Missingness-pattern-adaptive Learning with Incomplete Data

Yongshun Gong, *Member, IEEE*, Zhibin Li*, *Member, IEEE*, Wei Liu, *Senior Member, IEEE*, Xiankai Lu, *Member, IEEE*, Xinwang Liu, *Senior, IEEE*, Ivor W. Tsang, *Fellow, IEEE*, Yilong Yin*

Abstract—Many real-world problems deal with collections of data with missing values, e.g., RNA sequential analytics, image completion, video processing, etc. Usually, such missing data is a serious impediment to a good learning achievement. Existing methods tend to use a universal model for all incomplete data, resulting in a suboptimal model for each missingness pattern. In this paper, we present a general model for learning with incomplete data. The proposed model can be appropriately adjusted with different missingness patterns, alleviating competitions between data. Our model is based on observable features only, so it does not incur errors from data imputation. We further introduce a low-rank constraint to promote the generalization ability of our model. Analysis of the generalization error justifies our idea theoretically. In additional, a subgradient method is proposed to optimize our model with a proven convergence rate. Experiments on different types of data show that our method compares favorably with typical imputation strategies and other state-of-the-art models for incomplete data. More importantly, our method can be seamlessly incorporated into the neural networks with the best results achieved. The source code is released at https://github.com/YS-GONG/missingness-patterns.

Index Terms—Missingness patterns, adaptive learning, incomplete data classification, support vector machine, low-rank learning.

1 Introduction

Learning from incomplete data is of great practical and theoretical interest. Commonly, we are faced with incomplete data in many real-world applications, e.g., in condition-based monitoring, failure of a sensor will cause the absence of some records for a set of equipment [1]; in medical analysis, measurements on some subjects may be lost due to the lack of patient's compliance or unaffordable examination fees [2]; in urban computing problems, some areas or segments of traffic network may contain no data collectors [3]–[5]; and also there are inevitable dropouts in single-cell RNA sequencing data [6]–[8].

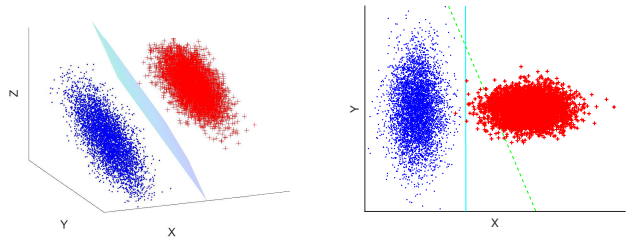
Currently, a typical strategy is to fill the missing attributes in advance and then feed the data into traditional machine learning models. Such missing attributes are commonly filled with zeros or means. K-nearest-neighbor-based method is also utilized to estimate the missing values for incomplete instances [9]. Probabilistic generative models such as Gaussian mixture model (GMM) [10] use expectation maximization (EM) algorithm to find the most probable completion. Multivariate imputation by chained equations (MICE) [11] is an iterative method of dealing with missing data under the assumption of missing at random (MAR). A limitation of the above imputation methods is that errors of imputation may propagate to the following machine learning processes. Another intuitive way is to delete incomplete instances in training and make some assumptions on missingness patterns [12], or tune the decision function for incomplete test data [13]. However, this limits the application of such models when all instances are incomplete.

Some methods process the missing data in a task-specific way. [14] proposed to use the EM algorithm to learn from incomplete data for a classifier. Similarly, [15] proposed a classification model that dealt with missing data by performing analytic integration with an estimated conditional density function. [16] avoided the imputation procedure by introducing instance-specific margins for large margin classifiers. [17] connected the matrix completion task with classification task in a transductive way, whereas [18] argued that completion was neither necessary nor sufficient for classification. They proposed a kernel method for incomplete data based on observed features. [19] used multiple imputations adaptively to improve the classification results. Apart from the methods mentioned above, many other works fall into this category [20], [21].

In addition to the above-mentioned methods, many neural networks can be utilized to process data with missing attributes [22]–[26]. However, they require complete instances in the learning phase. Only recently, [27] proposed a model that can be trained without complete data. They replaced the typical neuron's response in the first hidden layer by its expected value when data were incomplete. The missing data density was depicted by a Gaussian mixture model and trained together with the neural network.

The main shortcoming of previous methods is that they tend to use a universal model for all data, and thus ignore the inherent differences between data with different missingness patterns. Commonly, we use missingness patterns to indicate the locations of the missing entries. Samples may have varying subsets of observable features due to the inherent properties of the instances. Sometimes, a part of the features may not even be defined for some instances.

Yongshun Gong, Xiankai Lu and Yilong Yin are with the School of Software, Shandong University, China. (e-mail: {ysgong;luxiankai;ylyin}@sdu.edu.cn).
Zhibin Li is now with the Commonwealth Scientific and Industrial Research Organisation, Australia. (e-mail: Zhibin.Li@csiro.au).
Wei Liu and Ivor W. Tsang are with the Faculty of Engineering and Information Technology University of Technology Sydney, NSW 2007, Australia. (e-mail: {Wei.Liu;ivor.tsang}@uts.edu.au).
Xinwang Liu is with College of Computer, National University of Defense Technology, Changsha, 410073, China. (e-mail: xinwangliu@nudt.edu.cn).
Corresponding authors: Zhibin Li and Yilong Yin



(a) Features (x, y, z) are complete. (b) Feature z is missing.

Fig. 1: When all features (x, y, z) are observable, we have an optimal separating plane in (a). When only (x, y) are observable, the best separating line is the solid line in (b). The projection of optimal separating plane in (b) is the dashed line. If we train one model for both cases, we will probably end with a compromise of them and get an inferior result.

Accordingly, using the same model for these heterogeneous data limits the performance of the model, and imputation may lead to severe deviation. More importantly, the model could suffer from competition between data with different missingness patterns. We illustrate such a phenomenon in Fig. 1. For two sets of data labeled as "." and "+", when we have complete features of an instance, the best decision plane for classification is shown in Fig. 1. However, if we use the available features (x, y) to classify a point when feature z is missing, then use the coefficients of the decision plane in Fig. 1 (a) regarding (x, y) is not optimal (shown as the dashed line in Fig. 1 (b)). The best separating line, in this case, is the solid line as shown in Fig. 1 (b). These two patterns would compete against each other when training with incomplete data, leading to a suboptimal model for both cases. A straightforward way to minimize such influence is to learn different decision functions for each missingness pattern. However, for some missingness patterns, data can be insufficient for the training of the model, which causes difficulties in generalization. Motivated by the above discussions, we propose an adaptive learning model based on various missingness patterns for incomplete data. We summarize the main contributions and innovations of this paper as follows:

- To the best of our knowledge, the proposed method is the first attempt to provide an adaptive model that can apply associated decision functions to data with corresponding missingness patterns and does not require the imputation of missing data.
- We devise different models for data with various missingness patterns, while improving the generalization ability by a low-rank constraint. We also provide an efficient training approach for the non-convex optimization.
- We theoretically prove the generalization error bound and convergence property of our model, demonstrating the low-rank constraint can be helpful to reduce the error.
- Our method can be seamlessly incorporated into various neural networks with minimal modification of network architectures. We conduct extensive experiments on several real datasets with internally missing attributes, algorithm implemented in both linear and non-linear (neural networks) models show its superiority compared with other methods.

The remainder of this paper is organized as follows: Section 2 includes a literature review. Section 3 proposes our method. The theoretical analysis is given in Section 4. We also provide an efficient training procedure in Section 5. All experimental results are shown in Section 6. Finally, conclusions and future work are drawn in Section 7.

2 Related Work

In this section, we review the current studies with incomplete data. Generally, there are two categories in this field: learning after imputation and learning with incomplete data.

2.1 Learning After Imputation

A prevailing strategy is to fill the missing attributes in advance, and then the filled data can be fed into downstream tasks with traditional machine learning methods. In many real-world applications, missing attributes are commonly imputed by zeros or mean-values. An improved method is try to use K-nearest-neighbours of the incomplete instances to estimate the missing values [9].

Tensor decomposition is a widely used method to deal with the incomplete data problem [28] and has been applied into many applications [29]–[31]. For example, [32] leveraged Tucker decomposition for traffic prediction. They can achieve a comparatively accurate result even when the missing ratio of data is quite high. Liu et al. [33] proposed an model to impute missing data in tensors of visual data. There were three models proposed in the paper, they use a relaxation method to separate relationships and utilize the block coordinate descent (BCD) to find a globally optimal solution. Le et al. [34] devised a robust adaptive Canonical Polyadic decomposition method for dealing with high-order incomplete streaming tensors.

Multi-view learning usually need to face incomplete data problem [35]. Gong et al. [36], [37] developed a spatially related multi-view learning model with adaptive weight technique to address the incomplete data problem in Urban Statistical Data. Liu et al. [38] proposed an efficient and effective method LF-IMVC for the incomplete multi-view clustering problem. The proposed algorithm learns a consensus clustering matrix jointly, filling each missing values in the base matrix instead of completing kernel matrices, and optimizes the corresponding permutation matrices. Similar idea can be found in their following studies [39], [40]. The algorithm designed in [39] does not require that there be at least one complete base kernel matrix over all the samples, and different with traditional imputation process that complete the incomplete kernel matrices first. [40] developed a model named EE-IMVC focusing on imputing incomplete base matrices generated by incomplete views.

Other strategies such as Gaussian mixture model (GMM) utilizes expectation maximisation (EM) algorithm to find the most probable completion; adversarial joint-learning recurrent neural network is proposed for incomplete time

TABLE 1: Symbol description.

Symbols	Descriptions
$\mathbf{x}, y$	feature vectors and labels
$\mathbf{x^0}$	zero-filling for $\mathbf{x}$
$\mathbf{m}$	missingness pattern indicator
$\tilde{\mathbf{m}}$	augmented vector generated from $\mathbf{m}$
d ; d'	the numbers of dimensions $\mathbf{x}$ and $\mathbf{m}$
H	A dictionary for generating missingness-pattern specific functions
U, V	low-rank latent matrices decomposed from H
ξ_i	the slack variable for the margin
η_1 , η_2	regularization parameters
k	rank of U and V
$P_1, P_2, ..., P_n; l$	n different real polynomials in l real variables
$\alpha_s; T_s$	the step-size and number of iterations in stage s of the Restarted SubGradient method
e	the Euler number

series classification [41], where the adversarial network is used to encourage the network to complete missing data by distinguishing real and imputed values; Multivariate imputation by chained equations (MICE) [11] is an iterative method of dealing with missing data under the assumption of missing at random (MAR); and Kachuee et al. develop a generative approach to impute missing data and to measure class uncertainties arising from the distribution of missing values [42].

Compared to our method that can adaptively learn with incomplete data, the main disadvantage of the above imputation methods is that errors of imputation may propagate to the following machine learning models.

2.2 Learning with Incomplete Data

Methods learning with incomplete data can build a task-specific machine learning model to handle such incomplete information. An intuitive way is to delete incomplete instances in training, and make some assumptions for missingness in training [12], or tune the decision function for incomplete test data [13]. This limits the application of such models when most of instances are incomplete.

Ghahramani and Jordan [14] proposed to use EM approach to learn from incomplete data for classifier. Similarly, Williams et al. [15] proposed a classification model which dealt with missing data by performing analytic integration with an estimated conditional density function. Elhamifar et al. [43] cast the clustering of data with missing entries as clustering of complete data. Chen et al. [44] proposed a framework that can characterize both global and local consistencies in large-scale time series data. The developed graphical methods can perform probabilistic predictions and estimate uncertainty values without imputing those missing entries. Liu et al. [45] devised three algorithms to handle the situation where some channels of samples are missing. They can only classify each sample based on all observed channels, without imputation process involved.

Pelckmans et al. [46] defined a loss considering the uncertainty of predicted outputs. Under the assumption of missing completely at random, their method did not involve the imputation procedure. Chechik et al. [16] also avoided the imputation procedure by introducing an instance-specific margin for large margin classifier. Goldberg et al. [17] connected the matrix completion task with classification task in a transductive way, whereas Hazan et al. [18] argued that completion is neither necessary nor sufficient for classification. They proposed a kernel method for incomplete data based on observed features. Liu et al. [19] used multiple imputation adaptively to improve the classification results. Apart from above-mentioned methods, many other works fall into this category [20], [21]. Awawdeh et al. [47] designed a feature selection process to handle the missing values. Śmieja et al. [27] proposed a general approach for adapting neural networks to process incomplete data, which can learn from incomplete samples. They introduced input layer for processing missing data, where the typical neuron's response is replaced by its expected value when data were incomplete.

The main shortcoming of previous method is that they all use same model for different missingness patterns, and thus ignore the inherent differences carried by missingness patterns. Bullins et al. [48] analysed the limitation of such model under linear case with hinge loss. They gave a limit on the precision attainable when the learning algorithm was allowed to access only a limited number of attributes per example. A straightforward way to improve the lower error bound is to learn different decision functions for different missingness patterns, so for every decision function, the training data is relatively complete. However, for some missingness patterns, the data can be deficient for training a good model, and we may not see all possible missingness patterns in a training set. Hence, in this paper, we designed a general method for learning with incomplete data directly, where data of various missingness patterns are treated differently in the model level.

3 Missingness-Pattern Adaptive Model

We formulate our idea for binary classification, but it can also be extended to multi-class and regression tasks with associated objective functions. The main symbols used in this paper are summarized in Table 1, and Fig. 2 shows the flowchart of our method.

3.1 Linear Model

Given a data instance $(\mathbf{x}, \mathbf{m}, y)$ with feature vector $\mathbf{x} \in \mathbb{R}^d$, label $y \in \{-1, +1\}$ and $\mathbf{m} \in \mathbb{R}^{d'}$ an indicator vector represents its missingness pattern. Without any prior knowledge, $\mathbf{m}$ will be a d-dimensional binary vector. Each bit of $\mathbf{m}$ indicates the missingness of the corresponding bit in $\mathbf{x}$. The bit in $\mathbf{m}$ is set to 1 if the corresponding feature in $\mathbf{x}$ is observed; otherwise, the bit is set to 0. In some settings such as incomplete multi-view learning, features are missing group-wise, so $\mathbf{m}$ can serve as a group-wise indicator thus making d' much smaller than d.

In order to treat missingness patterns adaptively, the linear decision function can be formulated as:

$$f(\mathbf{x}) = g(\mathbf{m})\mathbf{x}^o, \tag{1}$$

where $\mathbf{x}^o \in \mathbb{R}^d$ denotes the $\mathbf{x}$ after zero-filling for the missing values. In this way, it is possible to apply different

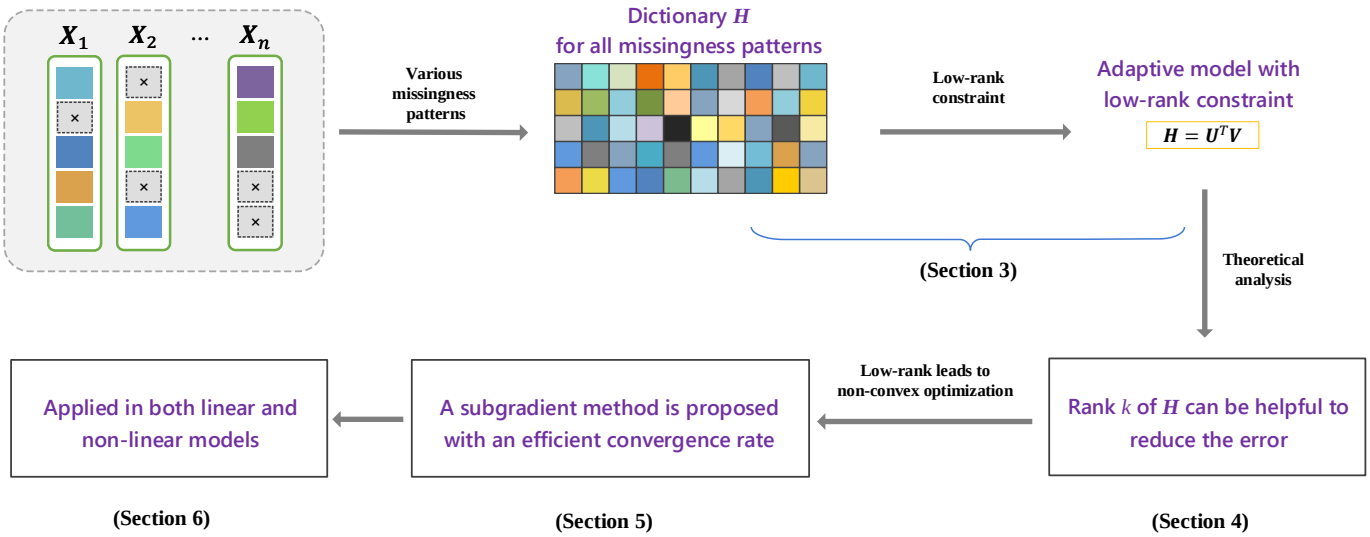


Fig. 2: The flowchart of our proposed method. In the learning process, given a set of samples with different missingness patterns, we provide a dictionary H for generating missingness pattern-specific functions. We then restrict H with a low-rank constraint that introduces correlations between models for different missingness patterns. After a rigorous generalization error bound analysis, we apply our method into both linear and non-linear models with efficient training process.

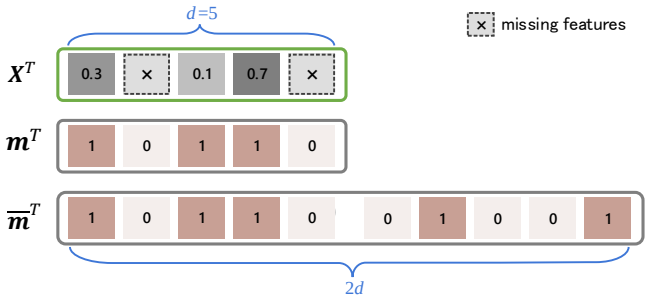


Fig. 3: examples for $\mathbf{x}$, $\mathbf{m}$ and $\bar{\mathbf{m}}$.

weight coefficients generated by $g(\mathbf{m})$ for data of different missingness patterns. g can be selected from a wide range of function classes. In this paper, we adopt a simple yet efficient form of $g(\mathbf{m})$ given by:

$$g(\mathbf{m}) = (H\bar{\mathbf{m}})^\top, \tag{2}$$

where $H \in \mathbb{R}^{d\times 2d'}$ serves as a dictionary for generating missingness-pattern-specific functions. $\bar{\mathbf{m}} = [\mathbf{m}^\top, (\mathbf{1} - \mathbf{m})^\top]^\top$ is an augmented vector generated by concatenating $\mathbf{m}$ and its element-wise logic NOT operation. The example of sample vector $\mathbf{x}$, corresponding $\mathbf{m}$ and $\bar{\mathbf{m}}$ is illustrated in Fig. 3. In doing this, for every distinct missingness pattern $\mathbf{m}$, we have a corresponding weight vector generated by $H\bar{\mathbf{m}}$. Notice that we use $\bar{\mathbf{m}}$ instead of $\mathbf{m}$ to ensure that for every missingness pattern we select a fixed number of elements from H. Bias terms could also be incorporated into Eq.(1) by appending a constant feature to $\mathbf{x}^o$ and extend $\mathbf{m}$ and H accordingly. Thus the bias terms can also be adaptively fitted to missingness patterns. For notational simplicity, we omit them in our formulas.

In the spirit of large margin classifier, we can define a modified learning objective which is specialized for incomplete data with the margins varying over different missingness patterns. Given a set of n labeled observations $\{(\mathbf{x}_i, \mathbf{m}_i, y_i)\}_{i=1}^n$, the learning objective is:

$$\begin{aligned} \min_{H} \quad & \frac{1}{n}\|M \odot (H\bar{M})\|_F^2 + \eta_1 \|H\|_F^2 + \frac{\eta_2}{n}\sum_{i=1}^{n} \ell(y_i, \bar{\mathbf{m}}_i^\top H^\top \mathbf{x}_i^o), \\ \text{s.t.} \quad & \text{rank}(H) \le k, \\ & M = [\mathbf{m}_1, \cdots, \mathbf{m}_n], \bar{M} = [\bar{\mathbf{m}}_1, \cdots, \bar{\mathbf{m}}_n], \end{aligned} \tag{3}$$

where $\ell(y, \hat{y}) \triangleq \max(0, 1 - y\hat{y})$ denotes the hinge loss. $\|\cdot\|_F$ and $\odot$ denotes the Frobenius norm and the Hadamard product respectively; ξ_i is the slack variable similar to that in Support Vector Machines; η_1, η_2 and k are hyper-parameters; $H \in \mathbb{R}^{d\times 2d'}$, $M \in \mathbb{R}^{d\times n}$, $\bar{M} \in \mathbb{R}^{2d'\times n}$.

Because each instance has its own observable part, we should optimize the margin regarding observable part only. Unlike in the complete data setting, where the margin optimization is associated with a regularization on the weight vector, we need vary the regularisation in incomplete data setting because the weight vectors vary over samples with different missingness patterns. This leads to the first term in Eq. (3), which is the approximate denominator for instance-based margins. We borrow this idea from [16] to ensure a fair optimization of margins. This is achieved through a mask matrix M to set the weights to zeros in $H\bar{M}$ corresponding to missing features. We also introduce η_1 to constraint the Frobenius norm of H and fix it to be a small constant.

Eq.(3) allows us to define a decision function for every missingness pattern while connecting them through a low-rank matrix H. The low-rank constraint introduces correlations between models for different missingness patterns, so that facilitates the learning of models related to some rare missingness patterns. In detail, we decompose H by

$U^\top V$ to restrict the rank of $H \leq k$, where $U \in \mathbb{R}^{k\times d}$ and $V \in \mathbb{R}^{k\times 2d'}$, then Eq.(3) can be converted as:

$$\begin{aligned}\min_{U,V} \quad & \frac{1}{n}\|M \odot (U^\top V \bar{M})\|_F^2 + \eta_1 \|U^\top V\|_F^2 \\ & + \frac{\eta_2}{n}\sum_{i=1}^{n} \ell(y_i, \bar{\mathbf{m}}_i^\top V^\top U \mathbf{x}_i^o), \\ \text{s.t.} \quad & M = [\mathbf{m}_1, \cdots, \mathbf{m}_n], \bar{M} = [\bar{\mathbf{m}}_1, \cdots, \bar{\mathbf{m}}_n].\end{aligned} \tag{4}$$

This learning objective is non-convex. The non-convexity naturally arise from the rank constraint in H. One may consider add more constraints on U or V (e.g., $UU^\top$ = I) to make the learning problem convex globally, but that will inevitably add the computation complexity and is indeed unnecessary, i.e., for the non-convex low-rank matrix problems, all local minima are also globally optimal [49]. Such constraints will not benefit to the performance of the proposed model. Nevertheless, we will show the learning objectives regarding U or V are convex respectively. We also present an efficient training algorithm in Section 5.

3.2 Generalize to Non-linear Model

Our idea can also be readily applied to many existing neural networks with some minimal modifications. Assume the output of a neural network with complete data can be expressed as:

$$\hat{y} = f(\mathbf{x}; \theta) \tag{5}$$

where θ denote parameters of the network with any non-linear activation functions. We can adjust the weight of observed features by missingness pattern, which gives the output:

$$\hat{y} = f((H\bar{\mathbf{m}}) \odot \mathbf{x}^o; \theta). \tag{6}$$

Considering the low-rank constraint, we decompose H by $U^\top V$. Then, the learning objective can be formulated as follows:

$$\min_{U,V,\theta} \sum_{i=1}^{n} \mathcal{L}\big(y_i, f((U^\top V \bar{\mathbf{m}}_i) \odot \mathbf{x}_i^o; \theta)\big), \tag{7}$$

where $\mathcal{L}$ is the loss function. Note that if we let $f(\cdot;\theta)$ in the non-linear case $\hat{y} = f((H\bar{\mathbf{m}}) \odot \mathbf{x}^o; \theta)$ be a element-wise sum function, it is then reduced to the linear case. That is, $\hat{y} = \mathbf{1}_d^\top \big((H\bar{\mathbf{m}}) \odot \mathbf{x}^o\big)$.

We incorporate the rank constrain by decomposing H into product of $U^\top$ and V with $U \in \mathbb{R}^{k\times d}$ and $V \in \mathbb{R}^{k\times 2d'}$. U and V would be learned together with the network's parameters θ in an end-to-end manner. The motivation behind the formula is clear and effective: we can adjust the importance of observed features when some other features are missing. Additionally, our non-linear model only introduces two learnable matrices $U \in \mathbb{R}^{k\times d}$ and $V \in \mathbb{R}^{k\times 2d}$. The number of extra parameters occupied a small proportion of the entire neural network.

4 Generalization Error Bound Analysis

In this section, we theoretically analyze the generalization error of our linear model. We give a rather general bound on the generalization error based on the growth function. This bound also supports the low-rank constraint in our model.

We firstly introduce some common settings in this section. A labeled training set is given by $D = \{(\mathbf{x}_i, \mathbf{m}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{X}$; $\mathcal{X}$ is a subset of $\mathbb{R}^d$, $y_i \in \{-1, +1\}$ and $\mathbf{m}_i \in \{0,1\}^{d'}$ represents the missingness indicator vector. We assume that training data are drawn independently and identically distributed (i.i.d.) according to some unknown distribution $\mathcal{D}$ and denote $D \sim \mathcal{D}$. The derived bound will be quite general since we do not assume the underlying missingness mechanism a prior. Let the hypothesis set $\mathcal{F}$ be a family of functions mapping $\mathcal{X}$ to $\{-1, +1\}$ defined by:

$$\mathcal{F} = \{\mathbf{x} \mapsto \mathrm{sign}((\mathrm{H}\bar{\mathbf{m}})^\top \mathbf{x}^{\mathrm{o}}) : \mathrm{rank}(\mathrm{H}) \leq \mathrm{k}\}, \tag{8}$$

The empirical error of a hypothesis $f \in \mathcal{F}$ over the training set D is defined as:

$$\widehat{R}_D(f) = \frac{1}{n}\sum_{i=1}^{n} 1_{f(\mathbf{x}_i)\neq y_i}, \tag{9}$$

where $1_{f(\mathbf{x}_i)\neq y_i} = 1$ if $f(\mathbf{x}_i) \neq y_i$ and 0 otherwise. The generalization error of f is defined by:

$$R_{\mathcal{D}}(f) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}}\left[1_{f(\mathbf{x})\neq y}\right]. \tag{10}$$

We start with a bound on the generalization error $R_{\mathcal{D}}(f)$ given by [50, Corollary 3.9]. For any $\delta > 0$, with probability at least $1-\delta$, for any $f \in \mathcal{F}$, we have:

$$R_{\mathcal{D}}(f) \leq \widehat{R}_D(f) + \sqrt{\frac{2\log \Pi_{\mathcal{F}}(n)}{n}} + \sqrt{\frac{\log \frac{1}{\delta}}{2n}}, \tag{11}$$

where $\Pi_{\mathcal{F}}(n)$ is the growth function for the hypothesis set $\mathcal{F}$ with n samples. The growth function $\Pi_{\mathcal{F}}(n)$ is the maximum number of distinct sign-patterns on n samples that can be produced with functions in $\mathcal{F}$. As a result, the generalisation error bound mainly relies on the growth function $\Pi_{\mathcal{F}}(n)$. Next, we will give the bound for $\Pi_{\mathcal{F}}(n)$and formal definition on $\Pi_{\mathcal{F}}(n)$.

We restate the following Lemma [51, Lemma 17] for bounding the growth function:

Lemma 4.1. Let $P_1, P_2, ..., P_n$ be n real polynomials in l real variables, and suppose the degree of each P_i does not exceed t. If $n \geq l$ then $s(P_1, P_2, ..., P_n) \leq 2(2e \cdot n \cdot t/l)^l$ with $s(P_1, P_2, ..., P_n)$ denotes the total number of sign-patterns of the polynomials $P_1, P_2, ..., P_n$; and e is the Euler number.

Lemma 4.1 provides a bound for sign patterns of polynomials. This bound assumes $P_i \neq 0$. This coincides with most of the practical cases. If we would like to consider a more complete setting that allows $P_i = 0$, we can set $\mathrm{sign}(0) = 1$ and follow the results in [52, Proposition 5.5] to obtain $s(P_1, P_2, ..., P_n) \leq (8e \cdot n \cdot t/l)^l$.

We then give the definition of the growth function $\Pi_{\mathcal{F}}(n)$ and its bound altogether in following theorem. Proof of this theorem will be based on Lemma 4.1.

Theorem 4.2. The growth function $\Pi_{\mathcal{F}}(n)$ of hypothesis set $\mathcal{F}$ on n samples is defined and bounded by:

$$\Pi_{\mathcal{F}}(n) = \max_{\{\mathbf{x}_1,...,\mathbf{x}_n\}\subseteq\mathcal{X}} \left| \left\{ \left(sign(f(\mathbf{x}_1)),...,sign(f(\mathbf{x}_1))\right) : f \in \mathcal{F} \right\} \right| \leq 2\left(\frac{2e\cdot n\cdot t}{l}\right)^l, \tag{12}$$

where $t = 2$ and $l = k(d + 2d')$.

Proof: Consider $f(\mathbf{x}_1), ..., f(\mathbf{x}_n)$ to be n real polynomials. Because $rank(H) \leq k$, H can be decomposed into product of $U^\top$ and V with $U \in \mathbb{R}^{k\times d}$ and $V \in \mathbb{R}^{k\times 2d'}$. Treat elements of U and V as variables, so that the degree of each polynomial $f(\mathbf{x}_i)$ is 2 and we have $k(d + 2d')$ variables. Apply Lemma 4.1 and we complete the proof. □

Corollary 4.2.1. For any $f \in \mathcal{F}$ and $\delta > 0$, following generalization error bound holds with probability at least $1 - \delta$:

$$R_{\mathcal{D}}(f) \leq \widehat{R}_D(f) + \sqrt{\frac{2k(d+2d')\log\frac{4e\cdot n}{k(d+2d')} + \log 4}{n}} + \sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \tag{13}$$

The rank k of H, the feature dimension d, the dimension d' of missingness indicator vector $\mathbf{m}$ and the sample size n jointly represent the upper bound of generalization error in above corollary. Clearly this bound decreases when sample size n increases. A low-dimensional feature vector $\mathbf{x}$ and a low-dimensional missingness pattern indicator vector $\mathbf{m}$ are both beneficial to the model generalization. It also shows that appropriately constrain the rank k of H can be helpful to reduce the error.

5 Efficient Training Procedure

The optimisation of Eq.(7) is based on stochastic gradient descent with PyTorch [53] implementation. We discuss the learning problem with regard to Eq.(4) in this section. It is non-convex due to the rank constraint. Notice that H can be decomposed as $H = U^\top V$ with $U \in \mathbb{R}^{k\times d}$ and $V \in \mathbb{R}^{k\times 2d'}$. Then the loss function associated with Eq.(4) is convex regarding U with fixed V and vice versa. We can optimize them alternatively until convergence. A straightforward way to minimize the loss function is through the subgradient method. We fix some subgradient oracles for U and V as:

$$g_U = \frac{2}{n}V\bar{M}\left(M^\top \odot \left(\bar{M}^\top V^\top U\right)\right) + 2\eta_1 VV^\top U - \frac{\eta_2}{n}\sum_{i\in\mathcal{I}_{sv}} y_i V\bar{\mathbf{m}}_i\mathbf{x}_i^{o\top}, \tag{14}$$

$$g_V = \frac{2}{n}U\left(M \odot \left(U^\top V\bar{M}\right)\right)\bar{M}^\top + 2\eta_1 UU^\top V - \frac{\eta_2}{n}\sum_{i\in\mathcal{I}_{sv}} y_i U\mathbf{x}_i^o\bar{\mathbf{m}}_i^\top, \tag{15}$$

Algorithm 1: Subroutine for optimizing U

Input: U_1^1, V, the number of stages S.
Output: U_{S+1}^1.
Initialization: $\epsilon_0 = F(U_1^1)$. Calculate C, γ, L_Φ, L_h, α_1, T_1.
for $s = 1$ *to* S **do**
  $\alpha_s = (\frac{1}{2})^{s-1}\alpha_1$; $T_s = 2^{s-1}T_1$;
  for $t = 1$ *to* T_s **do**
    Calculate U_s^{t+1} by Eq.(18);
  $U_{s+1}^1 = \arg\min_{U\in U_s^1,...,U_s^{T_s+1}} F(U)$;

where $\mathcal{I}_{sv}$ denotes indices of support vectors, i.e. samples with positive slack variables. Given the subgradients, we can optimize U with fixed V and optimize V with fixed U iteratively until convergence. The key factor that influences the overall convergence is the convergence rate of subroutines to optimize U and V, so we now discuss the convergence rate of optimization regarding U given V. For V, a similar result holds, and we omit the details here. Algorithm 1 presents the procedure for optimizing U.

Our loss function is non-Lipschitz and can not be guaranteed to be strongly-convex regarding U, as can be verified from its gradient given above. These are often required for deriving a convergence rate for subgradient methods. Inspired by [54] and [55], together with the Restarted SubGradient (RSG) method [56], we can give a ϵ approximate solution in $O(\frac{1}{\epsilon})$ iterations with our optimization strategy regarding U.

Our loss function has the form of $F(U) = \Phi(U) + h(U)$ with:

$$\Phi(U) = \frac{1}{n}\|M \odot (U^\top V\bar{M})\|_F^2 + \eta_1\|U^\top V\|_F^2, \tag{16}$$

$$h(U) = \frac{\eta_2}{n}\sum_{i=1}^{n}\ell(y_i, \bar{\mathbf{m}}_i^\top V^\top U\mathbf{x}_i^o), \tag{17}$$

where the hinge loss $\ell(y, \hat{y}) \triangleq \max(0, 1-y\hat{y})$. One can easily verify that $\Phi(U)$ has L_Φ-Lipschitz gradient and $h(U)$ is an L_h-Lipschitz function. Let α_s and T_s be the step-size and number of iterations in stage s. In each stage, we adopt the following update rule:

$$U_s^{t+1} = U_s^t - \alpha_s\frac{g_{U_s^t}}{\|g_{U_s^t}\|_F}, t = 1, \cdots, T_s, \tag{18}$$

and we choose $U_{s+1}^1 = \arg\min_{U\in U_s^1,...,U_s^{T_s+1}} F(U)$ for next stage. Let $\epsilon_0 = F(U_1^1)$ and F^* be the minima of $F(U)$. We have following theorem.

Theorem 5.1. With $\gamma = \max(\sqrt{8L_\Phi}, 8L_h)$, $C = \frac{1}{\eta_1\sigma_{min}^2(V)^+}$ where $\sigma_{min}(V)^+$ is the smallest non-zero singular value of V. In order to get U that satisfies $F(U) - F^* \leq \epsilon$, Algorithm 1 requires $S = \lceil log_2(\frac{\epsilon_0}{\epsilon})\rceil$ stages and $O\left(\frac{\sqrt{\eta_2}C\gamma}{\epsilon}\max(\sqrt{\frac{8}{9}L_\Phi\eta_2}, 8L_h)\right)$ iteration complexity where $\lceil a\rceil$ denotes the smallest integer not less than a.

For $s = 1, 2, \cdots, S$, the step-size α_s and number of iterations T_s are given by:

$$\begin{aligned} \alpha_s &= \frac{1}{2^{s-1}} \cdot \frac{\epsilon_0}{\gamma\sqrt{\eta_2}}, \\ T_s &= 2^{s-1} \cdot \lceil \frac{1}{\epsilon_0} \max(\sqrt{\frac{8L_\Phi}{9}}\eta_2 C\gamma, 8L_h\sqrt{\eta}_2 C\gamma) \rceil. \end{aligned} \tag{19}$$

Notice that Eq.(19) is applied when $\eta_2 \geq 1$. For $\eta_2 \leq 1$, we can simply set η_2 in Eq.(19) to 1 and share same convergence rate. Theorem 5.1 shows that our optimization strategy has sublinear convergence rate. Calculating the subgradient requires linear time regarding n, d, d' and square time regarding the rank k.

We give following lemma to prove theorem 5.1.

Lemma 5.2. Denote by $\mathcal{U}^*$ the optimal set contains all minimizers of F. Let U^* denote the element in $\mathcal{U}^*$ which is closest to U. The following holds:

$$\|U - U^*\|_F^2 \leq C(F(U) - F^*), \tag{20}$$

where a constant $C = \frac{1}{\eta_1 \sigma_{min}^2(V)^+}$ and F^* is the minimal value of F.

Proof: Our loss function regarding H has the form of:

$$\begin{aligned} K(H) =& \frac{1}{n}\|M \odot (H\bar{M})\|_F^2 + \eta_1\|H\|_F^2 \\ &+ \frac{\eta_2}{n}\sum_{i=1}^{n} \ell(y_i, \bar{\mathbf{m}}_i^\top H^\top \mathbf{x}_i^o). \end{aligned} \tag{21}$$

Clearly $K(H)$ is a ρ-strongly convex function with $\rho \geq 2\eta_1$. Following proof of [57, Theorem 8], the set of optimal solutions regarding minimizing $F(U)$ is:

$$\mathcal{U}^* = \{U : U^\top V = H^*\}, \tag{22}$$

where H^* denotes the unique minimizer of $K(H)$. Given V and U, by definition of U^* we have:

$$U^* = \min_{U' \in \mathcal{U}^*} \|U' - U\|_F^2. \tag{23}$$

From KKT conditions of Eq.(23) we know:

$$\mathbf{u}_i^* - \mathbf{u}_i + V\boldsymbol{\beta}_i = \mathbf{0}, \tag{24}$$

where $\mathbf{u}_i^*$, $\mathbf{u}_i$ and $\boldsymbol{\beta}_i$ denote i-th column vectors of U^*, U and related Lagrange multipliers respectively. This implies $\mathbf{u}_i^* - \mathbf{u}_i \in \mathrm{Im}(V)$. From Courant-Fischer theorem we know:

$$\|V^\top \mathbf{u}_i - V^\top \mathbf{u}_i^*\|_2 \geq \sigma_{min}(V)^+ \|\mathbf{u}_i - \mathbf{u}_i^*\|_2. \tag{25}$$

Apply Eq.(25) to every column of U we get:

$$\|V^\top U - V^\top U^*\|_F^2 \geq \sigma_{min}^2(V)^+ \|U - U^*\|_F^2. \tag{26}$$

By definition of strongly-convex function:

$$K(H_1) \geq K(H_2) + \langle k(H_2), H_1 - H_2\rangle + \frac{\rho}{2}\|H_1 - H_2\|_F^2, \tag{27}$$

where $k(H_2) \in \partial K(H_2)$ is any subgradient of K at H_2. Let $H_1 = U^\top V$ and $H_2 = (U^*)^\top V$, and notice that $K(U^\top V) = F(U)$. We have:

$$\begin{aligned} F(U) \geq F^* &+ \left\langle Vk((U^*)^\top V)^\top, U - U^* \right\rangle \\ &+ \frac{\rho\sigma_{min}^2(V)^+}{2}\|U - U^*\|_F^2, \end{aligned} \tag{28}$$

because $V\partial K((U^*)^\top V)^\top = \partial F(U^*)$. According to optimality conditions of subgradient method, we can choose $Vk((U^*)^\top V)^\top = \mathbf{0} \in \partial F(U^*)$. Thus,

$$\frac{\rho\sigma_{min}^2(V)^+}{2}\|U - U^*\|_F^2 \leq F(U) - F^*. \tag{29}$$

Because $\rho \geq 2\eta_1$,

$$\|U - U^*\|_F^2 \leq \frac{1}{\eta_1\sigma_{min}^2(V)^+}(F(U) - F^*), \tag{30}$$

which completes the proof. □

We adopt following update rule in stage s:

$$U_s^{t+1} = U_s^t - \alpha_s \frac{g_{U_s^t}}{\|g_{U_s^t}\|_F}, t = 1, \cdots, T_s. \tag{31}$$

Notice that our loss function has the form of:

$$F(U) = \Phi(U) + h(U), \tag{32}$$

where $\Phi(U)$ has L_Φ-Lipschitz gradient and $h(U)$ is an L_h-Lipschitz function. Then another useful Lemma is:

Lemma 5.3. With the update rule of Eq.(31), we have

$$\begin{aligned} &\min_{t=1\ldots T_s}\{F(U_s^t) - F^*\} \\ &\leq \frac{L_\Phi}{2}\left(\frac{\|U_s^1 - U^*\|_F^2}{2T_s\alpha_s} + \frac{\alpha_s}{2}\right)^2 \\ &+ 2L_h\left(\frac{\|U_s^1 - U^*\|_F^2}{2T_s\alpha_s} + \frac{\alpha_s}{2}\right) \end{aligned} \tag{33}$$

Lemma 5.3 is proved in [55] by firstly applying [55, Lemma 2.3] to our loss function $F(U)$ and then applying [55, Theorem 1.2].

We can use Lemma 5.2 and Lemma 5.3 to complete the proof of Theorem 5.1. Combine Lemma 5.2 and Lemma 5.3, we get:

$$\begin{aligned} &\min_{t=1\ldots T_s}\{F(U_s^t) - F^*\} \\ &\leq \frac{L_\Phi}{2}\left(\frac{CF(U_s^1)}{2T_s\alpha_s} + \frac{\alpha_s}{2}\right)^2 \\ &+ 2L_h\left(\frac{CF(U_s^1)}{2T_s\alpha_s} + \frac{\alpha_s}{2}\right) \end{aligned} \tag{34}$$

We assume that $F(U_s^1) \leq \eta_2$. This could be easily guaranteed by knowing that $F(0) \leq \eta_2$, and set $U = 0$ at initialization. When $\eta_2 \geq 1$, set the step size α_s and number of iteration T_s as:

$$\alpha_s = \frac{F(U_s^1)}{\gamma\sqrt{\eta_2}} \tag{35}$$

$$T_s = \lceil \frac{1}{F(U_s^1)} \max(\sqrt{\frac{8L_\Phi}{9}}\eta_2 C\gamma, 8L_h\sqrt{\eta}_2 C\gamma) \rceil. \tag{36}$$

We can obtain:

$$\min_{t=1,\ldots,T_s+1}\{F(U_s^t) - F^*\} \leq \frac{F(U_s^1)}{2}. \tag{37}$$

TABLE 2: Summary of datasets

Dataset	Instances	Features	% of Internally Missing
bands	539	19	5.38%
hepatitis	155	19	5.67%
horse	368	22	23.80%
mammographic	961	5	3.37%
pima	768	8	12.24%
MIC	1700	111	35.8%
Drive Diagnosis	58509	49	0%
MNIST	70000	784	0%
Avila	20867	10	0%

We choose the best U in stage s as the initial values in stage $s+1$ following:

$$U_{s+1}^1 = \arg \min_{U \in U_s^1, \ldots, U_s^{T_s+1}} F(U), \tag{38}$$

therefore we can get:

$$\|U_{s+1}^1 - U^*\|_F^2 \leq C(F(U_{s+1}^1) - F^*) \leq \frac{CF(U_s^1)}{2}. \tag{39}$$

Applying the inequality recursively, we obtain $\alpha_{s+1} = \frac{\alpha_s}{2}$, $T_{s+1} = 2T_s$, and

$$\min_{t=1,\ldots,T_S+1} \{F(U_S^t) - F^*\} \leq \frac{F(U_1^1)}{2^S}. \tag{40}$$

In order to get U that satisfies $F(U) - F^* \leq \epsilon$, Algorithm 1 in our paper requires $S = \lceil log_2(\frac{\epsilon_0}{\epsilon}) \rceil$ stages with $\epsilon_0 = F(U_1^1)$. Summing up the iterations for all stages and noticing that it is a geometric series, gives the iteration complexity $O(\frac{\sqrt{\eta_2}C\gamma}{\epsilon}\max(\sqrt{\frac{8}{9}L_\Phi\eta_2}, 8L_h))$.

When $\eta_2 < 1$, the iteration complexity becomes $O(\frac{C\gamma}{\epsilon}\max(\sqrt{\frac{8}{9}L_\Phi}, 8L_h))$ to satisfy $F(U) - F^* \leq \epsilon$. This can be verified by setting $\eta_2 = 1$ in Eq.(35), Eq.(36) and applying them recursively with Eq.(34). Thus we complete the proof of Theorem 5.1.

Computational Complexity. We discuss the time complexity of our proposed model here. The time complexity is mainly affected by our subgradient training methods, as shown in Eq.(14)-(15). It is apparent that the time complexity is governed by matrix multiplication operations and decision function complexity in each iteration. Calculating the subgradient incurs $O(nk(d+2d') + (d+2d')k^2)$ computational complexity, which is quadratic regarding the rank k and linear regarding n, d, d', so it can be easily calculate even for large number of samples and feature dimension. Notice that Eq.(14)-(15) require indices of the support vectors, which can be obtain through the decision function with $\mathcal{O}(nk(d+2d'))$ complexity.

6 Experiments

In this section, we present experiments on some real datasets with internally missing attributes as well as artificially missing entries. Table 2 summarizes the datasets.

6.1 Linear model

We apply our method learned through Eq.(4) on six real datasets retrieved from UCI repository [58] with **internally missing attributes**, those are the top six datasets in Table 2 (MIC indicates the Myocardial Infarction Complications dataset and we choose to predict Chronic Heart Failure). We randomly split those datasets into 70% for training and 30% for testing. First, we conduct experiments on the original dataset. Second, to consider a more general case, we randomly removed 30% of the values in the training sets and test sets. In this case, the missing rate would be higher than 30% for all datasets and the missingness mechanisms are more complex than the original datasets.

We considered methods with publicly available codes. We compared our method with the following baselines:

- **Flag**: This method added additional binary features to indicate whether a feature was missing for a given instance. The missing values were set to zero.
- **Zero**: This method sets missing values to zero.
- **Mean**: This method sets missing features to averages of corresponding features from other instances that were not missing.
- **KNN**: Missing features of an instance were filled with means of those features calculated from the K-nearest neighbors of this instance. The neighborhood was measured using Euclidean distance with observed features. The K was chosen from {3, 4, 5}.
- **GMM**: Missing values in the training set were filled in an iterative way between two steps: (1) learning a GMM with the filled data and (2) re-filling missing values using components' means, weighted by the posterior probabilities of related components generated the sample. For the test set, we used the learned GMM to iteratively fill the missing values until convergence according to step (2). We chose the number of the mixture components from {3, 4, 5}. This idea is similar to that in [10], [14], [16].
- **MICE**: MICE iteratively imputed one missing feature by regression based on other features [59]. We chose the linear regression to fit the models.
- **geom**: This method was proposed by [16]. It considers sample-specific margins. We used the iterative algorithm as suggested there with 5 iterations. The parameter C were selected from $\{10^{-5},...,10^5\}$.
- **karma**: This algorithm was presented in [18]. It trained a classifier under the low-rank assumption of data. The parameters γ and C were selected from $\{1, 2, 3, 4\}$ and $\{10^{-5},...,10^5\}$.
- **WEDGE**: WEDGE was designed to impute gene expression matrices via a biased low-rank matrix decomposition approach [60], we modified it to adapt to our datasets with incomplete values.
- **SMV-NMF**: A state-of-the-art missing data imputation method based on the spatio-temporal correlation learning [37]. It trained a classifier under the low-rank assumption of data. The implemented parameters are selected by grid search.

We combined the Flag, Zero, Mean, KNN, GMM and MICE with Support Vector Machines (SVM) and chose the

TABLE 3: Classification accuracy (mean±std) with original datasets. The best results are bold and the second best are underlined.

Method	Dataset					
	bands	hepatitis	horse	mammographic	pima	MIC
Flag	0.617±0.000	**0.872±0.000**	0.864±0.000	0.778±0.000	0.783±0.000	0.771±0.003
Zero	0.606±0.002	0.851±0.000	0.838±0.000	0.796±0.000	0.801±0.000	0.763±0.005
Mean	0.611±0.002	**0.872±0.000**	0.847±0.000	0.796±0.000	0.792±0.000	0.765±0.004
MICE	0.617±0.000	0.809±0.000	0.856±0.000	0.785±0.000	0.775±0.000	0.763±0.001
GMM	0.594±0.013	**0.872±0.021**	0.841±0.011	0.779±0.010	0.787±0.023	0.752±0.003
KNN	0.593±0.010	0.847±0.009	0.847±0.000	0.775±0.002	**0.805±0.000**	0.773±0.003
geom	0.605±0.000	**0.872±0.000**	0.865±0.000	0.789±0.000	0.792±0.000	0.758±0.005
karma	0.611±0.040	0.809±0.000	0.838±0.000	0.798±0.000	0.797±0.005	0.767±0.002
WEDGE	0.609±0.015	0.812±0.004	0.841±0.010	0.790±0.002	0.783±0.002	0.762±0.001
SNM-NMF	0.624±0.003	0.850±0.000	0.865±0.000	0.778±0.000	0.797±0.008	0.765±0.002
Ours	**0.678±0.005**	**0.872±0.000**	**0.876±0.007**	**0.799±0.001**	0.791± 0.002	**0.778± 0.000**

TABLE 4: Classification accuracy (mean±std) on datasets with additional missing values. The best results are bold and the second best are underlined.

Method	Dataset					
	bands	hepatitis	horse	mammographic	pima	MIC
Flag	0.583±0.006	0.845±0.016	0.825±0.007	0.764±0.006	0.737±0.022	0.773±0.004
Zero	0.597±0.022	0.842±0.017	0.816±0.015	0.761±0.018	0.736±0.010	0.759±0.003
Mean	0.586±0.008	0.843±0.017	0.822±0.013	0.774±0.009	0.740±0.002	0.761±0.003
MICE	0.575±0.027	0.774±0.044	0.712±0.041	0.772±0.016	0.738±0.023	0.764±0.002
GMM	0.572±0.021	0.825±0.037	0.805±0.013	0.768±0.012	0.742±0.021	0.764±0.005
KNN	0.592±0.016	0.812± 0.037	0.836±0.026	0.762±0.006	0.747±0.009	0.771±0.005
geom	0.575±0.023	0.834±0.025	0.819±0.022	0.762±0.009	0.742±0.006	0.764±0.002
karma	0.551±0.040	0.817±0.032	0.759±0.022	0.759±0.014	0.740±0.009	0.751±0.008
WEDGE	0.566±0.017	0.844±0.021	0.819±0.031	0.763±0.012	0.738±0.017	0.770±0.003
SVM-NMF	0.603±0.010	0.838±0.024	0.827±0.014	0.774±0.010	0.743±0.008	0.770±0.006
Ours	**0.648±0.021**	**0.868±0.009**	**0.840±0.011**	**0.776±0.010**	**0.756±0.006**	**0.780±0.004**

parameter C for SVM from $\{10^{-5},...,10^{5}\}$. Data were normalized to zero mean and unit covariance after imputation for imputation-based methods and normalized based on observed features for geom, karma and our method. We fixed $\eta_1 = 10^{-6}$ for our method. η_2 and k were chosen from $\{10^{-5},...,10^{5}\}$ and $\{2, 4, ..., d\}$ where d is the feature dimension of related dataset. All the hyper-parameters are selected based on 5-fold cross-validation on training sets.

We present experiments with the original datasets here in Table 3. The original datasets contains internally absent attributes. Our model consistently outperforms other baselines except on pima dataset. The performance gaps between all models are relatively small due to the low missing percentages.

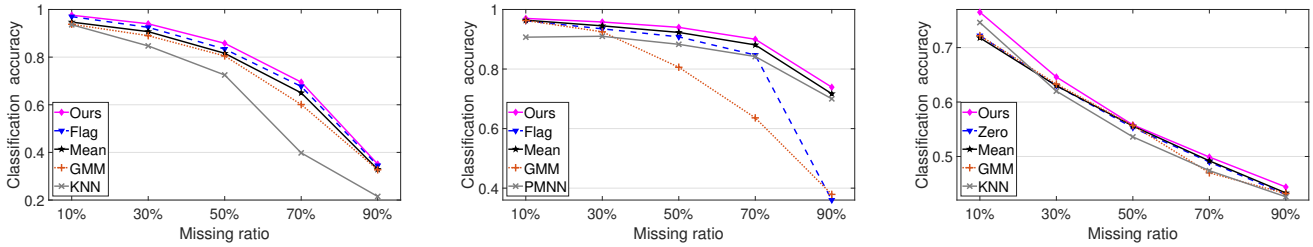
Experiment results in the general case (with additionally 30% data removed) are presented in Table 4. We repeated the experiments 5 times to report the classification accuracy. Our method achieved the best accuracy on all 5 datasets. In general, our method is better than imputation methods, because inaccurate imputation could deteriorate the downstream classification task. Our method also outperforms Flag, which indicates that simply adding the missingness pattern as additional features is not as good as our strategy. These datasets contain inherent missing features, and we also removed some values randomly. These factors make the missingness mechanism complicated and it is hard to learn a universal model that fits all these heterogeneous missingness patterns. Our method tries to adaptively apply the classifiers specialized to different missingness patterns, which makes it capable of learning some non-liner classifiers. This makes our model outperforms other baselines.

6.2 Non-Linear model

We compare our method with other baselines based on linear models (neural networks). The experiments were conducted on three real datasets.

Sensorless Drive Diagnosis dataset is retrieved from UCI repository [58]. The features are extracted from electric current drive signals. It consists of 11 classes indicating 11 different running conditions of the drive. There are 58509 instances and each instance has 49 features. The datasets were randomly split into 50% training set and 50% test set. We randomly selected 25% of the training data as the validation set.

MNIST [61] is a dataset for classification of handwritten digits. The dataset contains 784 features and has a training set of 60000 examples, and a test set of 10000 examples. We randomly selected 20% of the data in training set as the validation set.



(a) Results on dataset Sensorless Drive Diagnosis .

(b) Results on dataset MNIST.

(c) Results on dataset Avila.

Fig. 4: The average accuracy.

TABLE 5: Classification accuracy (mean±std) on Sensorless Drive Diagnosis dataset. The best results are bold and the second best are underlined.

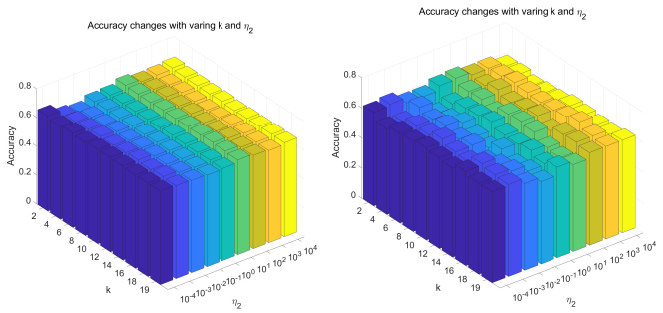
	Percentage of missing				
Method	10%	30%	50%	70%	90%
Zero	0.908±0.001	0.852±0.011	0.769±0.005	0.618±0.005	0.317±0.002
Mean	0.947±0.004	0.907±0.001	0.816±0.003	0.650±0.005	0.329±0.003
MICE	0.717±0.001	0.422±0.009	0.483±0.010	0.322±0.007	0.197±0.007
GMM	0.938±0.002	0.890±0.005	0.805±0.007	0.601±0.007	0.327±0.003
KNN	0.936±0.004	0.847±0.003	0.725±0.003	0.398±0.004	0.215±0.005
Flag	0.970±0.001	0.925±0.001	0.834±0.002	0.677±0.003	0.345±0.004
PMNN	0.733±0.001	0.886±0.001	0.781±0.001	0.649±0.002	0.318±0.001
Ours	**0.976±0.001**	**0.940±0.001**	**0.858±0.002**	**0.695±0.002**	**0.351±0.002**

TABLE 6: Classification accuracy (mean±std) on MNIST dataset. The best results are bold and the second best are underlined.

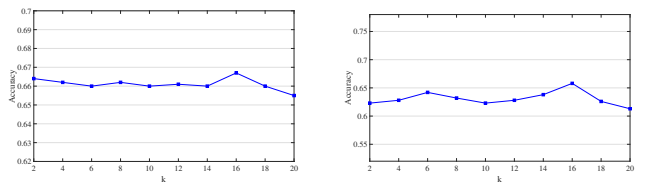
	Percentage of missing				
Method	10%	30%	50%	70%	90%
Zero	0.957±0.001	0.942±0.002	0.918±0.002	0.863±0.003	0.688±0.003
Mean	0.964±0.001	0.951±0.001	0.933±0.001	0.891±0.003	0.727±0.004
GMM	0.963±0.002	0.925±0.003	0.806±0.011	0.636±0.006	0.379±0.012
KNN	0.965±0.001	0.941±0.002	0.864±0.001	0.703±0.023	0.223±0.012
Flag	0.867±0.002	0.935±0.002	0.908±0.003	0.847±0.012	0.360±0.045
PMNN	0.907±0.001	0.910±0.002	0.883±0.003	0.842±0.002	0.700±0.004
Ours	**0.970±0.001**	**0.958±0.001**	**0.940±0.001**	**0.900±0.002**	**0.739±0.004**

TABLE 7: Classification accuracy (mean±std) on Avila dataset. The best results are bold and the second best are underlined.

	Percentage of missing				
Method	10%	30%	50%	70%	90%
Zero	0.722±0.002	0.630±0.005	0.553±0.006	0.496±0.004	0.433±0.002
Mean	0.718±0.005	0.630±0.005	0.556±0.003	0.492±0.003	0.433±0.002
MICE	0.717±0.005	0.618±0.005	0.435±0.007	0.422±0.002	0.412±0.001
GMM	0.722±0.004	0.633±0.004	**0.557±0.003**	0.470±0.002	0.432±0.002
KNN	0.746±0.004	0.620±0.002	0.536±0.006	0.474±0.003	0.426±0.002
Flag	0.713±0.005	0.630±0.004	0.555±0.003	0.491±0.002	0.433±0.002
PMNN	0.503±0.004	0.445±0.003	0.526±0.003	0.473±0.004	0.412±0.001
Ours	**0.765±0.002**	**0.646±0.004**	0.556±0.003	**0.499±0.003**	**0.444±0.001**

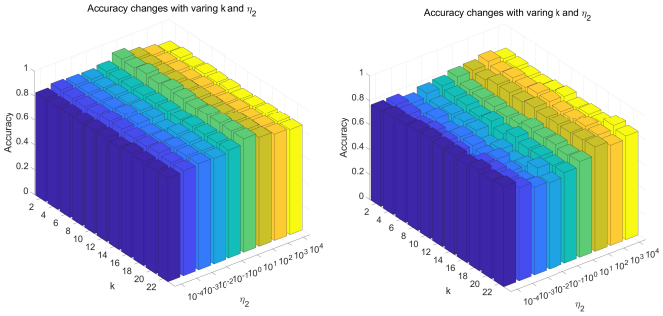


(a) Results on bands. (b) Results on hands (+30% missing values).

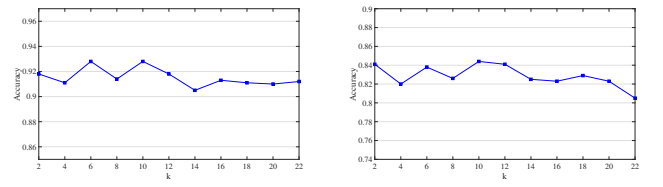


(c) Varying k on bands. (d) Varying k on bands (+30% missing values).

Fig. 5: Effect of Parameters on Dataset Bands.



(a) Results on horse. (b) Results on horse (+30% missing values).



(c) Varying k on horse . (d) Varying k on horse (+30% missing values).

Fig. 6: Effect of Parameters on Dataset Horse.

The **Avila** dataset was extracted from 800 images of the 'Avila Bible', an XII century giant Latin copy of the Bible. The prediction task consists in associating each pattern to a copyist, with the given 10 features. It consists of 12 classes and 20867 instances. Data have been normalized by using the Z-normalization method and divided into two subsets: a training set containing 10430 samples, and a test set containing 10437 samples. We randomly selected 25% of the training data as the validation set

karma and geom methods cannot be applied to neural networks so we omit them here. MICE cannot scale to MNIST dataset due to the high dimensionality of feature vectors. We compared an additional method proposed recently in [27] and named it **PMNN**. The number of components of GMM for PMNN was chosen from $\{3, 4, 5\}$. We did not compare with other neural networks for classification since they required complete instances for training. We compared all the baselines based on a multilayer perceptron (MLP) consists of 3 ReLU hidden layers with 100 neurons per layer. We used cross-entropy loss as the loss function in training. All hyper-parameters is selected based on the validation set. The range of hyper-parameters was similar to the linear model except that k was chosen from $\{2^1, 2^2, ..., 2^{\log_2 d}\}$ where d is the feature dimension. Because these datasets were complete, we randomly removed 10%, 30%, 50%, 70%, 90% of values in them. We repeated this procedure 5 times to report the classification accuracy with mean and standard deviation.

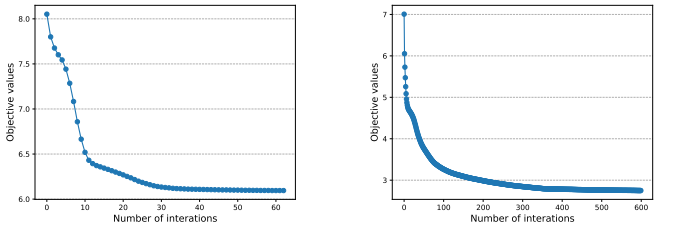
Figure 4 presents the average results on non-linear models. To keep image clear, we only draw top five methods, the comprehensive results are reported in Table 5 - 7. The tests drawn in Figure 4 demonstrate the superiority of our method with various missing ratios. Table 5 and Table 6 show the results of our method together with some baselines. The results show the advantage of our method over classical imputation methods and PMNN. Notice that PMNN produced a poor result when the missing ratio was low. PMNN requires to fit a GMM together with the neural network, but the GMM of PMNN is only trained with incomplete instances. Unlike GMM for imputation, where all data are used to fit the GMM, their model cannot be trained well when the percentage of missing is low. Flag shows good performance on Sensorless Drive Diagnosis dataset. However, its performance is limited on MNIST dataset. This indicates that the missingness patterns can be important in learning with incomplete data, but should be wisely incorporated into the model equation. Our model consistently outperforms other baselines, which verifies the effectiveness of our strategy to adjust the importance of present features by the missingness patterns.

Table 7 presents the experiment results on a smaller dataset Avila [62]. In general, our model performs better than other baselines, but compared to results on MNIST dataset, the improvements are relatively small. Because our method involves more parameters, it may require more data for the model learning.

6.3 Analysis of Parameters and Convergence

This section evaluates the performances of our proposed model by varying the critical parameters. As illustrated in Section 3.1 and 6.1, we fixed $\eta_1 = 10^{-6}$ to constraint the Frobenius norm of H. We here show the experimental results with various k and η_2 on datasets bands and horse, similar results can be gotten in other datasets. We discuss them jointly and pick them up by the grid search method.

Fig. 5 reveals the different accuracies with varying settings for k and η_2 on dataset bands. In general, our model is insensitive to k and η_2. When we increase η_2 from 10^{-4} to 10^4, the result improves at the beginning stage, and tends to stay stable at the range of $\{10^1, 10^4\}$. In particular,



(a) Conduct on the bands dataset. (b) Conduct on the horse dataset.

Fig. 7: Convergence rate.

our model achieves the best result when $k = 16$ and $\eta_2 = 10^1$, while it can get good performance if the k is set between 14 and 18. Fig. 6 reveals the effect of varying k and η_2 on dataset horse. Our model is insensitive to η_2 and k. We observe that the performance is stable when η_2 is ranged between 10^1 and 10^3. $\eta_2 = 10^1$ yields the best results. Generally speaking, when the k was less than full-rank, the performance was better. When we introduce more missing entries ($+30\%$), the trends are clearer that we attend at the best performance at $k = 16$ and $k = 10$ for bands and horse datasets, respectively. Our model is insensitive to k up to some values. Then the performance drops for overly-large k.

In summary, both parameters used in our model are benefit to the performance improvement. Moreover, our model is stable and easy fine-tuning because of the insensitivity for those parameters.

Figs. 7 (a) and (b) illustrate the convergence trends of our iterative model on both the above two datasets. It represents that our proposed efficient training algorithm can converge into a local solution in terms of the objective value in a small number of iterations.

7 Conclusion and Future Work

We proposed a general method for learning with incomplete data, where data of different missingness patterns are treated differently in model level. This idea can reduce the competition between data of different missingness patterns in training. In detail, we proposed a linear model that can be adaptively applied to data with different missingness patterns. And analysis of error bound justifies our method in the linear case. Our experiment results verified the effectiveness of our model empirically.

The dimension of missingness indicator vectors influences the computation complexity and generalization error, our future work will focus on how to develop a lower-dimension representation for them. Although we do not impute the missing data for our model, it does not conflict with imputation methods. How to combine various imputation methods with our model is another interesting future work.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 62202270, in part by the Natural Science Foundation of Shandong province, China, under Grant ZR2021QF034, in part by the Shandong Excellent Young Scientists Fund (Oversea) under Grant 2022HWYQ-044, and in part by the Open Fund of Beijing Key Laboratory of Traffic Data Analysis and Mining.

References

[1] Z. Li, J. Zhang, Q. Wu, Y. Gong, J. Yi, and C. Kirsch, "Sample adaptive multiple kernel learning for failure prediction of railway points," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2848–2856.

[2] K. J. Janssen, A. R. T. Donders, F. E. Harrell Jr, Y. Vergouwe, Q. Chen, D. E. Grobbee, and K. G. Moons, "Missing covariate data in medical research: to impute is better than to ignore," *Journal of clinical epidemiology*, vol. 63, no. 7, pp. 721–727, 2010.

[3] Y. Gong, Z. Li, J. Zhang, W. Liu, and Y. Zheng, "Online spatio-temporal crowd flow distribution prediction for complex metro system," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[4] Y. Gong, Z. Li, J. Zhang, W. Liu, B. Chen, and X. Dong, "A spatial missing value imputation method for multi-view urban statistical data." in *Proceedings of 29th International Joint Conference on Artificial Intelligence*, 2020, pp. 1310–1316.

[5] H. Qu, Y. Gong, M. Chen, J. Zhang, Y. Zheng, and Y. Yin, "Forecasting fine-grained urban flows via spatio-temporal contrastive self-supervision," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[6] W. V. Li and J. J. Li, "An accurate and robust imputation method scimpute for single-cell rna-seq data," *Nature communications*, vol. 9, no. 1, pp. 1–9, 2018.

[7] G. Eraslan, L. M. Simon, M. Mircea, N. S. Mueller, and F. J. Theis, "Single-cell rna-seq denoising using a deep count autoencoder," *Nature communications*, vol. 10, no. 1, pp. 1–14, 2019.

[8] Y. Cheng, Y. Gong, Y. Liu, B. Song, and Q. Zou, "Molecular design in drug discovery: a comprehensive review of deep generative models," *Briefings in Bioinformatics*, vol. 22, no. 6, p. bbab344, 2021.

[9] G. E. Batista and M. C. Monard, "A study of k-nearest neighbour as an imputation method." *HIS*, vol. 87, no. 251-260, p. 48, 2002.

[10] M. Ouyang, W. J. Welsh, and P. Georgopoulos, "Gaussian mixture clustering and imputation of microarray data," *Bioinformatics*, vol. 20, no. 6, pp. 917–923, 2004.

[11] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, pp. 1–68, 2010.

[12] O. Dekel, O. Shamir, and L. Xiao, "Learning to classify with missing and corrupted features," *Machine learning*, vol. 81, no. 2, pp. 149–178, 2010.

[13] J. Xia, S. Zhang, G. Cai, L. Li, Q. Pan, J. Yan, and G. Ning, "Adjusted weight voting algorithm for random forests in handling missing values," *Pattern Recognition*, vol. 69, pp. 52–60, 2017.

[14] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an em approach," in *Advances in neural information processing systems*, 1994, pp. 120–127.

[15] D. Williams, X. Liao, Y. Xue, and L. Carin, "Incomplete-data classification using logistic regression," in *Proceedings of the 22nd International Conference on Machine learning*. ACM, 2005, pp. 972–979.

[16] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller, "Max-margin classification of data with absent features," *Journal of Machine Learning Research*, vol. 9, no. Jan, pp. 1–21, 2008.

[17] A. Goldberg, B. Recht, J. Xu, R. Nowak, and J. Zhu, "Transduction with matrix completion: Three birds with one stone," in *Advances in neural information processing systems*, 2010, pp. 757–765.

[18] E. Hazan, R. Livni, and Y. Mansour, "Classification with low rank and missing data." in *ICML*, 2015, pp. 257–266.

[19] Z.-g. Liu, Q. Pan, J. Dezert, and A. Martin, "Adaptive imputation of missing values for incomplete pattern classification," *Pattern Recognition*, vol. 52, pp. 85–95, 2016.

[20] P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola, "Second order cone programming approaches for handling missing and uncertain data," *Journal of Machine Learning Research*, vol. 7, no. Jul, pp. 1283–1314, 2006.

[21] U. Dick, P. Haider, and T. Scheffer, "Learning from incomplete data with infinite imputations," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 232–239.

[22] I. Goodfellow, M. Mirza, A. Courville, and Y. Bengio, "Multi-prediction deep boltzmann machines," in *Advances in Neural Information Processing Systems*, 2013, pp. 548–556.

[23] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International Conference on Machine Learning*, 2018, pp. 5675–5684.

[24] S. C.-X. Li, B. Jiang, and B. Marlin, "Misgan: Learning from incomplete data with generative adversarial networks," in *International Conference on Learning Representations*, 2019.

[25] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6721–6729.

[26] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[27] M. Śmieja, L. u. Struski, J. Tabor, B. Zieliński, and P. a. Spurek, "Processing of missing data by neural networks," in *Advances in neural information processing systems*, 2018, pp. 2719–2729.

[28] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[29] P. Jing, Y. Su, X. Jin, and C. Zhang, "High-order temporal correlation model learning for time-series prediction," *IEEE transactions on cybernetics*, vol. 49, no. 6, pp. 2385–2397, 2018.

[30] H. Yin, H. Chen, X. Sun, H. Wang, Y. Wang, and Q. V. H. Nguyen, "Sptf: a scalable probabilistic tensor factorization model for semantic-aware behavior prediction," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 585–594.

[31] T. D. Pham and H. Yan, "Tensor decomposition of gait dynamics in parkinson's disease," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 8, pp. 1820–1827, 2017.

[32] H. Tan, G. Feng, J. Feng, W. Wang, Y.-J. Zhang, and F. Li, "A tensor-based method for missing traffic data completion," *Transportation Research Part C: Emerging Technologies*, vol. 28, pp. 15–27, 2013.

[33] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 208–220, 2012.

[34] K. Abed-Meraim, N. L. Trung, A. Hafiane *et al.*, "Robust tensor tracking with missing data and outliers: Novel adaptive cp decomposition and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4305–4320, 2022.

[35] C. Zhang, Y. Cui, Z. Han, J. T. Zhou, H. Fu, and Q. Hu, "Deep partial multi-view learning," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[36] Y. Gong, Z. Li, J. Zhang, W. Liu, B. Chen, and X. Dong, "A spatial missing value imputation method for multi-view urban statistical data," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 1310–1316.

[37] Y. Gong, Z. Li, J. Zhang, W. Liu, Y. Yin, and Y. Zheng, "Missing value imputation for multi-view urban statistical data via spatial correlation learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[38] X. Liu, X. Zhu, M. Li, L. Wang, C. Tang, J. Yin, D. Shen, H. Wang, and W. Gao, "Late fusion incomplete multi-view clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2410–2423, 2018.

[39] X. Liu, X. Zhu, M. Li, L. Wang, E. Zhu, T. Liu, M. Kloft, D. Shen, J. Yin, and W. Gao, "Multiple kernel k k-means with incomplete kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 5, pp. 1191–1204, 2019.

[40] X. Liu, M. Li, C. Tang, J. Xia, J. Xiong, L. Liu, M. Kloft, and E. Zhu, "Efficient and effective regularized incomplete multi-view clustering," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[41] Q. Ma, S. Li, and G. Cottrell, "Adversarial joint-learning recurrent neural network for incomplete time series classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[42] M. Kachuee, K. Karkkainen, O. Goldstein, S. Darabi, and M. Sarrafzadeh, "Generative imputation and stochastic prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[43] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[44] X. Chen and L. Sun, "Bayesian temporal factorization for multidimensional time series prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[45] X. Liu, L. Wang, X. Zhu, M. Li, E. Zhu, T. Liu, L. Liu, Y. Dou, and J. Yin, "Absent multiple kernel learning algorithms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 6, pp. 1303–1316, 2019.

[46] K. Pelckmans, J. De Brabanter, J. A. Suykens, and B. De Moor, "Handling missing values in support vector machine classifiers," *Neural Networks*, vol. 18, no. 5-6, pp. 684–692, 2005.

[47] S. Awawdeh, H. Faris, and H. Hiary, "Evoimputer: An evolutionary approach for missing data imputation and feature selection in the context of supervised learning," *Knowledge-Based Systems*, vol. 236, p. 107734, 2022.

[48] B. Bullins, E. Hazan, and T. Koren, "The limits of learning with missing data," in *Advances in Neural Information Processing Systems*, 2016, pp. 3495–3503.

[49] R. Ge, C. Jin, and Y. Zheng, "No spurious local minima in nonconvex low rank problems: A unified geometric analysis," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1233–1242.

[50] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[51] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, "Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks." *Journal of Machine Learning Research*, vol. 20, no. 63, pp. 1–17, 2019.

[52] N. Alon, "Tools from higher algebra," *Handbook of combinatorics*, vol. 2, pp. 1749–1783, 1995.

[53] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[54] I. Necoara, Y. Nesterov, and F. Glineur, "Linear convergence of first order methods for non-strongly convex optimization," *Mathematical Programming*, pp. 1–39, 2018.

[55] B. Grimmer, "Convergence rates for deterministic and stochastic subgradient methods without lipschitz continuity," *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 1350–1365, 2019.

[56] T. Yang and Q. Lin, "Rsg: Beating subgradient method without smoothness and strong convexity," *Journal of Machine Learning Research*, vol. 19, no. 6, pp. 1–33, 2018.

[57] I. Necoara, Y. Nesterov, and F. Glineur, "Linear convergence of first order methods for non-strongly convex optimization," *Mathematical Programming*, vol. 175, no. 1, pp. 69–107, 2019.

[58] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

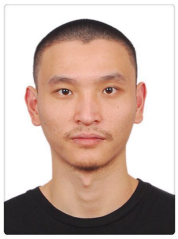
[59] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf, "Multiple imputation by chained equations: what is it and how does it work?" *International journal of methods in psychiatric research*, vol. 20, no. 1, pp. 40–49, 2011.

[60] Y. Hu, B. Li, W. Zhang, N. Liu, P. Cai, F. Chen, and K. Qu, "Wedge: imputation of gene expression values from single-cell rna-seq datasets using biased matrix decomposition," *Briefings in Bioinformatics*, vol. 22, no. 5, p. bbab085, 2021.

[61] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[62] C. De Stefano, M. Maniaci, F. Fontanella, and A. S. di Freca, "Reliable writer identification in medieval manuscripts through page layout features: The "avila" bible case," *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 99–110, 2018.

Yongshun Gong (SM'19-M'21) is an Associate Professor at School of Software, Shandong University, China. He received his Ph.D. degree from University of Technology Sydney in 2021. His principal research interest covers the data science and machine learning, in particular, the following areas: adaptive model; spatiotemporal data mining; urban flow prediction; recommender system and sequential pattern mining. He has published above 20 papers in top journals and refereed conference proceedings, including the IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics, IEEE Transactions on Multimedia, Pattern Recognition, NeurIPS, KDD, CIKM, AAAI and IJCAI.

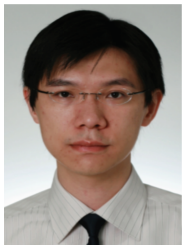


Zhibin Li obtained the Ph.D. degree from University of Technology Sydney in 2021. He is now a postdoctoral research fellow of the Commonwealth Scientific and Industrial Research Organisation, Australia. His principal research interest includes traffic analysis, factorization models, and theoretical analysis of machine learning. He has published 9 papers in top journals and refereed conference proceedings including TKDE, KDD2019 and NeurIPS2020.

Wei Liu (M'15-SM'20) is an Associate Professor and the Director of Robust Machine Learning Lab at the School of Computer Science, Faculty of Engineering and Information Technology, the University of Technology Sydney (UTS). Before joining UTS, he was a Research Fellow at the University of Melbourne and then a Machine Learning Researcher at NICTA. He obtained his PhD from the University of Sydney in 2011. He works in the areas of machine learning and data mining and has published more than 80 papers in the research topics of tensor factorization, adversarial learning, multimodal machine learning, graph mining, causal inference, and anomaly detection. He has won three best paper awards.

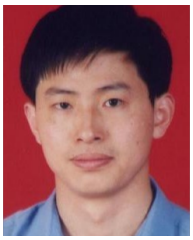
Xinwang Liu received his PhD degree from National University of Defense Technology (NUDT), China. He is now Professor at School of Computer, NUDT. His current research interests include kernel learning and unsupervised feature learning. Dr. Liu has published 70+ peer-reviewed papers, including those in highly regarded journals and conferences such as IEEE T-PAMI, IEEE T-KDE, IEEE T-IP, IEEE T-NNLS, IEEE T-MM, IEEE T-IFS, ICML, NeurIPS, CVPR, ICCV, AAAI, IJCAI, etc.

Xiankai Lu is a Research Professor in the School of Software, Shandong University. From 2018 to 2020, he was a research associate with Inception Institute of Artificial Intelligence, Abu Dhabi, UAE. He received the Ph.D. degree from Shanghai Jiao Tong University in 2018. Dr. Lu has published above 30 papers in top journals and refereed conference proceedings, such as TPAMI, TIP, TCSVT, CVPR, ICCV, ECCV, etc.



Ivor W. Tsang is Professor of Artificial Intelligence, at University of Technology Sydney. He is also the Research Director of the Australian Artificial Intelligence Institute. In 2019, his paper titled "Towards ultrahigh dimensional feature selection for big data" received the International Consortium of Chinese Mathematicians Best Paper Award. In 2020, Prof Tsang was recognized as the AI 2000 AAAI/IJCAI Most Influential Scholar in Australia for his outstanding contributions to the field of Artificial Intelligence between 2009 and 2019. His works on transfer learning granted him the Best Student Paper Award at International Conference on Computer Vision and Pattern Recognition 2010 and the 2014 IEEE Transactions on Multimedia Prize Paper Award. In addition, he had received the prestigious IEEE Transactions on Neural Networks Outstanding 2004 Paper Award in 2007.

Prof. Tsang serves as a Senior Area Chair for Neural Information Processing Systems and Area Chair for International Conference on Machine Learning, and the Editorial Board for Journal Machine Learning Research, Machine Learning, Journal of Artificial Intelligence Research, and IEEE Transactions on Pattern Analysis and Machine Intelligence.



Yilong Yin is the Director of the Machine Learning and Applications Group and a Distinguished Professor with Shandong University, Jinan, China. He received the Ph.D. degree from Jilin University, Changchun, China, in 2000. From 2000 to 2002, he was a Postdoctoral Fellow with the Department of Electronic Science and Engineering, Nanjing University, Nanjing, China. His research interests include machine learning, data mining, computational medicine, and biometrics. He has published above 100 papers in top journals and refereed conference proceedings including TKDE, TIP, TMM, ICML, IJCAI, etc.