AdaPipe: A Recommender System for Adaptive Computation Pipelines in Cyber-Manufacturing Computation Services

Xiaoyu Chen and Ran Jin*
Grado Department of Industrial and Systems Engineering, Virginia Tech, USA

***Abstract*—The industrial cyber-physical systems (ICPS) will accelerate the transformation of of ine data-driven modeling to fast computation services, such as computation pipelines for prediction, monitoring, prognosis, diagnosis, and control in factories. However, it is computationally intensive to adapt computation pipelines to heterogeneous contexts in ICPS in manufacturing. In this paper, we propose to rank and select the best computation pipelines to match contexts and formulate the problem as a recommendation problem. The proposed method Adaptive computation Pipelines (AdaPipe) considers similarities of computation pipelines from word embedding, and features of contexts. Thus, without exploring all computation pipelines extensively in a trial-and-error manner, AdaPipe ef ciently identi es top-ranked computation pipelines. We validated the proposed method with 60 bootstrapped data sets from three real manufacturing processes: thermal spray coating, printed electronics, and additive manufacturing. The results indicate that the proposed recommendation method outperforms traditional matrix completion, tensor regression methods, and a state-of-the-art personalized recommendation model.**

***Index Terms*—Computation pipeline, computing in cyber-physical systems, recommender system, smart factories.**

I. Introduction

An industrial cyber-physical system (ICPS) interconnects sensors, actuators, and many manufacturing equipment into a network, and integrates ubiquitous computation resources, such as Fog and Cloud to support data-driven decision-making [1]. The objective of ICPS in manufacturing is to improve efficiency and quality, control the cost, while enabling the flexibility to meet highly personalized manufacturing product and service needs. In order to provide effective data-driven decision-making supports, traditional offline data-driven modeling and statistical learning methods should be transformed to fast computation services for various objectives, such as fast quality modeling and prediction [2], monitoring [3], prognosis [4] and diagnosis [5] in ICPS.

Computation service is a concept originated from large-scale computation, such as cloud computing [6] and distributed computing [7]. The objective of computation services is to provide computation capability and algorithms as services to support data storage and analytics needs in various fields, such as cloud manufacturing [8], pervasive healthcare [9], large-scale deep learning [10], etc. Most of the mainstream computation services such as Apache Flink [11], its extension Alibaba Blink [12], and LARS [13] focus on providing the framework to support real-time computing.

In ICPS, the computation services must be accurate, reliable, responsive, and interoperable to be adaptive to heterogeneous manufacturing contexts. In this research, we define contexts as the contextualized data sets, frequently changed manufacturing settings (e.g., replacement of manufacturing equipment, changed manufacturing recipe), and customized manufacturing and computation specifications [1]. These frequently changed manufacturing contexts may cause sub-optimal computation algorithms due to the violation of assumptions. These algorithms directly lead to inaccurate predictions, which may result in out-of-control systems and cause major manufacturing failures and irreparable loss.

For example, a violation of *i.i.d.* assumption for modeling and prediction leads to high prediction error; and a violated assumption of underline distribution for a control chart in process monitoring may lead to high false alarm and mis-detection rate. Consequently, existing models and control charts can no longer be reliable. As another example, a selective laser melting (SLM) process requires high speed modeling algorithms to provide layer-to-layer quality prediction, which allows limited time to extensively explore all computation algorithms for prediction purpose. The time latency requirements may not be satisfied by a centralized algorithm [14].

For the same type of computation services in manufacturing, there are abundant choices of data analytical method options, such as data filtering and compression [15], dimension reduction [16], feature extraction [17], modeling methods [18] etc. Researches have been reported towards the objective of developing mathematical models based on sensor data to improve the quality and reliability of manufacturing processes since 1970s [19]. However, a typical paradigm for developing an effective data analytical method option is based on engineering knowledge for a specific manufacturing process and/or one's data-driven modeling experience, which requires large amount of trials. As a result, the inefficient trial-and-error studies prevent data analytics from being autonomous and responsive to be deployed in ICPS.

As a systematic way to explore existing data analytics

method options, a computation pipeline (pipeline for brief) is proposed as a sequence of method options from multiple steps [20] (see Figure 1). In this pipeline, the output from method option in Step-i is directly used as the input for method option in Step-$(i+1)$. Here *Step* is a collection of existing method options with the same functionality (e.g., feature extraction step as a collection of method options). By executing all candidate pipelines, the best pipeline associated with the lowest normalized root mean squared error (NRMSE) can be identified (i.e., highlighted in bold blue lines). However, exploring all pipelines leads to huge computation workload, while arbitrarily executing a few pipelines may not provide optimal performance. Thus, an efficient pipeline selection method is needed to minimize the optimal gap between the selected pipeline and the underline best pipeline.

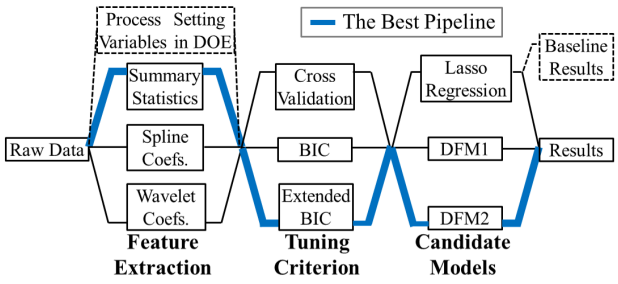


Fig. 1: Computation pipelines for modeling and prediction with three steps and three method options in each step [20]. The best pipeline is highlighted in bold blue.

In literature, automated machine learning (AutoML) methods have been investigated aiming at automatically building machine learning applications without extensive knowledge of statistics and machine learning in the last decade [21]. AutoML methods tackled the selection of machine learning methods (i.e., neural networks) in the format of computation pipelines or computation graphs from different perspectives. Most of AutoML methods focus on automatic hyperparameter optimization to automatically select best hyperparameters for neural networks [22], and neural architecture search to automate the design of architecture [23]. These methods promoted several commercial tools, such as Auto-WEKA [24], Auto-Sklearn [25], etc. However, the aforementioned methods adopt either greedy or random searching methods, hence requiring hours or even days to find a satisfactory computation pipeline.

Recently, collaborative filtering methods have been adopted to speedup the selection of computation pipelines for new data sets. Taking OBOE as an example [26], it defined a matrix of cross-validated errors of supervised learning models and data sets, and proposed to complete the missing results for a new data set by using a time-constrained matrix completion model. OBOE is designed to start from test runs on several initial models on new data set to convert a cold-start matrix completion problem to warm-start by filling in some entries for the empty row. It then sequentially executes models on the new data set based on design of experiment and update the completed matrix. However, directly providing a good computation model by predicting exact cross-validated errors may be biased by potential violation of the low-rank assumption for error matrix (e.g., the existence of outliers). Instead of sequentially predicting cross-validated errors, ranking the method options by pairwise comparisons can be more informative for users.

Therefore, in this research, we formulate the selection of pipeline as a recommendation problem to rank and suggest the pipelines based on performance. The objective is to efficiently rank and recommend the best pipeline associated with the best performance (i.e., lowest prediction error) to be adaptive to frequently changing manufacturing contexts. Therefore, we propose **AdaPipe**, which is a recommender system for **Ada**ptive computation **Pipe**lines in ICPS computation services. AdaPipe defines a sparse response matrix, where each row and each column correspond to a data set and a pipeline (i.e., one path in Figure 1 from data sourcing to models), respectively. And the (i, j)-th entry is defined as the statistical performance (e.g., prediction errors, time latency, etc.) of analyzing the i-th data set by using the j-th pipeline. This matrix is sparse in two scenarios, namely, (S1) arbitrarily missing entries: not all pipelines have been tested on one existing data set given limited time for trial-and-error modeling; and (S2) missing entire rows: a data set which is new to the recommender system has not been analyzed on any pipelines. Here S1 is a typical assumption of matrix completion (MC) and matrix factorization [27]. S2 is the well-known *cold-start* problem [28]. Different from traditional MC methods [27], AdaPipe makes recommendation by quantifying not only the implicit similarity among entries in the sparse response matrix, but also the explicit similarity from covariates (i.e., dense representation of method options and meta data). Thus, AdaPipe is expected to support computation services in an ICPS by efficiently providing accurate data analytics.

Different from MC in collaborative filtering [27], AdaPipe makes recommendation by quantifying not only the implicit similarity among entries in the sparse response matrix, but also the explicit similarity from covariates (i.e., dense representation of pipelines and meta data). Therefore, it contributes to current recommender systems in the following aspects: (1) AdaPipe uses unstructured descriptions to improve recommendation accuracy; (2) it does not assume unknown entries

to be arbitrarily missed; (3) it does not require large samples size (i.e., number of entries in sparse response matrix), due to the existence of covariates and the l-1 penalization; and (4) it is scalable when considering higher mode tensor covariates. On the other hand, it contributes to computation services in ICPS by (1) adapting computation pipelines to ICPS contexts by efficiently suggesting the best computation pipelines; and (2) enabling the flexibility in customized performance metrics (e.g., prediction error, time latency, weighted combination, etc.) according to computation needs. Thus, the proposed AdaPipe system is expected to support computation services in ICPS by providing accurate and responsive data analytics. AdaPipe differs from PRIME model [29] in two aspects: (1) AdaPipe adopt pairwise loss function for better ranking performance in computation pipeline recommendation problem; and (2) AdaPipe generalizes PRIME by adopting covariates tensor, which contains more information than covariates that are in vector format since it introduces interaction between data sets and computation pipelines via outer product.

The rest of this paper is organized as follows. In Section II, the proposed AdaPipe system is introduced. A case study in thermal spray coating (TSC), Aerosol jet printing (AJP), and fused deposition modeling (FDM) processes to validate AdaPipe is discussed in Section III, followed by results and discussion in Section IV. And conclusions are drawn in Section V.

II. The Proposed AdaPipe System

AdaPipe System consists of a tensor regression-based extended matrix completion (TEMC) model and two covariates generation machines. As presented in Figure 2, the TEMC model takes the covariates tensor and the sparse response matrix as input to complete the response matrix and further providing the ranking and recommendation of pipelines for both existing and new data sets. AdaPipe system assumes that: the text descriptions for a method option and the comparison results with benchmarks are available; pipelines share the same steps for a certain type of computation services; and the sparse response matrix to be completed has a linear relationship with a low-rank matrix and covariates.

A. TEMC Model

We firstly define and summarize the notations in Table I. Here, meta data $\boldsymbol{d}_i$ is defined as a vector of summary statistics extracting from $\mathbb{D}_i$, where $i = 1, 2, ..., m$; Embedded vector $\boldsymbol{e}_{j,k}$ is extracted via word2vec [30] for the method option in the j-th pipeline at the k-th step, where $j = 1, 2, ..., n$, $k = 1, 2, ..., K$, and dimension q is 50 in this research. Therefore, the j-th pipeline $\mathbb{P}_j$ can be represented as a informative dense vector $\boldsymbol{e}_j$ in real space, by concatenating the method option vector in a pre-defined order of steps: $\boldsymbol{e}_j = \text{concate}(\boldsymbol{e}_{j,1}, \boldsymbol{e}_{j,2}, ..., \boldsymbol{e}_{j,K})$. Based on these settings, the interaction (i.e., covariates) between the i-th data sets and the j-th pipelines is defined as the outer product of the i-th meta data vector $\boldsymbol{d}_i$ and the j-th pipeline vector $\boldsymbol{e}_j$: $\mathcal{X}_{:,:,i,j} = d_i \otimes \boldsymbol{e}_j$. Note that the covariates tensor $\mathcal{X}$ is not limited to four modes, other information related to the data set or pipeline can be incorporated as vectors in real space, which results in larger number of modes for $\mathcal{X}$. To formulate TEMC model, we define the (i, j)-th entry $y_{i,j}$ as the statistical performance for modeling the i-th data set by using the j-th pipeline. Hence, TEMC model is proposed as Equation (II.1):

$$\boldsymbol{Y} = \boldsymbol{R} + \langle \mathcal{B}, \mathcal{X} \rangle + \boldsymbol{E}, \tag{II.1}$$

where low-rank matrix $\boldsymbol{R}$ represents the similarity among the statistical performances in $\boldsymbol{Y}$; $\langle \mathcal{B}, \mathcal{X} \rangle = \mathcal{C} \in \mathbb{R}^{m,n}$, where $c_{i,j} = \sum_{k=1}^{p} \sum_{l=1}^{qK} \mathcal{B}_{k,l} \mathcal{X}_{k,l,i,j}$. In this way, the sparse response matrix $\boldsymbol{Y}$ is completed as $\widehat{\boldsymbol{Y}}$ by estimating $\widehat{\boldsymbol{R}}$ and $\widehat{\mathcal{B}}$. The key idea of this model is to explain the similarities among the data sets and pipelines by decomposing $\boldsymbol{Y}$ into a low-rank matrix $\boldsymbol{R}$ to quantify the implicit similarity, and a tensor regression (TR) term $\langle \mathcal{B}, \mathcal{X} \rangle$ to quantify the explicit similarity.

B. Proposed Estimator with Pairwise Loss

TEMC model is estimated by Problem (II.2):

$$\min L(\widehat{\boldsymbol{Y}}, \boldsymbol{Y}), \text{s.t. } \|\boldsymbol{R}\|_* \leq s, \ \|\mathcal{B}\|_1 \leq t, \tag{II.2}$$

where $L(\widehat{\boldsymbol{Y}}, \boldsymbol{Y})$ is a loss function, such as least square loss, pairwise loss [31], etc.; nuclear norm $\|\cdot\|_*$ is

TABLE I: Summary of notations

Notations	Definitions
$\mathbb{D}_i$	i-th raw data set
$\boldsymbol{d}_i$	i-th meta data $\mathbb{R}^p$
m, n, K	m data sets, n pipelines, K steps
p	Dimension of summary statistics
$\mathbb{P}_j$	j-th pipeline
$\boldsymbol{e}_{j,k}$	(j,k)-th embedded vector $\mathbb{R}^q$
q	Dimension of embedded vector
$\boldsymbol{e}_j$	j-th embedded pipeline
$\mathcal{X}$	Covariates tensor $\mathbb{R}^{p \times qK \times m \times n}$
$\boldsymbol{Y}$	Sparse response matrix $\mathbb{R}^{m \times n}$
$\boldsymbol{R}$	Low-rank matrix $\mathbb{R}^{m \times n}$
$\mathcal{B}$	Regression Coef. $\mathbb{R}^{p \times qk}$
$\boldsymbol{E}$	Error matrix $\mathbb{R}^{m \times n}$
$\mathcal{P}_\Omega(\cdot)$	Selector of non-empty entries
$vec(\cdot)$	Vectorization operator
$\langle \cdot, \cdot \rangle$	Inner product in real space
$\otimes$	Outer product $\boldsymbol{d}_i \otimes \boldsymbol{e}_j = \boldsymbol{d}_i \boldsymbol{e}_j^{\mathrm{T}}$
$\lambda_1, \lambda_2, s, t$	Tuning parameters
$\|\|\cdot\|\|_*, \|\|\cdot\|\|_1$	Nuclear norm and l-1 norm
$\mathcal{S}_\tau(\cdot)$	Sigular value soft-thresholding
$\mathcal{T}_\tau(\cdot)$	Wavelet thresholding

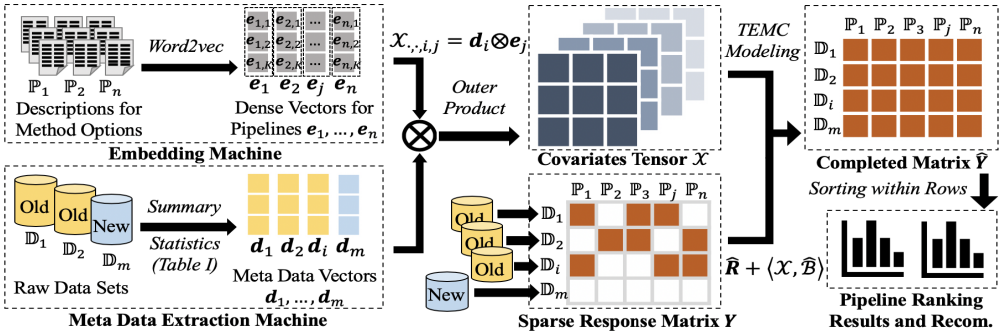


Fig. 2: An overview of AdaPipe system, which takes the vectorized pipelines and meta data as input to generate covariates in a tensor format. TEMC then predicts missing entries based on both the sparse response matrix and the covariates. In the end, the pipelines are ranked and suggested according to the completed matrix.

computed as $\|\boldsymbol{R}\|_* = \sum_{i=1}^{\min(m,n)} \sigma_i(\boldsymbol{R})$, here $\sigma_i(\boldsymbol{R})$ is the i-th singular value of $\boldsymbol{R}$ after performing singular value decomposition, to enforce low-rank structure of $\boldsymbol{R}$; l-1 norm $\|\cdot\|_1$ is computed as $\|\mathcal{B}\|_1 = \sum_{i=1}^{p}\sum_{j=1}^{qK} |\mathcal{B}_{i,j}|$ to control the sparsity of model coefficients $\mathcal{B}$; $s \geq 0$ and $t \geq 0$ are tuning parameters to control the amount of shrinkage, which can be selected by using cross validation. Specifically, if s becomes larger, implicit similarity will be smaller; and if t becomes larger, more covariates will be selected as significant factors.

We further investigate the form of the loss function $L(\widehat{\boldsymbol{Y}}, \boldsymbol{Y})$ in consideration of accurately ranking pipelines $\mathbb{P}_i$ for each data set. Pairwise loss function was reported to be used for *learning-to-rank* problem in information retrieval communities [31]. Therefore, we adopted pairwise loss (see Function (II.3)) in the proposed estimator to both consider pairwise comparison for higher ranking accuracy, which will be compared with least square loss in Section III.

$$\min_{\widehat{\mathcal{B}},\widehat{\boldsymbol{R}}} \sum_{i=1}^{m} \frac{1}{\Omega_i} \sum_{u}^{\Omega_i} \sum_{v}^{\Omega_i} [(y_{i,u} - y_{i,v}) - (\widehat{y}_{i,u} - \widehat{y}_{i,v})]^2, \tag{II.3}$$

where Ω_i is the index set of non-empty entries in the i-th row of sparse response matrix $\boldsymbol{Y}$ to standardize the pairwise loss for the i-th data set; and $|\cdot|$ is the cardinality of a set. Adopting pairwise loss provides the users with more informative recommendation results by ranking computation pipelines, especially when comparing with collaborative filtering methods that aimed at predicting exact cross-validated errors for methods.

C. ADMM Algorithm

To derive an efficient algorithm, We firstly show that such a pairwise loss function is equivalent to a quadratic matrix form in Proposition II.1. Therefore, close form solutions can be derived for the sub-problems detailed in Section III.C.

Proposition II.1. *Function (II.3) is equivalent to a quadratic matrix form, which is convex:* $[vec(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})]^{\mathrm{T}} \mathbb{L} [vec(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})]$, *given* $\mathbb{L} = \begin{bmatrix} \frac{1}{\Omega_1} L & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\Omega_m} L \end{bmatrix}$, $L_{u,v} = \begin{cases} \Omega_1 - 1, & \text{if } u = v \\ -1, & \text{otherwise} \end{cases}$. *See proof in Appendix A.*

Motivated by alternating direction method of multipliers (ADMM) [32], the augmented Lagrangian function is given by: $\mathcal{L}(\boldsymbol{R}, \mathcal{B}, \boldsymbol{C}, \mathcal{D}, \boldsymbol{U}, \mathcal{V}) = \frac{\mu_1}{2} \|\boldsymbol{R} - \boldsymbol{C}\|_{\mathrm{F}}^2 + \frac{\mu_2}{2} \|\mathcal{B} - \mathcal{D}\|_{\mathrm{F}}^2 + [vec(\mathcal{P}_\Omega(\boldsymbol{Y}) - \mathcal{P}_\Omega(\widehat{\boldsymbol{Y}}))]^{\mathrm{T}} \mathbb{L} [vec(\mathcal{P}_\Omega(\boldsymbol{Y}) - \mathcal{P}_\Omega(\widehat{\boldsymbol{Y}}))] + \lambda_1 \|\boldsymbol{C}\|_* + \lambda_2 \|\mathcal{D}\|_1 + \langle \boldsymbol{U}, \boldsymbol{R} - \boldsymbol{C} \rangle + \langle \mathcal{V}, \mathcal{B} - \mathcal{D} \rangle$, where $\boldsymbol{C} = \boldsymbol{R}$ and $\mathcal{D} = \mathcal{B}$ are two linear constraints associated with two dual variables $\boldsymbol{U}$ and $\mathcal{V}$, respectively; $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are the tuning parameters for $\|\boldsymbol{C}\|_*$ and $\|\mathcal{D}\|_1$, respectively; $\mu_1 > 0$ and $\mu_2 > 0$ are two parameters which influence the convergence speed; and the term $\frac{\mu_1}{2} \|\boldsymbol{R} - \boldsymbol{C}\|_{\mathrm{F}}^2$ and term $\frac{\mu_2}{2} \|\mathcal{B} - \mathcal{D}\|_{\mathrm{F}}^2$ penalize violations of the aforementioned linear constraints. The ADMM method [32] is investigated to decouple the non-differentiable terms by alternating among the minimization sub-problems as shown in Problem (II.4):

$$\begin{aligned}
\boldsymbol{R}^{k+1} &:= \min_{\boldsymbol{R}} L + \langle \boldsymbol{U}^k, \boldsymbol{R} - \boldsymbol{C}^k \rangle + \frac{\mu_1}{2} \|\boldsymbol{R} - \boldsymbol{C}^k\|_{\mathrm{F}}^2, \\
\mathcal{B}^{k+1} &:= \min_{\mathcal{B}} L + \langle \mathcal{V}^k, \mathcal{B} - \mathcal{D}^k \rangle + \frac{\mu_2}{2} \|\mathcal{B} - \mathcal{D}^k\|_{\mathrm{F}}^2, \\
\boldsymbol{C}^{k+1} &:= \min_{\boldsymbol{C}} \lambda_1 \|\boldsymbol{C}\|_* + \langle \boldsymbol{U}^k, \boldsymbol{R}^{k+1} - \boldsymbol{C} \rangle + \\
&\quad \frac{\mu_1}{2} \|\boldsymbol{R}^{k+1} - \boldsymbol{C}\|_{\mathrm{F}}^2, \\
\mathcal{D}^{k+1} &:= \min_{\mathcal{D}} \lambda_2 \|\mathcal{D}\|_1 + \langle \mathcal{V}^k, \mathcal{B}^{k+1} - \mathcal{D} \rangle + \\
&\quad \frac{\mu_2}{2} \|\mathcal{B}^{k+1} - \mathcal{D}\|_{\mathrm{F}}^2, \\
\boldsymbol{U}^{k+1} &= \boldsymbol{U}^k + \mu_1 (\boldsymbol{R}^{k+1} - \boldsymbol{C}^{k+1}), \\
\mathcal{V}^{k+1} &= \mathcal{V}^k + \mu_2 (\mathcal{B}^{k+1} - \mathcal{D}^{k+1}),
\end{aligned} \tag{II.4}$$

where $L = [vec(\mathcal{P}_\Omega(\boldsymbol{Y}) - \mathcal{P}_\Omega(\widehat{\boldsymbol{Y}}))]^{\mathrm{T}}\mathbb{L}[vec(\mathcal{P}_\Omega(\boldsymbol{Y}) - \mathcal{P}_\Omega(\widehat{\boldsymbol{Y}}))]$, and k is the index of iteration.

Denote $\boldsymbol{x} = vec(\mathcal{X})$, $\boldsymbol{y} = vec(\boldsymbol{Y})$, $\boldsymbol{r} = vec(\boldsymbol{R})$, $\boldsymbol{b} = vec(\mathcal{B})$, $\boldsymbol{v} = vec(\mathcal{V})$, and the subscript $(\cdot)_\Omega$ as the selector to only contain the element according to the index of the non-empty entries in $\boldsymbol{Y}$. Denote $\mathcal{S}_\tau(\cdot)$ as the singular value soft-thresholding operator introduced by [33]. Denote a singular value decomposition (SVD) of matrix $\boldsymbol{C}$ $\mathbb{R}^{m\times n}$ as $\boldsymbol{C} = \boldsymbol{A}\Sigma\boldsymbol{B}^*$, where $\Sigma = \mathrm{diag}(\ \sigma_i\ _{1\le i\le r})$. Then $\mathcal{S}_\tau(\boldsymbol{C}) = \mathrm{diag}(\ (\sigma_i - \tau)_+\ _{1\le i\le r})$, where $(\sigma_i - \tau)_+ = \max(0, \sigma_i - \tau)$. As another soft-thresholding operator in Algorithm 1, $\mathcal{T}_\tau(\cdot)$ was introduced by [34] as wavelet thresholding. It can be defined as $\mathcal{T}_\lambda(\boldsymbol{w}) = [t_\lambda(w_1), t_\lambda(w_2), ...]^{\mathrm{T}}$, where $t_\lambda(w_i) = \mathrm{sgn}(w_i)\ (w_i - \lambda)_+$.

Algorithm 1 Solver for TEMC

Initialize $\boldsymbol{R}^0$, $\mathcal{B}^0$, $\boldsymbol{C}^0$, $\mathcal{D}^0$, $\boldsymbol{U}^0$, and $\mathcal{V}^0$.
repeat
$\boldsymbol{r}^{k+1} = (2\mathbb{L} + \mu_1\boldsymbol{I})^{-1}(\mu_1\boldsymbol{c}^k - \boldsymbol{u}^k + 2\mathbb{L}\boldsymbol{y} - 2\mathbb{L}vec(\ \widehat{\mathcal{B}}^k, \mathcal{X}\ ))$, then reshape to $\boldsymbol{R}^{k+1}$,
$\boldsymbol{\beta}^{k+1} = (\boldsymbol{x}_\Omega^{\mathrm{T}}\mathbb{L}_\Omega\boldsymbol{x}_\Omega + \frac{\mu_2}{2}\boldsymbol{I}_\Omega)^{-1}[\frac{1}{2}(\mu_2\boldsymbol{d}^k - \boldsymbol{v}^k) + \boldsymbol{x}_\Omega^{\mathrm{T}}\mathbb{L}_\Omega(\boldsymbol{y}_\Omega - \boldsymbol{r}_\Omega^{k+1})]$, then reshape to $\mathcal{B}^{k+1}$,
$\boldsymbol{C}^{k+1} = \mathcal{S}_{\frac{\lambda_1}{\mu_1}}(\boldsymbol{R}^{k+1} + \frac{\boldsymbol{U}^k}{\mu_1})$,
$\boldsymbol{d}^{k+1} = \mathcal{T}_{\frac{\lambda_2}{\mu_2}}(\boldsymbol{\beta}^{k+1} + \frac{\boldsymbol{v}^k}{\mu_2})$, then reshape to $\mathcal{D}^{k+1}$,
$\boldsymbol{U}^{k+1} = \boldsymbol{U}^k + \mu_1(\boldsymbol{R}^{k+1} - \boldsymbol{C}^{k+1})$,
$\mathcal{V}^{k+1} = \mathcal{V}^k + \mu_2(\mathcal{B}^{k+1} - \mathcal{D}^{k+1})$,
until
Convergence: $\frac{L^{k+1}-L^k}{L^k} \le tol$.

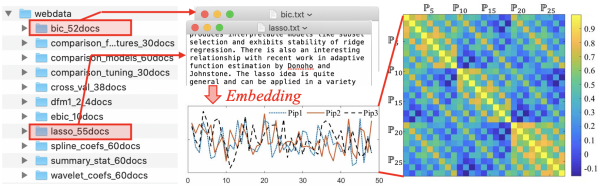
The pseudo code for the proposed algorithm is summarized in Algorithm 1. This algorithm is guaranteed to converge based on Theorem II.1 with a unique solution.

Theorem II.1. *Suppose at least one optimal solution for Algorithm 1 exists and is defined as* $(\boldsymbol{R}^*, \mathcal{B}^*, \boldsymbol{C}^*, \mathcal{D}^*, \boldsymbol{U}^*, \text{and } \mathcal{V}^*)$. *Under the conditions that* $\lambda_1 \ge 0$, $\lambda_2 \ge 0$, $\mu_1 > 0$, $\mu_2 > 0$, *the following convergence property holds:* $\lim_k L(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}^k) + \lambda_1\ \boldsymbol{R}^k\ _* + \lambda_2\ \mathcal{B}^k\ _1 = L(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}^*) + \lambda_1\ \boldsymbol{R}^*\ _* + \lambda_2\ \mathcal{B}^*\ _1$, *solution is unique.*

D. Covariates Generation Machine

As shown in Figure 2, the covariates generation machine consists *embedding machine* for pipeline embedding, and the *meta data extraction machine* to generate meta data vectors from existing and new data sets. The generated covariates will be used for model estimation by using Algorithm 1.

1) Embedding Machine: The embedding machine includes a web crawler and a word2vec embedding neural network provided by Gensim [35] coded in Python program language. The web crawler collects and parses websites and documents as a corpus of text documents, which includes detailed descriptions of and comparisons among different method options in computation pipelines (see structure of the corpus in Figure 3(a), and examples of extracted documents in Figure 3(b). Afterwards, this corpus will serve as the input to a word2vec neural network to embed the method options as dense vectors in real space with a certain length (i.e., 16 in this research). The embedded vectors should be informative to quantify the similarity and dissimilarity among method options within one step in pipelines. Thus, the vector representation of a pipeline $\boldsymbol{e}_j$ can be generated by concatenating the vectors of corresponding method options in the same order as they form the pipeline (i.e., a $16 \times 3 = 48$-dimensional vector). A visualization of cosine similarity among the embedded vector representations of all 27 computation pipelines is presentation in Figure 3(right).



(a) Structure of scraped corpus (b) Example documents and embedded pipeline vectors (c) Cosine similarity matrix among vectors

Fig. 3: An illustration of prepared descriptions for method options, where (a) presents the structure of scraped corpus as input for embedding machine, including descriptions (i.e., text documents parsed from scraped HTML and PDF files, e.g., see examples in (b)-top) of each method option in computation pipelines, and comparison studies among method options in the same step. (b)-bottom visualize the embedded pipeline vectors for three pipelines. (c) visualizes the cosine similarities among embedded pipelines for 27 computation pipelines.

2) Meta Data Extraction Machine: The meta data extraction machine generates a vector of summary statistics from a data set according to predefined summary statistic operator lists. In this research, without loss of generality, we categorize a data set in an ICPS into three subset of variables: (1) process setting variables, (2) *in situ* process variables, and (3) response variables. Three summary statistic operator lists for three variable categories are reported in Table II. Therefore, the meta data vector $\boldsymbol{d}_i$ for a data set can be generated by concatenating three vectors extracted from three variable categories.

An outer product will then be used to generate the informative interactions between data sets and pipelines (i.e., covariates) as $\mathcal{X}_{:,:,i,j} = d_i \otimes \boldsymbol{e}_j$ $\mathbb{R}^{p\times qK\times m\times n}$, for $i = 1, 2, ..., m$, and $j = 1, 2, ..., n$. Note that the information for covariates is not limited to the pipeline embeddings and the meta data from data sets, other

TABLE II: Summary statistic operator lists

Process Setting Vars.	*in situ* Process Vars.	Response Vars.
Number of rows	Number of rows	Mean value
Number of columns	Number of columns	Std. value
Mean of mean values	Mean of mean values	Range
Std. of mean values	Std. of mean values	Kurtosis
Mean of Kurtosis	Mean of Kurtosis	Skewness
Mean of Skewness	Mean of Skewness	

related information can also be vectorized in real space and serve as new modes for the covariates tensor $\mathcal{X}$.

III. A Real Case Study

Six initial data sets were extracted from TSC, AJP, and FDM processes as summarized in Table III, where map sets identify the variable relationships among process setting variables and *in situ* process variables. The map sets are necessary for several data fusion-based method options included in the pipelines [20]. Here, a process setting variable is defined as a scalar value which sets a condition for a process (e.g., temperature setting for furnace); and an *in situ* process variable is collected during the process by sensor system in a time series format. Two types of changing manufacturing contexts can be identified in Table III: (1) Data Sets $1 \sim 4$ represent the changing of quality management plans with the same process variables but different response variables; and (2) Data Sets 1, 5, and 6 represent the changing of processes with totally different process and response variables, which identify the differences among six manufacturing data sets. To stress the proposed AdaPipe system, we applied bootstrapping [36] by re-sampling to increase the number of data sets. As a result, ten new data sets are bootstrapped with 90% observations from each data set in Table III. Thus, in total 60 data sets are generated. Meta data $\boldsymbol{d}_i \quad \mathbb{R}^{17}$, where $i = 1, \ldots, 60$ are extracted from these 60 data sets via the 17 summary statistic operators in Table II.

In this study, the pipelines (see Figure 1) are adopted from [20] to model the aforementioned 60 data sets, where data fusion model 1 (DFM1), and DFM2 were recently proposed in [20]. Figure 1 presents in total 27 (i.e., $3 \times 3 \times 3$) candidate pipelines. Based on 60 bootstrapped data sets and 27 pipelines, the sparse response matrix $\boldsymbol{Y} \quad \mathbb{R}^{60\times 27}$ can be extracted in two sets of cases, i.e., warm-start and cold-start, to compare the statistical performance of the AdaPipe with three benchmark models: MC, TR and PRIME models. PRIME model and estimator [29] are introduced in Model (III.1)

$$\begin{aligned} &\textbf{Model:} && \boldsymbol{Y} = \boldsymbol{R} + \mathcal{A}(\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{E}, \\ &\textbf{Estimator:} && \underset{\boldsymbol{R},\boldsymbol{\beta}}{\text{minimize}} \quad \boldsymbol{Y} - \boldsymbol{R} - \mathcal{A}(\boldsymbol{X}\boldsymbol{\beta}) \ {}_{\text{F}}^{2} \\ &&& \text{subject to} \quad \boldsymbol{R} \ {}_{*} \leq s, \ \ \boldsymbol{\beta} \ {}_{1} \leq t, \end{aligned} \tag{III.1}$$

where a least square loss function was applied for point estimation with a nuclear norm and a l-1 norm penalization. Since this model takes covariates matrix $\boldsymbol{X}$ as input instead of the covariates tensor $\mathcal{X}$, we redefined the covariates as $\boldsymbol{X}_{h,:} = \text{concat}(\boldsymbol{d}_i, \boldsymbol{e}_j) \quad \mathbb{R}^{p+qK}$, where $h = m(i-1)+j$ and $h \quad 1, 2, ..., mn$.

In warm-start cases, $\boldsymbol{Y}$ is extracted under the assumption of arbitrarily missing entries with five levels of missing rates $\rho_f \quad 0.1, 0.3, 0.5, 0.7, 0.9$, $f = 1, ..., 5$, by setting arbitrarily selected $\rho_f \times mn$ entries in $\boldsymbol{Y}$ to be 0. For cold-start cases, $\boldsymbol{Y}$ is extracted under the assumption of missing an entire row (i.e., a new data set has not been modeled by any pipeline) by setting a row of entries to be 0 in $\boldsymbol{Y}$. Therefore, in total five warm-start cases and a cold-start case are extracted.

Since the objective of AdaPipe system is to rank and recommend the pipelines for data sets, we propose to use the minimal number of top-ranked pipelines to reach the best statistical accuracy (i.e., NRMSE in this research) for the i-th data set with a tolerance level T in percentage ($AM_i(T)$) as the performance metric. Assuming y_{i,j^*} to be the optimal NRMSE for pipelines on the i-th data set, $AM_i(T)$ can be shown as $AM_i(T) = \min_{j \quad 1,\ldots,27} (1 +$

TABLE III: Summary of six manufacturing data sets [20]

Data Sets	Process Vars. (Types)	Resp. Vars. (Obs.)	Map Sets
1	1) Stand-off distance (Setting) 2) Surface speed (Setting) 3) Current of the torch (Setting) 4) Primary gas flow rate (Setting)	Porosity(40)	
	5) Secondary gas flow rate (Setting)		1-13
	6) Carrier gas flow rate (Setting)		1-14
	7) Traverse rate (Setting)		1-15
2	8) Particle velocity (*in situ*)	Roughness(40)	2-21
	9) Particle temperature (*in situ*)		3-11
	10) Correlation (*in situ*)		3-12
	11) Head temperature (*in situ*)		3-16
	12) Plume temperature (*in situ*)		3-20
3	13) Total intensity (*in situ*)	Curvature(40)	3-21
	14) Peak intensity (*in situ*)		4-19
	15) Half width (*in situ*)		5-19
	16) Substrate temperature (*in situ*)		6-19
4	17) Laser displacement (*in situ*) 18) Arc current (*in situ*) 19) Gas flow rate 1&2 (*in situ*) 20) Arc voltage (*in situ*) 21) Surface temperature (*in situ*)	Depo-rate(40)	
5	1) Process speed (Setting)	Resistance(90)	1-5
	2) Atomizer gas flow rate (Setting)		1-6
	3) Shealth gas flow rate (Setting)		2-5
	4) Atomizer power voltage (Setting)		3-5
	5) Current (*in situ*)		4-5
	6) Nozzle location – x, y (*in situ*)		
6	1) Nozzle travel speed (Setting)	Surf-Rough.(48)	
	2) Nozzle temperature (Setting)		1-3
	3) Vibration signals (on the nozzle) – x, y, z (*in situ*)		1-4
	4) Vibration signals (on the bed) – x, y, z (*in situ*)		2-5
	5) Nozzle temperature (*in situ*)		2-6
	6) Bed temperature (three sensors on the bed) (*in situ*)		

$T)y_{i,j^*} - y_{i,j}$. For example, $AM_1(0\%) = 5$ indicates that executing the top-5 ranked pipelines is adequate to reach the lowest NRMSE; and $AM_1(10\%) = 3$ indicates that executing the top-3 ranked pipelines is adequate to achieve a satisfactory NRMSE which is 10% greater than the lowest possible NRMSE.

The proposed TEMC model is simultaneously estimated and used for prediction since it is an unsupervised learning method. Two 10-fold cross-validation (CV) was implemented to select the tuning parameters λ_1 and λ_2 for the proposed model. For model evaluation, different training-testing splitting strategies were used for warm-start and cold-start scenarios. Namely, for warm-start scenarios, $(1-\rho_f)mn$ training samples were arbitrarily selected from the entries in sparse response matrix $\boldsymbol{Y}$, and the remaining entries are treated as testing samples. Besides, ten replicates were conducted with different testing samples for each missing rate ρ_f $0.1, 0.3, 0.5, 0.7, 0.9$. For cold-start scenarios, 60-fold leave-one-data-set-out cross-validation was adopted to select the training and testing samples. Specifically, entries in 59 out of 60 rows in $\boldsymbol{Y}$ were selected as training samples, and the entries in the remaining one row were selected as testing samples. This procedure was repeated for 60 times to ensure that every row in $\boldsymbol{Y}$ can be selected as testing samples in one cross-validation fold. The recommendation results are compared with three benchmark models in the same sets of cases.

IV. Results and Discussion

The average values and standard errors of the performance metric $AM_i(T)$ over CV folds is summarized in Table IV, where the best performance (i.e., the lowest mean values and the lowest standard errors) are highlighted in **bold**. It can be concluded that the propose AdaPipe system outperforms three benchmark models by executing the lowest number of pipelines to achieve the underline best performance. For tolerance level $T = 0\%$ under the worst cases (i.e., warm-start case when missing rate $\rho = 0.9$, and cold-start case), the AdaPipe system saves $1 - \frac{8.733}{27} = 67.66\%$ and $1 - \frac{8.983}{27} = 66.73\%$ computation workloads. And the AdaPipe system saves $1 - \frac{4.917}{27} = 81.79\%$ and $1 - \frac{4.533}{27} = 83.21\%$ computation workloads for tolerance level $T = 10\%$. It can be observed that the ranking performances of AdaPipe in warm-start scenarios with ρ $0.1, 0.3, 0.5, 0.7$ outperform those in cold-start scenarios. The reason is that relatively lower sparsity in $\boldsymbol{Y}$ can support better estimation of the low-rank structure, which directly leads to higher ranking accuracy. However, the ranking performances of AdaPipe in warm-start scenarios when $\rho = 0.9$ are worse than those in cold-start scenarios, since only the execution results for, on average, $(1 - 0.9) \times 27 = 2.7$ computation pipelines are available in each data set. As a result, the low-rank structure in $\boldsymbol{Y}$ is hard to be accurately estimated when compared with the estimation in cold-start scenarios.

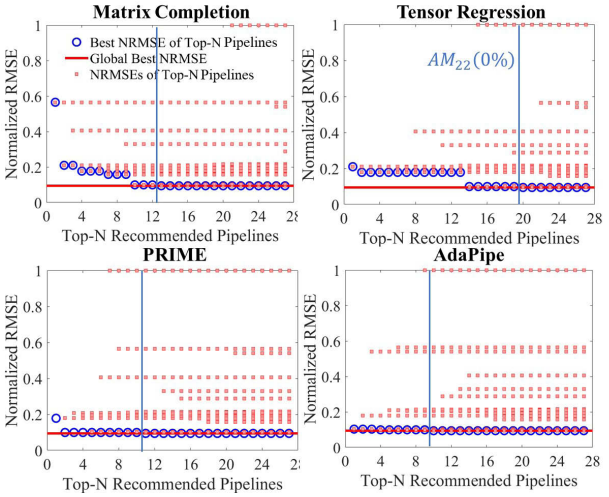


Fig. 4: An example of ranked pipelines under cold-start case for the 22-th data set, where the blue circles identify the lowest NRMSE among Top-N recommended pipelines; the red line is the global lowest NRMSE among all 27 pipelines; the red squares show the NRMSEs for Top-N recommended pipelines; and the vertical blue line shows the performance metric $AM_{22}(0\%)$.

An example of ranked computation pipelines for the 22-th data set is presented in Figure 4. It can be observed from Figure 4 that (1) the Top-1 recommended computation pipeline by AdaPipe has the closest NRMSE to the global best NRMSE; (2) the AdaPipe yields the lowest $AM_{22}(0\%)$ among four models; and (3) the AdaPipe presents the best ranking performance by investigating the trends and distribution of the red dots.

When comparing performance among benchmark models, the matrix completion does not perform well under both warm-start and cold-start cases due to limited sample size, which results in full rank $\boldsymbol{Y}$. Tensor regression significantly outperforms matrix completion method since it does not require large sample size, and it quantifies the information contained in the covariates. However, better ranking and recommendation performance of PRIME model indicates that the implicit similarity existed in the low-rank matrix $\boldsymbol{R}$ decomposed from $\boldsymbol{Y}$ can significantly improve the ranking and recommendation performance. Therefore, the best ranking and recommendation performance provided by AdaPipe may be attributed to the following reasons: (1) pairwise loss function provides the capability to rank the computation pipelines by comparing statistical prediction errors in pairs, but ignoring the alignment between predicted responses and true responses; (2) low-rank matrix $\boldsymbol{R}$

TABLE IV: Average values and standard errors (within parenthesis) of AdaPipe system and three benchmark models evaluated on 60 bootstrapped data sets and 27 computation pipelines. The significantly best performance is highlighted in **bold**.

Tol. T	Methods	Warm-start					Cold-start
		$\rho = 0.9$	$\rho = 0.7$	$\rho = 0.5$	$\rho = 0.3$	$\rho = 0.1$	
0%	Matrix Completion	12.367 (1.363)	11.017 (1.481)	10.767 (1.536)	12.800 (1.415)	11.467 (1.587)	12.400 (1.311)
	Tensor Regression	12.017 (1.488)	11.300 (1.348)	10.100 (1.466)	8.317 (1.282)	8.467 (1.240)	10.033 (1.398)
	PRIME	**9.833 (1.217)**	**8.583 (1.204)**	**8.267 (1.021)**	**6.283 (0.856)**	6.233 (0.948)	**8.150 (1.198)**
	AdaPipe	**8.733 (1.333)**	**8.367 (1.364)**	**7.917 (1.295)**	**5.483 (1.180)**	**2.900 (0.765)**	**8.983 (1.327)**
10%	Matrix Completion	7.033 (1.050)	6.267 (1.224)	6.450 (1.292)	8.083 (1.182)	6.867 (1.266)	8.800 (1.258)
	Tensor Regression	**5.650 (0.938)**	6.800 (1.035)	6.983 (1.176)	5.617 (0.912)	5.700 (0.880)	7.183 (1.110)
	PRIME	**5.000 (0.963)**	5.600 (1.009)	5.233 (0.888)	3.667 (0.582)	3.017 (0.737)	**4.683 (0.860)**
	AdaPipe	**4.917 (0.890)**	**3.867 (0.935)**	**3.833 (0.866)**	**2.183 (0.633)**	**1.650 (0.368)**	**4.533 (0.861)**

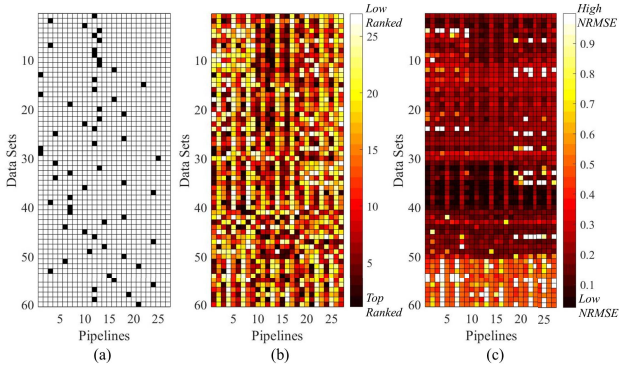


Fig. 5: Computation pipelines ranked for 60 data sets, where (a) shows the best computation pipelines associated with the lowest prediction NRMSEs in black blocks; (b) presents the ranking results, where darker block represents higher rank; (c) reports the NRMSEs and darker block identifies lower prediction error.

and covariates $\mathcal{X}$ effectively quantify the implicit and explicit similarities, which jointly contributes to accurate ranking and recommendation performance; and (3) the embedded dense vector representations are informative to identify the similarity and difference among computation pipelines.

To interpret the effectiveness of low-rank regularization in TEMC model, Figure 5(a) presents the visualization of the best computation pipelines associated with lowest prediction NRMSEs. The NRMSEs and heatmaps of ranking results are presented in Figure 5(a) and (b), respectively. It can be observed from two heatmaps that linear dependencies exist among columns and rows. Note that the distribution of bootstrapped data sets are not identical, which generates similar-but-non-identical covariates by meta data extraction machine. Therefore, the NRMSEs between two bootstrapped data sets (i.e., two adjacent rows in Figure 5(a)) may not be identical. The aforementioned linear dependencies represent the low-rank structure of response matrix $\mathbf{Y}$, which can be decomposed into low-rank matrix $\mathbf{R}$ and can be captured by nuclear norm regularization. Assuming $\mathbf{Y}$ itself to be low-rank can be ambiguous since the underline best computation pipelines for 60 data sets shows no low-rank structures. Therefore, pure matrix completion model does not perform as well as TEMC model.

V. Conclusions

To match the contexts and computation pipelines is critical for providing effective computation services in ICPS in manufacturing. However, exploring all candidate pipelines is inefficient for fast computation services. Therefore, we propose an AdaPipe system that integrates a covariates generation machine and a newly proposed TEMC model to accurately rank and recommend pipelines for different contexts. A case study in real TSC, AJP, and FDM processes showed that AdaPipe outperforms MC, TR, and PRIME in both warm-start and cold-start cases in supporting accurate and responsive computation services. The proposed AdaPipe can be applied in other areas as a recommender system to make accurate recommendations in extreme cases.

In the future, AdaPipe can be generalized for other types of computation services in ICPS, such as process monitoring, prognosis, diagnosis, and control. Besides, a distributed recommender system will be investigated for similar-but-non-identical manufacturing processes in

ICPS. And it will help understand the boundaries and application scope of existing pipelines.

VI. Acknowledgement

This research is partially supported by NSF CMMI-1634867.

Appendix

A. Proof of Proposition II.1

Proof. For the pairwise loss function $L(\boldsymbol{Y}, \widehat{\boldsymbol{Y}})$, let $\epsilon_{i,u} = y_{i,u} - \widehat{y}_{i,u}$, it can be rewritten as

$$\begin{aligned} L(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) &= \sum_{i=1}^{m} \frac{1}{2\,\Omega_i} \sum_{u\ \Omega_i} \sum_{v\ \Omega_i} (\epsilon_{i,u} - \epsilon_{i,v})^2 \\ &= \sum_{i=1}^{m} \frac{1}{\Omega_i} (\ \Omega_i \sum_{u\ \Omega_i} \epsilon_{i,u}^2 - \sum_{u\ \Omega_i} \sum_{v\ \Omega_i} \epsilon_{i,u}\epsilon_{i,v}). \end{aligned}$$

Denoting $L_{u,v} = \begin{cases} \Omega_1 - 1, & \text{if } u = v \\ -1, & \text{otherwise} \end{cases}$, and $\mathbb{L} = \begin{bmatrix} \frac{1}{\Omega_1} L & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\Omega_m} L \end{bmatrix}$, The pairwise loss function can be rewritten as:

$$\begin{aligned} PL(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) &= \sum_{i=1}^{m} \frac{1}{\Omega_i} \boldsymbol{\epsilon}_i^{\mathrm{T}} L \boldsymbol{\epsilon}_i, \\ &= [vec(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})]^{\mathrm{T}} \mathbb{L} [vec(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})]. \end{aligned}$$

Therefore, this pairwise loss function has a equivalent form of $[vec(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})]^{\mathrm{T}} \mathbb{L} [vec(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})]$, which is convex. □

References

[1] X. Chen, L. Wang, C. Wang, and R. Jin, "Predictive offloading in mobile-fog-cloud enabled cyber-manufacturing systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 167–172.

[2] D. Wang, J. Liu, and R. Srinivasan, "Data-driven soft sensor approach for quality prediction in a refining process," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 11–17, 2009.

[3] Z. Ge and J. Chen, "Plant-wide industrial process monitoring: A distributed modeling framework," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 310–321, 2015.

[4] H. Liao, E. A. Elsayed, and L.-Y. Chan, "Maintenance of continuously monitored degrading systems," *European Journal of Operational Research*, vol. 175, no. 2, pp. 821–835, 2006.

[5] J.-H. Zhou, C. K. Pang, F. L. Lewis, and Z.-W. Zhong, "Intelligent diagnosis and prognosis of tool wear using dominant feature identification," *IEEE transactions on Industrial Informatics*, vol. 5, no. 4, pp. 454–464, 2009.

[6] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of ec2 cloud computing services for scientific computing," in *International Conference on Cloud Computing*. Springer, 2009, pp. 115–131.

[7] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "Cosmos: computation offloading as a service for mobile devices," in *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2014, pp. 287–296.

[8] F. Tao, Y. Cheng, L. Da Xu, L. Zhang, and B. H. Li, "Cciot-cmfg: cloud computing and internet of things-based cloud manufacturing service system," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1435–1442, 2014.

[9] B. Xu, L. Da Xu, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in iot-based information system for emergency medical services," *IEEE Transactions on Industrial informatics*, vol. 10, no. 2, pp. 1578–1586, 2014.

[10] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4665–4673, 2018.

[11] P. Mika, "Flink: Semantic web technology for the extraction and analysis of social networks," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 2-3, pp. 211–223, 2005.

[12] X. Jiang, "A year of blink at alibaba: Apache flink in large scale production," May 2017, https://www.dataversity.net/year-blink-alibaba/. Last accessed 27 January 2019. [Online]. Available: https://www.dataversity.net/year-blink-alibaba/

[13] H. Beck, M. Dao-Tran, and T. Eiter, "Lars: A logic-based framework for analytic reasoning over streams," *Artificial Intelligence*, vol. 261, pp. 16–70, 2018.

[14] Y. Zhang, L. Wang, X. Chen, and R. Jin, "Fog computing for distributed family learning in cyber-manufacturing modeling," in *2019 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2019.

[15] Q. Hu, D. Yu, and Z. Xie, "Information-preserving hybrid data reduction based on fuzzy-rough techniques," *Pattern recognition letters*, vol. 27, no. 5, pp. 414–423, 2006.

[16] I. K. Fodor, "A survey of dimension reduction techniques," *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory*, vol. 9, pp. 1–18, 2002.

[17] S. Abe, "Feature selection and extraction," in *Support Vector Machines for Pattern Classification*. Springer, 2010, pp. 331–341.

[18] A. Jayal, F. Badurdeen, O. Dillon Jr, and I. Jawahir, "Sustainable manufacturing: Modeling and optimization challenges at the product, process and system levels," *CIRP Journal of Manufacturing Science and Technology*, vol. 2, no. 3, pp. 144–152, 2010.

[19] S. Wu, "Dynamic data system: a new modeling approach," *Journal of Engineering for Industry*, vol. 99, no. 3, pp. 708–714, 1977.

[20] X. Chen and R. Jin, "Data fusion pipelines for autonomous smart manufacturing," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 1203–1208.

[21] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning*. Springer, 2019.

[22] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. Feb, pp. 281–305, 2012.

[23] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[24] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 826–830, 2017.

[25] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in neural information processing systems*, 2015, pp. 2962–2970.

[26] C. Yang, Y. Akimoto, D. W. Kim, and M. Udell, "Oboe: Collaborative filtering for automl model selection," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1173–1183.

[27] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[28] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2002, pp. 253–260.

[29] X. Chen, N. Lau, and R. Jin, "Prime: A personalized recommendation for information visualization methods via extended matrix completion," *ACM Transactions on Interactive Intelligent Systems*, 2019, (Under second round review).

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[31] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 129–136.

[32] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

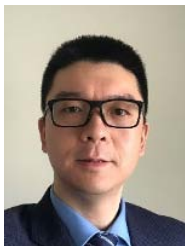
[33] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[34] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[35] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[36] C. Z. Mooney, R. D. Duval, and R. Duvall, *Bootstrapping: A nonparametric approach to statistical inference*. Sage, 1993, no. 94-95.

Xiaoyu Chen is a Ph.D. candidate in the Grado Department of Industrial and Systems Engineering at Virginia Tech. He received a B.E. degree from School of Optoelectronics at Beijing Institute of Technology, China in 2015. He previously worked as Research assistant and Grado teaching assistant at Virginia Tech. His research focused on human-AI collaboration and Fog-Cloud computing in smart manufacturing systems. Mr. Chen is a member of Institute for Operations Research and the Management Sciences (INFORMS), Institute of Industrial and Systems Engineering (IISE), and IEEE. His awards and honors include Doctoral Student of the Year in the Grado Department of Industrial and Systems Engineering at Virginia Tech, 2019.



Ran Jin is an Associate Professor and the Director of Laboratory of Data Science and Visualization at the Grado Department of Industrial and Systems Engineering at Virginia Tech. He received his Ph.D. degree in Industrial Engineering from Georgia Tech, Atlanta, his Master's degrees in Industrial Engineering, and in Statistics, both from the University of Michigan, Ann Arbor, and his bachelor's degree in Electronic Engineering from Tsinghua University, Beijing. His research focuses on machine learning in manufacturing, manufacturing computation services, and cognitive-based interactive visualization. He is currently serving as an Associate Editor for IISE Transactions and an Associate Editor for ASME Transactions, Journal of Manufacturing Science and Engineering. He has been working with many leading manufacturing companies in aerospace, semiconductor, personal care, optical fiber industries. He is a member of IEEE. For more information about Dr. Jin, please visit his faculty website at Virginia Tech: https://ise.vt.edu/ran-jin.