Lori: Local Low-rank Response Imputation for Automatic Configuration of Contextualized Artificial Intelligence

Xiaoyu Chen, *Member, IEEE*, and Ran Jin, *Member, IEEE*

***Abstract*— Artificial Intelligence (AI) has played an important role for data-driven decision making in complex engineering problems. However, there has been a huge waste of efforts to configure AI methods (e.g., to select preprocessing and modeling methods, etc.), catering to different contexts (e.g., data analytics objectives, data distributions, etc.). In current practice, data scientists need to manually configure the AI methods in trial-and-errors according to a specific context, including determining the different options of the pipeline components and evaluating the advantages and limitations of an AI method. In this paper, we propose a Local Low-rank Response Imputation (Lori) method, which will automatically configure AI methods to specific contexts by completing a sparse context-pipeline response matrix. Different from the traditional recommendation systems, Lori performs multivariate partition of the entire context-pipeline response matrix based on the principal Hessian directions of the low-rank imputed response matrix. Thus, the partitioned local low-rank response matrices can be closely modeled to automatically match the AI methods with the data sets. A small-scale and a large-scale case studies in three manufacturing processes demonstrated the merits of the proposed Lori method.**

***Index Terms*— Computation Pipelines, Matrix Completion, Principal Hessian Direction, Regression Tree**

I. Introduction

The advancement of machine learning and statistical learning methods have dramatically promoted the deployment of data-driven decision-making in many systems. For example, a cybermanufacturing system (CMS) interconnects manufacturing facilities, processes, and systems by various sensors, actuators, and Fog-Cloud computation units to provide advanced data analytics towards multiple objectives (e.g., quality modeling and prediction, monitoring, prognosis, diagnosis, etc.). In healthcare systems, the deployment of AI methods in Internet of Things (IoT) enables the monitoring of body health status [1], detection of human falls [2], heart disease prediction [3], etc. However, as tremendous method configuration options from data sourcing, feature extraction, dimension reduction, tuning criteria, to model estimation become available, it has posed increasingly huge human and computation workloads on evaluating different configurations of these method options in a trial-and-error manner. On the other hand, the application boundaries of these configurations have not been well understood, hence preventing fast adaptation from one configuration of method options to another in response to the frequent changes in contexts. Here, we define a context as the dataset produced from a changed manufacturing condition or analytical objective. For one example, after adjusting a manufacturing process setting variable, the underlying data distributions will also change, which creates a new context. The computation pipelines should be automatically configured to this new context by learning from the dataset produced from such context. As another example, when the analytical objective changed from one quality variable to another, the computation pipeline should be configured to match such changing objective. Therefore, the concept of automatic configuration for AI methods has been proposed as a responsive service by systematically integrating many method options in different steps in sequences as computation pipelines [4].

In recent years, the concept of automatic machine learning (AutoML) has been attracting great interests from both researchers and practitioners to automatically configure AI methods [5]. AutoML yields an intelligent way to optimize the AI pipeline for a given context to reach the best possible prediction accuracy. Researchers and practitioners started to tackle this research question by automating the searching process of ML structures. On the one hand, neural architecture search (NAS) was initially created as one direction of AutoML to automate the searching in the space of neural network architectures [6]. The key idea of NAS is to greedily search for and assemble blocks of neural nets (e.g., fully connected blocks, convolutional blocks, recurrent blocks, etc.) to construct a neural network. Consequently, NAS usually requires huge computational powers to achieve satisfactory performance; e.g., 800 GPUs were used to trained the networks for 28 days until the optimal structure was identified [7]. Then NASNet, an advanced algorithm based on NAS to reduce the computational budget. However, NASNet still used 500 GPU in 5 days to get the best structure [8]. On the other hand, recommender systems (RecSys) have been created to efficient rank the ML pipelines for a given dataset/task without iterative searching [4]. However, the RecSys-based approaches are not scalable for large-scale AI configuration tasks, calling for a more effective and efficient algorithm. Before introducing details, we summarize the notations in Table I.

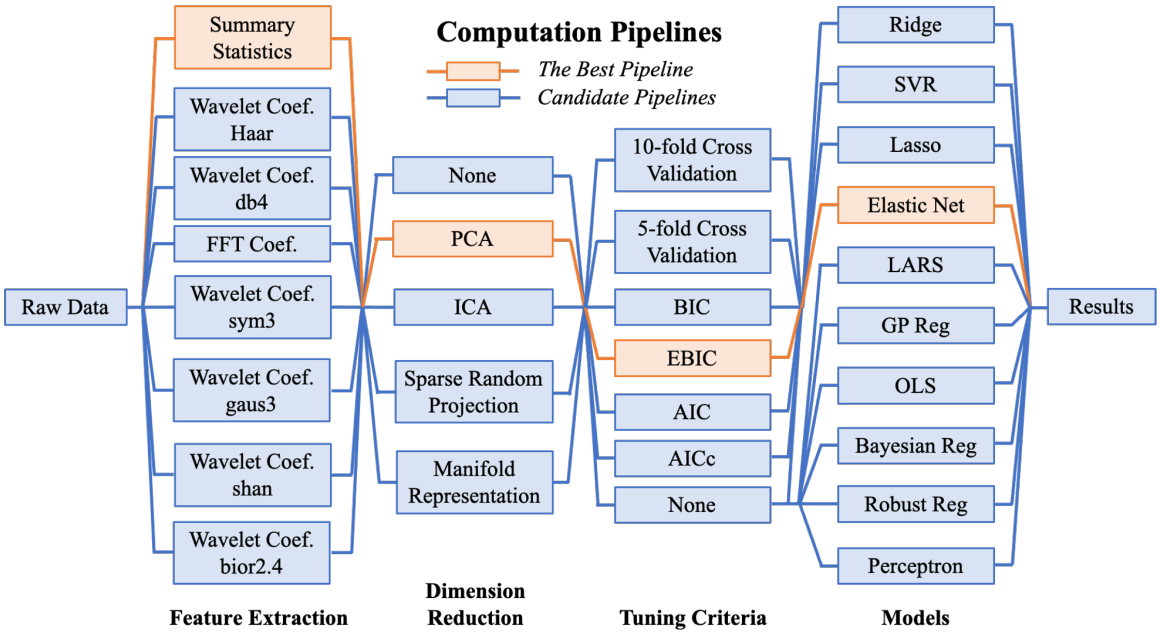


Fig. 1. An example of 1360 computation pipelines for supervised learning problems with four steps including feature extraction, dimension reduction, tuning criteria, and models with multiple method options, where the best pipeline for a given context is highlighted in orange (light) color. Here, one path that links all four steps from raw data to results represents one computation pipeline. Before executing all the pipelines on the raw data and comparing their results, the best pipeline cannot be directly identified. Note of abbreviations: fast Fourier transformation (FFT), principal component analysis (PCA), independent component analysis (ICA), Bayesian information criterion (BIC), extended BIC (EBIC), Akaike information criterion (AIC), small-sample corrected AIC (AICc), support vector regression (SVR), least angle regression (LARS), Gaussian process regression (GP Reg), ordinary least squares (OLS).

TABLE I
NOTATIONS

Notations	Descriptions
m, n, p	Number of contexts, pipelines, and covariates
$\boldsymbol{X}$	Covariates matrix in $\mathbb{R}^{mn\times p}$
$\boldsymbol{Y}$	Response matrix in $\mathbb{R}^{m\times n}$
$\boldsymbol{E}$	Error matrix in $\mathbb{R}^{m\times n}$
$\boldsymbol{R}$	Low-rank matrix in $\mathbb{R}^{m\times n}$
$\boldsymbol{B}$	Model coefficient vector in $\mathbb{R}^{p}$
$\mathcal{A}$	Linear map $\mathcal{A}$: $\mathbb{R}^{mn\times 1}\rightarrow\mathbb{R}^{m\times n}$
$\boldsymbol{y}, \boldsymbol{\epsilon}, \boldsymbol{r}$	$\mathcal{A}^{-1}(\boldsymbol{Y}), \mathcal{A}^{-1}(\boldsymbol{E}), \mathcal{A}^{-1}(\boldsymbol{R})$
s, t	Positive tuning parameters
$\boldsymbol{e}$	Gaussian noise vector $\boldsymbol{\epsilon}=\boldsymbol{r}+\boldsymbol{e}$
$\boldsymbol{H}$	Expected Hessian matrix
$\boldsymbol{\Sigma}_{\boldsymbol{X}}$	Covariance matrix of $\boldsymbol{X}$
$\boldsymbol{\Sigma}_{\boldsymbol{yXX}}$	Averaged covariance matrix weighted by $\boldsymbol{y}$
$\boldsymbol{\Sigma}_{\boldsymbol{\epsilon XX}}$	Averaged covariance matrix weighted by $\boldsymbol{\epsilon}$
$\boldsymbol{\Sigma}_{\boldsymbol{eXX}}$	Averaged covariance matrix weighted by $\boldsymbol{e}$
$\boldsymbol{\Sigma}_{\boldsymbol{rXX}}$	Averaged covariance matrix weighted by $\boldsymbol{r}$
$\boldsymbol{b}_j$	The j-th pHd
$\boldsymbol{\lambda}_j$	The j-th eigen value
$\mathbf{pHd}_j$	The j-th principal direction

A. Adaptive Computation Pipelines

As an example, **Ada**ptive Computation **Pipe**lines (AdaPipe)-based computation service described in [4] was proposed with two key components: 1) the computation pipeline, which is defined as a sequence of method options for multiple steps from feature extraction to models as presented in Figure 1. For example, the sequence "*Summary Statistics* $\Rightarrow$ *Principal Component Analysis* $\Rightarrow$ *Extended Bayesian Information Criterion* $\Rightarrow$ *Elastic Net*" in orange highlights one pipeline; 2) a recommender system (RecSys) to formulate the computation pipeline selection as a recommendation problem by effectively and efficiently suggest the best computation pipelines to a give manufacturing contexts via an extended matrix completion (EMC) model. The EMC model and estimator is shown in Model (1):

$$\begin{aligned} &\textbf{Model:} && \boldsymbol{Y} = \boldsymbol{R} + \mathcal{A}(\boldsymbol{XB}) + \boldsymbol{E}, \\ &\textbf{Estimator:} && \min_{\boldsymbol{R},\boldsymbol{B}} L(\widehat{\boldsymbol{Y}}, \boldsymbol{Y}) \\ &&& \text{subject to } ||\boldsymbol{R}||_* \le s, \\ &&& \quad ||\boldsymbol{B}||_1 \le t, \\ &&& \quad \widehat{\boldsymbol{Y}} = \widehat{\boldsymbol{R}} + \mathcal{A}(\boldsymbol{X}\widehat{\boldsymbol{B}}), \end{aligned} \tag{1}$$

where $\boldsymbol{Y}, \boldsymbol{R} \in \mathbb{R}^{m\times n}$ are the context-pipeline response matrix and a low-rank matrix, respectively, with m contexts and n computation pipelines; $\boldsymbol{X} \in \mathbb{R}^{mn\times p}$ is the p-dimensional covariates matrix that is concatenated from the covariates of contexts and the covariates of computation pipelines; Here, the covariates are generated following the embedding machine and metadata extraction machine detailed in [4] to preserve the similarities among contexts and among pipelines. $\boldsymbol{B} \in \mathbb{R}^{p\times 1}$ is the p-dimensional model coefficient vector for the linear regression term; $\mathcal{A}(\cdot)$ is a linear mapping function that maps from $\mathbb{R}^{mn\times 1}$ to $\mathbb{R}^{m\times n}$; $L(\widehat{\boldsymbol{Y}}, \boldsymbol{Y})$ is a loss function, such as least square loss function, pairwise loss function [4], etc.; $||\cdot||_*$ is the nuclear norm (also known as trace norm), which is 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 $\boldsymbol{R}$; $||\cdot||_1$ is the

l-1 norm, which is the sum of absolute value of the elements in $\boldsymbol{B}$ to control the sparsity of model coefficients; $s \geq 0$ and $t \geq 0$ are tuning parameters to control the amount of shrinkages.

Unlike existing collaborative filtering models in RecSys, EMC model not only considers the explicit similarity expressed in the context-pipeline response matrix (i.e., the low-rank matrix completion term $\boldsymbol{R}$), but also quantifies the implicit similarities contained in the covariates of contexts and computation pipelines (i.e., a linear regression term $\boldsymbol{XB}$). In [4], an embedding machine and a meta-data extraction machine were investigate to generate informative covariates to distinguish the pipelines and covariates, hence benefit the recommendation accuracy. This AdaPipe system was validated in a small-scale pipeline recommendation problem with 60 bootstrapped datasets from three manufacturing processes and 27 computation pipelines to represent a pipeline selection problem in 60 different manufacturing contexts. However, the scalability of this AdaPipe system has not been well studied in large-scale problems, i.e., when thousands of candidate computation pipelines are available. For example, by count the paths from *Raw Data* to *Results*, Figure 1 presents in total $8 \times 5 \times (7 \times 4 + 1 \times 6) = 1360$ computation pipelines. Such a large-scale problem can pose significant challenges to both the responsiveness and scalability of AdaPipe system as an online computation service.

B. Challenges of Responsiveness and Scalability

The challenges in responsiveness and scalability can be attributed to 1) the huge time costs in searching for the best two tuning parameters in a large-scale AdaPipe system when time costs for one estimation increase, and 2) the lack of information contained in relatively low dimensional covariates to distinguish thousands of computation pipelines. Specifically, as more candidate computation pipelines are considered for recommendation, the dimensions of context-pipeline response matrix will be significantly increased, which leads to a cubic growth (i.e., $\mathcal{O}(n^3)$ for singular value decomposition, SVD [9]) of computation complexity for the low-rank matrix completion term $\boldsymbol{R}$ in AdaPipe. On the other hand, higher dimensional covariates will be required to better distinguish the candidate computation pipelines by accurately representing the similarities and dissimilarities. The increased dimension of p-dimensional covariates will not only lead to a cubic growth (i.e., $\mathcal{O}(p^3)$) for matrix inversion of computation complexity for the linear regression term $\boldsymbol{XB}$, but also put significantly higher requirements on the pipeline embedding machine to distinguish increasingly higher number of pipelines. However, simply maintaining the low dimensionality of covariates directly lead to an one-to-one mapping problem, in other words, covariates from different computation pipelines may result in similar responses for some contexts. Thus, the one-to-one mapping between covariates and entries in the context-pipeline response matrix cannot be established. Therefore, a responsive and scalable RecSys is in great needs to reduce the time costs in parameter tuning and address the one-to-one mapping issue.

Motivated by the challenges for AdaPipe system, the objective of this research is to investigate a RecSys called "Lori" for the large-scale computation pipeline recommendation problem with sufficient responsiveness and scalability. Previous work investigated three directions towards large-scale recommendation problem: 1) to develop efficient parallel or distributed solvers and mechanisms for specific recommendation models, such as parallel stochastic matrix factorization [10], [11], distributed matrix completion algorithm [12], [13], etc.; 2) to propose more efficient recommendation models, such as to improve the SVD-based matrix completion model by singular value thresholding [14], the Riemannian optimization-based manifold learning for matrix completion [15], etc. and 3) to investigate local low-rank properties of a large response matrix and complete it in the decomposed low-rank sub-matrices [16]. However, most of the studies in the aforementioned directions focused on collaborative filtering methods [17]–[19] without the consideration of covariates. Thus, these methods cannot directly benefit from the extra information contained in the covariates, hence limiting their contributions to the computation pipeline recommendation problem. However, the third research direction in local low-rank approximation motivated us to reformulate the AdaPipe system from a matrix partition perspective.

C. Local Low-rank Matrix Partition Problem

Partitioning the context-pipeline response matrix will not only provide efficient RecSys by decomposing a large-scale problem into a finite set of small-scale problems, but also address the one-to-one mapping issue while maintaining the low dimensionallity of the covariates. Because for small-scale problems with limited number of contexts and computation pipelines, the information provided by covariates is, to some degree, sufficient to distinguish computation pipelines. The state-of-the-art studies in local low-rank matrix approximation propose to directly identify the local low-rank sub-matrices without applying elementary transformations on row and column switching [20]. The differences of these RecSys lie in the identification methods of low-rank sub-matrices, e.g., random generation [16], anchor points selection [21], kernel smoothing nearest-neighbors [22], etc. However, the low-rank sub-matrices identified by existing methods and their recommendation accuracy highly depend on the organization of rows and columns. For example, in a MovieLens dataset [23], where each row represents one user, and each column corresponds to a movie, if movies that belong to one cluster (e.g, action movies) and users shared the same interests are adjacent in the user-movie rating matrix, identifying the local low-rank sub-matrices is feasible. However, if the users and movies (i.e., rows and columns) are randomly ordered, a local low-rank structure may not be easily identified without any elementary transformation.

Unfortunately, random order of contexts and computation pipelines is often the case in a computation pipeline recommendation problem. As presented in Figure 2, the left matrix is a small-scale full-rank context-pipeline response matrix $\boldsymbol{Y} \in \mathbb{R}^{60 \times 27}$ with each the (i, j)-th entry representing the normalized root-mean-square-error (NRMSE) for the i-th context tested on the j-th computation pipeline. Patterns

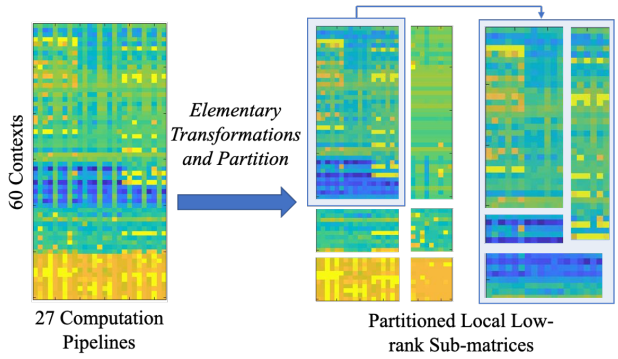


Fig. 2. An illustration of partitioning local low-rank sub-matrices after applying elementary transformations on the original context-pipeline response matrix.

can be easily observed in columns and rows, which help the identification of linear operators I_L and I_R for the elementary transformation $I_L \boldsymbol{Y} I_R$, such that the columns and rows can be switched to form local low-rank structures. The right half of Figure 2 shows the local low-rank sub-matrices partitioned from the original full-rank context-pipeline response matrix. Although the integration of elementary transformations and local low-rank sub-matrices partitioning seems to be a promising approach for RecSys, directly optimizing the linear operators I_L and I_R requires to efficiently solve a quadratic integer programming problem with m^2+n^2 binary decision variables. Besides the infeasibility to solve such an optimization problem in linear time, this optimization problem is not well-defined since the partitioning process may be stochastic, which leads to uncertainties in objective functions that minimizes the sum of ranks (or equivalently the nuclear norm approximations) of the sub-matrices. Therefore, a computationally-feasible method is desired to partition the response matrix with the consideration of maximizing the recommendation accuracy.

In this research, we propose to formulate the local low-rank matrix approximation problem for computation pipeline recommendation as a treed RecSys based on the EMC model [4], [24]. The core idea is to investigate a tree splitting method to partition the original context-pipeline response matrix by considering both the similarities in covariates and the local low-rank structures in the response matrix. The proposed tree splitting method aims at performing multivariate splitting. Because a univariate splitting method cannot guarantee the generation of local low-rank sub-matrices, and cannot address the one-to-one mapping issue. Specifically, as presented in Figure 3(left), splitting based on only one variable (perpendicular to either x_1 or x_2 axis) may easily result in two full-rank matrix. Splitting according to the identified curvature (i.e., from multiple variables) can potentially lead to local low-rank matrices. However, a multivariate splitting both on covaraites and responses requires a reasonable guess for the missing entries of the response matrix. Therefore, the Lori employs a newly proposed multivariate splitting method based on the principal Hessian directions (pHds) [25] of the covariates and a low-rank imputed response matrix. Here, the low-rank imputed response matrix is equivalent to the low-rank matrix $\boldsymbol{R}$ in Model 1, which represents the purely low-rank structure after subtracting random noises $\boldsymbol{E}$ and a hyperplane, $\boldsymbol{XB}$, from the imputed response matrix $\boldsymbol{Y}$. The proposed multivariate partition method can guarantee that the splitted sub-matrices are low-rank in the pHd space, hence can be well modeled by a linear regression tree with one-to-one mapping problems addressed. Besides, the Lori does not require to fine-tune the EMC model by searching for the best combination of tuning parameters s and t in Model 1, since it does not relies on the exact solution of $\boldsymbol{R}$. In this way, the partitioning of response matrix and the maximization of recommendation accuracy can be simultaneously achieved in one-shot in a responsive and scalable manner especially for a large-scale computation pipeline recommendation problem.

Differing from the existing local low-rank matrix approximation methods that generates local regions of the response matrix, the proposed Lori automatically reorganizes (i.e., equivalently applies elementary transformations to) the response matrix and partitions the matrix into low-rank sub-matrices with irregular shapes. Hence, the Lori contributes to the RecSys methodologies in the following aspects:

- The low-rank assumption of existing matrix completion methodologies is relaxed by defining the local low-rank properties in a pHd space.
- Lori provides an efficient local low-rank sub-matrices partitioning method based on a regression tree, which can serve as a new methodology to address many industrial challenges that can be formulated as matrix completion problems.
- Lori addresses the one-to-one mapping issue in large-scale matrix completion problems when considering both covariates and response matrix, hence providing more robust modeling performance.

We also summarize the following contributions to specific transformative applications in industrial informatics:

- The proposed automatic AI configuration method prevents industrial decision makers from time-consuming exploration by traversing all candidate machine learning method options in a trial-and-error manner.
- Both small-scale and large-scale case studies demonstrates Lori's superior performance in configuring AI towards better modeling accuracy, especially in cold-start scenarios when no prior AI evaluation record is available.
- Lori can broadly benefit many industrial informatics applications by providing novel local low-rank completion methodology, such as anomaly detection, predictive maintenance, demand forecasting, missing data imputation, etc.

In addition, the regression tree methodologies are also advanced by proposing a novel multivariate splitting method based on imputed responses. Thus, it can be easily extended to other regression problem by imputing the responses using many low-rank approximations, such as matrix factorization [26], basis expansion [27], etc.

The remaining part of this paper is organized as follows. In Section III, the proposed Lori is introduced. A small-scale and a large-scale computation pipeline recommendation case

studies to validate the Lori are discussed in Section III and Section IV, respectively. Section VII draws the conclusions and provides potential future directions.

II. Related Work

Literature review from three perspectives are performed in this section, namely, major RecSys methodologies, computation pipeline RecSys, local low-rank matrix approximation, and the tree splitting methods for regression trees.

A. Computation Pipelines and Recommendation

The concept of pipeline for machine learning originates from software engineering as a way to organize machine learning method options in a sequence for the convenience of programming. For example, *Scikit-learn*, which is a machine learning library for Python programming language, proposed a pipeline of transforms with a final estimator to assemble several steps that can be cross-validated together while setting different parameters [28]. Similarly, the widely adopted deep learning platforms Google® *Tensorflow* [29] and Pytorch [30] promoted the idea of computational graph as deep learning pipeline to organize deep learning operations and layers into graph. These concepts of pipeline enhance the readiness and traceability to use machine learning and deep learning method options. However, user needs to test the performance of pipelines with many trials according to their experience and expertise. For large-scale distributed data analytics, [28] developed *KeystoneML* to optimize the computation and communication workloads for advanced machine learning pipelines via a greedy method. However, the prediction accuracy of the pipeline execution results may not be optimal. As such, [31] proposed an adaptive data fusion pipeline methodology based on learning-to-rank to rank data fusion pipelines according to their predicted statistical accuracy (i.e., NRMSEs). Though performing well on three manufacturing data sets, this methodology requires exploratory computation for feature extraction step. Different from [31], the construction of computation pipelines for AdaPipe enables the flexibility for different type of computation services since the steps can be readily redefined or reorganized according to different method options. For instance, a feature extraction step can be eliminated in computation pipelines if functional models, such as [32], are investigated.

B. Matrix Completion and Ranking

Recommender systems are typically applied to user-item interaction completion problems. Specifically, organizing users' ratings for multiple items (e.g., movies, musics, books, etc.) as a user-item matrix with arbitrarily missing entries, the models in recommender systems provide predictions for missing entries in this matrix by assuming the its low-rank property, and then make recommendations based on the predicted ratings [33]. For example, Netflix® recommendation challenge investigated models to accurately suggest the right movie to the right user according to the predicted ratings for missing entries in a large sparse user-movie rating matrix [34]. In the last two decades, various techniques have been proposed for Recommender systems, such as collaborative filtering which generates recommendations by similar items liked by a user and similar users who liked the same item [35]; content-based filtering which provides recommendations by similarity among contents of items [36]; knowledge-based filtering which assumes the existence of users' or items' background knowledge and recommends according to this knowledge [37]; social network-based filtering which suggests the items according to the social network between users [38]; and hybrid methods which integrate the aforementioned techniques to enhance their recommendation performance [39]. Among the aforementioned techniques, collaborative filtering has been extensively studied by using three types of models, i.e., matrix completion [40], matrix factorization [41], and neural network-based models [42] for real-world recommendation applications. These methods typically require relatively large sample size to achieve desirable performance, since the similarity among entries of the user-item matrix can not be easily quantified with few sample size. Moreover, the aforementioned RecSys that involve covariates cannot address the one-to-one mapping issue.

C. Local Low-rank Matrix Approximation

In recent years, local low-rank matrix approximation methods have been developed aiming to completing the arbitrarily missed entries in a user-item rating matrix in RecSys. A notable attempt in local low-rank matrix approximation was made by [43] to randomly generate sub-matrices from the original rating matrix, then to complete these sub-matrices as predictions for the arbitrarily missed entries. This method cannot guarantee the low-rank property of the generated sub-matrices, since the selected users may not share similar interests, while the selected items may from different categories. Since then, the focuses of existing studies are on the generation of local low-rank sub-matrices with well-established collaborative filtering methods (e.g., low-rank matrix completion [40], matrix factorization [41], etc.) to complete these sub-matrices. For example, [16] proposed to approximate the original rating matrix as a smoothed weighted sum of sub-matrices, which are nearest neighbors of each entry in the original rating matrix. This research was further improved by strategically selecting anchor points to generate the local low-rank sub-matrices with certain dimensions [21], [44]. Along with this smoothed weighting direction, [22] investigated a co-clustering method to partition the original rating matrix into a set of submatrices with overlaps, hence a weighted and ensemble local low-rank approximation model was proposed; and [45] proposed an additive model by identifying local low-rank structures based on a Bayesian co-clustering method. However, depending on collaborative filtering methods, the aforementioned methods only consider to partition the original rating matrix by considering the explicit similarities expressed by the rating matrix itself. The implicit similarities contained in covariates are generally ignored or not available for use, thus, existing methods cannot support the matrix partitioning on both covariates and the rating matrix.

D. Tree Splitting Methods

Tree splitting methods for regression problems provide another potential direction for partition, specifically, in the space defined on covariates or transformed covaraites. In general, existing splitting methods can be categorized into two groups: 1) univariate splitting methods, and 2) multivariate splitting methods. Researches towards univariate splitting direction aims at investigating new splitting criteria to effectively identify one variable and threshold at one time for binary splitting in the space defined on covariates [46]. Many univariate splitting criteria have been proposed, for example, [47] proposed classification and regression trees (CART) with two optional splitting criteria, i.e., gini impurity measure and twoing criterion. Besides, [48] proposed to identify splits based on a likelihood ratio statistic; Mean posterior improvement was investigated as a splitting criterion by [49]; Chi-squared criterion was studied for classification trees in [50] and regression trees [51]. The major limitation of the aforementioned criteria lies in testing a single variable at a node, which typically results in much larger tree structures than multivariate splitting [52].

For multivariate splitting criteria, in general, transformations are applied to the covariates to create a new space and grow a tree in this space. For example, linear discriminant analysis (LDA) was adopted to transform the covariates for a multivariate splitting as described in [53], which demonstrated the advantage of multivariate splitting over CART in significantly smaller tree sizes without growing then pruning. LDA has also been used for other type of multivariate splitting methods, see examples in [54]–[56]. Besides, a pHd-based multivariate splitting method were proposed by [25] as a second order method to better detect the curvature effects in the expected Hessian matrix, hence yielding superior performance. Unlike other tree models, [25] provided a successful attempt to generate splits by jointly consider the covariates and the responses. One limitation of this approach lies in its dependence on the accurate estimation of pHds, in other words, the tree structure and the performance are sensitive to pHds as pointed out in a sensitivity study [57]. Specifically, noisy pHds may have significant impacts on the growth of a tree. Besides, this method is not applicable when missing responses exist, which is the case for the computation pipeline recommendation problem. Therefore, a robust pHd-based multivariate splitting method with imputed responses is desired for our problem.

III. Lori

We propose Lori for responsive and scalable computation services with the following assumptions: 1) covariates for the contexts and computation pipelines are available in Lori; and 2) the context-pipeline response matrix has local low-rank structures after applying some elementary matrix transformations on rows and columns. In general, Assumption 1 holds for almost all the scenarios in a computation pipeline recommendation problem. Because covariates can be readily extracted either from the embedding machine and meta-data extraction machine [4] or simply from the factorized matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ via non-negative matrix factorization $\boldsymbol{Y} = \boldsymbol{UV}$ [26]. And Assumption 2 is a relaxed assumption for low-rank matrix completion methods that typically require the existence of a global low-rank structure in the response matrix, which may not be true especially for small-scale problems. The Assumption 2 will be validated in both a small-scale and a large-scale recommendation case studies in Section V and Section V, respectively. In this section, we will firstly introduce a novel robust pHd method defined on the a EMC model, then detail Lori.

A. Robust Principal Hessian Directions

We firstly revisit the original pHd method proposed by [58], which is a direct application of Stein's Lemma [59]. Following the notations in Model 1, we firstly define $\boldsymbol{y} = \mathcal{A}^{-1}(\boldsymbol{Y})$, where $\mathcal{A}^{-1}(\cdot)$ is a inverse linear mapping function that maps from $\mathbb{R}^{mn\times 1}$ back to $\mathbb{R}^{m\times n}$; $\boldsymbol{r} = \mathcal{A}^{-1}(\boldsymbol{R})$ as the vectorized low-rank matrix; $\boldsymbol{\epsilon} = \mathcal{A}^{-1}(\boldsymbol{E})$ as the vectorized error matrix. The expected Hessian matrix is then defined as $\bar{\boldsymbol{H}} = \mathbb{E}[\partial^2 m(\boldsymbol{X})/\partial \boldsymbol{X}\partial \boldsymbol{X}^T]$, where $m(\boldsymbol{X}) = \mathbb{E}[\boldsymbol{Y}|\boldsymbol{X}]$ is the conditional mean function. According to the Stein's Lemma, if $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma_X})$, then $\bar{\boldsymbol{H}}$ can be derived as Equation 2:

$$\bar{\boldsymbol{H}} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{yXX}}\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}, \tag{2}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{yXX}} = \frac{1}{mn}\sum_{i=1}^{mn}(y_i - \bar{y})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$, $\boldsymbol{x}_i$ is the i-th row in $\boldsymbol{X}$ representing one sample, $\bar{y}$ and $\bar{\boldsymbol{x}}$ are the mean values for $\boldsymbol{y}$ and $\boldsymbol{X}$, respectively. Note that the discussion in [58] extends to elliptic distributions of $\boldsymbol{X}$ and other weaker condition of linearity. Hence, the normality in $\boldsymbol{X}$ is not strongly required to produce good estimation of pHds. Observing that subtracting a hyperplane from $\boldsymbol{Y}$ will not change the expected Hessian matrix, a residual-based ($\boldsymbol{\epsilon}$-based) pHd variant can be defined on the residuals $\boldsymbol{\epsilon}$ of a multiple linear regression model $\boldsymbol{y} = \boldsymbol{XB} + \boldsymbol{\epsilon}$:

$$\bar{\boldsymbol{H}}^{(\boldsymbol{\epsilon})} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{\epsilon XX}}\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}, \tag{3}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon XX}} = \frac{1}{mn}\sum_{i=1}^{mn}\epsilon_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$.

Based on Equation 3, the directions can be found following an eigenvalue decomposition of $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon XX}}$ with respected to $\boldsymbol{\Sigma}_{\boldsymbol{X}}$:

$$\begin{aligned}\boldsymbol{\Sigma}_{\boldsymbol{\epsilon XX}}\boldsymbol{b}_j^{(\boldsymbol{\epsilon})} &= \lambda_j^{(\boldsymbol{\epsilon})}\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{b}_j^{(\boldsymbol{\epsilon})}, \text{ for } j = 1, \ldots, p,\\ |\lambda_1^{(\boldsymbol{\epsilon})}| &\leq \cdots \leq |\lambda_p^{(\boldsymbol{\epsilon})}|,\end{aligned} \tag{4}$$

where $\boldsymbol{b}_j^{(\boldsymbol{\epsilon})}$ is the j-th eigen vector (see more theoretical analysis in [58] and discussions in [60]. We can then define the $\boldsymbol{\epsilon}$-based pHd as $\mathbf{pHd}_j^{(\boldsymbol{\epsilon})} = \boldsymbol{X}\boldsymbol{b}_j^{(\boldsymbol{\epsilon})}$.

pHd aims at finding the directions along with the response surface in a reduced dimension space (r.d.s.). For example, as shown in the left panel of Figure 3, the residual surface of a linear regression (i.e., $\boldsymbol{Y} - \mathcal{A}(\boldsymbol{XB})$ in our notation) implies a curvature, which can be identified by the pHd method. By splitting this residual surface along with the first pHd, each branch can be well approximated by a hyperplane (i.e., a linear regressor) defined on the space which is expanded on covariates $(\boldsymbol{x}_1, \boldsymbol{x}_2)$.

However, as pointed out in Section 2.4, two important issues prevent the direct adoption of the pHd method in the matrix completion application, namely, 1) the solution of the $\boldsymbol{\epsilon}$-based

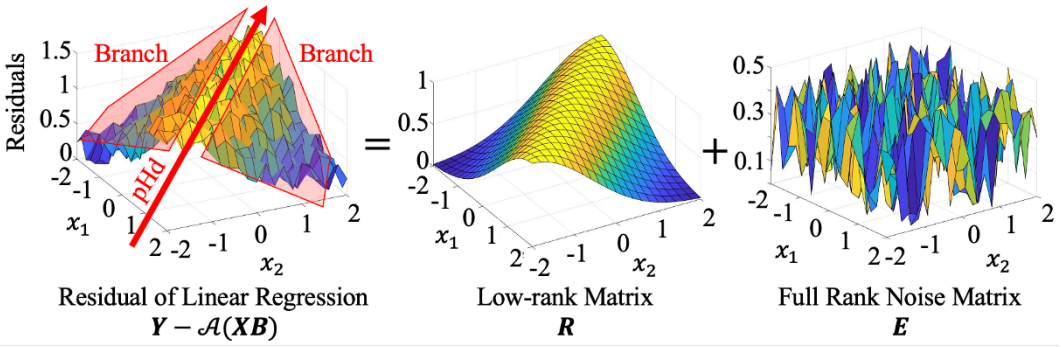


Fig. 3. A toy example of multivariate splitting on curvature via pHds and its robust enhancement based on the extended matrix completion model in Model 1.

pHds is sensitive to the noises in $\boldsymbol{y}$, and 2) the estimation of $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon XX}}$ requires fully observed responses without missing values. Therefore, in this paper, a robust pHd method is proposed based on the EMC Model 1 with least square loss function $L(\widehat{\boldsymbol{Y}},\boldsymbol{Y}) = \frac{1}{2}\left\|\mathcal{P}_\Omega(\boldsymbol{Y}) - \mathcal{P}_\Omega(\boldsymbol{R}) - \mathcal{P}_\Omega\left(\mathcal{A}\left(\boldsymbol{XB}\right)\right)\right\|_{\mathrm{F}}^2$, which can be efficiently solved by an alternating direction method of multipliers (ADMM) solver as proposed in [24]. The key idea to make pHds robust to noise and missing values in responses is to separate the curvatures from Gaussian noises, and to provide a reasonable guess for the missing entries in the context-pipeline response matrix. The EMC model addresses this issue by decomposing the residuals of a linear regression term $\boldsymbol{Y} - \mathcal{A}(\boldsymbol{XB})$ to be an additive model of a low-rank matrix that represents curvatures, and a full-rank matrix that contains identically and independently distributed Gaussian noises. Estimating Model 1 results in a low-rank matrix $\hat{\boldsymbol{R}}$ with missing entries completed, hence imputing the responses. Therefore, the robust pHd method (i.e., $\boldsymbol{r}$-based pHd) can be proposed as:

$$\bar{\boldsymbol{H}}^{(\boldsymbol{r})} = \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{rXX}}\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}, \tag{5}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{rXX}} = \frac{1}{mn}\sum_{i=1}^{mn} r_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$. Then, the eigenvalue decomposition described in Equation 4 can be used to find the robust directions. As an illustration of the robust pHd method, Figure 3 provides a toy example, where the residual matrix (Left) of the linear regression can be further decomposed as a low-rank matrix (Middle) adding a full-rank noise matrix (Right) via the estimation of Model 1. Hence, the proposed robust pHd method can reduce the impacts of noise on estimating pHds.

The difference between the robust pHds and the $\boldsymbol{\epsilon}$-based pHds is analyzed by Theorem 1, which also provide a close-form solution to investigate the impacts of noise on the $\boldsymbol{\epsilon}$-based pHds.

Theorem 1: Let $\boldsymbol{\Sigma}_{\boldsymbol{rXX}}\boldsymbol{b}_j^{(\boldsymbol{r})} = \lambda_j^{(\boldsymbol{r})}\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{b}_j^{(\boldsymbol{r})}$ identifies the robust pHds, $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon XX}}\boldsymbol{b}_j^{(\boldsymbol{\epsilon})} = \lambda_j^{(\boldsymbol{\epsilon})}\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{b}_j^{(\boldsymbol{\epsilon})}$ identifies the $\boldsymbol{\epsilon}$-based pHds, and $\boldsymbol{\epsilon} = \boldsymbol{r} + \boldsymbol{e}$, $\lambda_j^{(\boldsymbol{\epsilon})}$ and $\boldsymbol{b}_j^{(\boldsymbol{\epsilon})}$ can be expressed as:

$$\begin{aligned}
\lambda_j^{(\boldsymbol{\epsilon})} &= \lambda_j^{(\boldsymbol{r})} + {\boldsymbol{b}_j^{(\boldsymbol{r})}}^T\boldsymbol{\Sigma}_{\boldsymbol{eXX}}\boldsymbol{b}_j^{(\boldsymbol{r})},\\
\boldsymbol{b}_j^{(\boldsymbol{\epsilon})} &= \boldsymbol{b}_j^{(\boldsymbol{r})}\left(1 - \frac{1}{2}{\boldsymbol{b}_j^{(\boldsymbol{r})}}^T\boldsymbol{\Sigma}_{\boldsymbol{eXX}}\boldsymbol{b}_j^{(\boldsymbol{r})}\right)\\
&\quad + \sum_{i\neq j}^{p}\frac{{\boldsymbol{b}_j^{(\boldsymbol{r})}}^T\boldsymbol{\Sigma}_{\boldsymbol{eXX}}\boldsymbol{b}_i^{(\boldsymbol{r})}}{\lambda_j^{(\boldsymbol{r})} - \lambda_i^{(\boldsymbol{r})}}\boldsymbol{b}_j^{(\boldsymbol{r})},
\end{aligned} \tag{6}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{eXX}} = \frac{1}{mn}\sum_{i=1}^{mn} e_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$ is a randomly weighted variance-covariance matrix, and $e_i \sim \mathcal{N}(0,\sigma^2)$.

Therefore, an asymptotic relation exists between the $\boldsymbol{\epsilon}$-based pHds and the proposed robust pHds as described in Corollary 2.

Corollary 2: Let each element e_i in $\boldsymbol{e}$ identically and independently follows $\mathcal{N}(0,\sigma^2)$ with constant variance σ^2, the following asymptotic property exists:

$$\begin{aligned}
\lim_{\sigma^2\to 0^+}\lambda_j^{(\boldsymbol{\epsilon})} &\to \lambda_j^{(\boldsymbol{r})},\\
\lim_{\sigma^2\to 0^+}\boldsymbol{b}_j^{(\boldsymbol{\epsilon})} &\to \boldsymbol{b}_j^{(\boldsymbol{r})}.
\end{aligned} \tag{7}$$

These asymptotic properties also proof the robustness of the proposed robust pHd method, which serves as the basis for a multivariate splitting method. Theorem 1 and Corollary 2 jointly guarantee that the pHds can be robustly identified even from a noisy full-rank matrix by Lori.

B. Multivariate Splitting Method for Treed Recommender System

A multivariate splitting method can be proposed based on the robust pHds to develop a treed RecSys for computation pipeline recommendation. The key idea is to redefine the local low-rank property in the effective dimension reduction (e.d.r., [58]) space expanded by the robust pHds by existing impurity measures, such as entropy, gini index, chi-squred criterion, etc. Then, the context-pipeline response matrix can be partitioned as the growth of a regression tree that performs traditional univariate splitting in the e.d.r. space, which is equivalent to multivariate splitting in the original covariates-response space. These splits are well generated since splitting on robust pHds (i.e., the directions of curvatures) directly partition the response surface into local "flat" surfaces that can be well

Algorithm 1 Lori with Multivariate Splitting Method

1: Estimate $\hat{\boldsymbol{R}}$, and $\hat{\boldsymbol{B}}$ in Model 1 via an ADMM solver detailed in [24],
2: Compute $\boldsymbol{r} = \mathcal{A}(\boldsymbol{R})$, $\boldsymbol{\Sigma}_{\boldsymbol{rXX}} = \frac{1}{mn}\sum_{i=1}^{mn} r_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$, $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \frac{1}{mn-1}\boldsymbol{X}^T\boldsymbol{X}$,
3: Solve for $\lambda_j^{(\boldsymbol{r})}$ and $\boldsymbol{b}_j^{(\boldsymbol{r})}$ for $j = 1, \ldots, p$ via the generalized eigenvalue decomposition of $\boldsymbol{\Sigma}_{\boldsymbol{rXX}}$ with respected to $\boldsymbol{\Sigma}_{\boldsymbol{X}}$,
4: Transform $\boldsymbol{X}$ to the e.d.r. space by $\mathbf{pHd}_j = \boldsymbol{X}\boldsymbol{b}_j^{(\boldsymbol{r})}$, for $j = 1, \ldots, p$,
5: Organize new covariates in expanded e.d.r. space by $\tilde{\boldsymbol{X}} = (\mathbf{pHd}_1, \ldots, \mathbf{pHd}_p, \boldsymbol{r})$,
6: Grow linear regression trees on $\{\mathcal{P}_\Omega(\tilde{\boldsymbol{X}}), \mathcal{P}_\Omega(\boldsymbol{y})\}$ based on any univariate splitting linear regression tree methods, here operator $\mathcal{P}_\Omega(\cdot)$ select training samples from non-empty entries in $\boldsymbol{Y}$,
7: Predict the missing entries by using the trained tree model.

approximated by linear regression as hyperplanes. However, such a splitting method may not help to address the one-to-one mapping issue in each local low-rank partition, since the curvature effects may hinder the one-to-one relationship between robust pHds and responses. Therefore, we further expand the e.d.r. space by including the estimation of curvature $\hat{\boldsymbol{R}}$, such that splits generated on $\hat{\boldsymbol{R}}$ will prevent the one-to-one mapping problem.

The full algorithm for the proposed Lori is then summarized in Algorithm 1. Note that the the selection of tuning parameters λ_1 and λ_2 will not have significant impacts on the recommendation accuracy of the proposed Lori. Because the recommendation accuracy is majorly contributed by the linear regression tree, which is not sensitive to the scale of the robust pHds and $\boldsymbol{r}$. Besides, the trend of curvatures express by $\hat{\boldsymbol{R}}$ will not change too much along with tuning parameters in a reasonable range. According to some numerical studies, $\lambda_1 = 2$, and $\lambda_2 = 1$ are suggested for large-scale applications to avoid comprehensive tuning via cross-validations. For large-scale applications, another strategy to save computation power is to only use a small proportion of the robust pHds according to eigenvalues to organize the new covariates in the expanded e.d.r. space. The reason is that the first few robust pHds can typically explain more than 90% variance, hence the rest robust pHds are likely to be less informative.

IV. Small-scale Case Study

We firstly validate the proposed Lori in the same small-scale case study in [4] with three manufacturing process and 60 bootstraped manufacturing datasets to represent sixty manufacturing contexts. This case study employed 27 computation pipelines, as presented in Figure 4, with three steps from feature extraction, tuning criterion, to candidate models, and three method options for each step.

The objective of this case study is to identify the best computation pipeline, which provides the lowest prediction errors (i.e., the lowest NRMSEs) among all the candidate pipelines, for each manufacturing context. Therefore, after

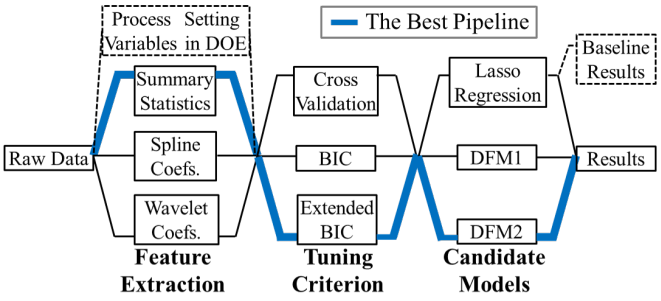


Fig. 4. Computation pipelines for the small-scale case study with three steps and three method options in each step. The best pipeline is highlighted in bold blue. (Redrawn from [4] with authors' permission).

obtaining the true context-pipeline response matrix $\boldsymbol{Y}_{\text{true}} \in \mathbb{R}^{60\times 27}$ without missing entries, we generated two scenarios to evaluate the Lori, namely, 1) warm-start scenario, and 2) cold-start scenario. Here the warm-start scenario refers to arbitrarily entry missing with certain probabilities in $\boldsymbol{Y}_{\text{true}}$, specifically, five cases (i.e., five $\boldsymbol{Y}$ matrices) are generated with missing probabilities $\rho_f \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, for $f = 1, \ldots, 5$ by setting arbitrarily selected $\rho_f \times mn$ entries in $\boldsymbol{Y}_{\text{true}}$ to 0. This is the scenario that most of existing RecSys are developed upon, e.g., the MovieLens datasets include user-movie rating matrices with arbitrarily missed entries [23]. In the cold-start scenario, $\boldsymbol{Y}$ is generated from $\boldsymbol{Y}_{\text{true}}$ with entire missing rows, which corresponds to a common manufacturing scenario that no computation pipelines have been executed before recommendation.

For both scenarios, different training-testing splitting methods are adopted to generate replicates. Specifically, for the f-th case in warm-start scenario, ten replicates are generated with $(1-\rho_f)\times mn$ training samples arbitrarily selected from $\boldsymbol{Y}_{\text{true}}$ for each replicate, and the rest $\rho_f \times mn$ entries are set to be zeros. In cold-start scenario, 60-fold leave-one-fold-out cross-validation is employed, such that the i-th row in $\boldsymbol{Y}_{\text{true}}$ can be left out for testing in the i-th fold, while the rest rows are used for training. Thus, in total 60 replicates are created for the cold-start scenario. To evaluate the prediction accuracy, root-mean-square errors (RMSE) defined in Equation 8 was used.

$$RMSE = \sqrt{\frac{\sum_{(i,j)\in\bar{\Omega}}(\hat{y}_{i,j} - y_{i,j})^2}{|\bar{\Omega}|}}, \tag{8}$$

where $\bar{\Omega}$ is the index set of the testing samples, and $|\cdot|$ is the cardinality of a set. In addition, to evaluate the ranking accuracy, Normalized Discounted Cumulative Gain of the top-3 items (NDCGk) was adopted, defined in Equation 9.

$$\begin{aligned} DCG_3 &= \sum_{i=1}^{3} \frac{2^{r_{i,z}}}{\log_2(r+1)}, \\ NDCG_3 &= \frac{DCG_3}{IDCG_3}, \end{aligned} \tag{9}$$

where $r_{i,z}, z \in \{1, ..., n\}$ is a binary variable for the actual rank z of the predicted top-i pipelines; And the $IDCG_3$ represents the ideal discounted cumulative gain, evaluating the cumulative gain of the optimal ranking. Five benchmark

models are selected, namely, the well-tuned EMC model, a proximal nuclear norm minimization-based matrix completion model [61], a Lasso linear regression model [62], a popular neural collaborative filtering (NCF) model [18], [63], and a standard non-negative matrix factorization (NNMF) model [26]. The Lasso linear regression model only considers the covariates, the matrix completion and the NNMF models only consider the response matrix, and the EMC model jointly considers both. The NCF model will be trained by using the covariates of contexts as the users' vectors, and using the covariates of computation pipelines as the items' vectors following the original definition in [18], [63]. The comparison with the selected benchmark methods will validate the advantages of the robust pHds-based multivariate splitting Lori. Note that none of the local low-rank approximation models were selected as benchmark, because they were designed for matrix completion without consideration of covariates.

Covariates for each entry in $\boldsymbol{Y}_{\text{true}}$ is generated by convolving the context features (i.e., 17-dimensional) from the meta-data generation machine and the pipeline features (i.e., 48-dimensional) from the embedding machine in AdaPipe system [4]. Convolution is adopted here to create two-way interactions between contexts and computation pipelines. As a result, $\boldsymbol{X} \in \mathbb{R}^{1620\times 64}$ are generated in this case study. For an alternative way mentioned in Section 3, covariates can also be generated by factorizing matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ via non-negative matrix factorization $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{V}$ [26].

The results for small-scale case study are summarised in Table II with the significantly lowest averaged NRMSEs highlighted in **bold**. It can be easily observed that the the proposed Lori model yields the best over benchmark models, especially when being tested in the cold-start scenario. Statistical significance can be The superior performance can be mainly attributed to the relaxation from global low-rank structure to local low-rank structure. Specifically, in such a small-scale scenario with 60 contexts and 27 pipelines, the global low-rank assumption does not exist (i.e., $\text{rank}(\boldsymbol{Y}) = 27$). Thus, matrix completion model and EMC model cannot perform well due to the violated assumptions. However, as presented in Figure 2, local low-rank structures exist after performing certain elementary matrix transformations to switch rows and columns. Therefore, the proposed Lori outperforms benchmark methods by identifying and modeling these "implicit" local low-rank structures in the expanded e.d.r. space.

We also investigated the singular values of the original response matrix and one example of the local low-rank matrix. As presented in Figure 5, the original response matrix is, in fact, a full-rank matrix whose singular values slowly decrease; while the highlighted submatrix on the bottom presents a local low-rank structure with rapidly decreasing singular values. This finding justifies the reason why the proposed Lori model achieves the best performance over benchmarks that assume a global low-rank property.

V. Large-scale Case Study

A large-scale case study is conducted to evaluate the responsiveness and scalability of the proposed Lori. In this study,

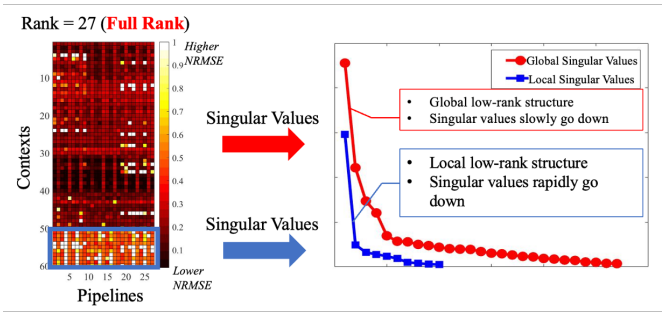


Fig. 5. Singular values of the original response matrix (left) and one example of the local low-rank matrix (bottom left within the blue box).

81 manufacturing contexts were randomly generated from a validated simulation model of a set of similar-but-non-identical manufacturing processes by varying process setting variables and *in situ* process variables. 1360 computation pipelines as graphically demonstrated in Figure 1 serve as the candidates to be recommended. Hence, after executing 1360 candidate pipelines on all 81 data sets, the ground truth $\boldsymbol{Y}_{\text{true}} \in \mathbb{R}^{81\times 1360}$ can be obtained, where the (i, j)-th entry in $\boldsymbol{Y}_{\text{true}}$ records the NRMSE after testing the j-th computation pipeline on the i-th context. The true response matrix is graphically presented in Figure 6, where both global and local low-rank structures can be identified.

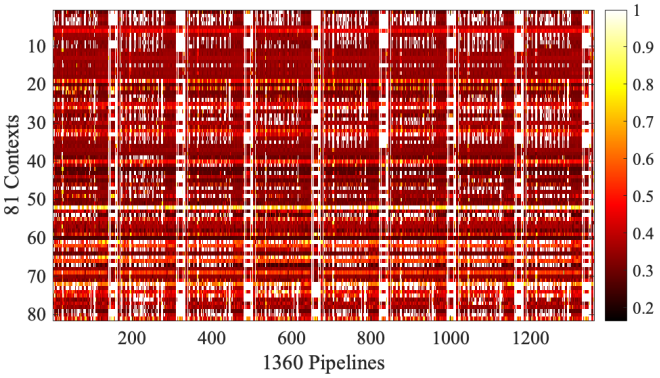


Fig. 6. A visualization of the true context-pipeline response matrix of the large-scale case study with 81 contexts and 1360 computation pipelines.

To validate whether the proposed method can address the one-to-one mapping issue, we did not increase the dimension of covariates to create the aforementioned one-to-one mapping issue. Specifically, covariates are generated from the same number of features, which leads to a 64-dimensional covariates matrix $\boldsymbol{X} \in \mathbb{R}^{(81\times 1360)\times 64}$. The same case study setup as is adopted for the large-scale case study, thus resulting in the testing results summarized in Table IV. The Lori maintained its superior performance over benchmark models in both scenarios.

We further investigate the one-to-one mapping issue in this large-scale recommendation problem by visualizing the covariates-response relationship in the e.d.r. space expanded on the robust pHds. As presented in Figure 7(a), the one-to-multiple mapping can be easily observed in the pHd-y

TABLE II
AVERAGE RMSES AND STANDARD ERRORS (WITHIN PARENTHESIS) OF THE PROPOSED LORI AND FIVE BENCHMARK MODELS IN THE SMALL-SCALE CASE STUDY, WHERE THE SIGNIFICANTLY LOWEST AVERAGED RMSES ARE HIGHLIGHTED IN **BOLD**.

Models	Warm Start					Cold Start
	$\rho = 0.1$	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	
Lori	**0.043 (0.010)**	**0.072 (0.015)**	**0.086 (0.007)**	**0.098 (0.011)**	**0.118 (0.017)**	**0.038 (0.012)**
EMC	0.111 (0.008)	0.117 (0.009)	0.121 (0.007)	0.128 (0.009)	0.137 (0.003)	0.122 (0.004)
Matrix Completion	0.149 (0.018)	0.175 (0.009)	0.185 (0.006)	0.187 (0.003)	0.187 (0.001)	0.140 (0.001)
Lasso	0.135 (0.010)	0.128 (0.007)	0.130 (0.005)	0.130 (0.002)	0.175 (0.020)	0.124 (0.016)
NCF	0.117 (0.012)	0.136 (0.020)	0.142 (0.015)	0.150 (0.021)	0.163 (0.019)	0.112 (0.020)
NNMF	0.143 (0.015)	0.156 (0.008)	0.178 (0.005)	0.179 (0.003)	0.178 (0.003)	0.140 (0.001)

TABLE III
AVERAGE $\boldsymbol{NDCG_3}$ AND STANDARD ERRORS (WITHIN PARENTHESIS) OF THE PROPOSED LORI AND FIVE BENCHMARK MODELS IN THE SMALL-SCALE CASE STUDY, WHERE THE SIGNIFICANTLY LOWEST AVERAGED $\boldsymbol{NDCG_3}$ ARE HIGHLIGHTED IN **BOLD**.

Models	Warm Start					Cold Start
	$\rho = 0.1$	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	
Lori	**0.732 (0.030)**	**0.706 (0.045)**	**0.659 (0.047)**	**0.592 (0.091)**	**0.587 (0.098)**	**0.727 (0.068)**
EMC	0.621 (0.084)	0.579 (0.086)	0.552 (0.102)	0.508 (0.106)	0.453 (0.126)	0.494 (0.121)
Matrix Completion	0.259 (0.086)	0.208 (0.075)	0.193 (0.051)	0.162 (0.039)	0.159 (0.028)	0.204 (0.021)
Lasso	0.325 (0.080)	0.317 (0.057)	0.303 (0.051)	0.258 (0.036)	0.249 (0.095)	0.342 (0.084)
NCF	0.649 (0.065)	0.604 (0.092)	0.587 (0.083)	0.552 (0.101)	0.518 (0.095)	0.628 (0.082)
NNMF	0.283 (0.052)	0.276 (0.039)	0.239 (0.029)	0.198 (0.023)	0.194 (0.020)	0.204 (0.021)

TABLE IV
AVERAGE RMSES AND STANDARD ERRORS (WITHIN PARENTHESIS) OF THE PROPOSED LORI AND FIVE BENCHMARK MODELS IN THE LARGE-SCALE CASE STUDY, WHERE THE SIGNIFICANTLY LOWEST AVERAGED RMSES ARE HIGHLIGHTED IN **BOLD**.

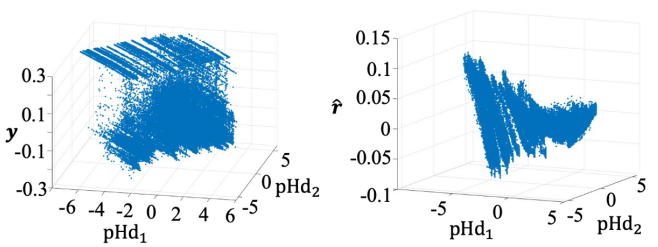
Models	Warm Start					Cold Start
	$\rho = 0.1$	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	
Lori	**0.064 (0.002)**	**0.082 (0.002)**	**0.099 (0.002)**	**0.116 (0.003)**	**0.145 (0.004)**	**0.051 (0.002)**
EMC	0.102 (0.003)	0.121 (0.001)	0.146 (0.003)	0.160 (0.001)	0.165 (0.002)	0.134 (0.002)
Matrix Completion	0.144 (0.000)	0.170 (0.000)	0.198 (0.000)	0.229 (0.000)	0.253 (0.000)	0.246 (0.000)
Lasso	0.125 (0.002)	0.129 (0.001)	0.176 (0.001)	0.194 (0.000)	0.205 (0.000)	0.232 (0.001)
NCF	0.095 (0.018)	0.105 (0.011)	0.128 (0.021)	0.137 (0.020)	0.152 (0.032)	0.092 (0.012)
NNMF	0.138 (0.000)	0.154 (0.000)	0.183 (0.000)	0.198 (0.000)	0.203 (0.000)	0.246 (0.000)

TABLE V
AVERAGE $\boldsymbol{NDCG_3}$ AND STANDARD ERRORS (WITHIN PARENTHESIS) OF THE PROPOSED LORI AND FIVE BENCHMARK MODELS IN THE LARGE-SCALE CASE STUDY, WHERE THE SIGNIFICANTLY LOWEST AVERAGED $\boldsymbol{NDCG_3}$ ARE HIGHLIGHTED IN **BOLD**.

Models	Warm Start					Cold Start
	$\rho = 0.1$	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$	
Lori	**0.542 (0.017)**	**0.519 (0.032)**	**0.485 (0.030)**	**0.462 (0.023)**	**0.438 (0.020)**	**0.557 (0.010)**
EMC	0.452 (0.023)	0.437 (0.031)	0.406 (0.023)	0.378 (0.021)	0.335 (0.016)	0.424 (0.015)
Matrix Completion	0.173 (0.000)	0.156 (0.000)	0.142 (0.000)	0.128 (0.000)	0.121 (0.000)	0.118 (0.000)
Lasso	0.238 (0.017)	0.211 (0.012)	0.196 (0.009)	0.188 (0.009)	0.175 (0.008)	0.230 (0.001)
NCF	0.394 (0.088)	0.371 (0.067)	0.313 (0.085)	0.294 (0.090)	0.238 (0.093)	0.384 (0.053)
NNMF	0.190 (0.000)	0.178 (0.000)	0.165 (0.000)	0.142 (0.000)	0.130 (0.000)	0.118 (0.012)

relation, thus limiting the performance of benchmark models, especially the Lasso linear regression model. The best way to reduce such a one-to-multiple mapping problem to one-to-one mapping is to split on the response $\boldsymbol{y}$ following certain directions. However, splitting on $\boldsymbol{y}$ requires the knowledge of the ground truth $\boldsymbol{Y}_{\text{true}}$, which is contradicted to the matrix completion problem.

By adopting the proposed Lori, Figure 7(b) presents the identified curvatures in the estimated low-rank matrix $\hat{\boldsymbol{R}}$ in the same e.d.r. space. By jointly splitting on the e.d.r. space and $\hat{\boldsymbol{r}}$ following Algorithm 1, the one-to-one mapping can be established by partitioning the expanded e.d.r. space into local low-rank spaces. The execution of the Lori only take less than two minutes on a Lenovo Thinkpad T460p laptop with an i7-6820HQ processor @2.7GHz, while the EMC and Lasso models took over 20 minutes (on average) for hyperparam-



(a) One-to-one mapping not satisfied

(b) Enable one-to-one mapping by estimating curvatures

Fig. 7. Scatter plots to demonstrate the one-to-one mapping issue, where (a) visualizes the covariates-response relationship in the e.d.r. space expanded on the robust pHds, and (b) represents the curvatures identified by the estimated low-rank matrix $\hat{\boldsymbol{R}}$ in the same space.

eter tuning. In summary, the proposed Lori demonstrated its advantages in providing responsive and scalable computation services for cybermanufacturing systems.

VI. Note to Practitioners

To deploy and interpret Lori in practice, the following steps are recommended:

- (Initialization) Define the response matrix as a context-pipeline matrix by filling in the performance scores (e.g., RMSEs for regression analysis and accuracies for classification) collected from historical evaluation records. For new context (i.e., in the form of dataset), append a new empty row to the response matrix. Covariates for both the contexts and the pipelines can be generated based on the process described in [4].
- (Lori Execution) Both covariates and response matrix can then be filled into Algorithm 1, which will result in a completed response matrix.
- (Results Interpretation) In the completed response matrix, rank the predicted performance scores within the row(s) that correspond to the target context. The top-ranked pipelines can then be executed to find the truly best one.
- (Incremental) After initial execution of Lori, for any new context, it can generate a new row in the response matrix; for any new pipeline, it can generate a new column in the response matrix. Then the Lori can be estimated again to identify the best pipeline.

VII. Conclusions and Future Work

Recommendation of computation pipelines prevents the wastes of time and efforts for both researchers and practitioners to explore numerous AI methods in a trial-and-error manner. Existing RecSys for computation pipeline recommendation problems suffer from responsiveness and scalability challenges, hence being not applicable for large-scale problems. This research investigates a Lori to effectively and efficiently identify local low-rank structures by response imputation and perform multivariate splitting for linear regression trees in an expanded e.d.r. space which is defined on both the robust pHds and the estimated low-rank matrix from the EMC model. This method successfully addresses the one-to-one mapping issue as the pipeline recommendation problem scale increases from small to large with reduced computation complexity. Therefore, it can support both small- and large-scale pipeline recommendation problems with high responsiveness and scalability.

A few limitations were identified. First, the uncertainty of pipeline performance estimation on a given context was not quantified. For computation pipelines with classical machine learning methods, this is not a significant problem as most of the estimators can reach global optimality. However, the situation becomes severe for deep neural network pipelines/graphs, since the model estimation is associated with large uncertainties. Such uncertainty may direct influence the recommendation performance. Second, Lori relies on an initial estimation of the EMC model, whose computational complexity can increase exponentially along with increasing number of pipelines.

This research points out several future directions. First, we will generalize the RecSys to both vector and tensor responses by investigating new response imputation techniques, so that the responsiveness and scalability can benefit more real-world applications other than RecSys. Second, more efficient covariates generation techniques will be investigated to provide low-dimensional but highly informative covariates to better distinguish contexts and computation pipelines. Thirdly, Bayesian online learning techniques will be investigated to enable the adaptive sampling capability of the Lori, which can be sequentially improved as more entries of a context-pipeline response matrix become available in an online manner. Moreover, although the advantages of Lori were demonstrated in computational pipelines of classical machine learning methods, Lori is designed to enable generalizability to other AI configuration problems. For example, with meta data extracted from both datasets and deep learning models, Lori can be readily used for deep learning model configuration, following the similar schemes discussed in [64].

References

[1] S. Vimal, Y. H. Robinson, S. Kadry, H. V. Long, and Y. Nam, "Iot based smart health monitoring with cnn using edge computing," *Journal of Internet Technology*, vol. 22, no. 1, pp. 173–185, 2021.

[2] A. H. Fakhrulddin, X. Fei, and H. Li, "Convolutional neural networks (cnn) based human fall detection on body sensor networks (bsn) sensor data," in *2017 4th international conference on systems and informatics (ICSAI)*. IEEE, 2017, pp. 1461–1465.

[3] V. Shankar, V. Kumar, U. Devagade, V. Karanth, and K. Rohitaksha, "Heart disease prediction using cnn algorithm," *SN Computer Science*, vol. 1, pp. 1–8, 2020.

[4] X. Chen and R. Jin, "Adapipe: A recommender system for adaptive computation pipelines in cyber-manufacturing computation services," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[5] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-based systems*, vol. 212, p. 106622, 2021.

[6] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–34, 2021.

[7] Q. Wang, Y. Ming, Z. Jin, Q. Shen, D. Liu, M. J. Smith, K. Veeramachaneni, and H. Qu, "Atmseer: Increasing transparency and controllability in automated machine learning," in *Proceedings of the 2019 CHI conference on human factors in computing systems*, 2019, pp. 1–12.

[8] S. Narkar, Y. Zhang, Q. V. Liao, D. Wang, and J. D. Weisz, "Model lineupper: Supporting interactive model comparison at multiple levels for automl," in *26th International Conference on Intelligent User Interfaces*, 2021, pp. 170–174.

[9] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra*. Springer, 1971, pp. 134–151.

[10] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 201–226, 2013.

[11] J. Yu, G. Zhou, C. Li, Q. Zhao, and S. Xie, "Low tensor-ring rank completion by parallel matrix factorization," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 7, pp. 3020–3033, 2020.

[12] C. Teflioudi, F. Makari, and R. Gemulla, "Distributed matrix completion," in *2012 ieee 12th international conference on data mining*. IEEE, 2012, pp. 655–664.

[13] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems*, vol. 36, no. 5, pp. 11–20, 2020.

[14] 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.

[15] B. Vandereycken, "Low-rank matrix completion by riemannian optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1214–1236, 2013.

[16] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," in *International conference on machine learning*, 2013, pp. 82–90.

[17] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[18] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, "Neural collaborative filtering vs. matrix factorization revisited," in *Proceedings of the 14th ACM Conference on Recommender Systems*, 2020, pp. 240–248.

[19] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[20] H. Liu, L. Jing, Y. Qian, and J. Yu, "Adaptive local low-rank matrix approximation for recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 4, pp. 1–34, 2019.

[21] M. Zhang, B. Hu, C. Shi, and B. Wang, "Local low-rank matrix approximation with preference selection of anchor points," in *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017, pp. 1395–1403.

[22] C. Chen, D. Li, Y. Zhao, Q. Lv, and L. Shang, "Wemarec: Accurate and scalable recommendation through weighted and ensemble matrix approximation," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 303–312.

[23] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[24] 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*, 2020, (To appear).

[25] K.-C. Li, H.-H. Lue, and C.-H. Chen, "Interactive tree-structured regression via principal hessian directions," *Journal of the American Statistical Association*, vol. 95, no. 450, pp. 547–560, 2000.

[26] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2000.

[27] D. Gross, "Recovering low-rank matrices from few coefficients in any basis," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1548–1566, 2011.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[29] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[30] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," 2017.

[31] 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.

[32] H. Sun, S. Huang, and R. Jin, "Functional graphical models for manufacturing process modeling," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1612–1621, 2017.

[33] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[34] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA, 2007, p. 35.

[35] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender systems handbook*. Springer, 2015, pp. 77–118.

[36] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[37] S. Trewin, "Knowledge-based recommender systems," *Encyclopedia of library and information science*, vol. 69, no. Supplement 32, p. 180, 2000.

[38] J. He and W. W. Chu, "A social network-based recommender system (snrs)," in *Data mining for social network data*. Springer, 2010, pp. 47–74.

[39] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[40] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, p. 717, 2009.

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

[42] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[43] L. W. Mackey, M. I. Jordan, and A. Talwalkar, "Divide-and-conquer matrix factorization," in *Advances in neural information processing systems*, 2011, pp. 1134–1142.

[44] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio, "Llorma: Local low-rank matrix approximation," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 442–465, 2016.

[45] A. Beutel, A. Ahmed, and A. J. Smola, "Accams: Additive co-clustering to approximate matrices succinctly," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 119–129.

[46] L. Breiman, "Some properties of splitting criteria," *Machine Learning*, vol. 24, no. 1, pp. 41–47, 1996.

[47] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[48] A. Ciampi, C.-H. Chang, S. Hogg, and S. McKinney, "Recursive partition: A versatile method for exploratory-data analysis in biostatistics," in *Biostatistics*. Springer, 1987, pp. 23–50.

[49] P. C. Taylor and B. W. Silverman, "Block diagrams and splitting criteria for classification trees," *Statistics and Computing*, vol. 3, no. 4, pp. 147–161, 1993.

[50] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica sinica*, pp. 815–840, 1997.

[51] W.-Y. Loh, "Regression tress with unbiased variable selection and interaction detection," *Statistica sinica*, pp. 361–386, 2002.

[52] C. E. Brodley and P. E. Utgoff, "Multivariate decision trees," *Machine learning*, vol. 19, no. 1, pp. 45–77, 1995.

[53] W.-Y. Loh and N. Vanichsetakul, "Tree-structured classification via generalized discriminant analysis," *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 715–725, 1988.

[54] O. T. Yildiz and E. Alpaydin, "Linear discriminant trees," in *ICML*, 2000, pp. 1175–1182.

[55] G. H. John, "Robust linear discriminant trees," in *Learning From Data*. Springer, 1996, pp. 375–385.

[56] J. Gama, "Functional trees," *Machine learning*, vol. 55, no. 3, pp. 219–250, 2004.

[57] L. Heng-Hui, "A study of sensitivity analysis on the method of principal hessian directions," *Computational Statistics*, vol. 16, no. 1, pp. 109–130, 2001.

[58] K.-C. Li, "On principal hessian directions for data visualization and dimension reduction: Another application of stein's lemma," *Journal of the American Statistical Association*, vol. 87, no. 420, pp. 1025–1039, 1992.

[59] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *The annals of Statistics*, pp. 1135–1151, 1981.

[60] R. D. Cook, "Principal hessian directions revisited," *Journal of the American Statistical Association*, vol. 93, no. 441, pp. 84–94, 1998.

[61] N. Perraudin, V. Kalofolias, D. Shuman, and P. Vandergheynst, "Unlocbox: A matlab convex optimization toolbox for proximal-splitting methods," *arXiv preprint arXiv:1402.0779*, 2014.
[62] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
[63] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.
[64] C. Yang, Y. Akimoto, D. Kim, and M. Udell, "Oboe: Collaborative filtering for automl initialization," *arXiv preprint arXiv:1808.03233*, 2018.