Multi-Dimensional Classification via Selective Feature Augmentation

Bin-Bin Jia[1,2,3] Min-Ling Zhang[1,3]

[1]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China

[2] College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou 730050, China

[3] Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China

Abstract: In multi-dimensional classification (MDC), the semantics of objects are characterized by multiple class spaces from different dimensions. Most MDC approaches try to explicitly model the dependencies among class spaces in output space, while the recently proposed feature augmentation strategy which aims at manipulating feature space has also been shown as an effective solution for MDC. However, existing feature augmentation approaches only focus on designing holistic augmented features to be appended with the original features, while better generalization performance could be achieved by exploiting multiple kinds of augmented features. In this paper, we propose the *selective feature augmentation* strategy which focuses on how to synergize multiple kinds of augmented features. Specifically, by assuming that only part of the augmented features is pertinent and useful for each dimension's model induction, we derive a classification model which can fully utilize the original features while conduct feature selection for the augmented features. To validate the effectiveness of the proposed strategy, we generate three kinds of simple augmented features based on standard *k*NN, weighted *k*NN, and maximum margin techniques respectively. Comparative studies show that the proposed strategy achieves superior performance against both state-of-the-art MDC approaches and its degenerated versions with either kind of augmented features.

Keywords: Machine learning, multi-dimensional classification, feature augmentation, feature selection, class dependencies.

1 Introduction

Traditional supervised learning tasks usually characterize the semantics of objects with one output variable, i.e., single-output learning, among which multi-class classification is one of the most important learning frameworks. However, in some real-world applications, it is better to use multiple output variables to characterize the rich semantics of objects, which results in the problem of multi-output learning[1]. Here, when the type of each output variable is restricted to discrete-valued, then the *multi-dimensional classification* (MDC) framework is obtained[2,3]. Under the MDC setting, each object is represented by a single instance while associated with multiple class variables, each of which corresponds to a specific class space characterizing the object's semantics along one specific dimension. Specifically, the MDC problem widely exists in many application scenarios, such as bioinformatics[4,5], text classification[6,7], computer vision [8-10], resource allocation[11], etc. Fig. 1 shows an illustrative example of MDC on vehicle classification.

Formally speaking, let $\mathcal{X} \in \mathbb{R}^d$ be the d-dimensional feature space and $\mathcal{Y} = C_1 \times C_2 \times \cdots \times C_q$ be the output space. Here, $\mathcal{Y}$ corresponds to the Cartesian product of q class spaces $C_j = \{c_1^j, c_2^j, \ldots, c_{K_j}^j\}$ $(1 \le j \le q)$ which consists of K_j possible classes respectively. Given a set of MDC training examples $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \mid 1 \le i \le m\}$, where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^\top \in \mathcal{X}$ is a d-dimensional feature vector and $\boldsymbol{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iq}]^\top \in \mathcal{Y}$ is the q-dimensional class vector associated with $\boldsymbol{x}_i$ with each element $y_{ij} \in C_j$, the MDC task aims to learn a predictive model $f : \mathcal{X} \mapsto \mathcal{Y}$ from $\mathcal{D}$ which can return a proper class vector $f(\boldsymbol{x}_*) \in \mathcal{Y}$ for unseen instance $\boldsymbol{x}_*$.

Fig. 1 An illustrative example of multi-dimensional classification: Vehicle classification. For a vehicle, it can be classified from the *type* dimension (with possible classes *car*, *SUV*, *bus*, *truck*, etc.), from the *brand* dimension (with possible classes *Audi*, *Benz*, *YUTONG*, *JAC*, etc.), and from the *color* dimension (with possible classes *black*, *white*, *red*, *blue*, etc.). Here, Fig.(a) is *a red Audi car*, Fig.(b) is a *black Benz SUV*, Fig.(c) is a *red YUTONG bus*, and Fig.(d) is a *white JAC truck*.

It is obvious that the MDC problem can be solved dimension by dimension, i.e., training a multi-class classifier for each class space. However, this independent decomposition strategy does not consider potential dependencies among class spaces which might impact the generalization performance of the resulting model. The MDC problem can also be solved by a single multi-class classifier, where each distinct class combination is regarded as a new class. However, this powerset-like strategy cannot consider class combinations not appearing in the training

Research Article
Manuscript received date; accepted date; published online date
Recommended by Associate Editor xxxx

set and usually suffers high computational complexity due to possible large number of classes. In fact, one of the key challenges for MDC studies is how to model dependencies among class spaces in appropriate ways. Existing works mainly focus on modeling class dependencies in output space, such as capturing pairwise class dependencies[12-14], specifying chaining order over class spaces[15, 16], learning a directed acyclic graph (DAG) structure for class spaces[17-19], and partitioning class spaces into groups[2], etc. Recently, feature augmentation strategy which aims at manipulating feature space has been shown as an effective solution for MDC. This strategy enriches the original feature space with a set of new features which are generated by making use of some well-established techniques, e.g., *k*NN[20] or deep learning[21]. Existing works only focus on how to design more informative augmented features while it might be beneficial to exploit multiple kinds of augmented features generated by making use of different techniques. In this paper, we propose the *selective feature augmentation* strategy which makes the first attempt towards how to synergize multiple kinds of augmented features. The strategy is abbreviated as SFAM, i.e., *Selective Feature Augmentation for Multi-dimensional classification*, in the following parts of this paper for brevity. Specifically, SFAM assumes only part of augmented features are pertinent and useful for each dimension's model induction. To validate the effectiveness of SFAM, three simple kinds of augmented features are generated by making use of standard *k*NN, weighted *k*NN, and maximum margin techniques respectively. After that, for each dimension, SFAM derives a classification model which can take full advantage of the original features via ℓ_2 regularization and conduct feature selection for the augmented features via ℓ_1 regularization (i.e., selective feature augmentation). Experimental results demonstrate that SFAM achieves superior performance against both state-of-the-art MDC approaches and its degenerated versions with either kind of augmented features.

The rest of this paper is organized as follows. Firstly, related works on multi-dimensional classification are briefly discussed. Secondly, technical details of SFAM are introduced. Thirdly, experimental results of comparative studies are reported. Finally, we conclude this paper.

2 Related Work

The most related learning framework to multi-dimensional classification is the widely studied multi-label classification (MLC)[22-24], which can be regarded as a special case of MDC when the type of class variable in each dimension is restricted to binary-valued. However, MDC usually assumes *heterogeneous* class spaces which are used to characterize the rich semantics of objects from different dimensions, while MLC usually assumes *homogeneous* class space in which multiple concepts are relevant to the polysemous objects.

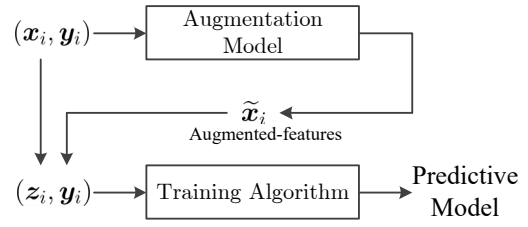
The MDC problem can be solved via independent decomposition strategy, where a total of q multi-class classifiers are learned independently, one per dimension. However, this intuitive strategy ignores possible dependencies among class spaces and the induced model would be suboptimal. An improved strategy is learning the q multi-class classifiers in a chaining order, where predictions of preceding classifiers are used as extra features by the subsequent ones[15,16]. However, the chaining order would largely affect the generalization performance while determining an optimal one is NP-hard. The MDC problem can also be solved via powerset transformation strategy where a single multi-class classifier is learned by regarding all distinct class combinations in training set as new classes. However, this intuitive strategy cannot consider class combinations not appearing in training set and usually suffers high computational complexity due to large number of new classes. An improved strategy is partitioning the class spaces into groups according to conditional dependencies[2]. However, the combinatorial nature still exists which leads to that the deficiencies cannot be fully addressed. A family of MDC models called multi-dimensional Bayesian network classifier[25] aim at learning different kinds of DAG structures over class spaces to explicitly model the class dependencies. However, determining DAG structures is computationally demanding and only nominal features can be tackled generally. The class dependencies can also be modeled in a two-level strategy[12-14], where pairwise dependencies are captured in the first level and then high-order dependencies are captured in the second-level based on the predictions from the first level. However, capturing pairwise dependencies needs $\mathcal{O}(q^2)$ complexity which is very time-consuming.

The aforementioned strategies mainly focus on directly modeling class dependencies in the output space, while the KRAM approach[20] attempts to manipulate the feature space of MDC examples via feature augmentation by making use of *k*NN techniques. Helpful discriminative information is expected to be brought into feature space which would facilitate the subsequent MDC model induction. Based on deep learning techniques, the LEFA approach[21] further generates better augmented features which can depict the inter-class dependencies and the intra-class exclusiveness simultaneously. However, these approaches simply treat the original and augmented features equally which might be less reasonable due to different characteristics of different features. Moreover, it is usually easier to design multiple kinds of simple augmented features than a terrific one, and it might be beneficial to consider synergizing the discriminative information residing in different kinds of augmented features. Fig. 2 shows an intuitive comparison between existing feature augmentation techniques and the proposed one in this paper. Existing works usually employ general MDC algorithms to accomplish the training phase, while the proposed one designs a novel training algorithm which can accomplish the selective feature augmentation phase.

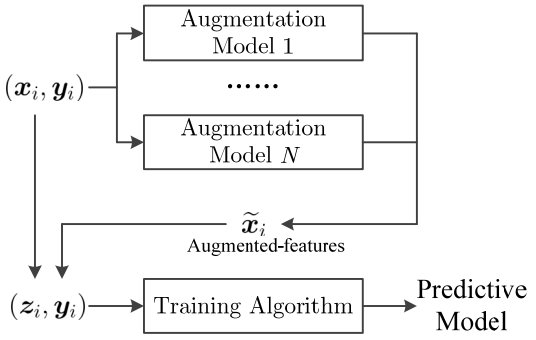
3 Technical Details of SFAM

This section presents how we implement the *selective*

feature augmentation strategy. In other words, the technical details in this section correspond to the part of *Training Algorithm* in Fig. 2(b). For any instance $\boldsymbol{x}_i$, let $\widetilde{\boldsymbol{x}}_i$ be the concatenation of all the corresponding augmented features generated by the N augmentation models, and $\boldsymbol{z}_i$ corresponds to the concatenation of $\boldsymbol{x}_i$ and $\widetilde{\boldsymbol{x}}_i$, i.e., $\boldsymbol{z}_i = [\boldsymbol{x}_i; \widetilde{\boldsymbol{x}}_i]$, in the following, we derive a regularized classification model which fully utilizes original features $\boldsymbol{x}_i$ while employs feature selection mechanism over augmented features $\widetilde{\boldsymbol{x}}_i$.



(a) Existing feature augmentation



(b) Selective feature augmentation

Fig. 2 An intuitive comparison of the training phase between existing feature augmentation techniques and the proposed one in this paper. Here, $\boldsymbol{z}_i$ is the concatenation of $\boldsymbol{x}_i$ and $\widetilde{\boldsymbol{x}}_i$. Specifically, existing feature augmentation techniques focus on designing better *Augmentation Model* to generate more informative augmented features, while this paper focuses on designing a novel *Training Algorithm* which aims at synergizing multiple kinds of augmented features.

For simplicity, we employ the one-vs-rest decomposition strategy for each dimension where a classification model with both ℓ_1 and ℓ_2 regularization is derived to solve the decomposed binary classification problems. Specifically, for the *a*th decomposed binary classification problem in the *j*th dimension, we determine the optimal model $(\boldsymbol{w}_{ja}^*, b_{ja}^*)$ as follows:

$$(\boldsymbol{w}_{ja}^*, b_{ja}^*) = \arg\min_{\boldsymbol{w}_{ja}, b_{ja}} \; L(\boldsymbol{w}_{ja}, b_{ja}) + \lambda R(\boldsymbol{w}_{ja}) \tag{1}$$

where λ is a trade-off parameter. The first term $L(\boldsymbol{w}_{ja}, b_{ja})$ denotes the empirical loss function. In this paper, we simply employ the cross-entropy loss which is defined as follows:

$$L(\boldsymbol{w}_{ja}, b_{ja}) = -\sum_{i=1}^{m} \Big[l_i^{ja} \cdot \ln h_{\boldsymbol{w}_{ja}, b_{ja}}(\boldsymbol{z}_i) + (1 - l_i^{ja}) \cdot \ln(1 - h_{\boldsymbol{w}_{ja}, b_{ja}}(\boldsymbol{z}_i)) \Big]$$

Here, $l_i^{ja} = \mathbf{1}_{y_{ij} = c_a^j}$, the predicate $\mathbf{1}_\pi$ returns 1 if π holds and 0 otherwise, and $h_{\boldsymbol{w}_{ja}, b_{ja}}(\boldsymbol{z}_i)$ is the logistic function which is defined as follows:

$$h_{\boldsymbol{w}_{ja}, b_{ja}}(\boldsymbol{z}_i) = \frac{1}{1 + e^{-(\langle \boldsymbol{w}_{ja}, \boldsymbol{z}_i \rangle + b_{ja})}}$$

where $\langle \cdot, \cdot \rangle$ returns the inner product of two vectors. The second term $R(\boldsymbol{w}_{ja})$ denotes the regularization term which is defined as follows:

$$R(\boldsymbol{w}_{ja}) = \|\boldsymbol{\theta}_{ja}\|_2^2 + \|\widetilde{\boldsymbol{\theta}}_{ja}\|_1$$

where $\boldsymbol{\theta}_{ja}$ and $\widetilde{\boldsymbol{\theta}}_{ja}$ denote the first d elements and the last remaining elements of $\boldsymbol{w}_{ja}$ respectively, i.e., $\boldsymbol{w}_{ja} = [\boldsymbol{\theta}_{ja}; \widetilde{\boldsymbol{\theta}}_{ja}]$. It is worth noting that the ℓ_2 regularization corresponds to the original features while the ℓ_1 regularization corresponds to the augmented features. By doing this, we employ feature selection mechanism over the augmented features to synergize multiple kinds of augmented features in a better way, while the original features are still fully utilized. In other words, the *selective feature augmentation* strategy is implemented here.

To optimize the problem (1), we solve one of the three sets of parameters $\{\boldsymbol{\theta}_{ja}\}$, $\{\widetilde{\boldsymbol{\theta}}_{ja}\}$and $\{b_{ja}\}$ alternatingly, while the remaining parameters are fixed.

(a) Optimizing w.r.t. $\{\boldsymbol{\theta}_{ja}\}$ when $\{\widetilde{\boldsymbol{\theta}}_{ja}\}$ and $\{b_{ja}\}$ are fixed: When $\{\widetilde{\boldsymbol{\theta}}_{ja}\}$ and $\{b_{ja}\}$ are fixed, the optimization problem (1) can be equivalently reformulated as follows:

$$\min_{\boldsymbol{\theta}_{ja}} \; L_1(\boldsymbol{\theta}_{ja}) + \lambda \|\boldsymbol{\theta}_{ja}\|_2^2 \tag{2}$$

where

$$L_1(\boldsymbol{\theta}_{ja}) = -\sum_{i=1}^{m} \Big[l_i^{ja} \cdot \ln h_{\boldsymbol{\theta}_{ja}}(\boldsymbol{z}_i) + (1 - l_i^{ja}) \cdot \ln(1 - h_{\boldsymbol{\theta}_{ja}}(\boldsymbol{z}_i)) \Big]$$

and

$$h_{\boldsymbol{\theta}_{ja}}(\boldsymbol{z}_i) = \frac{1}{1 + e^{-(\langle \boldsymbol{\theta}_{ja}, \boldsymbol{x}_i \rangle + C_1^i)}}$$

Here, $C_1^i = \langle \widetilde{\boldsymbol{\theta}}_{ja}, \widetilde{\boldsymbol{x}}_i \rangle + b_{ja}$ is a constant which is not dependent on variables $\boldsymbol{\theta}_{ja}$. In this paper, we use gradient descent to solve the optimization problem (2). Specifically, let $\Omega_1(\boldsymbol{\theta}_{ja})$ be the objective function, the gradient is given as follows:

$$\frac{\partial \Omega_1(\boldsymbol{\theta}_{ja})}{\partial \boldsymbol{\theta}_{ja}} = -\sum_{i=1}^{m} (l_i^{ja} - h_{\boldsymbol{\theta}_{ja}}(\boldsymbol{z}_i)) \boldsymbol{x}_i + 2\lambda \boldsymbol{\theta}_{ja}$$

(b) Optimizing w.r.t. $\{\widetilde{\boldsymbol{\theta}}_{ja}\}$ when $\{\boldsymbol{\theta}_{ja}\}$ and $\{b_{ja}\}$ are fixed: When $\{\boldsymbol{\theta}_{ja}\}$ and $\{b_{ja}\}$ are fixed, the optimization problem (1) can be equivalently reformulated as follows:

$$\min_{\widetilde{\boldsymbol{\theta}}_{ja}} \; L_2(\widetilde{\boldsymbol{\theta}}_{ja}) + \lambda \|\widetilde{\boldsymbol{\theta}}_{ja}\|_1 \tag{3}$$

where

$$L_2(\widetilde{\boldsymbol{\theta}}_{ja}) = -\sum_{i=1}^{m}\Big[l_i^{ja}\cdot \ln h_{\widetilde{\boldsymbol{\theta}}_{ja}}(\boldsymbol{z}_i)+ (1-l_i^{ja})\cdot \ln(1-h_{\widetilde{\boldsymbol{\theta}}_{ja}}(\boldsymbol{z}_i))\Big]$$

and

$$h_{\widetilde{\boldsymbol{\theta}}_{ja}}(\boldsymbol{z}_i) = \frac{1}{1+e^{-(\langle \widetilde{\boldsymbol{\theta}}_{ja}, \widetilde{\boldsymbol{x}}_i\rangle + C_2^i)}}$$

Here, $C_2^i = \langle \boldsymbol{\theta}_{ja}, \boldsymbol{x}_i\rangle + b$ is a constant which is not dependent on variables $\{\widetilde{\boldsymbol{\theta}}_{ja}\}$. In this paper, we use accelerated proximal gradient method[26] to solve it.

Theorem 1. *For the derivable function $L_2(\widetilde{\boldsymbol{\theta}}_{ja})$, $\nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja})$ is Lipschitz continuous and the Lipschitz constant is:*

$$L_f = \frac{1}{4}\sum_{i=1}^{m}\|\widetilde{\boldsymbol{x}}_i\|_2^2 \tag{4}$$

where ∇ denotes the differential operator.

Proof. For $\nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja})$, it can be calculated as:

$$\nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja}) = \frac{\partial L_2(\widetilde{\boldsymbol{\theta}}_{ja})}{\partial \widetilde{\boldsymbol{\theta}}_{ja}} = -\sum_{i=1}^{m}(l_i^{ja} - h_{\widetilde{\boldsymbol{\theta}}_{ja}}(\boldsymbol{z}_i))\widetilde{\boldsymbol{x}}_i$$

Given any $\widetilde{\boldsymbol{\theta}}_{ja}$ and $\widetilde{\boldsymbol{\theta}}'_{ja}$, we have:

$$\begin{aligned}
&\left\|\nabla L_2(\widetilde{\boldsymbol{\theta}}'_{ja}) - \nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja})\right\|_2 \\
&\quad= \left\|\sum_{i=1}^{m}\left(h_{\widetilde{\boldsymbol{\theta}}'_{ja}}(\boldsymbol{z}_i) - h_{\widetilde{\boldsymbol{\theta}}_{ja}}(\boldsymbol{z}_i)\right)\widetilde{\boldsymbol{x}}_i\right\|_2 \\
&\quad\le \sum_{i=1}^{m}\left\|h_{\widetilde{\boldsymbol{\theta}}'_{ja}}(\boldsymbol{z}_i) - h_{\widetilde{\boldsymbol{\theta}}_{ja}}(\boldsymbol{z}_i)\right\|_2\|\widetilde{\boldsymbol{x}}_i\|_2 \\
&\quad\le \sum_{i=1}^{m}\frac{1}{4}\left\|\langle\widetilde{\boldsymbol{\theta}}'_{ja}, \widetilde{\boldsymbol{x}}_i\rangle - \langle\widetilde{\boldsymbol{\theta}}_{ja}, \widetilde{\boldsymbol{x}}_i\rangle\right\|_2\|\widetilde{\boldsymbol{x}}_i\|_2 \\
&\quad\le \sum_{i=1}^{m}\frac{1}{4}\left\|\langle\widetilde{\boldsymbol{\theta}}'_{ja} - \widetilde{\boldsymbol{\theta}}_{ja}, \widetilde{\boldsymbol{x}}_i\rangle\right\|_2\|\widetilde{\boldsymbol{x}}_i\|_2 \\
&\quad\le \frac{1}{4}\sum_{i=1}^{m}\left\|\widetilde{\boldsymbol{\theta}}'_{ja} - \widetilde{\boldsymbol{\theta}}_{ja}\right\|_2\|\widetilde{\boldsymbol{x}}_i\|_2^2
\end{aligned}$$

Here, the second "$\le$" is due to the truth that the Lipschitz constant of logistic function equals $\frac{1}{4}$. Then, it is easy to know:

$$\frac{\left\|\nabla L_2(\widetilde{\boldsymbol{\theta}}'_{ja}) - \nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja})\right\|_2}{\left\|\widetilde{\boldsymbol{\theta}}'_{ja} - \widetilde{\boldsymbol{\theta}}_{ja}\right\|_2} \le \frac{1}{4}\sum_{i=1}^{m}\|\widetilde{\boldsymbol{x}}_i\|_2^2$$

which completes the proof. □

According to Theorem 1, given any initial value $\widetilde{\boldsymbol{\theta}}_{ja}^{(t)}$ of $\widetilde{\boldsymbol{\theta}}_{ja}$, let $\Delta\widetilde{\boldsymbol{\theta}}_{ja} = \widetilde{\boldsymbol{\theta}}_{ja} - \widetilde{\boldsymbol{\theta}}_{ja}^{(t)}$, the following inequation always holds:

$$\left\|\nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja}) - \nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja}^{(t)})\right\|_2 \le L_f\left\|\Delta\widetilde{\boldsymbol{\theta}}_{ja}\right\|_2$$

Then, the quadratic approximation of $L_2(\widetilde{\boldsymbol{\theta}}_{ja})$ around $\widetilde{\boldsymbol{\theta}}_{ja}^{(t)}$ can be given as follows:

$$\begin{aligned}
\widehat{L}_2(\widetilde{\boldsymbol{\theta}}_{ja}) &\simeq L_2(\widetilde{\boldsymbol{\theta}}_{ja}^{(t)}) + \langle\nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja}^{(t)}), \Delta\widetilde{\boldsymbol{\theta}}_{ja}\rangle + \frac{L_f}{2}\left\|\Delta\widetilde{\boldsymbol{\theta}}_{ja}\right\|_2^2 \\
&= \frac{L_f}{2}\left\|\widetilde{\boldsymbol{\theta}}_{ja} - \boldsymbol{u}^{(t)}\right\|_2^2 + C_{L_f}
\end{aligned}$$

where $C_{L_f} = -\frac{1}{2L_f}\left\|\nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja}^{(t)})\right\|_2^2 + L_2(\widetilde{\boldsymbol{\theta}}_{ja}^{(t)})$ is a constant which is not dependent on variables $\widetilde{\boldsymbol{\theta}}_{ja}$, and

$$\boldsymbol{u}^{(t)} = \widetilde{\boldsymbol{\theta}}_{ja}^{(t)} - \frac{1}{L_f}\nabla L_2(\widetilde{\boldsymbol{\theta}}_{ja}^{(t)}) \tag{5}$$

According to the descent lemma[27], the approximation is an upper bound of the original function, i.e., $L_2(\widetilde{\boldsymbol{\theta}}_{ja}) \le \widehat{L}_2(\widetilde{\boldsymbol{\theta}}_{ja})$ always holds. Therefore, we can minimize the original function by iteratively minimizing the approximation. Plugging the above approximation into the optimization problem (3), we can obtain the following iterative equation:

$$\begin{aligned}
\widetilde{\boldsymbol{\theta}}_{ja}^{(t+1)} &= \arg\min_{\widetilde{\boldsymbol{\theta}}_{ja}}\frac{L_f}{2}\left\|\widetilde{\boldsymbol{\theta}}_{ja} - \boldsymbol{u}^{(t)}\right\|_2^2 + \lambda\left\|\widetilde{\boldsymbol{\theta}}_{ja}\right\|_1 \\
&= \mathrm{soft}(\boldsymbol{u}^{(t)}, \frac{\lambda}{L_f})
\end{aligned} \tag{6}$$

Here, $\mathrm{soft}(\cdot,\cdot)$ is the (element-wise) soft-thresholding function which is defined as follows:

$$\mathrm{soft}(x,\mu) = \begin{cases} x-\mu & \text{if } x > \mu \\ x+\mu & \text{if } x < -\mu \\ 0 & \text{otherwise.} \end{cases}$$

In [28], it is shown that the convergence rate of iterative equation in (6) can be improved to $O(t^{-2})$ from $O(t^{-1})$ if we replace $\widetilde{\boldsymbol{\theta}}_{ja}^{(t)}$ in (5) with the following $\boldsymbol{v}_{ja}^{(t)}$:

$$\boldsymbol{v}_{ja}^{(t)} = \widetilde{\boldsymbol{\theta}}_{ja}^{(t)} + \frac{r_{t-1}-1}{r_t}(\widetilde{\boldsymbol{\theta}}_{ja}^{(t)} - \widetilde{\boldsymbol{\theta}}_{ja}^{(t-1)}) \tag{7}$$

where $r_0 = r_1 = 1$ and $r_t = \frac{1+\sqrt{1+4r_{t-1}^2}}{2}$ when $t > 1$.

(c) Optimizing w.r.t. $\{b_{ja}\}$ when $\{\boldsymbol{\theta}_{ja}\}$ and $\{\widetilde{\boldsymbol{\theta}}_{ja}\}$ are fixed: When $\{\boldsymbol{\theta}_{ja}\}$ and $\{\widetilde{\boldsymbol{\theta}}_{ja}\}$ are fixed, the optimization problem (1) can be equivalently reformulated as follows:

$$\min_{b_{ja}} \quad L_3(b_{ja}) \tag{8}$$

where

$$L_3(b_{ja}) = -\sum_{i=1}^{m}\Big[l_i^{ja}\cdot \ln h_{b_{ja}}(\boldsymbol{z}_i)+ (1-l_i^{ja})\cdot \ln(1-h_{b_{ja}}(\boldsymbol{z}_i))\Big]$$

and

$$h_{b_{ja}}(\boldsymbol{z}_i) = \frac{1}{1+e^{-(b_{ja}+C_3^i)}}$$

Here, $C_3^i = \langle \boldsymbol{w}_{ja}, \boldsymbol{z}_i\rangle$ is a constant which is not dependent on variable $\{b_{ja}\}$. In this paper, we use gradient descent to solve it. Specifically, the gradient of the objective function (i.e., $L_3(b_{ja})$) is given as follows:

$$\frac{\partial L_3(b_{ja})}{\partial b_{ja}} = -\sum_{i=1}^{m}(l_i^{ja} - h_{b_{ja}}(\boldsymbol{z}_i))$$

Algorithm 1 The proposed SFAM approach.

Input: $\mathcal{D}$: MDC training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i) \mid 1 \le i \le m\}$
λ: trade-off parameter in (1)
$\mathcal{A}_n$: augmentation model n ($1 \le n \le N$)
$\boldsymbol{x}_*$: unseen instance

Output: $\boldsymbol{y}_*$: predicted class vector for $\boldsymbol{x}_*$

1: $\widetilde{\mathcal{D}} = \varnothing$;
2: **for** $i = 1$ to m **do**
3: **for** $n = 1$ to N **do**
4: Generate the nth augmented features $\boldsymbol{\Delta}_n^{\boldsymbol{x}_i}$ for the ith training example $\boldsymbol{x}_i$ via $\mathcal{A}_n$;
5: **end for**
6: $\widetilde{\mathcal{D}} = \widetilde{\mathcal{D}} \cup (\boldsymbol{z}_i, \boldsymbol{y}_i)$ where $\boldsymbol{z}_i = [\boldsymbol{x}_i; \widetilde{\boldsymbol{x}}_i]$, $\widetilde{\boldsymbol{x}}_i$ is the concatenation of $\boldsymbol{\Delta}_1^{\boldsymbol{x}_i}, \ldots, \boldsymbol{\Delta}_N^{\boldsymbol{x}_i}$;
7: **end for**
8: **for** $j = 1$ to q **do**
9: **for** $a = 1$ to K_j **do**
10: Initialize $\boldsymbol{w}_{ja} = [\boldsymbol{\theta}_{ja}; \widetilde{\boldsymbol{\theta}}_{ja}] = \boldsymbol{0}$, $b_{ja} = 0$;
11: **Repeat**
12: Update $\boldsymbol{\theta}_{ja}$ by solving the optimization problem (2) via gradient descent;
13: Initialize $\widetilde{\boldsymbol{\theta}}_{ja}^{(0)} = \widetilde{\boldsymbol{\theta}}_{ja}^{(1)} = \widetilde{\boldsymbol{\theta}}_{ja}$, $r_0 = r_1 = 1$, $t = 1$;
14: **Repeat**
15: Obtain $\boldsymbol{v}_{ja}^{(t)}$ according to (7);
16: Compute $\boldsymbol{u}^{(t)} = \boldsymbol{v}_{ja}^{(t)} - \frac{1}{L_f}\nabla L_2(\boldsymbol{v}_{ja}^{(t)})$;
17: Obtain $\widetilde{\boldsymbol{\theta}}_{ja}^{(t+1)}$ according to (6);
18: Compute $r_{t+1} = \frac{1+\sqrt{1+4r_t^2}}{2}$;
19: $t = t + 1$;
20: **Until** Convergence
21: Update $\widetilde{\boldsymbol{\theta}}_{ja}$ with $\widetilde{\boldsymbol{\theta}}_{ja}^{(t)}$
22: Update b_{ja} by solving the optimization problem (8) via gradient descent;
23: **Until** Convergence
24: **end for**
25: **end for**
26: Obtain $\boldsymbol{x}_*$'s augmented features $\widetilde{\boldsymbol{x}}^* = [\boldsymbol{\Delta}_1^{\boldsymbol{x}_*}; \ldots; \boldsymbol{\Delta}_N^{\boldsymbol{x}_*}]$
27: **for** $j = 1$ to q **do**
28: Determine the class y_{*j} according to (9);
29: **end for**
30: Return $\boldsymbol{y}_* = [y_{*1}, y_{*2}, \ldots, y_{*q}]^\top$;

As the above three alternating optimizing steps converge, we can obtain the optimal values of $\boldsymbol{w}_{ja}$ and b_{ja}. For unseen instance $\boldsymbol{x}_*$, let $\widetilde{\boldsymbol{x}}_*$ be its augmented features, then its class label in the jth dimension can be determined based on the augmented instance $\boldsymbol{z}_* = [\boldsymbol{x}_*; \widetilde{\boldsymbol{x}}_*]$ as follows:

$$y_{*j} = c_{\hat{a}}^j, \text{ where } \hat{a} = \arg\max_{1 \le a \le K_j} \langle \boldsymbol{w}_{ja}, \boldsymbol{z}_* \rangle + b_{ja} \quad (9)$$

The complete procedure of the proposed SFAM approach is summarized in Algorithm 1. Firstly, SFAM transforms the original MDC training set $\mathcal{D}$ into $\widetilde{\mathcal{D}}$ by augmenting each instance's feature space (steps 1-7). After that, the predictive model is induced via a classification model with both ℓ_2 and ℓ_1 regularization (steps 8-25), where ℓ_2 regularization and the bias term are updated via gradient descent and ℓ_1 regularization is updated via accelerated proximal gradient method. Finally, the class vector of unseen instance is predicted based on the augmented features as well (steps 26-30). As shown in Algorithm 1, it is worth noting that SFAM should be regarded as a general framework and can be coupled with any kind of augmented-features, while this paper only aims at investigating the feasibility of synergizing the different kinds of augmented features.

4 Experiments

This section conducts comparative studies and the obtained experimental results clearly validate the superiority and effectiveness of SFAM. Firstly, Subsection 4.1 introduces the experimental setup including the employed benchmark data sets, the evaluation metrics and the compared approaches. Then, Subsection 4.2 reports the detailed experimental results with statistical tests. Finally, Subsection 4.3 further investigates SFAM's algorithmic design and parameter sensitivity.

Table 1 Characteristics of benchmark data sets.

Data Set	*#Exam.*	*#Dim.*	*#Labels/Dim.*	*#Features*
Flare1	323	3	3,4,2	10x
Enb	768	2	2,4	6n
WQplants	1060	7	4	16n
WQanimals	1060	7	4	16n
WaterQuality	1060	14	4	16n
BeLaE	1930	5	5	1n, 44x
Voice	3136	2	4,2	19n
TIC2000	9822	3	6,4,2	83x
Adult	18419	4	7,7,5,2	5n, 5x
Default	28779	4	2,7,4,2	14n, 6x

4.1 Experimental Setup

1) Benchmark Data Sets

In this paper, a total of ten benchmark data sets are collected for comparative studies. Table 1 summarizes the detailed characteristics of all benchmark data sets, including *number of examples* (#Exam.), *number of dimensions* (#Dim.), *number of class labels per dimension*

(#Labels/Dim.), and *number of features* (#Features). Here, for the *#Labels/Dim.* column, if all dimensions contain the same number of class labels, then only this number is recorded, otherwise, the number of class labels per dimension is recorded in turn; for the *#Features* column, n and x denote numeric and nominal features respectively.

2) Evaluation Metrics

In this paper, a total of three widely-used evaluation metrics are employed for performance evaluation, including *Hamming Score* (HS), *Exact Match* (EM) and *Sub-Exact Match* (SEM) [2, 13, 14, 20, 29]. Specifically, given the test set $\mathcal{S}=\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\mid 1\le i\le p\}$, for the MDC model $f:\mathcal{X}\mapsto\mathcal{Y}$ to be evaluated, let $\hat{\boldsymbol{y}}_i=f(\boldsymbol{x}_i)=[\hat{y}_{i1},\hat{y}_{i2},\ldots,\hat{y}_{iq}]^\top$ be the predicted class vector for $\boldsymbol{x}_i$ while the ground-truth one is $\boldsymbol{y}_i=[y_{i1},y_{i2},\ldots,y_{iq}]^\top$, then the number of correctly predicted dimensions corresponds to $r^{(i)}=\sum_{j=1}^{q}\mathbf{1}_{y_{ij}=\hat{y}_{ij}}$. The detailed definitions of the metrics are given as follows:

$$\mathrm{HS}_{\mathcal{S}}(f)=\frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}\cdot r^{(i)}$$

$$\mathrm{EM}_{\mathcal{S}}(f)=\frac{1}{p}\sum_{i=1}^{p}\mathbf{1}_{r^{(i)}=q}$$

$$\mathrm{SEM}_{\mathcal{S}}(f)=\frac{1}{p}\sum_{i=1}^{p}\mathbf{1}_{r^{(i)}\ge q-1}$$

For the three metrics, it is easy to know that the *larger* the values, the better the performance. Ten-fold cross validation is conducted over each benchmark data set, where both the mean metric value as well as standard deviation are recorded for comparative studies.

3) Compared Approaches

In this paper, a total of six state-of-the-art MDC approaches are employed as compared approaches, including BR, CP, ECC[16], ESC[2], gMML[29] and SEEM [13]:

- BR solves the MDC problem via training a number of multi-class classifiers independently, one per dimension. BR serves as the baseline when all possible class dependencies are ignored.
- CP transforms the MDC problem into a single multi-class classification problem by treating the whole output space as a compound one, where each distinct class combination in training set is regarded as a new class. CP serves as the baseline when all possible class dependencies in training set are considered, but overfitting might occur because CP cannot return class combinations not appearing in training set.
- ECC solves the MDC problem via training a chain of multi-class classifiers, one per dimension, where predictions of preceding classifiers on the chain are used as extra features by the subsequent ones.
- ESC preprocesses the MDC problem via partitioning the class variables into super-class, where each super-class is used as a compound class variable.
- gMML works by learning a regression model for each class label as well as a Mahalanobis metric which can shorten the distance between the regression outputs and ground-truth label vector.
- SEEM models the class dependency via a two-level strategy, where the pairwise and high-order class dependencies are modeled in the first and second level respectively.

For BR, CP, ECC, ESC and SEEM, the multi-class base learner is implemented via LIBLINEAR[30] with the parameter setting "L2-regularized logistic regression (primal)" for fair comparison. Following [2], for ensemble approaches ECC and ESC, a total of 10 base models are trained over 67% examples randomly selected from training set, and the predictive results are combined via majority voting. For gMML and SEEM, the recommended parameters are used according to respective literatures.

To validate the effectiveness of the selective feature augmentation strategy, for the proposed SFAM approach, a total of three *simple* kinds of augmented features are generated, where two of them are generated by making use of standard and weighted *k*NN techniques respectively, and the remaining one are generated by making use of maximum margin techniques. Specifically, the two kinds of *k*NN-based augmented features are generated by KRAM[20]. To be more specific, for each instance $\boldsymbol{x}$, let $\mathcal{N}_k(\boldsymbol{x})=\{i_r\mid 1\le r\le k\}$ be the set of indices for the k nearest neighbors of $\boldsymbol{x}$ identified in training set $\mathcal{D}$, we can define an indicating vector $\boldsymbol{v}_{ja}^{\boldsymbol{x}}=[v_{ja}^{\boldsymbol{x}}(1),v_{ja}^{\boldsymbol{x}}(2),\ldots,v_{ja}^{\boldsymbol{x}}(k)]^\top\in\{0,1\}^k$ which is defined as follows:

$$v_{ja}^{\boldsymbol{x}}(r)=\mathbf{1}_{y_{i_rj}=c_a^j}\quad(1\le r\le k,\ i_r\in\mathcal{N}(\boldsymbol{x}))$$

Here, $1\le a\le K_j, 1\le j\le q$. $\boldsymbol{y}_{i_r}=[y_{i_r1},\ldots,y_{i_rq}]^\top$ corresponds to the class vector of the neighboring MDC example $\boldsymbol{x}_{i_r}$ for $\boldsymbol{x}$. Based on $\boldsymbol{v}_{ja}^{\boldsymbol{x}}$, the following discrete version of statistics $\boldsymbol{\delta}_j^{\boldsymbol{x}}=[\delta_{j1}^{\boldsymbol{x}},\delta_{j2}^{\boldsymbol{x}},\ldots,\delta_{jK_j}^{\boldsymbol{x}}]^\top$ can be defined w.r.t. the jth class space:

$$\delta_{ja}^{\boldsymbol{x}}=\langle\mathbb{I}_k,\boldsymbol{v}_{ja}^{\boldsymbol{x}}\rangle\quad(1\le a\le K_j)$$

where $\mathbb{I}_k$ is a column vector of all ones with length k. By concatenating all the q counting statistics vectors, the first *k*NN-augmented feature vector $\boldsymbol{\Delta}_{k\mathrm{NN1}}^{\boldsymbol{x}}$ based on standard *k*NN techniques for $\boldsymbol{x}$ can be obtained:

$$\boldsymbol{\Delta}_{k\mathrm{NN1}}^{\boldsymbol{x}}=\left[\boldsymbol{\delta}_1^{\boldsymbol{x}},\boldsymbol{\delta}_2^{\boldsymbol{x}},\ldots,\boldsymbol{\delta}_q^{\boldsymbol{x}}\right]^\top\tag{9}$$

Moreover, let $\boldsymbol{\beta}=[1,1/\sqrt{2},\cdots,1/\sqrt{k}]^\top$, a bias vector $\boldsymbol{\zeta}_j^{\boldsymbol{x}}=[\zeta_{j1}^{\boldsymbol{x}},\zeta_{j2}^{\boldsymbol{x}},\ldots,\zeta_{jK_j}^{\boldsymbol{x}}]^\top$ is defined as follows:

$$\zeta_{ja}^{\boldsymbol{x}}=\frac{\langle\boldsymbol{\beta},\boldsymbol{v}_{ja}^{\boldsymbol{x}}\rangle-min(\boldsymbol{v}_{ja}^{\boldsymbol{x}})}{max(\boldsymbol{v}_{ja}^{\boldsymbol{x}})-min(\boldsymbol{v}_{ja}^{\boldsymbol{x}})}(\zeta_{max}-\zeta_{min})+\zeta_{min}$$

Here, ζ_{max} and ζ_{min} are two hyper-parameters and set as 0.5 and 0 respectively, and $max(\boldsymbol{v}_{ja}^{\boldsymbol{x}})=\sum_{r=1}^{\delta_{ja}^{\boldsymbol{x}}}\beta(r)$, $min(\boldsymbol{v}_{ja}^{\boldsymbol{x}})=\sum_{r=r0}^{k}\beta(r)$ where $r0=k-\delta_{ja}^{\boldsymbol{x}}+1$ and $\beta(r)$ denotes the *r*th element of weight vector $\boldsymbol{\beta}$. Then, the second *k*NN-augmented feature vector $\boldsymbol{\Delta}_{k\mathrm{NN2}}^{\boldsymbol{x}}$ based on weighted *k*NN techniques for $\boldsymbol{x}$ can be obtained:

$$\boldsymbol{\Delta}_{k\mathrm{NN2}}^{\boldsymbol{x}}=\boldsymbol{\Delta}_{k\mathrm{NN1}}^{\boldsymbol{x}}+\left[\boldsymbol{\zeta}_1^{\boldsymbol{x}},\boldsymbol{\zeta}_2^{\boldsymbol{x}},\ldots,\boldsymbol{\zeta}_q^{\boldsymbol{x}}\right]^\top\tag{10}$$

For the maximum margin-augmented features, SFAM

employs the real-valued predictions which are returned by the multi-class support vector machine to generate the maximum margin-augmented features. Specifically, SFAM solves the following maximum margin formulation[31] for the *j*th dimension ($1 \le j \le q$):

$$\min_{\mathbf{P}_j,\xi_j} \ \frac{1}{2}\sum_{a=1}^{K_j}\|\boldsymbol{p}_{ja}\|_2^2+\gamma\cdot\sum_{i=1}^{m}\xi_{ji}$$
$$\text{s.t.} \ \ \langle \boldsymbol{p}_{ja_i},\boldsymbol{x}_i\rangle-\langle \boldsymbol{p}_{ja},\boldsymbol{x}_i\rangle\ge e_{ji}^{a}-\xi_{ji},$$
$$1\le a\le K_j,\ 1\le i\le m$$

where $\mathbf{P}_j=[\boldsymbol{p}_{j1},\boldsymbol{p}_{j2},\ldots,\boldsymbol{p}_{jK_j}]\in\mathbb{R}^{d\times K_j}$ is the weight matrix to be determined, and $\xi_j=[\xi_{j1},\xi_{j2},\ldots,\xi_{jm}]^\top\in\mathbb{R}^{m\times 1}$ is the slack variable vector. Suppose $y_{ij}=c_{a_i}^j$, then $e_{ji}^a=1$ if $y_{ij}\ne c_a^j$ (i.e., $a_i\ne a$) and 0 otherwise. Furthermore, γ is a trade-off parameter. Let $\mathbf{P}=[\mathbf{P}_1,\mathbf{P}_2,\ldots,\mathbf{P}_q]$, the maximum margin-augmented feature vector $\boldsymbol{\Delta}_{\mathrm{SVM}}^{\boldsymbol{x}}$ for each instance $\boldsymbol{x}$ can be defined as follows:

$$\boldsymbol{\Delta}_{\mathrm{SVM}}^{\boldsymbol{x}}=\mathbf{P}^\top\boldsymbol{x} \tag{11}$$

Then, the MDC training set $\mathcal{D}$ can be transformed into:

$$\widetilde{\mathcal{D}}=\{(\boldsymbol{z}_i,\boldsymbol{y}_i)\mid 1\le i\le m\},\ \text{where}\ \ \boldsymbol{z}_i=[\boldsymbol{x}_i;\widetilde{\boldsymbol{x}}_i]$$

where $\widetilde{\boldsymbol{x}}_i=[\boldsymbol{\Delta}_{k\mathrm{NN1}}^{\boldsymbol{x}_i};\boldsymbol{\Delta}_{k\mathrm{NN2}}^{\boldsymbol{x}_i};\boldsymbol{\Delta}_{\mathrm{SVM}}^{\boldsymbol{x}_i}]$ denotes the augmented features of $\boldsymbol{x}_i$. Here, we reiterate that we make use of standard *k*NN, weighted *k*NN, and maximum margin techniques to generate the augmented feature only for the purpose of simplicity. The experiments in this paper mainly aims at validating the effectiveness of the selective feature augmentation strategy. In the future, it is interesting to further investigate synergizing multiple kinds of augmented features which are generated by making use of more advanced techniques such as deep learning[21].

Table 2 Experimental results (mean±std. deviation) of each MDC approach. In addition, the performance rank on each data set is also shown in the parentheses.

Data Set	SFAM	BR	CP	ECC	ESC	gMML	SEEM
	Hamming Score						
Flare1	.925±.035(2)	.925±.034(2)	.923±.033(7)	.926±.034(1)	.925±.035(2)	.925±.034(2)	.925±.032(2)
Enb	.865±.036(1)	.774±.023(3)	.764±.031(5)	.773±.034(4)	.754±.029(6)	.742±.027(7)	.777±.031(2)
WQplants	.666±.016(1)	.658±.014(3)	.649±.016(7)	.654±.016(5)	.653±.016(6)	.655±.015(4)	.661±.023(2)
WQanimals	.641±.012(1)	.631±.013(3)	.628±.013(7)	.629±.013(6)	.631±.014(3)	.630±.015(5)	.635±.015(2)
WQ	.653±.012(1)	.644±.011(3)	.625±.011(7)	.642±.012(5)	.642±.014(5)	.643±.013(4)	.646±.014(2)
BeLaE	.441±.013(1)	.427±.017(2)	.383±.023(7)	.424±.021(3)	.420±.022(4)	.417±.020(5)	.416±.020(6)
Voice	.947±.009(1)	.900±.012(3)	.898±.011(4)	.896±.012(6)	.897±.011(5)	.842±.009(7)	.910±.011(2)
TIC2000	.946±.003(1)	.915±.006(3)	.905±.006(6)	.915±.006(3)	.915±.006(3)	.895±.007(7)	.917±.005(2)
Adult	.724±.005(1)	.721±.004(2)	.709±.004(6)	.720±.003(4)	.710±.005(5)	.705±.004(7)	.721±.004(2)
Default	.671±.003(3)	.669±.003(5)	.669±.004(5)	.670±.003(4)	.672±.004(1)	.666±.004(7)	.672±.003(1)
	Exact Match						
Flare1	.824±.073(1)	.821±.075(4)	.817±.068(7)	.824±.073(1)	.824±.073(1)	.821±.075(4)	.818±.075(6)
Enb	.729±.071(1)	.548±.045(3)	.529±.063(5)	.546±.069(4)	.508±.057(6)	.483±.053(7)	.554±.063(2)
WQplants	.100±.037(1)	.092±.033(5)	.093±.031(3)	.092±.034(5)	.093±.036(3)	.092±.035(5)	.096±.034(2)
WQanimals	.058±.013(5)	.058±.017(5)	.065±.018(1)	.059±.017(4)	.064±.019(2)	.062±.023(3)	.049±.022(7)
WQ	.009±.006(1)	.005±.008(4)	.000±.000(7)	.005±.008(4)	.005±.008(4)	.006±.008(3)	.009±.006(1)
BeLaE	.028±.011(1)	.021±.008(7)	.026±.014(3)	.023±.010(5)	.027±.009(2)	.022±.009(6)	.026±.011(3)
Voice	.897±.018(1)	.809±.023(3)	.807±.021(4)	.802±.022(6)	.803±.019(5)	.699±.017(7)	.831±.020(2)
TIC2000	.846±.009(1)	.762±.017(3)	.739±.017(6)	.762±.017(3)	.762±.016(3)	.706±.018(7)	.770±.015(2)
Adult	.284±.010(5)	.275±.008(6)	.317±.010(1)	.287±.007(4)	.312±.011(2)	.230±.009(7)	.289±.010(3)
Default	.185±.006(4)	.181±.007(6)	.194±.008(1)	.185±.006(4)	.187±.007(3)	.177±.007(7)	.190±.009(2)
	Sub-Exact Match						
Flare1	.954±.039(5)	.957±.039(2)	.954±.039(5)	.957±.039(2)	.954±.039(5)	.957±.039(2)	.960±.033(1)
Enb	1.00±.000(1)	1.00±.000(1)	1.00±.000(1)	1.00±.000(1)	1.00±.000(1)	1.00±.000(1)	1.00±.000(1)
WQplants	.292±.035(1)	.286±.044(3)	.285±.052(5)	.285±.053(5)	.282±.049(7)	.286±.053(3)	.287±.042(2)
WQanimals	.246±.034(1)	.229±.030(5)	.232±.032(3)	.226±.026(7)	.231±.029(4)	.227±.033(6)	.241±.029(2)
WQ	.061±.022(1)	.047±.019(6)	.034±.017(7)	.048±.022(4)	.048±.019(4)	.049±.024(3)	.050±.025(2)
BeLaE	.137±.024(1)	.134±.025(2)	.117±.019(7)	.130±.025(3)	.128±.024(5)	.130±.020(3)	.125±.022(6)
Voice	.998±.003(1)	.991±.006(2)	.989±.006(5)	.989±.008(5)	.991±.007(2)	.985±.011(7)	.990±.006(4)
TIC2000	.993±.002(1)	.983±.003(4)	.978±.002(6)	.984±.003(2)	.984±.003(2)	.978±.003(6)	.982±.003(5)
Adult	.690±.007(1)	.685±.009(2)	.637±.007(7)	.679±.008(4)	.644±.007(6)	.669±.008(5)	.680±.006(3)
Default	.604±.007(1)	.601±.006(4)	.594±.008(6)	.600±.007(5)	.604±.008(1)	.593±.008(7)	.604±.007(1)

4.2 Experimental Results

Table 2 reports the detailed experimental results with the performance rank shown in the parentheses. Moreover, *Wilcoxon signed-ranks test*[32] (at 0.05 significance level) serves as the statistical tool to show whether SFAM achieves better performance against the compared approaches over the whole benchmark data sets, and the corresponding test results are summarized in Table 3.

According to the reported experimental results, we can make the following observations:

- Among all the 30 cases (10 data sets×3 evaluation metrics), SFAM ranks first in 24 cases, ranks second in 1 cases, ranks third in 1 cases, ranks fourth in 1 cases, ranks fifth in 5 cases, and never ranks last.
- BR solves the MDC problem by dealing with each dimension independently, where potential class dependencies are fully ignored. Although SFAM also induces classification models for each dimension independently, the class dependencies can be considered by the augmented features[20]. It is shown that SFAM achieves superior performance against BR in terms of each metrics, which reveals that considering class dependencies is important for learning MDC models.
- Both ECC and gMML explicitly consider the class dependencies, where a chaining order over class spaces or a Mahalanobis metric is employed to accomplish this task. It is shown that SFAM also achieves superior performance against ECC and gMML in terms of each metrics, which validates the superiority of SFAM's selective feature augmentation strategy.
- CP solves the MDC problem by dealing with all dimensions jointly via powerset transformation, which can be viewed as optimizing *Exact Match*. ESC and SEEM can be regarded as two improved versions of CP, where class spaces are grouped into super-classes according to conditional dependencies or each pair of class spaces are considered in the first level learning. It is shown that SFAM still achieves comparable performance against CP, ESC and SEEM in terms of *Exact Match*, and superior performance against CP, ESC and SEEM in terms of *Hamming Score* and *Sub-Exact Match*.

Table 3 Wilcoxon signed-ranks test for SFAM against each compared approach where the p-values at 0.05 significance level are also shown in the brackets.

SFAM against	Evaluation Metric		
	HS	EM	SEM
BR	**win**[3.91e-03]	**win**[3.91e-03]	**win**[7.81e-03]
CP	**win**[1.95e-03]	**tie**[2.75e-01]	**win**[7.81e-03]
ECC	**win**[3.91e-03]	**win**[3.91e-02]	**win**[7.81e-03]
ESC	**win**[7.81e-03]	**tie**[2.62e-01]	**win**[1.56e-02]
gMML	**win**[3.91e-03]	**win**[9.77e-03]	**win**[7.81e-03]
SEEM	**win**[9.77e-03]	**tie**[7.42e-02]	**win**[3.91e-02]

4.3 Further Analysis

1) Effectiveness of Algorithmic Design

In this paper, SFAM generates three kinds of augmented features according to (9), (10) and (11) respectively. To further investigate the effectiveness of SFAM's algorithmic design, we also compare SFAM with its three degenerated versions which generate either kind of augmented features. The one with discrete version of *k*NN-augmented features in (9) is denoted as DeV1, which is also known as the KRAM_d approach[20], the one with continuous version of *k*NN-augmented features in (10) is denoted as DeV2, which is also known as the KRAM_c approach[20], and the another one with maximum margin-augmented features in (11) is denoted as DeV3. It is worth noting that the baseline BR actually serves as another degenerated version without any kind of augmented features, whose experimental results have been reported and analyzed in Subsection 4.2.

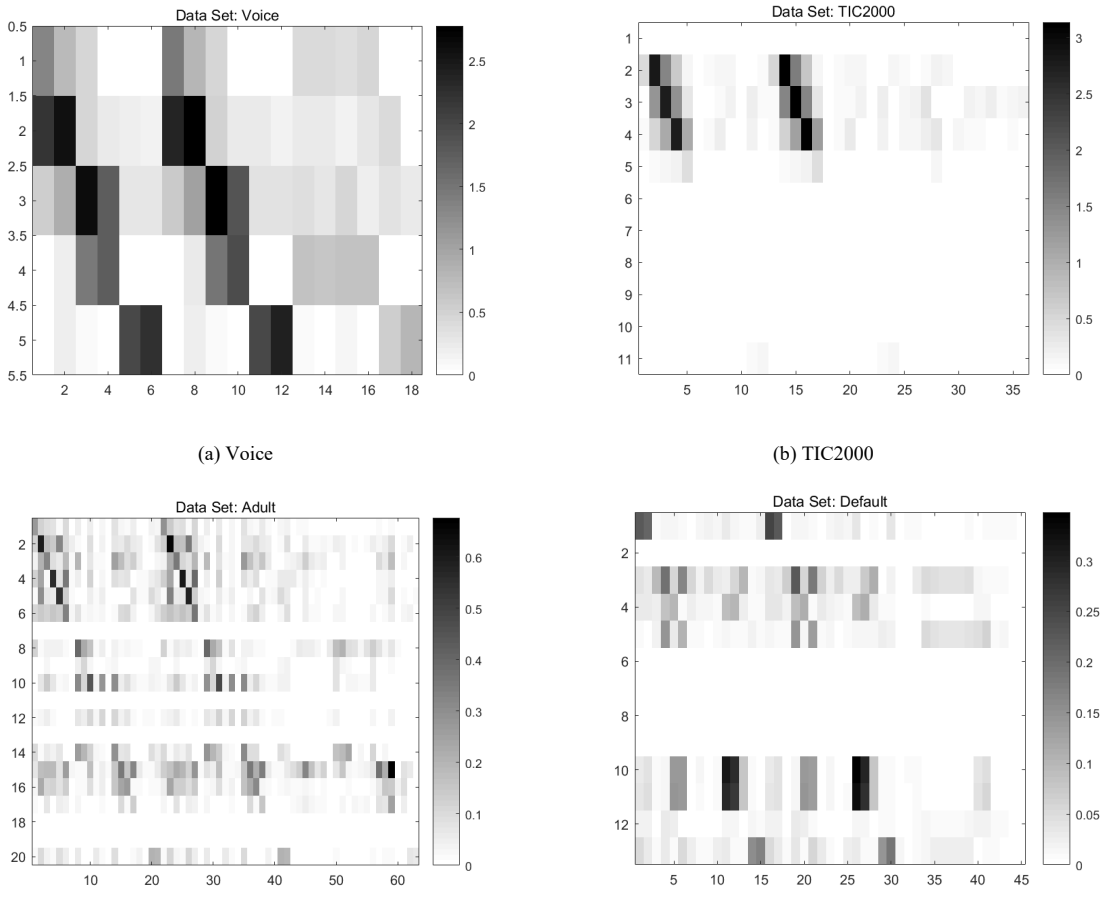
Table 4 Experimental results (mean±std. deviation) of SFAM and its two degenerated versions. In addition, the performance rank on each data set is also shown in the parentheses.

Data Set	Hamming Score			
	SFAM	DeV1	DeV2	DeV3
Flare1	.925±.035(1)	.924±.036(4)	.925±.035(1)	.925±.035(1)
Enb	.865±.036(1)	.852±.036(2)	.848±.043(3)	.807±.029(4)
WQpla.	.666±.016(1)	.664±.016(2)	.664±.016(2)	.659±.015(4)
WQani.	.641±.012(1)	.641±.015(1)	.641±.015(1)	.632±.013(4)
WQ	.653±.012(1)	.650±.011(3)	.652±.010(2)	.645±.012(4)
BeLaE	.441±.013(1)	.436±.008(2)	.434±.011(4)	.435±.019(3)
Voice	.947±.009(1)	.945±.010(3)	.946±.010(2)	.913±.010(4)
TIC2000	.946±.003(2)	.946±.003(2)	.947±.003(1)	.914±.006(4)
Adult	.724±.005(1)	.724±.004(1)	.724±.005(1)	.721±.004(4)
Default	.671±.003(1)	.671±.003(1)	.671±.003(1)	.669±.003(4)
Data Set	Exact Match			
	SFAM	DeV1	DeV2	DeV3
Flare1	.824±.073(1)	.821±.079(4)	.824±.073(1)	.824±.073(1)
Enb	.729±.071(1)	.705±.072(2)	.696±.086(3)	.613±.058(4)
WQpla.	.100±.037(1)	.097±.038(2)	.097±.037(2)	.095±.034(4)
WQani.	.058±.013(3)	.058±.011(3)	.061±.016(1)	.059±.018(2)
WQ	.009±.006(1)	.008±.005(2)	.008±.005(2)	.005±.008(4)
BeLaE	.028±.011(1)	.025±.007(2)	.024±.007(3)	.024±.008(3)
Voice	.897±.018(1)	.893±.020(3)	.894±.021(2)	.834±.018(4)
TIC2000	.846±.009(3)	.847±.008(2)	.849±.008(1)	.760±.018(4)
Adult	.284±.010(1)	.284±.009(1)	.284±.010(1)	.274±.008(4)
Default	.185±.006(2)	.185±.006(2)	.186±.006(1)	.181±.006(4)
Data Set	Sub-Exact Match			
	SFAM	DeV1	DeV2	DeV3
Flare1	.954±.039(1)	.954±.039(1)	.954±.039(1)	.954±.039(1)
Enb	1.00±.000(1)	1.00±.000(1)	1.00±.000(1)	1.00±.000(1)
WQpla.	.292±.035(3)	.294±.037(1)	.294±.036(1)	.289±.042(4)
WQani.	.246±.034(1)	.244±.031(2)	.243±.035(3)	.231±.032(4)
WQ	.061±.022(1)	.059±.019(2)	.058±.020(3)	.048±.020(4)
BeLaE	.137±.024(1)	.133±.021(4)	.134±.022(2)	.134±.023(2)
Voice	.998±.003(1)	.998±.003(1)	.998±.003(1)	.993±.006(4)
TIC2000	.993±.002(1)	.992±.002(3)	.993±.002(1)	.983±.003(4)
Adult	.690±.007(1)	.690±.007(1)	.689±.007(3)	.685±.008(4)
Default	.604±.007(1)	.604±.007(1)	.604±.006(1)	.601±.007(4)

Table 5 Wilcoxon signed-ranks test for SFAM against its two variants where the p-values at 0.05 significance level are also shown in the brackets.

SFAM against	Evaluation Metric		
	HS	EM	SEM
DeV1	**win**[3.13e-02]	**win**[4.69e-02]	**tie**[3.13e-01]
DeV2	**tie** [1.25e-01]	**tie** [3.52e-01]	**tie** [1.88e-01]
DeV3	**win**[3.91e-03]	**win**[7.81e-03]	**win**[7.81e-03]

Detailed experimental results are shown in Table 4. Table 5 summarizes the test results of *Wilcoxon signed-ranks test* (at 0.05 significance level). It is shown that SFAM achieves superior performance against DeV1 in terms of *Hamming Score* and *Exact Match*, and DeV3 in terms of all metrics. For DeV2, although SFAM achieves comparable performance against it in terms of all metrics, as shown in Table 4, among the 19 cases where the performance of SFAM is different with DeV2, there are 14 cases where the performance of SFAM is better than DeV2. These results clearly validate that SFAM can identify the pertinent and useful features from the three kinds of augmented features. Besides, it is shown that both DeV1 and DeV2 achieve similar results compared to SFAM, possible reason is that the two kinds of *k*NN-augmented features contain more useful discriminative information than maximum margin-augmented features. Nonetheless, SFAM is able to utilize all the available information in hand to achieve better generalization performance.



(a) Voice (b) TIC2000

(c) Adult (d) Default

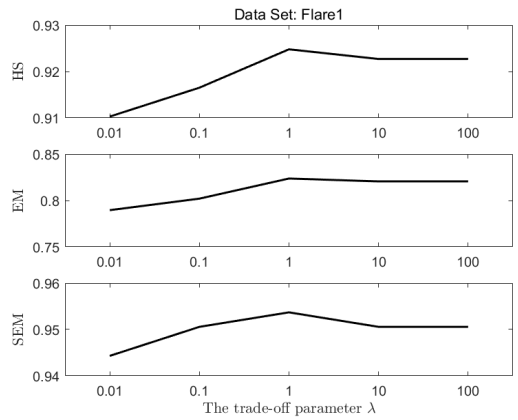
Fig. 3 The weight matrix (absolute value) of the learned model w.r.t. the ℓ_1 regularization

Furthermore, Fig.3 shows the weight matrix (absolute value) of the learned model w.r.t. the ℓ_1 regularization. Specifically, following the notations in Section III, Fig.3 shows the absolute value of weight matrix $\widetilde{\boldsymbol{\Theta}} = [\widetilde{\boldsymbol{\theta}}_{11}, \ldots, \widetilde{\boldsymbol{\theta}}_{1K_1}, \ldots, \widetilde{\boldsymbol{\theta}}_{q1}, \ldots, \widetilde{\boldsymbol{\theta}}_{qK_q}]^\top$ for data set *Voice*, *TIC2000*, *Adult*, and *Default*. For each figure, each row corresponds to the binary classification model of one class label, and the first third of all columns correspond to the discrete version of *k*NN-augmented features in (9), the middle third of all columns to the continuous version of *k*NN-augmented features in (10), and the last third of all columns to the maximum margin-augmented features in (11). It is shown that, for each third of all columns, the diagonal element usually takes the largest value in its corresponding row. Note that each element in all the three kinds of augmented features (i.e., each column in Fig.3) corresponds to one class label, and each binary classification model (i.e., each row in Fig.3) also corresponds to one class label. In other words, the largest value corresponds to the augmented feature w.r.t. its own class label. It is also shown that each

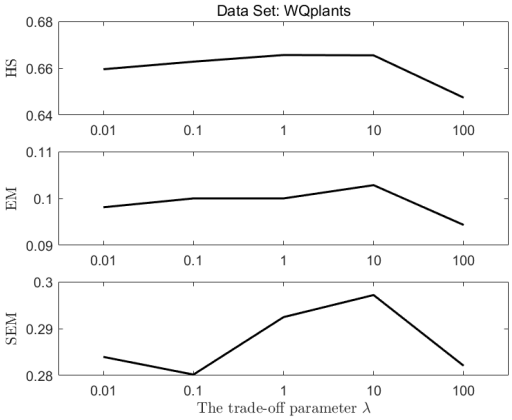
binary classification model is only related to part of augmented features, where the model weights w.r.t. the two kinds of *k*NN-augmented features are usually larger than the model weights w.r.t. the maximum margin-augmented features. This observation further supports the aforementioned conjecture that the two kinds of *k*NN-augmented features contain more useful discriminative information than maximum margin-augmented features. Besides, we can also observe that the model weights w.r.t. the ℓ_1 regularization for some binary classification models are almost all zero, which means that not all binary classification models rely on augmented features.

2) Parameter Sensitivity Analysis

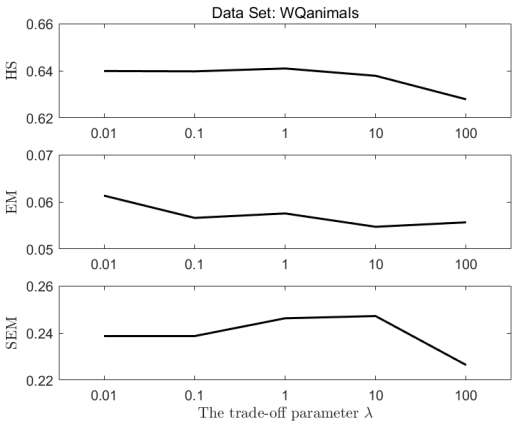
The regularized classification model (1) has one trade-off parameter λ. In this subsection, we investigate how the performance of SFAM changes with different values of λ. Fig.4 illustrates SFAM's performance fluctuation when λ ranges in {0.01, 0.1, 1, 10, 100} over data sets *Flare1*, *WQplants*, *WQanimals* and *BeLaE*. It is shown that the performance of SFAM degenerates with either small or large value of λ generally, and $\lambda = 1$ is usually a better choice. Therefore, we fix λ as 1 in all the previous comparative studies.



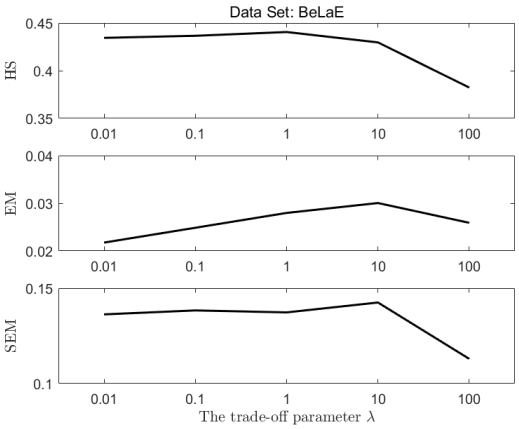
(a) Flare1



(b) WQplants



(c) WQanimals



(d) BeLaE

Fig. 4 Performance SFAM changes as λ ranges in {0.01, 0.1, 1, 10, 100}

5 Conclusions

Feature augmentation has been shown as an effective strategy for solving the MDC problem. Existing works only focus on how to generate better augmented features, while it might be beneficial to exploit multiple kinds of augmented features generated by making use of different techniques. This paper makes a first attempt towards how to synergize the discriminative information residing in multiple kinds of augmented features. Accordingly, a novel strategy named *selective feature augmentation* is proposed which assumes that only part of the augmented features is pertinent and useful for each dimension's model induction. Comparative studies clearly validate the effectiveness of the proposed strategy.

Current feature augmentation works simply concatenate the original and augmented features, though the proposed SFAM has treated them differently via different regularization terms. In fact, the original and augmented features (even different kinds of augmented features) can be regarded as features from different views[33]. In the future, other ensemble strategies borrowing from multi-view learning can also be used instead of merely using the concatenation operation. Besides, this paper only generates two simple kinds of augmented features to

validate the proposed selective feature augmentation strategy, it is also deserved to investigate generating more kinds of augmented features.

References

[1] D. Xu, Y. Shi, I. W. Tsang, Y. Ong, C. Gong, and X. Shen. Survey on multi-output learning. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 37, no. 7, pp. 2409–2429, 2020.

[2] J. Read, C. Bielza, and P. Larranaga. Multi-dimensional classification with super-classes. *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1720–1733, 2014.

[3] B.-B. Jia and M.-L. Zhang. Maximum margin multi-dimensional classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2021, in press

[4] J. D. Rodrıguez, A. Perez, D. Arteta, D. Tejedor, and J. A. Lozano. Using multidimensional Bayesian network classifiers to assist the treatment of multiple sclerosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1705–1715, 2012.

[5] H. Borchani, C. Bielza, C. Toro, and P. Larranaga. Predicting human immunodeficiency virus inhibitors using multi-dimensional Bayesian network classifiers. *Artificial Intelligence in Medicine*, vol. 57, no. 3, pp. 219–229, 2013.

[6] H. Shatkay, F. Pan, A. Rzhetsky, and W. J. Wilbur. Multi-dimensional classification of biomedical text: Toward automated, practical provision of high-utility text to diverse users. *Bioinformatics*, vol. 24, no. 18, pp. 2086–2093, 2008.

[7] F. Serafino, G. Pio, M. Ceci, and D. Malerba. Hierarchical multidimensional classification of web documents with multiwebclass. In *Proceedings of the 18th International Conference on Discovery Science*, Banff, AB, Canada, 2015, pp. 236–250.

[8] Z. Lian, Y. Li, J. Tao, J. Huang, and M. Niu. Expression Analysis Based on Face Regions in Real-world Conditions. *International Journal of Automation and Computing*, vol. 17, no. 1, pp. 96-107, 2020.

[9] Z. He, L. Zhang, F. Liu. DiscoStyle: Multi-level Logistic Ranking for Personalized Image Style Preference Inference. *International Journal of Automation and Computing*, vol. 17, no. 5, pp. 637-651, 2020.

[10] Y. Zhang, X. Shi, S. Mi, X. Yang. Image captioning with transformer and knowledge graph. *Pattern Recognition Letters*, vol. 143, pp. 43-49, 2021.

[11] A. H. Al Muktadir, T. Miyazawa, P. Martinez-Julia, H. Harai, and V. P. Kafle. Multi-target classification based automatic virtual resource allocation scheme. *IEICE Transactions on Information and Systems*, vol. 102, no. 5, pp. 898–909, 2019.

[12] J. Arias, J. A. Gamez, T. D. Nielsen, and J. M. Puerta. A scalable pairwise class interaction framework for multidimensional classification. *International Journal of Approximate Reasoning*, vol. 68, pp. 194–210, 2016.

[13] B.-B. Jia and M.-L. Zhang. Multi-dimensional classification via stacked dependency exploitation. Science China Information Sciences, vol. 63, no. 12, 2020, Article 222102.

[14] B.-B. Jia and M.-L. Zhang. MD-KNN: An instance-based approach for multi-dimensional classification. In *Proceedings of the 25th International Conference on Pattern Recognition*, Milan, Italy, 2021, pp. 126–133.

[15] J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larranaga. Bayesian chain classifiers for multidimensional classification. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, Barcelona, Spain, 2011, pp. 2192–2197.

[16] J. Read, L. Martino, and D. Luengo. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, vol. 47, no. 3, pp. 1535–1546, 2014.

[17] C. Bielza, G. Li, and P. Larranaga. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, vol. 52, no. 6, pp. 705–727, 2011.

[18] J. H. Bolt and L. C. van der Gaag. Balanced sensitivity functions for tuning multi-dimensional Bayesian network classifiers. *International Journal of Approximate Reasoning*, vol. 80, pp. 361–376, 2017.

[19] M. Benjumeda, C. Bielza, and P. Larranaga. Tractability of most probable explanations in multidimensional Bayesian network classifiers. *International Journal of Approximate Reasoning*, vol. 93, pp. 74–87, 2018.

[20] B.-B. Jia and M.-L. Zhang. Multi-dimensional classification via *k*NN feature augmentation. *Pattern Recognition*, vol. 106, 2020, Article 107423.

[21] H. Wang, C. Chen, W. Liu, K. Chen, T. Hu, and G. Chen. Incorporating label embedding and feature augmentation for multi-dimensional classification. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, New York, NY, USA, 2020, pp. 6178–6185.

[22] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.

[23] E. Gibaja and S. Ventura. A tutorial on multilabel learning. *ACM Computing Surveys*, vol. 47, no. 3, 2015, Article 52.

[24] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng. Binary relevance for multi-label learning: An overview. *Frontiers of Computer Science*, vol. 12, no. 2, pp. 191–202, 2018.

[25] S. Gil-Begue, C. Bielza, and P. Larranaga. Multi-dimensional Bayesian network classifiers: A survey. *Artificial Intelligence Review*, vol. 54, pp. 519–559, 2021.

[26] J. Huang, G. Li, Q. Huang, and X. Wu. Learning label-specific features and class-dependent labels for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3309–3323, 2016.

[27] H. H. Bauschke, J. Bolte, and M. Teboulle. A descent lemma beyond Lipschitz gradient continuity: First-order methods revisited and applications. *Mathematics of Operations Research*, vol. 42, no. 2, pp. 330–348, 2017.

[28] A. Beck and M. Teboulle. A fast iterative shrinkage thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[29] Z. Ma and S. Chen. Multi-dimensional classification via a metric approach. *Neurocomputing*, vol. 275, pp. 1121–1131, 2018.

[30] R. Fan, K. Chang, C. Hsieh, X. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[31] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines.

Journal of Machine Learning Research, vol. 2, pp. 265–292, 2001.

[32] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[33] J. Zhao, X. Xie, X. Xu, and S. Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, vol. 38, pp.43–54, 2017.