JYPLS and domain-invariant PLS are combinations of heterogeneous DA and statistical methods. However, these two variants assume that there is only one source domain and one target domain. In real-world problems such as wafer production, which has multiple domains, these methods are not applicable because the number of samples in each source domain is also limited; thus, the expected model needs to transform information from multiple domains. There are other methods that consider multiple domains in DA including multisource TCA (Grubinger et al. 2017) and multisource Generative Adversarial Networks (GAN) (Zhao et al. 2017). The former method focuses on learning mapping matrices, while the latter method is based on the neural network and needs many samples from each domain, which are also not suitable for the small-batch process.

In general, there have been a few attempts to combine statistical models and domain adaptation with multiple domains in the application of small-batch processes. In

this study, a novel domain adaptation regression method is proposed to extend the DA-JYPLS method to cases with multiple domains. Similar to JYPLS and some heterogeneous DA methods, the present method does not require the observations from all source domains to have anything in common except for having the same response variables, i.e. $\mathcal{Y}_S = \mathcal{Y}_T$. However, more shared information can be learned and the present method is more effective if the sources are more similar. Additionally, there is no restriction on the number of either the source domains or the target domains, which increases the potential applications of the proposed method. In real-world cases, the proposed method can be used to predict the quality of a new batch in a small-batch process using information from multiple historical batches.

The remainder of this paper is organised as follows. In Section 2, the proposed model and the relevant parameter estimation steps are presented in detail. Examples including numerical experiments and a real-world application problem are presented in Section 3 and Section 4, respectively, to verify the efficiency of the proposed method Finally, conclusions are drawn in Section 5.

2. Model

The proposed multisource DA-JYPLS model is shown in detail in Section 2.1. The parameter estimation procedures and the techniques for selecting hyperparameters are presented in Section 2.2 and Section 2.3, respectively.

2.1. Multisource domain adaptation joint-Y PLS

In the domain adaptation problem with multiple domains, we assume that there are $K(\geq 2)$ relative domains and that there is no restriction on the number of source domains and target domains. $D_k = \{(\boldsymbol{x}_{i,k}, \boldsymbol{y}_{i,k})\}_{i=1}^{n_k}$ denotes the pairs of input and output data from the k_{th} domain with the corresponding input and output matrices, which are denoted as $\boldsymbol{X}_k^{=}[\boldsymbol{x}_{1,k}^T, \boldsymbol{x}_{2,k}^T, \cdots, \boldsymbol{x}_{n_k,k}^T] \in \mathbb{R}^{n_k \times d_{X_k}}$ and $\boldsymbol{Y}_k^{=}[\boldsymbol{y}_{1,k}^T, \boldsymbol{y}_{2,k}^T, \cdots, \boldsymbol{y}_{n_k,k}^T] \in \mathbb{R}^{n_k \times d_Y}$, respectively. JYPLS assumes that $\boldsymbol{Y}_k$ matrices lie in the common latent space that is defined by the joint loading matrix $\boldsymbol{Q}_J$, which is the plane described by all $\boldsymbol{Y}_k$s. The only restriction for JYPLS is that all $\boldsymbol{Y}_k$s must have the same variables defining their columns. The model does not have any restrictions on $\boldsymbol{X}_k$s, the number of the columns in $\boldsymbol{X}_k$s, or the nature of the variables defining them. The loading matrices, $\boldsymbol{W}_k$s, are defined as the direction of variation in $\boldsymbol{X}_k$s' space that are most correlated with $\boldsymbol{Q}_J$. $\boldsymbol{T}_k$s are defined as the scoring matrices of $\boldsymbol{X}_k$s on the loading matrices, $\boldsymbol{W}_k$s. The model also assumes $\boldsymbol{X}_k$s to have a common latent space. The loading matrices, $\boldsymbol{W}_k$s, scoring matrices, $\boldsymbol{T}_k$s, together with the weight matrices defined by $\boldsymbol{P}_k$s, should have different forms because the numbers of dimensions in $\boldsymbol{X}_k$s are different (Jia, Zhang, and You 2020).

Similar to JYPLS, the model part of the proposed method is given as follows:

$$\boldsymbol{Y}_J = \begin{bmatrix} \boldsymbol{Y}_1 \\ \vdots \\ \boldsymbol{Y}_K \end{bmatrix} = \begin{bmatrix} \boldsymbol{T}_1 \\ \vdots \\ \boldsymbol{T}_K \end{bmatrix} \boldsymbol{Q}_J^T + \boldsymbol{E}_{Y_J} \tag{1}$$

$$\boldsymbol{X}_k = \boldsymbol{T}_k \boldsymbol{P}_k^T + \boldsymbol{E}_{X_k} \quad k = 1, \ldots, K. \tag{2}$$

$$\boldsymbol{T}_k = \boldsymbol{X}_k \boldsymbol{W}_k \quad k = 1, \ldots, K. \tag{3}$$

where $\boldsymbol{E}_{X_k}$ are residual matrices of input variables of the k_{th} domain and $\boldsymbol{E}_{Y_J}$ is the residual matrix of the joint output variables. In contrast to the JYPLS model, the objective function for the loading vector $\boldsymbol{w}_J$ is given in Equation (4) in its general form:

$$\max_{\boldsymbol{w}_J} cov\{\boldsymbol{t}_J, \boldsymbol{Y}_J\} - \mu_1 \cdot dist(\boldsymbol{t}_1, \cdots, \boldsymbol{t}_K) - \mu_2 \cdot Loss_{Locality} \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{t}_k = \boldsymbol{X}_k \boldsymbol{w}_k \quad k = 1, \ldots, K$$

$$||\boldsymbol{w}_J|| = 1$$

$$\boldsymbol{t}_k \in \mathbb{R}^{n_k \times 1}, \ \boldsymbol{w}_k \in \mathbb{R}^{d_{X_k} \times 1}$$

where $\boldsymbol{t}_J^T = [\boldsymbol{t}_1^T, \ldots, \boldsymbol{t}_K^T]$ is the joint vector of the first scoring vectors of the domains, $\boldsymbol{w}_J^T = [\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_K^T]$ is the joint vector of the first weight vectors of the domains, and μ_1, μ_2 are parameters for the two penalty terms.

In the objective function (4), the first term is similar to the JYPLS model, maximising the covariances between the scoring vector and the output data and the second term is the extension form for minimising the distribution distances of scoring vectors of K domains. Recall that under the assumption of domain adaptation, the marginal distributions of the projection of the input data of different domains in the common latent feature space might be different (i.e. $p(\boldsymbol{t}_S) \neq p(\boldsymbol{t}_T)$), and the shared loading matrix $\boldsymbol{Q}_J$ cannot effectively suit them. Therefore, adding the term of minimising the distribution distance of $\boldsymbol{t}_k$s would increase the explanatory ability of the model, which is a commonly used technique in DA methods. The specific form of the second form is given as follows:

$$\begin{aligned} dist(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_K) &= Var\{Var(\boldsymbol{t}_1), \ldots, Var(\boldsymbol{t}_K)\} \\ &= \frac{1}{K-1} \sum_{i=1}^{K} \left(Var(\boldsymbol{t}_i) - \frac{1}{K} \sum_{i=1}^{K} Var(\boldsymbol{t}_i) \right)^2 \\ &= \frac{1}{K-1} \boldsymbol{w}_J^T \left(\sum_{i=1}^{K} \boldsymbol{A}_i \boldsymbol{w}_J \boldsymbol{w}_J^T \boldsymbol{A}_i \right) \boldsymbol{w}_J \end{aligned} \tag{5}$$

where $\boldsymbol{A}_i$ is a K-block diagonal matrix with the jth($j = 1, \ldots, K$) block expressed as

$$A_i^{(j)} = \begin{cases} \dfrac{\boldsymbol{X}_j^T \boldsymbol{X}_j}{n_j - 1} \cdot \left(-\dfrac{1}{K}\right) j \neq i \\ \dfrac{\boldsymbol{X}_j^T \boldsymbol{X}_j}{n_j - 1} \cdot \left(\dfrac{K-1}{K}\right) j = i \end{cases} \tag{6}$$

Note that this form of evaluating distribution difference is defined as the squared value of the difference of the variances, which is similar to the CORAL distance (Sun and Saenko 2016). Conversely, distances between multiple domains are typically in the form of $dist(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_K) = \sum_{i=1}^{K} \sum_{i \neq j} dist(\boldsymbol{t}_i, \boldsymbol{t}_j)$. There are other terms for evaluating the distribution difference in domain adaptation, such as the maximum mean discrepancy (MMD) (Smola et al. 2007a, 2007b) and Earth mover's distance (EMD) (Rubner, Tomasi, and Guibas 2000), which are not suitable for use in the model because the means of the vectors are the same after normalisation.

In contrast to DA-JYPLS and domain invariant PLS, which consider only one source and one target domain, in the present work with multiple domains with different features, learning the scoring vector by only minimising their marginal distribution differences may destroy their geometrical properties within domains. When source domains have little in common, the distributions' distance-minimisation term might 'overlearn' the shared information and therefore make the extracted features from different domains 'over similar'. The resulting projection matrix usually destroys the geometric properties within domains, where the geometric properties in the present work are called 'locality'. Therefore, to make the model more effective even in cases in which there are outlier domains that have little similarity with other domains, a penalty form is added in the objective function as the locality loss. The specific definition of the form comes from the idea in TCA (Pan, 2011) and is given as:

$$Loss_{Locality} = \sum_{k=1}^{K} \sum_{i,j=1}^{n_k} \boldsymbol{t}_i - \boldsymbol{t}_j{}^2 \boldsymbol{D}_{k(i,j)} = \boldsymbol{w}_J^T \boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X} \boldsymbol{w}_J \tag{7}$$

where n_k is the number of samples in the kth domain, and $\boldsymbol{D}_{k(i,j)}$ is the element of the matrix $\boldsymbol{D}_k$ in the ith row and jth column, which is defined as:

$$\boldsymbol{D}_{k(i,j)} = \begin{cases} \exp\left(-\dfrac{\boldsymbol{x}_{k,i} - \boldsymbol{x}_{k,j}{}^2}{\sigma_k^2}\right) & \text{if } \boldsymbol{x}_{k,i} \in \mathcal{N}_{l_k}(\boldsymbol{x}_{k,j}) \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

where $\boldsymbol{x}_{k,j}$ is the jth input sample in the k_{th} domain and $\mathcal{N}_{l_k}(\boldsymbol{x}_{k,j})$ is the set of l_k-nearest neighbours of $\boldsymbol{x}_{k,j}$ within the kth domain. Note that the loss term is defined as a weighted summation of the quadratic form of differences between observations in the latent feature space within domains. The respective weight is negatively correlated with the differences between observations in the original feature space within domains. Minimisation of the summation will make the elements with lower weights greater than those with higher weights. Therefore, distances between observations in the latent feature space are comparatively larger if they have larger distances in the original feature space with lower weights, which keeps the geometric properties (locality) within domains. To further explain the effect of the locality loss form, a simple example is considered where the same number of points (observations) are stochastically chosen in two ellipses that represent two source domains. The lengths of the minor axis are equal, while the major axis is not, which is shown in Figure 2. If we aim to choose the 1-D latent feature space, only minimising the distribution difference in the latent space would obtain x_2. This would lose all the information about x_1, which has more unique information within domains. Adding the consideration of locality preservation would make the resulting latent space a combination of x_1 and x_2, which keeps both the shared and unique information. The locality loss can be further derived as Equation (9):

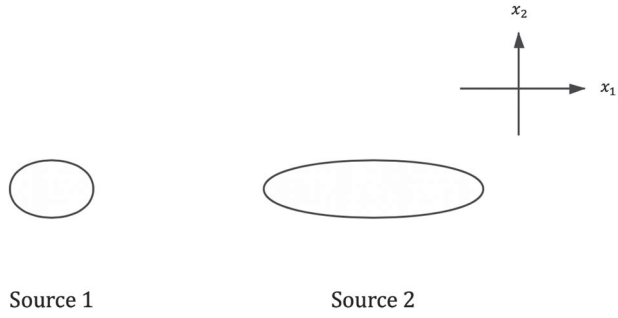

Figure 2. Illustrated example of the locality preservation term.

$$Loss_{Locality} = \sum_{k=1}^{K} \sum_{i,j=1}^{n_k} ||\boldsymbol{t}_i - \boldsymbol{t}_j||^2 \boldsymbol{D}_{k(i,j)} = \boldsymbol{w}_J^T \boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X} \boldsymbol{w}_J \tag{9}$$

where $\boldsymbol{X} = diag\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_K\}$ is the diagonal matrix of the input data and $\boldsymbol{L} = diag\{\boldsymbol{L}_1, \ldots, \boldsymbol{L}_K\}, \boldsymbol{L}_k = \boldsymbol{S}_k - \boldsymbol{D}_k$, where $\boldsymbol{L}_k$ is an $n_k \times n_k$ diagonal matrix with the ith element is the sum of elements in the ith row of $\boldsymbol{D}_k$ (i.e. $\boldsymbol{S}_{k(i,i)} = \sum_{j=1}^{n_k} \boldsymbol{D}_{k(i,j)}$).

Combining these three terms in the objective function, the final optimisation problem for finding the first components for scoring and weight vectors can be

written as:

$$\max_{\boldsymbol{w}_J} \left\{ cov\{\boldsymbol{t}_J, \boldsymbol{Y}_J\} - \mu_1 Var\{Var(\boldsymbol{t}_i)\} - \mu_2 \sum_{k=1}^{K} \sum_{i,j=1}^{n_k} ||\boldsymbol{t}_i - \boldsymbol{t}_j||^2 \boldsymbol{D}_{k(i,j)} \right\} \quad (10)$$

$$\text{s.t.} \quad \boldsymbol{t}_k = \boldsymbol{X}_k \boldsymbol{w}_k \quad k = 1, \ldots, K$$

$$||\boldsymbol{w}_J|| = 1$$

2.2. Parameter estimation

Having defined the model and the objective function, the parameters in the proposed model should be estimated from the available data in the domains. Substituting Equations (5) and (9) into objective function (10), the final objective function can be written as:

$$\max_{\boldsymbol{w}_J} \boldsymbol{w}_J^T \boldsymbol{M} \boldsymbol{w}_J - \lambda_1 \boldsymbol{w}_J^T \left(\sum_{i=1}^{K} \boldsymbol{A}_i \boldsymbol{w}_J \boldsymbol{w}_J^T \boldsymbol{A}_i \right) \boldsymbol{w}_J - \lambda_2 \boldsymbol{w}_J^T \boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X} \boldsymbol{w}_J \quad (11)$$

$$\text{s.t.} \quad ||\boldsymbol{w}_J|| = 1$$

where $\boldsymbol{M} \in \mathbb{R}^{(n_1+\cdots+n_K)\times(n_1+\cdots+n_K)}$; the block of the ith row and the jth column are $\boldsymbol{M}_{i,j} \in \mathbb{R}^{n_i \times n_j} = \boldsymbol{X}_i^T \boldsymbol{Y}_i \boldsymbol{Y}_j^T \boldsymbol{X}_j$ $(i, j = 1, \ldots, K)$; and λ_1 and λ_2 are the adjusted penalty hyperparameters. According to the Lagrange slack theorem, the optimisation problem (11) can be reformulated as follows by adding a Lagrange multiplier λ_3:

$$\max_{\boldsymbol{w}_J} L(\boldsymbol{w}_J) = \boldsymbol{w}_J^T \boldsymbol{M} \boldsymbol{w}_J - \lambda_1 \boldsymbol{w}_J^T \left(\sum_{i=1}^{K} \boldsymbol{A}_i \boldsymbol{w}_J \boldsymbol{w}_J^T \boldsymbol{A}_i \right) \boldsymbol{w}_J - \lambda_2 \boldsymbol{w}_J^T \boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X} \boldsymbol{w}_J - \lambda_3 (\boldsymbol{w}_J^T \boldsymbol{w}_J - 1) \quad (12)$$

Taking the derivative of Equation (12) with respect to $\boldsymbol{w}_J$ and setting it equal to zero, the following equation is obtained:

$$\boldsymbol{M} \boldsymbol{w}_J = 2\lambda_1 \left(\sum_{i=1}^{K} \boldsymbol{A}_i \boldsymbol{w}_J \boldsymbol{w}_J^T \boldsymbol{A}_i \right) \boldsymbol{w}_J + \lambda_2 \boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X} \boldsymbol{w}_J + \lambda_3 \boldsymbol{w}_J \quad (13)$$

Note that Equation (13) contains the high-order term of $\boldsymbol{w}_J$, and we cannot obtain the analytical form of $\boldsymbol{w}_J^*$, which is the reason why we used Newton's method to obtain an approximate solution. The result $\boldsymbol{w}_J^*$ is actually a local optimum for the optimisation problem (12). However, if the initial value of Newton's method is randomised, the resulting local optimum solution might be worse than $\boldsymbol{w}_J^{JYPLS}$, which violates the expectation. Therefore, using $\boldsymbol{w}_J^{JYPLS}$ as the initial value for iteration can make the result $\boldsymbol{w}_J^*$ a comparatively closer local optimum solution to $\boldsymbol{w}_J^{JYPLS}$, which is more likely to obtain a better solution in the optimisation Equation (12) than $\boldsymbol{w}_J^{JYPLS}$, although it might not be the global optimum solution. Note that $\boldsymbol{w}_J^{JYPLS}$ is the solution to optimisation problem (14) as follows:

$$\max_{\boldsymbol{w}_J} cov(\boldsymbol{t}_J, \boldsymbol{Y}_J) = \boldsymbol{w}_J^T \begin{bmatrix} \boldsymbol{X}_1^T \cdots 0 \\ \vdots \ddots \vdots \\ 0 \cdots \boldsymbol{X}_K^T \end{bmatrix} \begin{bmatrix} \boldsymbol{Y}_1 \\ \vdots \\ \boldsymbol{Y}_K \end{bmatrix} \times \begin{bmatrix} \boldsymbol{Y}_1 \\ \vdots \\ \boldsymbol{Y}_K \end{bmatrix}^T \begin{bmatrix} \boldsymbol{X}_1 \cdots 0 \\ \vdots \ddots \vdots \\ 0 \cdots \boldsymbol{X}_K \end{bmatrix} \boldsymbol{w}_J \quad (14)$$

$$\text{s.t.} \quad ||\boldsymbol{w}_J|| = 1$$

In Newton's method, the original function and its derivative are given as:

$$G(\boldsymbol{w}_J) = \boldsymbol{M} \boldsymbol{w}_J - 2\lambda_1 \left(\sum_{i=1}^{K} \boldsymbol{A}_i \boldsymbol{w}_J \boldsymbol{w}_J^T \boldsymbol{A}_i \right) \boldsymbol{w}_J - \lambda_2 \boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X} \boldsymbol{w}_J - \lambda_3 \boldsymbol{w}_J \quad (15)$$

$$G'(\boldsymbol{w}_J) = \boldsymbol{M} - 2\lambda_1 \sum_{i=1}^{K} (\boldsymbol{w}_J^T \boldsymbol{A}_i \boldsymbol{w}_J + 2\boldsymbol{A}_i \boldsymbol{w}_J \boldsymbol{w}_J^T) \boldsymbol{A}_i - \lambda_2 \boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X} - \lambda_3 \boldsymbol{I}. \quad (16)$$

1. We initialise $\boldsymbol{w}_J^{(0)}$ as $\boldsymbol{w}_J^{JYPLS}$, and iteratively update $\boldsymbol{w}_J$ as:

$$\boldsymbol{w}_J^{(i+1)} = \boldsymbol{w}_J^{(i)} - [G'(\boldsymbol{w}_J^{(i)})]^{-1} G(\boldsymbol{w}_J^{(i)}) \quad (17)$$

until convergence (i.e. $|\boldsymbol{w}_J^{(i+1)} - \boldsymbol{w}_J^{(i)}| < \epsilon$). The detailed algorithm for the proposed multisource DA-JYPLS model is described as follows:

1. Mean centre and unit-scale input matrix $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_K$, $\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_K$
   For $i = 1 : A$, repeat step 2–7. Initialise $\boldsymbol{Y}_{J1} = \boldsymbol{Y}_J$ and $\boldsymbol{X}_{k1} = \boldsymbol{X}_k (k = 1 \cdots K)$
2. Obtain the scoring vector $\boldsymbol{w}_{Ji}$ by solving the optimisation problem (11)
3. Normalise $\boldsymbol{w}_{Ji}$ by $\boldsymbol{w}_{Ji} = \frac{\boldsymbol{w}_{Ji}}{||\boldsymbol{w}_{Ji}||}$, and obtain $\boldsymbol{w}_{ki} i = 1 \cdots K$
4. Regress to obtain the scoring vector $\boldsymbol{t}_{ki}$ by $\boldsymbol{t}_{ki} = \boldsymbol{X}_k^T \boldsymbol{w}_{ki} (\boldsymbol{w}_{ki}^T \boldsymbol{w}_{ki})^{-1}$
5. Regress to obtain the loading vector $\boldsymbol{q}_{Ji}$ by $\boldsymbol{q}_{ki} = \boldsymbol{Y}_J^T \boldsymbol{t}_{Ji} (\boldsymbol{t}_{Ji}^T \boldsymbol{t}_{Ji})^{-1}$
6. Calculate $\boldsymbol{p}_{ki}$ by $\boldsymbol{p}_{ki} = \boldsymbol{X}_k^T \boldsymbol{t}_{ki} (\boldsymbol{t}_{ki}^T \boldsymbol{t}_{ki})^{-1}$

7. Deflate the matrices $\boldsymbol{X}_k, \boldsymbol{Y}_k$ by equations $\boldsymbol{X}_{k,i+1} = \boldsymbol{X}_{ki} - \boldsymbol{t}_{ki}\boldsymbol{p}_{ki}^T$, $\boldsymbol{Y}_{k,i+1} = \boldsymbol{Y}_{ki} - \boldsymbol{t}_{ki}\boldsymbol{q}_{ji}^T$

The final projection matrices are $\boldsymbol{W}_k^* = \boldsymbol{W}_k(\boldsymbol{P}_k^T\boldsymbol{W}_k)^{-1}$ and the final prediction for test data in the kth domain is $\hat{\boldsymbol{y}}_k = \boldsymbol{x}_{k,new}\boldsymbol{W}_k^*\boldsymbol{Q}_j^T$, where $\boldsymbol{x}_{k,new}$ is the new observation of input data in the kth domain.

2.3. Determining the hyperparameters

In the proposed model, there are many hyperparameters that must be determined before the final estimation for the weight and scoring matrices. σ_k^2 in Equation (8) is the bandwidth of the exponentials and should be determined by the scales of the input data in the kth domain to make the elements of D_k properly disperse. l_k is the number of chosen nearest neighbours in the kth domain and should be selected properly according to the number of samples in the domain. In this work, $l_k = 0.5\times$ number of training samples in the respective domain. A sensitivity analysis of l_k and detailed interpretation are given in Section 3. Similar to DA-JYPLS, λ_1 and λ_2 are selected from a finite candidate set and are tuned by cross-validation for the training data of the domains. In the following numerical simulations and real-world examples, candidate sets $\Omega_{\lambda_1} = \Omega_{\lambda_2} = \{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$. Note that the combination of these two parameters in the optimisation problem when extracting each component could be different. Additionally, $|\Omega_{\lambda_1}| = |\Omega_{\lambda_2}| = 6$ is not a large number and is chosen in this work to simplify computations because the number of candidate combinations of λ_1 and λ_2 is $|\Omega|^{2A}$, where A is the number of latent variables. Therefore, the computation would be costly if denser candidate sets for λ_1 and λ_2 are chosen. The Lagrange multiplier λ_3, is set equal to the JYPLS model (i.e. the largest eigenvalue of the matrix $\boldsymbol{M}$ in Equation (11)).

3. Numerical simulations

In this section, numerical experiments are conducted to verify the efficiency of the proposed method. The comparison models included in the experiments are the JYPLS model and the separate PLS model, which operates multiple traditional PLS methods on the samples of each domain. Note that the parameters of these two models are estimated using the NIPALS algorithm, which has been shown to be stable and accurate. Apart from the variants of PLS, principal component regression (PCR) and support vector machines (SVMs) are also serve as comparing methods. These two methods are also utilised separately on each domain.

The measurement index that is used to determine the penalty parameters used in this study is the leave-one-out cross-validation (CV) with the mean absolute error (MAE). MAE is also used to compare the efficiency of the different models. These two measurements would be specifically defined in different situations. As mentioned above, there are no constraints on the number of source domains or the target domains; however, in scenarios with different combinations of these two types of domains, their goals are different, and thus, the evaluation index should be adjusted correspondingly. In this study, two specific scenarios are designed to verify the stability and efficiency of the prediction performance of the proposed model: multiple source domains with one target domain and multiple target domains with no source domain. In the present method, theoretically, data from different domains $(\boldsymbol{X}_k, \boldsymbol{Y}_k)$ are treated equally, and the only difference between the source domains and the target domains is the number of observations (sufficient information or not). The two scenarios are designed to fit real-world application situations. In Scenario 1, where there are multiple source domains and a single target domain, a new batch (target domain) of products is in process, and there are many finished batches (source domain) from which we can learn information. Additionally, since the historical batches are finished, there is no need to transfer knowledge from other domains to help model the processes of source domains. In Scenario 2, where there are multiple target domains, several new batches are in process, and there are no historical batches (source domains) to learn from. Therefore, in the application's view, the in-process batches (which can be improved) are classified into target domains and historical batches (which cannot be improved) are classified into source domains. Usually, the information from the source domains is more sufficient but shared information can be learned and transferred from both source and target domains. The corresponding MAEs for CV and comparisons between the two scenarios are defined below.

Scenario 1: multiple source domains and one target domain

$$\mathrm{MAECV} = \frac{1}{n_T^{(train)}} \sum_{i=1}^{n_T^{(train)}} \sum_{j=1}^{d_Y} |y_{T,i,j} - \hat{y}_{T,-i,j}| \tag{18}$$

$$\mathrm{MAE} = \frac{1}{n_T^{(test)}} \sum_{i=1}^{n_T^{(test)}} \sum_{j=1}^{d_Y} |y_{T,\,i,j} - \hat{y}_{T,i,j}| \tag{19}$$

where $n_T^{(train)}$ and $n_T^{(test)}$ are the numbers of training samples and testing samples in the target domain, respectively; d_Y is the dimension of the output data; $y_{T,i,j}$ is the

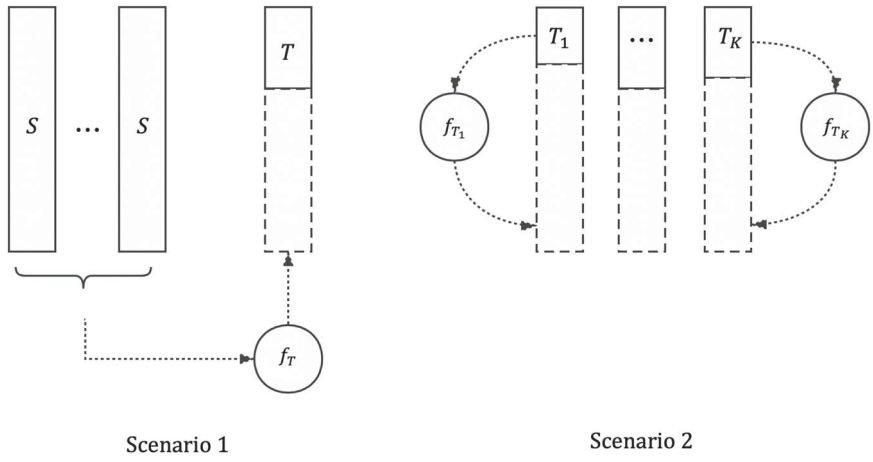

Figure 3. Design of the two scenarios. (observed data are symbolised as solid boxes and unobserved data are symbolised as dotted boxes; $f\cdot$ is the functional relationship for the corresponding domain).

real value of the jth element of the ith sample in the target domain; $\hat{y}_{T,-i,j}$ is the predicted value of the jth element of the ith sample in the model without using the ith sample in the training samples of the target domain; and $\hat{y}_{T,i,j}$ is the predicted value of the jth element of the ith sample in the testing samples of the target domain. Note that in Scenario 1, the prediction ability of the source domains is not included because in this scenario, the target domain is typically associated with the in-operation processes, and source domains are associated with historical processes and thus not important compared to the target domain.

Scenario 2: multiple target domains and no source domain

$$\text{MAECV} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{n_k^{(train)}}\sum_{i=1}^{n_k^{(train)}}\sum_{j=1}^{d_Y}|y_{k,i,j}-\hat{y}_{k,-i,j}| \tag{20}$$

$$\text{MAE} = \frac{1}{K}\sum_{k}^{K}\frac{1}{n_k^{(test)}}\sum_{i=1}^{n_k^{(test)}}\sum_{j=1}^{d_Y}|y_{k,\,i,j}-\hat{y}_{k,i,j}| \tag{21}$$

where K is the number of the target domains; $n_k^{(train)}$, $n_k^{(test)}$ are the numbers of training samples and the testing samples of the k_{th} domain, respectively; and $y_{k,i,j}$, $\hat{y}_{k,-i,j}$ and $\hat{y}_{k,i,j}$ have similar meanings to Scenario 1. Note that in Scenario 2, because all domains are target domains, the prediction ability in every domain is considered in this multitask problem. The design of these two scenarios is shown in Figure 3, where Scenario 1 focuses on the transfer of historical information from the source domains, and Scenario 2 focuses on the shared information between multiple target domains.

The settings of the input data and the functional relationship are given below. In this study, the generation steps of the two designed scenarios are the same, and the differences are the number of training samples of each domain. If the number of samples from a particular domain is sufficient, then this domain is grouped as the source domain; otherwise, it is grouped as the target domain. In the following experiments, the number of domains is $K = 3$, and the total number of samples in each domain is $N_k = N = 100$. For the domains serving as the source domain, $n_k^{(train)} = N_k, n_k^{(test)} = 0$, and for the target domains, $n_k^{(train)} = p_{train}\cdot N_k, n_k^{(test)} = (1-p_{train})\cdot N_k$, where p_{train} is the proportion of the training samples for the target domains. The data matrices are generated by following the simulation study from the previous DA-JYPLS work (Jia, Zhang, and You 2020). The number of dimensions of the latent space for the input matrices is denoted as d_l. The rows of scoring matrices $\boldsymbol{T}_k \in \mathbb{R}^{N_k\times d_l}$s are generated with different Gaussian distributions, and the input data matrices $\boldsymbol{X}_k \in \mathbb{R}^{N_k\times d_{X_k}}$s are generated by using Equation (2). The output data matrices $\boldsymbol{Y}_k \in \mathbb{R}^{N_k\times d_Y}$s are obtained by employing Equation (1). In this study, elements of $\boldsymbol{P}_k \in \mathbb{R}^{d_{X_k}\times d_l}$s and $\boldsymbol{Q}_J \in \mathbb{R}^{d_Y\times d_l}$ are generated from standard Gaussian distributions, and elements of $\boldsymbol{E}_{X_k} \in \mathbb{R}^{N_k\times d_{X_k}}$s and $\boldsymbol{E}_{Y_J} \in \mathbb{R}^{N_k\times d_Y}$ are generated from Gaussian distributions with a 0 mean and 0.1 variance. The number of dimensions for matrices and the distribution of $\boldsymbol{T}_k$s in each domain are listed in Table 1. The similarity of $\boldsymbol{T}_k$s' distributions between domains stands for their similarity of the functional relationship of input and output variables according to this simulation setting. The number of latent variables in these PLS variants and PCR are set to d_l.

The results of Scenario 1 with different training proportions p_{train} are shown in Figure 4 and Table 2 with 100 replications.

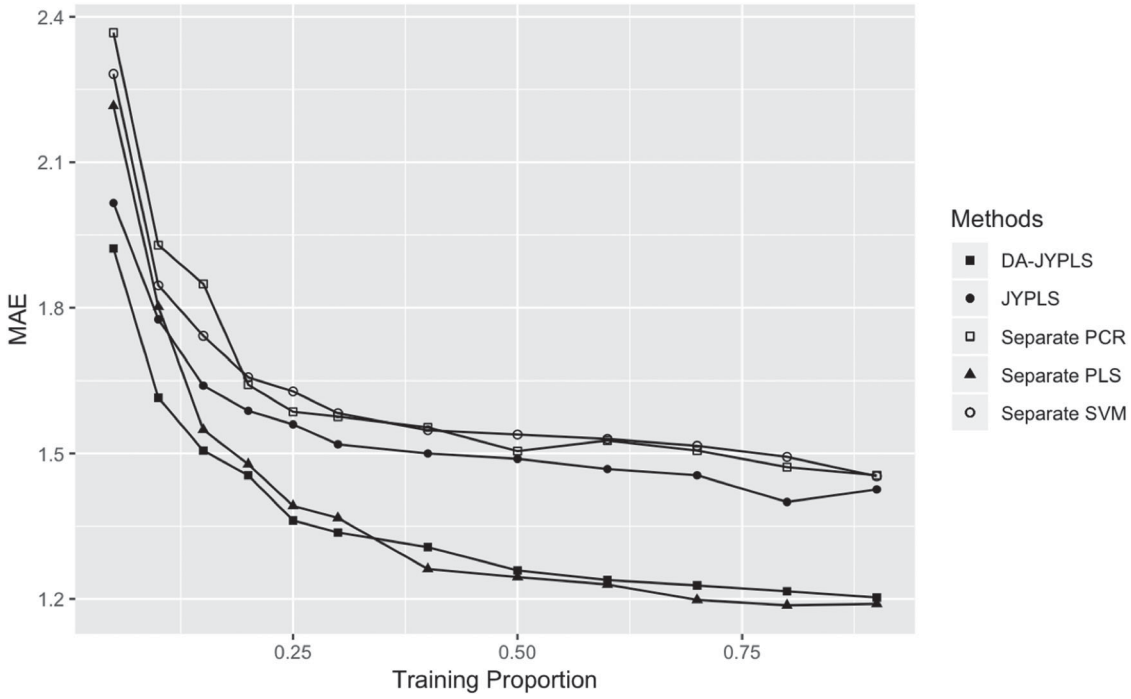

Figure 4. Mean MAEs with different training proportions produced by the five methods in Scenario 1.

Table 1. Some simulation settings of the domains.

Matrices	Domain 1 (S)	Domain 2 (S)	Domain 3 (T)
Distribution of $\boldsymbol{T}_k$	$N(0.9 \times 1^{d_l}, \boldsymbol{I}_{d_l})$	$N(1.1 \times 1^{d_l}, \boldsymbol{I}_{d_l})$	$N(1.0 \times 1^{d_l}, \boldsymbol{I}_{d_l})$
d_{X_k}	6	8	7
d_Y		2	
d_l		4	

It is shown that the prediction accuracy of the proposed method is greater than that of the other four methods when the training proportion is relatively small (i.e. $p_{train} \leq 0.3$). Note that the proposed method is better than the JYPLS method in all p_{train} values, which agrees with the theoretical advantages of the proposed method. Due to the penalty terms in Equation (5), the proposed method effectively learns the similarity between the domains. However, because the number of training samples is sufficient, the proposed model is marginally less accurate than the separate PLS model because a difference between the coefficients between domains exists, as shown in Equation (24). The potential reasons for this phenomenon are that when the number of training samples of the target domain is small, the effects of the learned similarity of the proposed model are greater than the effects of the differences of the coefficients between domains; the relationship also reverses when the number of training samples is sufficient to construct an accurate PLS model using only information from the target domain. Note that the designed functional relationship is linear, which is not complex, and the corresponding intersection point of the training proportion is between 0.30 and 0.35. In theory, the intersection

Table 2. Mean MAEs and standard deviations in parentheses with different training proportions produced by the five methods in Scenario 1.

p_{train}	JYPLS	DA-JYPLS	Separate PLS	Separate PCR	Separate SVM
0.05	2.016 (0.807)	**1.922 (1.235)**	2.216 (0.785)	2.367 (0.969)	2.282 (0.931)
0.1	1.776 (0.362)	**1.615 (0.556)**	1.802 (0.447)	1.929 (0.485)	1.846 (0.417)
0.15	1.640 (0.258)	**1.506 (0.392)**	1.549 (0.309)	1.849 (0.447)	1.742 (0.298)
0.2	1.588 (0.252)	**1.455 (0.403)**	1.478 (0.232)	1.642 (0.304)	1.657 (0.246)
0.25	1.560 (0.222)	**1.362 (0.398)**	1.392 (0.216)	1.586 (0.291)	1.628 (0.229)
0.3	1.519 (0.168)	**1.337 (0.338)**	1.367 (0.202)	1.576 (0.258)	1.583 (0.205)
0.4	1.500 (0.167)	1.307 (0.295)	**1.262 (0.152)**	1.554 (0.257)	1.548 (0.166)
0.5	1.489 (0.179)	1.259 (0.292)	**1.245 (0.179)**	1.505 (0.243)	1.539 (0.171)
0.6	1.468 (0.182)	1.239 (0.219)	**1.230 (0.157)**	1.527 (0.253)	1.530 (0.186)
0.7	1.455 (0.156)	1.228 (0.178)	**1.198 (0.142)**	1.506 (0.252)	1.516 (0.201)
0.8	1.400 (0.159)	1.216 (0.165)	**1.187 (0.162)**	1.472 (0.282)	1.493 (0.197)
0.9	1.426 (0.245)	1.223 (0.290)	**1.190 (0.239)**	1.455 (0.424)	1.453 (0.247)

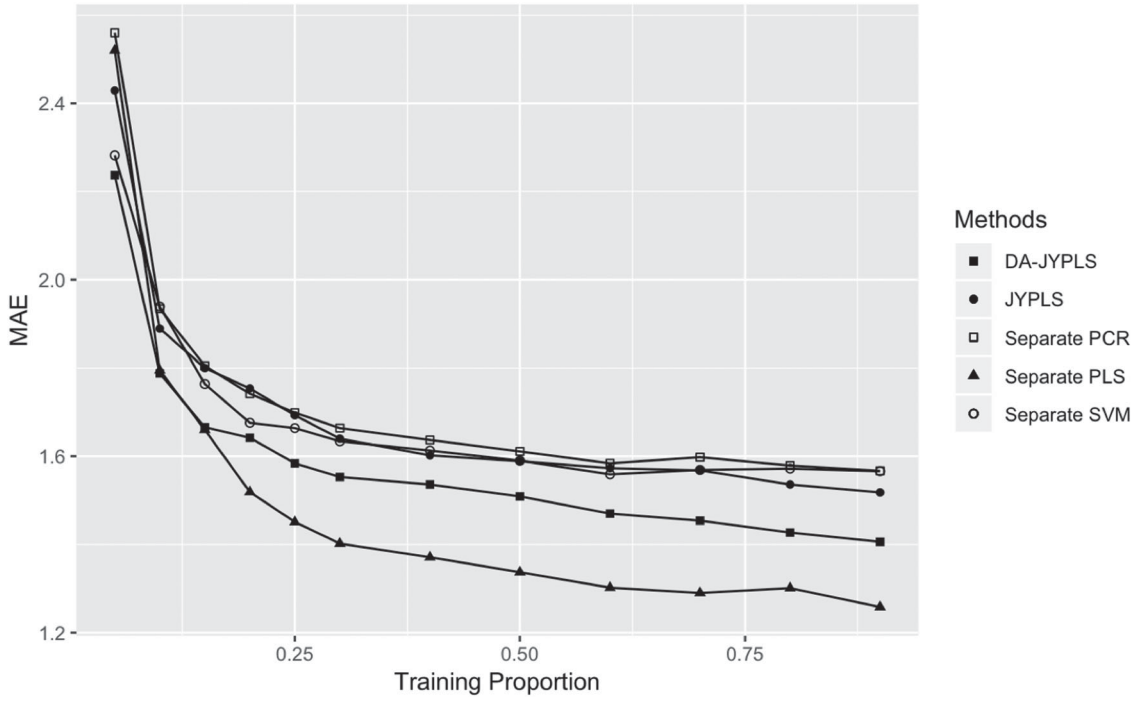

Figure 5. Mean MAEs with different training proportions produced by the five methods in Scenario 2.

point of the training proportion might be relevant to the corresponding absolute number of training samples and the other settings of the functional relationship. The variants of the PLS methods perform better than PCR and SVM in all p_{train} values, which might result from the fact that the simulation data matrices are generated according to the JYPLS model.

In Scenario 2 with multiple target domains, the settings are the same except all three domains are target domains (i.e. p_{train} is utilised in all domains). The results are shown in Figure 5 and Table 3. The overall trend of the MAE curves in Scenario 2 is similar to those in Scenario 1: the proposed method achieves the best performance when p_{train} is relatively small and achieves better performance than the JYPLS method in all p_{train} values. In Scenario 2, the difference between the proposed method and the JYPLS method is smaller than that in Scenario 1. The intersection point of p_{train} between the proposed method and the separate PLS model is also smaller than that in Scenario 1 (i.e. between 0.1 and 0.15). This result shows that the effects of domain adaptation decrease in cases with multiple target domains because there are fewer training samples in all domains and therefore less shared information to be learned. In general, the proposed method is effective in cases in which the available samples of the target domain are limited, specifically in cases where there are source domains to be learned from.

Recall that l_k is an important hyperparameter in the proposed method. In this work, sensitivity analysis of l_k is given to provide practical guidance on the selection of l_k. The settings and the data generation steps are the same as in Scenario 1. The training proportion of the target domain is set to $p_{train} = 0.15$, which is a comparatively suitable situation for DA-JYPLS to be utilised. To determine the potential influence of l_k, it is set to $l_k = p_{l_k} n_k^{(train)}$ in each domain. The results are summarised in Figure 6. The performance of the proposed method is better when p_{l_k} is neither too large nor too small; in this case, $p_{l_k}^* = 0.5$. Theoretically, p_{l_k} affects the chosen number of nearest neighbours and further influences the effect of the locality loss form in the optimisation problem. When p_{l_k} is too small, the chosen neighbours are not enough to reflect the local distribution of variables in the original feature space. On the other hand, when p_{l_k} is large, the local distribution becomes 'global' in the domain and keeping this global information within the domain would contradict the shared information learned from other domains. In practice, the method would be sensitive to outliers within domains when p_{l_k} is chosen to be too large.

4. Real example study

In this section, the proposed model is applied to the problem of wafer quality prediction to further demonstrate its effectiveness. The complete production process primarily consists of physical and chemical machining

Table 3. Mean MAEs and standard deviations in parentheses with different training proportions produced by the five methods in Scenario 2.

p_{train}	JYPLS	DA-JYPLS	Separate PLS	Separate PCR	Separate SVM
0.05	2.429 (0.476)	**2.237 (0.850)**	2.520 (0.719)	2.560 (0.583)	2.493 (0.356)
0.1	1.889 (0.269)	**1.788 (0.447)**	1.795 (0.263)	1.934 (0.248)	1.939 (0.225)
0.15	1.800 (0.185)	1.666 (0.370)	**1.660 (0.257)**	1.805 (0.213)	1.764 (0.152)
0.2	1.754 (0.165)	1.642 (0.365)	**1.519 (0.209)**	1.742 (0.183)	1.676 (0.125)
0.25	1.693 (0.141)	1.584 (0.390)	**1.451 (0.159)**	1.699 (0.170)	1.664 (0.134)
0.3	1.640 (0.137)	1.553 (0.327)	**1.402 (0.148)**	1.664 (0.150)	1.634 (0.120)
0.4	1.602 (0.138)	1.536 (0.270)	**1.371 (0.106)**	1.637 (0.131)	1.613 (0.105)
0.5	1.589 (0.126)	1.509 (0.204)	**1.337 (0.116)**	1.611 (0.104)	1.590 (0.093)
0.6	1.573 (0.121)	1.470 (0.242)	**1.302 (0.109)**	1.584 (0.125)	1.559 (0.110)
0.7	1.568 (0.125)	1.454 (0.256)	**1.290 (0.097)**	1.598 (0.136)	1.569 (0.119)
0.8	1.536 (0.129)	1.427 (0.249)	**1.301 (0.128)**	1.579 (0.158)	1.572 (0.117)
0.9	1.518 (0.196)	1.406 (0.267)	**1.258 (0.169)**	1.567 (0.191)	1.566 (0.153)

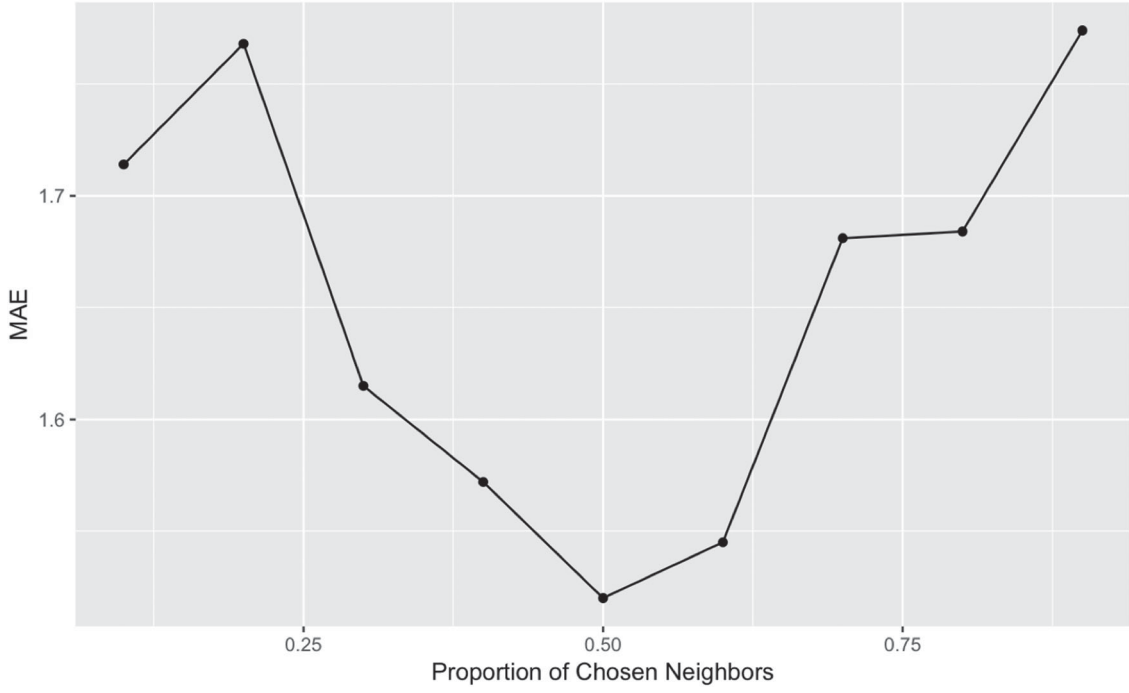

Figure 6. Mean MAEs with different p_{l_k} values in the proposed method.

and processing from single crystals of silicon into wafers, which are the fundamental raw materials in semiconductor manufacturing. The geometric shapes and certain physical properties of wafers have a direct effect on downstream production and are thus used to estimate wafer quality. Therefore, it is helpful to accurately predict these characteristics during the process. Unfortunately, due to the different physical properties of wafers, the relationships of certain process variables to the characteristics of each single crystal of silicon are similar but not indifferent. The number of training wafers in process from single crystals of silicon available to construct an effective model is expected to be small to reduce costs. Because the in-process wafers are typically from the same single crystal of silicon, this quality prediction problem is similar to Scenario 1 in Section 3. Therefore, this problem considers a single crystal of silicon as a batch (domain), and the single crystal of silicon in process is the target domain, while the finished single crystals of silicon serve as the source domains. A brief introduction of the wafer production process is shown in Figure 7. The process consists of five primary stages, and a quality inspection step is performed after each stage, which generates the observations of quality variables.

With the available data, there is historical information from two single crystals of silicon, which are denoted as Domain S-1 and Domain S-2 and contain 389 and 379 wafers, respectively. There are also 100 observations for the single crystal of silicon in process, which is denoted as Domain T. In this study, 6 important quality indices that are evaluated after the final stage are considered to be the output variables, most of which are defined by Semiconductor Equipment and Materials International as industrial standards; these indices include Centre

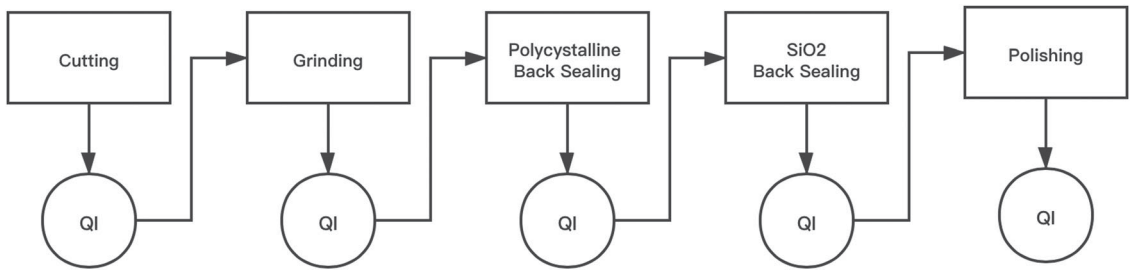

Figure 7. Wafer production process. QI: Quality Inspection.

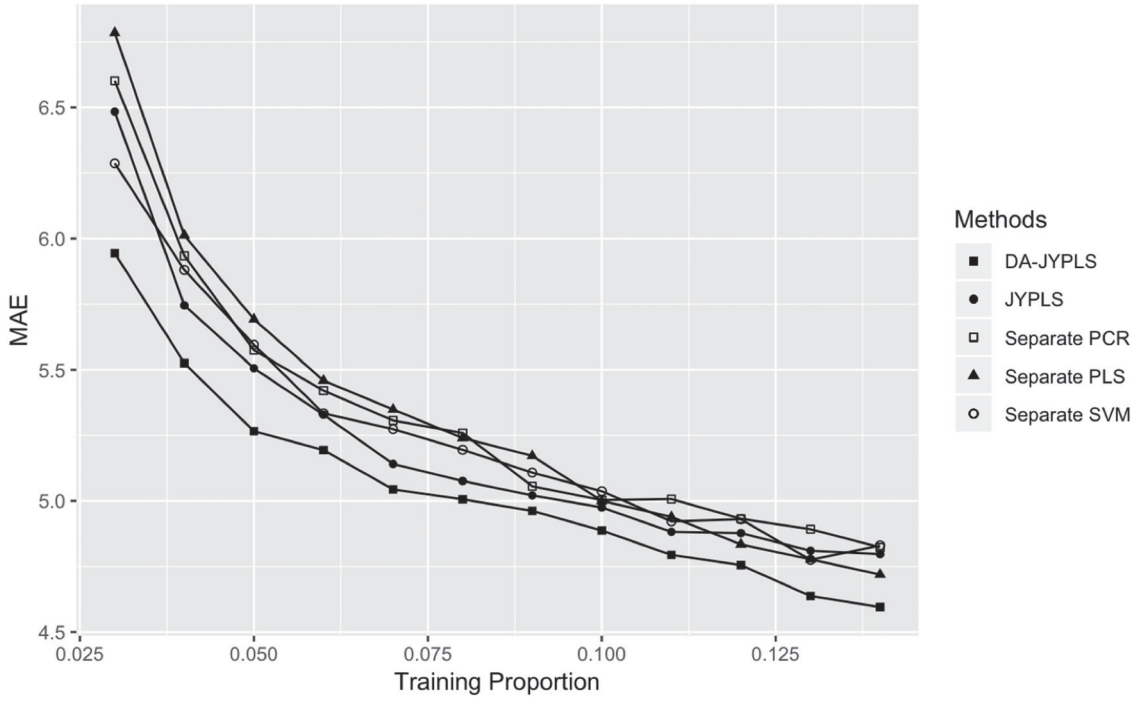

Figure 8. Mean MAEs with different training proportions produced by the three methods in the real wafer-production process.

Thickness (CTRTHK), Total Thick Variation (TTV), Total Indicator Reading (TIR), Site Total Indicator Reading (STIR), Bow and Warp. There are missing data and omitted quality inspections, and therefore, the corresponding number of input variables inspected in the first four stages of the domains are different, which are listed in Table 4. The quality variables inspected in the first four stages contain certain geometric and physical properties, such as taper and resistivity. Note that the different circumstances of the available input variables correspond to the heterogeneous assumption that the domains have different input feature spaces. In the conclusion of the numerical experiments, the advantages of the proposed method should be apparent in situations with fewer training samples in the target domain. Therefore, to evaluate the proposed model, the chosen range of the training proportion of Domain T is from 0.03–0.15 with a step length equal to 0.01, and all samples from Domain S-1 and Domain S-2 are used as the training samples of the source domains. The results are shown in Figure 8 and Tables 5 and 6, where the hyperparameters are chosen as follows: the bandwidth parameter in Equation (14) is $\sigma_{S_1}^2 = \sigma_{S_2}^2 = \sigma_T^2 = 1$.

Table 4. Available number of input variables from each domain.

Domain Index	Available Stages	Number of quality variables per stage	Total number of input variables
S-1	1–4	12	48
S-2	1–4	15	60
T	1–3	15	45

The results show that the proposed method enhances the prediction performances compared to the other methods in the chosen range of the training proportions of the source data. Comparisons of the first extracted latent feature's experimental distributions of the three domains are shown in Figure 9. The proposed method succeeds in making observations from different domains

Table 5. Mean MAEs and standard deviations in parentheses produced by the five methods in the real wafer production process.

p_{train}	JYPLS	DA-JYPLS	Separate PLS	Separate PCR	Separate SVM
0.03	6.484 (1.32)	**5.944 (1.60)**	6.785 (1.34)	6.602 (1.20)	6.287 (1.14)
0.04	5.745 (0.98)	**5.525 (1.48)**	6.012 (0.96)	5.934 (0.93)	5.880 (1.02)
0.05	5.506 (0.87)	**5.266 (1.44)**	5.692 (0.80)	5.574 (0.77)	5.595 (0.84)
0.06	5.329 (0.76)	**5.194 (1.20)**	5.459 (0.74)	5.421 (0.69)	5.334 (0.90)
0.07	5.141 (0.70)	**5.044 (1.14)**	5.349 (0.68)	5.307 (0.74)	5.274 (0.70)
0.08	5.076 (0.72)	**5.006 (0.86)**	5.240 (0.64)	5.259 (0.69)	5.195 (0.78)
0.09	5.021 (0.68)	**4.962 (0.75)**	5.172 (0.61)	5.056 (0.64)	5.108 (0.72)
0.1	4.975 (0.70)	**4.887 (0.73)**	5.000 (0.59)	5.003 (0.65)	5.037 (0.66).
0.11	4.882 (0.69)	**4.794 (0.70)**	4.939 (0.58)	5.007 (0.69)	4.922 (0.61)
0.12	4.877 (0.65)	**4.755 (0.72)**	4.834 (0.55)	4.932 (0.65)	4.930 (0.66)
0.13	4.810 (0.69)	**4.637 (0.89)**	4.777 (0.55)	4.892 (0.66)	4.775 (0.64)
0.14	4.797 (0.66)	**4.595 (0.69)**	4.719 (0.56)	4.823 (0.68)	4.830 (0.70)

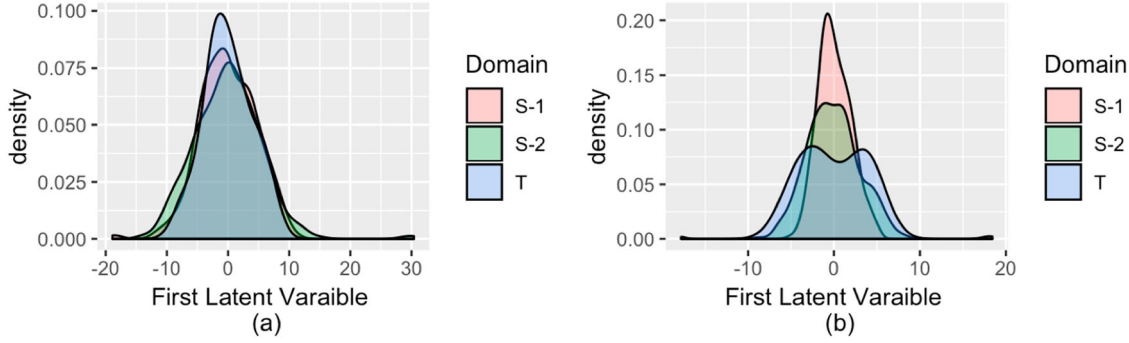

Figure 9. Comparisons of the empirical distributions between the source and target domains in the latent variable space in the wafer example when $p_{train} = 0.1$. (a) The proposed multisource DA-JYPLS method, (b) JYPLS method.

Table 6. R^2_{pred}s of the PLS variants for the wafer production process when $p_{train} = 0.1$.

	JYPLS	DA-JYPLS	Separate PLS
R^2_{pred}	0.740	**0.751**	0.737

that are then distributed similarly in the latent space compared to JYPLS. When physical processes are also similar in producing different batches (domains) of wafers, more shared information can be learned because of the similarity of distributions, and therefore, the performance of quality prediction can be improved. With many hyperparameters that need to be tuned, the proposed method is indeed less stable since the deviation of MAEs is relatively larger than that of the other methods. However, this decrease in stability of the proposed method is acceptable from an overall perspective of the better performance achieved in the mean of MAEs.

Based on these analyses of the real-world problem, when training samples of the in-process batch are limited in the small-batch processes, the proposed model achieves better performance by learning the shared information between batches than some other methods without using historical information.

5. Conclusions

In this study, a multisource domain adaptation JYPLS approach is proposed to predict quality during small-batch production processes. The proposed method is an extension of the original DA-JYPLS model that uses multiple numbers of source and target domains, which is more meaningful in real-world applications. Additionally, a regularised term of maintaining the local properties within each domain is added to make the proposed model robust in various situations. The results of numerical experiments and a real-world case study about wafer production verify the efficiency of the proposed method in situations with few training samples in the target domain compared to JYPLS and the traditional PLS models. In future research, certain extensions of the model should be developed to construct nonlinear functional relationships to fit more complex processes.

Acknowledgements

The authors thank the editor and referees for helping to improve this article.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by National Natural Science Foundation of China [grant numbers 71932006 and 71731008].

Notes on contributors

Dengyu Li is currently a PhD student at Department of Industrial Engineering, Tsinghua University. He received his B.S. degree in Industrial Engineering from Tsinghua University in 2020. His research focuses on data-driven methods for quality prediction and control.

Kaibo Wang is a professor in the Department of Industrial Engineering, Tsinghua University, Beijing, China. He received his BS and MS degrees in Mechatronics from Xi'an Jiaotong University, Xi'an, China, and his PhD in Industrial Engineering and Engineering Management from the Hong Kong University of Science and Technology, Hong Kong. His research focuses on statistical quality control and data-driven system modelling, monitoring, diagnosis, and control, with a special emphasis on the integration of engineering knowledge and statistical theories for solving problems from the real industry.

Data availability statement

The data is not available due to commercial restrictions. Due to the nature of this research, participants of this study did not agree for their data to be shared publicly, so supporting data is not available.

ORCID

Kaibo Wang http://orcid.org/0000-0001-9888-4323

References

Boser, B., I. Guyon, and V. Vapnik. 1996. "A Training Algorithm for Optimal Margin Classifier." *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* 5, doi:10.1145/130385.130401.

Cheng, L., F. Tsung, and A. Wang. 2017. "A Statistical Transfer Learning Perspective for Modeling Shape Deviations in Additive Manufacturing." *IEEE Robotics and Automation Letters* 1–1. doi:10.1109/LRA.2017.2713238.

Chu, F., X. Cheng, R. Jia, F. Wang, and M. Lei. 2018. "Final Quality Prediction Method for New Batch Processes Based on Improved JYKPLS Process Transfer Model." *Chemometrics and Intelligent Laboratory Systems* 183), doi:10.1016/j.chemolab.2018.10.004.

De Vin, L., J. De Vries, and T. Streppel. 2000. "Process Planning for Small Batch Manufacturing of Sheet Metal Parts." *International Journal of Production Research* 38: 4273–4283. doi:10.1080/00207540050205082.

Duan, L., D. Xu, and S. Chang. 2012. *Exploiting Web Images for Event Recognition in Consumer Videos: A Multiple Source Domain Adaptation Approach*.

Garcia-Munoz, S., J. MacGregor, and T. Kourti. 2005. "Product Transfer Between Sites Using Joint-Y PLS." *Chemometrics and Intelligent Laboratory Systems* 79: 101–114. doi:10.1016/j.chemolab.2005.04.009.

Geladi, P., and B. R. Kowalski. 1986. "Partial Least-Squares Regression: A Tutorial." *Analytica Chimica Acta* 185: 1–17. doi:10.1016/0003-2670(86)80028-9.

Gopalan, R., R. Li, and R. Chellappa. 2011. *Domain Adaptation for Object Recognition: An Unsupervised Approach*.

Grubinger, T., A. Birlutiu, H. Schöner, T. Natschläger, and T. Heskes. 2017. "Multi-Domain Transfer Component Analysis for Domain Generalization." *Neural Processing Letters* 46: 845–855. doi:10.1007/s11063-017-9612-8.

Hoerl, A. E., and R. W. Kennard. 1970. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics* 8: 27–51.

Jia, R., S. Zhang, and F. You. 2020. "Transfer Learning for End-product Quality Prediction of Batch Processes Using Domain-adaption Joint-Y PLS." *Computers & Chemical Engineering* 140: 106943. doi:10.1016/j.compchemeng.2020.106943.

Linares, G., and M. V. Mederos. 1981. "Principal Components Regression." *Revista Ciencias Matemáticas* 2: 51–62.

Long, M., J. Wang, G. Ding, S. J. Pan, and P. S. Yu. 2013. "Adaptation Regularization: A General Framework for Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering* 26: 1076–1089.

Long, M., J. Wang, G. Ding, S. Pan, and P. Yu. 2014. "Adaptation Regularization: A General Framework for Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering* 26: 1076–1089. doi:10.1109/TKDE.2013.111.

Long, M., J. Wang, G. Ding, J. Sun, and P. Yu. 2013. *Transfer Feature Learning with Joint Distribution Adaptation*.

Muandet, K., D. Balduzzi, and B. Schölkopf. 2013. *Domain Generalization via Invariant Feature Representation*. ICML.

Nikzad, R., W. Zellinger, E. Lughofer, and S. Saminger-Platz. 2018. "Domain-Invariant Partial Least Squares Regression." *Analytical Chemistry* 90, doi:10.1021/acs.analchem.8b00498.

Pan, S., I. Tsang, J. Kwok, and Q. Yang. 2011. "Domain Adaptation via Transfer Component Analysis." *IEEE Transactions on Neural Networks* 22 (2): 199–210.

Rubner, Y., C. Tomasi, and L. J. Guibas. 2000. "The Earth Mover's Distance as a Metric for Image Retrieval." *International Journal of Computer Vision* 40 (2): 99–121. doi:10.1023/A:1026543900054.

Smola, A., A. Gretton, L. Song, and B. Schlkopf. 2007a. "A Hilbert Space Embedding for Distributions." Paper presented at the International Conference on Algorithmic Learning Theory.

Smola, A., A. Gretton, L. Song, and B. Schölkopf. 2007b. *A Hilbert Space Embedding for Distributions*.

Sun, B., and K. Saenko. 2016. *Deep CORAL: Correlation Alignment for Deep Domain Adaptation*.

Wang, Z., Q. Liu, H. Chen, and X. Chu. 2020. "A Deformable CNN-DLSTM Based Transfer Learning Method for Fault

Diagnosis of Rolling Bearing Under Multiple Working Conditions." *International Journal of Production Research*, 1–15. doi:10.1080/00207543.2020.1808261.

Wang, C., and S. Mahadevan. 2011. *Heterogeneous Domain Adaptation Using Manifold Alignment*.

Wang, K., and F. Tsung. 2020. "Bayesian Cross-product Quality Control via Transfer Learning." *International Journal of Production Research*, 1–19. doi:10.1080/00207543.2020.1845413.

Weiss, K., T. Khoshgoftaar, and D. Wang. 2016. "A Survey of Transfer Learning." *Journal of Big Data* 3, doi:10.1186/s40537-016-0043-6.

Yao, Y., Y. Zhang, X. Li, and Y. Ye. 2020. "Discriminative Distribution Alignment: A Unified Framework for Heterogeneous Domain Adaptation." *Pattern Recognition* 101: 107165. doi:10.1016/j.patcog.2019.107165.

Yin, S., S. Ding, A. Sari, and H. Hao. 2012. "Data-driven Monitoring for Stochastic Systems and its Application on Batch Process." *International Journal of Systems Science – IJSySc* 44: 1–11. doi:10.1080/00207721.2012.659708.

Zhang, S., F. Wang, D. He, and R. Jia. 2011. "Real-time Product Quality Control for Batch Processes Based on Stacked Least-squares Support Vector Regression Models." *Computers & Chemical Engineering* 36, doi:10.1016/j.compchemeng.2011.05.015.

Zhao, H., S. Zhang, G. Wu, J. Costeira, J. Moura, and G. Gordon. 2017. *Multiple Source Domain Adaptation with Adversarial Training of Neural Networks*.