MD-KNN: An Instance-based Approach for Multi-Dimensional Classification

Bin-Bin Jia*†‡ and Min-Ling Zhang*‡
*School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
†College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou 730050, China
‡Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China
Email: jiabb@seu.edu.cn, zhangml@seu.edu.cn

***Abstract*—Multi-dimensional classification (MDC) deals with the problem where each instance is associated with multiple class variables, each of which corresponds to a specific class space. One of the mainstream solutions for MDC is to adapt traditional machine learning techniques to deal with MDC data. In this paper, a first attempt towards adapting instance-based techniques for MDC is investigated, and a new approach named MD-KNN is proposed. Specifically, MD-KNN identifies unseen instance's k nearest neighbors and obtains its corresponding kNN counting statistics for each class space, based on which maximum a posteriori (MAP) inference is made for each pair of class spaces. After that, the class label w.r.t. each class space is determined by synergizing predictions from the learned classifiers via consulting empirical kNN accuracy. Comparative studies over ten benchmark data sets clearly validate MD-KNN's effectiveness.**

I. INTRODUCTION

In traditional multi-class classification, each example is represented by a single instance and associated with a single class variable. However, there are many learning tasks where the simplifying assumption does not fit well as real-world objects usually have rich semantics and should be classified along different dimensions [1], [2], [3], [4], [5], [6], [7], [8], [9]. For example, in news categorization, we should simultaneously classify a news document from the `topic` dimension (with possible classes *Sci&Tech*, *social*, *politics*, *sports*, etc.), from the `mood` dimension (with possible classes *good news*, *neutral news*, *bad news*), and from the `zone` dimension (with possible classes *domestic*, *intra-/inter-continental*, etc.), etc. To deal with this kind of problems, one natural solution is to associate multiple class variables with the object to explicitly express its semantics, which yields the multi-dimensional classification framework [10], [11], [12]. Compared to multi-class classification, each MDC example is also represented by a single instance while associated with multiple class variables. Here, each class variable corresponds to one specific class space characterizing the object's semantics from one dimension.

Formally speaking, denote the input (feature) space by $\mathcal{X} = \mathbb{R}^d$, and the output space by $\mathcal{Y} = C_1 \times C_2 \times \cdots \times C_q$. Here, the output space corresponds to the Cartesian product of q class spaces and each class space C_j $(1 \leq j \leq q)$ consists of K_j possible class labels, i.e., $C_j = \{c_1^j, c_2^j, \ldots, c_{K_j}^j\}$. Given the MDC training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \mid 1 \leq i \leq m\}$ with m training examples, for each example $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}$, $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^\top \in \mathcal{X}$ is a d-dimensional feature vector and $\boldsymbol{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iq}]^\top \in \mathcal{Y}$ is the class vector associated with $\boldsymbol{x}_i$, where each component y_{ij} is one possible class label in C_j, i.e., $y_{ij} \in C_j$. The task of multi-dimensional classification is to learn a mapping function $f : \mathcal{X} \mapsto \mathcal{Y}$ from $\mathcal{D}$ which can predict a class vector $f(\boldsymbol{x}_*) \in \mathcal{Y}$ for unseen instance $\boldsymbol{x}_*$.

To solve the MDC problem, one main strategy is to adapt popular learning techniques to directly deal with MDC data. In [13], [14], [15], [16], [17], Bayesian network is adapted for MDC where three subgraphs are learned specifically to model the dependencies among feature variables, class variables, and feature-class variables. In [11], a metric learning approach is designed for MDC where a Mahalanobis distance metric is learned which can make the distance between linear regression outputs of one example and its ground-truth class label vector closer. In [18], maximum margin technique is adapted for MDC where not only classification margins on individual class variable are maximized, but also dependencies among class variables are considered.

Existing approaches focus on adapting parametric learning techniques to deal with MDC problem, while instance-based techniques have been shown as effective solutions to learn from objects with rich semantics [19], [20], [21], [22], [23], [24]. In this paper, we make a first attempt to adapt instance-based techniques for multi-dimensional classification, and propose a novel approach named MD-KNN, i.e., *Multi-Dimensional k-Nearest Neighbor*. Firstly, MD-KNN identifies unseen instance's k nearest neighbors in training set, and obtains its kNN counting statistics w.r.t. each class space. Then, for each pair of class spaces, maximum a posteriori (MAP) inference is made based on the obtained kNN counting statistics w.r.t. both class spaces. Finally, the class label w.r.t. each class space is determined by synergizing predictions from corresponding pairwise class spaces via consulting empirical kNN accuracy. Experimental results over ten benchmark data sets show that MD-KNN achieves superior performance against other state-of-the-art MDC approaches.

The rest of this paper is organized as follows. Firstly, related works on MDC are briefly discussed. Secondly, technical details of the proposed approach are introduced. Thirdly, experimental results of comparative studies are reported. Finally, we conclude this paper.

II. Related Work

The most related learning framework to multi-dimensional classification is multi-label classification (MLC) [25], [26], [27], and both of them can be regarded as one possible instantiation of multi-output learning [28] where each object is associated with multiple output variables. Intuitively, the difference between MLC and MDC lies in the type of output variables, which are binary-valued for MLC while discrete-valued for MDC. However, the essential difference between MLC and MDC lies in whether the class space is *homogeneous* or *heterogenous*. MLC usually assumes homogeneous class space where each label represents the relevancy of one concept, while MDC assumes heterogenous class space where each class variable corresponds to one specific class space which characterizes the object's semantics along one specific dimension [10], [12], [29].

Due to powerful modeling capabilities of probabilistic graphical model, Bayesian network has been adapted to solve MDC problems [30], [31], [32], [13]. Specifically, a total of three subgraphs are learned, i.e., class subgraph, feature subgraph, and bridge subgraph, which respectively model the dependencies among feature variables, class variables, and feature-class variables. Different structure types of subgraphs result in different solutions which form a family of MDC approaches called Multi-dimensional Bayesian network Classifier (MBC). Recent works further explore different MBC structures or efficient structure learning strategies [14], [15], [16], [17]. However, structure learning for Bayesian network is still challenging and only data set with discrete features can be tackled. In [11], the proposed gMML approach alternately learns linear regression models for each class label as well as a Mahalanobis distance metric to solve MDC problem effectively, where the Mahalanobis distance metric can make the distance between linear regression outputs of one example and its ground-truth class label vector closer. In [18], the proposed M^{3}MDC approach adapts maximum margin technique to solve MDC problem, which not only maximizes classification margins on individual class variable via one-vs-one decomposition, but also considers dependencies among class variables via covariance regularization.

Other than these approaches which adapt existing learning techniques to deal with MDC data, there are also some works which solve the MDC problem by transforming it into other well-established learning frameworks. By focusing on each class space one by one, the MDC problem can be intuitively transformed into a number of independent or a chain of successive multi-class classification problems [33], [34], one per class space, while by focusing on all class spaces at the same time, the MDC problem can be intuitively transformed into a single multi-class classification problem. By grouping the class spaces into super-classes [10], or enriching the original feature space with augmented features [12], [35], the MDC problem can be transformed into a new MDC problem, which is expected to facilitate follow-up learning procedure after tailored transformations.

III. The MD-kNN Approach

For any instance $\boldsymbol{x}$, let $\mathcal{N}_k(\boldsymbol{x}) = \{i_t \mid 1 \le t \le k\}$ denote the set of indices for $\boldsymbol{x}$'s k nearest neighbors identified in training set $\mathcal{D}$. Here, Euclidean distance is used to measure the similarities between two instances. Then, the following *k*NN counting statistics $\boldsymbol{\delta}_j^{\boldsymbol{x}} = [\delta_{j1}^{\boldsymbol{x}}, \delta_{j2}^{\boldsymbol{x}}, \ldots, \delta_{jK_j}^{\boldsymbol{x}}]$ can be defined for the j-th class space:

$$\delta_{ja}^{\boldsymbol{x}} = \sum_{i_t \in \mathcal{N}_k(\boldsymbol{x})} \mathbb{I}(y_{i_t j}, c_a^j) \quad (1 \le a \le K_j, 1 \le j \le q) \tag{1}$$

Here, $\boldsymbol{y}_{i_t} = [y_{i_t 1}, y_{i_t 2}, \ldots, y_{i_t q}]^\top$ corresponds to the class vector of the neighboring MDC example $\boldsymbol{x}_{i_t}$ of $\boldsymbol{x}$. $\mathbb{I}(\pi_1, \pi_2)$ returns 1 if π_1 is identical with π_2 and 0 otherwise. Therefore, $\delta_{ja}^{\boldsymbol{x}}$ records the number of $\boldsymbol{x}$'s neighboring MDC examples which has class label c_a^j in the j-th class space. According to Eq.(1), it is easy to verify that $\sum_{a=1}^{K_j} \delta_{ja}^{\boldsymbol{x}} = k$ holds.

The task of MDC is to learn a mapping function from $\mathcal{X}$ to $\mathcal{Y} = C_1 \times C_2 \times \cdots \times C_q$, rather than q independent mapping functions from $\mathcal{X}$ to C_j $(1 \le j \le q)$ respectively. This means that we should consider dependencies among class spaces $C_1, C_2, \ldots, C_q$ when inducing an MDC predictive model. Generally speaking, due to limited number of examples in training set, dependencies among fewer class spaces can be modeled more reliably than dependencies among many class spaces. Therefore, MD-kNN only aims at considering second-order (or pairwise) class dependencies. Specifically, MD-kNN aims at predicting the class vector of unseen instance $\boldsymbol{x}_*$ by utilizing MAP rule based on its *k*NN counting statistics. For the r-th and s-th class space $(1 \le r < s \le q)$, the class label $c_{a_r}^r \in C_r$ and $c_{a_s}^s \in C_s$ with maximum posteriori probability $\mathbb{P}(\phi_{rs}(c_{a_r}^r, c_{a_s}^s) \mid \boldsymbol{x}_*^{rs})$ will be returned as the predicted labels for $\boldsymbol{x}_*$'s r-th and s-th class space. Here, $\phi_{rs}(\cdot,\cdot)$ is some injective function from Cartesian product of C_r and C_s to natural numbers, and $\boldsymbol{x}_*^{rs} = [\boldsymbol{\delta}_r^{\boldsymbol{x}_*}, \boldsymbol{\delta}_s^{\boldsymbol{x}_*}]$. Intuitively, the posteriori probability can be *explicitly* estimated from training set $\mathcal{D}$ as follows:

$$\begin{aligned} &\mathbb{P}\left(\phi_{rs}(c_{a_r}^r, c_{a_s}^s) \mid \boldsymbol{x}_*^{rs}\right) \\ &= \frac{\sum_{i=1}^{m} \mathbb{I}(\phi_{rs}(y_{ir}, y_{is}), \phi_{rs}(c_{a_r}^r, c_{a_s}^s)) \cdot \mathbb{I}(\boldsymbol{x}_i^{rs}, \boldsymbol{x}_*^{rs})}{\sum_{i=1}^{m} \mathbb{I}(\boldsymbol{x}_i^{rs}, \boldsymbol{x}_*^{rs})} \end{aligned} \tag{2}$$

where $\boldsymbol{x}_i^{rs} = [\boldsymbol{\delta}_r^{\boldsymbol{x}_i}, \boldsymbol{\delta}_s^{\boldsymbol{x}_i}]$. However, due to the limited number of training examples and large number of possible different $\boldsymbol{x}_*^{rs}$, there are usually few $\boldsymbol{x}_i^{rs}$s in training set $\mathcal{D}$ which are identical with $\boldsymbol{x}_*^{rs}$, i.e, the denominator in Eq.(2) is usually very small which leads to that Eq.(2) can't work properly. Specifically, denote the number of possible different $\boldsymbol{x}_*^{rs}$ by $N(k, K_r, K_s)$, we have

$$N(k, K_r, K_s) = N(k, K_r) \cdot N(k, K_s) \tag{3}$$

where

$$N(u, v) = \begin{cases} u + 1, & \text{when } v = 2 \\ \sum_{i=0}^{u} N(i, v-1), & \text{when } v > 2 \end{cases}$$

In this paper, MD-KNN uses an alternative way where $\mathbb{P}\left(\phi_{rs}(c^r_{a_r}, c^s_{a_s}) \mid \boldsymbol{x}^{rs}_*\right)$ is estimated *implicitly* via a classifier g_{rs}, which is trained over the following multi-class data set:

$$\mathcal{D}^{\mathrm{MAP}}_{rs} = \{(\boldsymbol{x}^{rs}_i, \phi_{rs}(y_{ir}, y_{is})) \mid 1 \le i \le m\} \tag{4}$$

i.e., $g_{rs} = \mathcal{M}(\mathcal{D}^{\mathrm{MAP}}_{rs})$. Here, $\mathcal{M}$ corresponds to the employed multi-class training algorithm. According to the definition of $\phi_{rs}(\cdot,\cdot)$, the set of new classes in $\mathcal{D}^{\mathrm{MAP}}_{rs}$ corresponds to $\Phi(\mathcal{D}^{\mathrm{MAP}}_{rs}) = \{\phi_{rs}(y_{ir}, y_{is}) \mid 1 \le i \le m\}$. Denote the composition of functions ϕ^{-1}_{rs} and g_{rs} by h_{rs}, i.e., $h_{rs} = \phi^{-1}_{rs} \circ g_{rs}$, where $\phi^{-1}_{rs}(\cdot)$ is the inverse function of $\phi_{rs}(\cdot,\cdot)$. For unseen instance $\boldsymbol{x}_*$, based on its *k*NN counting statistics $\boldsymbol{x}^{rs}_* = [\boldsymbol{\delta}^{\boldsymbol{x}_*}_r, \boldsymbol{\delta}^{\boldsymbol{x}_*}_s]$, its class label w.r.t. r-th and s-th class space can be predicted by h_{rs}, i.e., $[y^{rs}_{*r}, y^{rs}_{*s}] = h_{rs}(\boldsymbol{x}^{rs}_*) = \phi^{-1}_{rs}(g_{rs}(\boldsymbol{x}^{rs}_*))$. It is easy to know that there are a total of $\binom{q}{2}$ different $\mathcal{D}^{\mathrm{MAP}}_{rs}$ and h_{rs}, based on which we can obtain $q-1$ predictive outputs for each class space. Denote the $q-1$ predictive outputs w.r.t. the j-th class space by $\boldsymbol{y}^{\boldsymbol{x}_*}_j$ $(1 \le j \le q)$:

$$\boldsymbol{y}^{\boldsymbol{x}_*}_j = \left[y^{\boldsymbol{x}_*}_j(1), y^{\boldsymbol{x}_*}_j(2), \ldots, y^{\boldsymbol{x}_*}_j(q-1)\right]^\top \tag{5}$$

where

$$y^{\boldsymbol{x}_*}_j(i) = \begin{cases} y^{ij}_{*j}, & \text{when } 1 \le i \le j-1 \\ y^{j(i+1)}_{*j}, & \text{when } j \le i \le q-1 \end{cases}$$

Note that each element in $\boldsymbol{y}^{\boldsymbol{x}_*}_j$ is obtained by one different classifier. Intuitively, the predictive ability of one classifier might vary when it is employed to classify different examples. Therefore, we aim to choose the element in $\boldsymbol{y}^{\boldsymbol{x}_*}_j$ which corresponds to the classifier with best ability in classifying the unseen instance $\boldsymbol{x}_*$. Specifically, for classifier h_{rs}, we firstly obtain all the corresponding *k*NN counting statistics $\boldsymbol{x}^{rs}_{i_t}$ of $\boldsymbol{x}_*$'s k nearest neighbors, where $1 \le t \le k$, $i_t \in \mathcal{N}_k(\boldsymbol{x}_*)$, and $\boldsymbol{x}^{rs}_{i_t} = [\boldsymbol{\delta}^{\boldsymbol{x}_{i_t}}_r, \boldsymbol{\delta}^{\boldsymbol{x}_{i_t}}_s]$. Then, predictive outputs can be returned for each neighboring example, i.e., $[\hat{y}^{rs}_{i_t r}, \hat{y}^{rs}_{i_t s}] = h_{rs}(\boldsymbol{x}^{rs}_{i_t})$. Finally, we can estimate h_{rs}'s predictive accuracy $\eta^{rs}_r(\boldsymbol{x}_*)$ and $\eta^{rs}_s(\boldsymbol{x}_*)$ w.r.t. the r-th and s-th class space respectively:

$$\begin{aligned} \eta^{rs}_r(\boldsymbol{x}_*) &= \tfrac{1}{k}\textstyle\sum_{t=1}^{k} \mathbb{I}(\hat{y}^{rs}_{i_t r}, y_{i_t r}) \\ \eta^{rs}_s(\boldsymbol{x}_*) &= \tfrac{1}{k}\textstyle\sum_{t=1}^{k} \mathbb{I}(\hat{y}^{rs}_{i_t s}, y_{i_t s}) \end{aligned} \tag{6}$$

The estimated two *k*NN accuracies can be approximately regarded as h_{rs}'s ability in classifying the r-th and s-th class space of $\boldsymbol{x}_*$. Denote the $q-1$ predictive accuracies corresponding to each element in $\boldsymbol{y}^{\boldsymbol{x}_*}_j$ by $\boldsymbol{\eta}^{\boldsymbol{x}_*}_j$:

$$\boldsymbol{\eta}^{\boldsymbol{x}_*}_j = \left[\eta^{\boldsymbol{x}_*}_j(1), \eta^{\boldsymbol{x}_*}_j(2), \ldots, \eta^{\boldsymbol{x}_*}_j(q-1)\right]^\top \tag{7}$$

where

$$\eta^{\boldsymbol{x}_*}_j(i) = \begin{cases} \eta^{ij}_j(\boldsymbol{x}_*), & \text{when } 1 \le i \le j-1 \\ \eta^{j(i+1)}_j(\boldsymbol{x}_*), & \text{when } j \le i \le q-1 \end{cases}$$

The final predicted class vector of $\boldsymbol{x}_*$, denoted by $\boldsymbol{y}_* = [y_{*1}, y_{*2}, \ldots, y_{*q}]^\top$, can be returned as follows:

$$y_{*j} = y^{\boldsymbol{x}_*}_j(I), \text{ where } I = \mathop{\arg\max}_{1 \le i \le q-1} \eta^{\boldsymbol{x}_*}_j(i) \tag{8}$$

Algorithm 1 The pseudo-code of MD-KNN

Input: The MDC training set $\mathcal{D}$, the number of nearest neighbors considered *k*, the multi-class training algorithm $\mathcal{M}$, the unseen instance $\boldsymbol{x}_*$;
Output: The predicted class vector $\boldsymbol{y}_*$ for $\boldsymbol{x}_*$;

1: Initialize $\mathcal{D}^{\mathrm{MAP}}_{rs}$ $(1 \le r < s \le q)$ as empty set;
2: **for** $i = 1$ to m **do**
3: Identify $\boldsymbol{x}_i$'s k nearest neighbors in $\mathcal{D}$ and store their indices in $\mathcal{N}_k(\boldsymbol{x}_i)$;
4: **for** $r = 1$ to $q-1$ **do**
5: **for** $s = r+1$ to q **do**
6: Obtain vector $\boldsymbol{\delta}^{\boldsymbol{x}_i}_r$ and $\boldsymbol{\delta}^{\boldsymbol{x}_i}_s$ according to Eq.(1);
7: $\mathcal{D}^{\mathrm{MAP}}_{rs} = \mathcal{D}^{\mathrm{MAP}}_{rs} \cup (\boldsymbol{x}^{rs}_i, \phi_{rs}(y_{ir}, y_{is}))$, where $\boldsymbol{x}^{rs}_i = [\boldsymbol{\delta}^{\boldsymbol{x}_i}_r, \boldsymbol{\delta}^{\boldsymbol{x}_i}_s]$;
8: **end for**
9: **end for**
10: **end for**
11: **for** $r = 1$ to $q-1$ **do**
12: **for** $s = r+1$ to q **do**
13: Train g_{rs} over $\mathcal{D}^{\mathrm{MAP}}_{rs}$, i.e., $g_{rs} = \mathcal{M}(\mathcal{D}^{\mathrm{MAP}}_{rs})$;
14: **end for**
15: **end for**
16: Identify $\boldsymbol{x}_*$'s k nearest neighbors in $\mathcal{D}$ and store their indices in $\mathcal{N}_k(\boldsymbol{x}^*)$;
17: Obtain $\boldsymbol{x}_*$'s q *k*NN counting statistics $\boldsymbol{\delta}^{\boldsymbol{x}_*}_j$ $(1 \le j \le q)$ according to Eq.(1);
18: Initialize $\boldsymbol{y}^{\boldsymbol{x}_*}_j, \boldsymbol{\eta}^{\boldsymbol{x}_*}_j$ $(1 \le j \le q)$ as empty vector;
19: **for** $r = 1$ to $q-1$ **do**
20: **for** $s = r+1$ to q **do**
21: Obtain $\eta^{rs}_r(\boldsymbol{x}_*)$ and $\eta^{rs}_s(\boldsymbol{x}_*)$ according to Eq.(6);
22: $\boldsymbol{\eta}^{\boldsymbol{x}_*}_r = [\boldsymbol{\eta}^{\boldsymbol{x}_*}_r,\ \eta^{rs}_r(\boldsymbol{x}_*)], \boldsymbol{\eta}^{\boldsymbol{x}_*}_s = [\boldsymbol{\eta}^{\boldsymbol{x}_*}_s, \eta^{rs}_s(\boldsymbol{x}_*)]$;
23: $[y^{rs}_{*r}, y^{rs}_{*s}] = h_{rs}(\boldsymbol{x}^{rs}_*)$, where $\boldsymbol{x}^{rs}_* = [\boldsymbol{\delta}^{\boldsymbol{x}_*}_r, \boldsymbol{\delta}^{\boldsymbol{x}_*}_s]$;
24: $\boldsymbol{y}^{\boldsymbol{x}_*}_r = [\boldsymbol{y}^{\boldsymbol{x}_*}_r,\ y^{rs}_{*r}], \boldsymbol{y}^{\boldsymbol{x}_*}_s = [\boldsymbol{y}^{\boldsymbol{x}_*}_s, y^{rs}_{*s}]$;
25: **end for**
26: **end for**
27: **for** $j = 1$ to q **do**
28: Predict y_{*j} according to Eq.(8);
29: **end for**
30: Return $\boldsymbol{y}_* = [y_{*1}, \ldots, y_{*q}]$.

In summary, Algorithm 1 presents the complete procedure of MD-KNN. Firstly, a total of $\binom{q}{2}$ different $\mathcal{D}^{\mathrm{MAP}}_{rs}$ are generated based on training set $\mathcal{D}$ (steps 1-10), over which a total of $\binom{q}{2}$ classifiers are trained respectively for each pair of class spaces (steps 11-15). After that, for each class space pair, the corresponding two empirical *k*NN accuracies and predictive outputs for unseen instance $\boldsymbol{x}_*$ are obtained (steps 16-26). Finally, for each class space of $\boldsymbol{x}_*$, the predicted class label is returned by synergizing $q-1$ candidates according to the corresponding empirical *k*NN accuracies (steps 27-30).

Computational complexity. For the employed multi-class training algorithm $\mathcal{M}$, let $\mathcal{F}(m,d,N)$ and $\mathcal{F}'(m,d,N)$ denote the training and testing complexity of $\mathcal{M}$, where m, d, N corresponds to the number of examples, number of features

and number of class labels respectively. According to Algorithm 1, the training complexity of MD-KNN corresponds to $\mathcal{O}(m\cdot(m+q^2\cdot K)+q^2\cdot\mathcal{F}(m,2K,K^2))$, and the testing complexity corresponds to $\mathcal{O}(m+q\cdot K+q^2\cdot\mathcal{F}'(k,2K,K^2)+q)$. Here, K represents the maximum number of class labels in each class space, i.e., $K=\max\{K_1,K_2,\ldots,K_q\}$. Note that q represents the number of class spaces (dimensions), each of which corresponds to one semantic space. Generally, it is not reasonable to assume too many semantic spaces and the number of class spaces in each MDC data set is at moderate size. Besides, both steps 4-9, steps 11-15 in training phase and steps 19-26 in testing phase can be run in parallel if there are enough computing resources.

IV. EXPERIMENTS

A. Experimental Setup

1) Data Sets: In this paper, a total of ten real-world MDC data sets are collected for comparative studies. To the best of our knowledge, seven of these data set are firstly employed in MDC researches [10], [11], [12], [13], [18], [35]. Table I summarizes the detailed characteristics of these data sets, including *number of examples* (#Exam.), *number of class spaces* (#Dim.), *number of class labels per class space* (#Labels/Dim.),[1] and *number of features* (#Features).

Specifically, `Flare1`, `WaterQuality`, `Scm20d`, `Rf1`, `Scm1d` are adapted from multi-target regression tasks,[2] `Thyroid`, `Adult` are adapted from UCI data sets,[3] and `Pain`, `Fera`, `Disfa` are adapted from copula ordinal regression tasks [36], [37].[4]

2) Evaluation Metrics: In this paper, a total of three evaluation metrics are used to measure the generalization performance of MDC approaches, i.e., *Hamming Score* (HS), *Exact Match* (EM) and *Sub-Exact Match* (SEM). Specifically, denote the test set by $\mathcal{S}=\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\mid 1\le i\le p\}$, where $\boldsymbol{y}_i=[y_{i1},y_{i2},\ldots,y_{iq}]^\top$ is the ground-truth class vector associated with $\boldsymbol{x}_i$. Denote the MDC model to be evaluated by $f:\mathcal{X}\mapsto\mathcal{Y}$, the predictive class vector for $\boldsymbol{x}_i$ is $\hat{\boldsymbol{y}}_i=f(\boldsymbol{x}_i)=[\hat{y}_{i1},\hat{y}_{i2},\ldots,\hat{y}_{iq}]^\top$, then the number of class spaces predicted correctly can be obtained, i.e., $r^{(i)}=\sum_{j=1}^{q}[\![y_{ij}=\hat{y}_{ij}]\!]$. Here, the predicate $[\![\pi]\!]$ returns 1 if π holds and 0 otherwise. The three metrics can be defined as follows:

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

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

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

[1]If all class spaces have the same number of class labels, then only this number is recorded; Otherwise, the number of class labels in each class space is recorded in turn.

[2]http://mulan.sourceforge.net/datasets-mtr.html

[3]http://archive.ics.uci.edu/ml/index.php

[4]https://github.com/RWalecki/copula_ordinal_regression/

TABLE I
CHARACTERISTICS OF THE EXPERIMENTAL MDC DATA SETS.

Data Set	#Exam.	#Dim.	#Labels/Dim.	#Features†
Flare1	323	3	3,4,2	$10x$
WaterQuality	1060	14	4	$16n$
Scm20d	8966	16	4	$61n$
Rf1	8987	8	4,4,3,4,4,3,4,3	$64n$
Thyroid	9172	7	5,5,3,2,4,4,3	$7n$, $22x$
Pain	9734	10	2,5,4,2,2,5,2,5,2,2	$136n$
Scm1d	9803	16	4	$280n$
Disfa	13095	12	5,5,6,3,4,4,5,4,4,4,6,4	$136n$
Fera	14052	5	6	$136n$
Adult	18419	4	7,7,5,2	$5n$,$5x$

† n, x denote numeric and nominal features respectively.

For all metrics, the *larger* the values the better the performance. Ten-fold cross-validation is conducted over all experimental data sets, where both mean metric value and standard deviation are recorded for performance evaluation.

3) Compared Approaches: In this paper, a total of five state-of-the-art MDC approaches are utilized as compared approaches, including Binary Relevance (BR), Class Powerset (CP), Ensembles of Classifier Chains (ECC), Ensembles of Super Class classifiers (ESC) [10], and a metric learning approach for MDC (gMML) [11]. Specifically, BR trains q independent multi-class classifiers while ECC trains q successive multi-class classifiers in a chain, where predictions of preceding classifiers in the chain will be used as extra features for subsequent ones. CP trains a single multi-class classifier by treating each distinct class combination as a new class, while ESC firstly groups class spaces into super-classes and then CP is used for each super-class. gMML alternately learns linear regression models for each class label as well as a Mahalanobis distance metric to solve MDC problem effectively. The recommended parameters in respective literatures are used for all approaches. For MD-KNN, BR, CP, ECC and ESC, LIBSVM with linear kernel [38] is used as the base multi-class classifier. As shown in Algorithm 1, the only parameter *k* of MD-KNN is set to 10.

B. Experimental Results

Table II shows the detailed experimental results of each MDC approach. Moreover, to show whether MD-KNN achieves significantly different performance to other compared approaches, pairwise t-test is conducted based on ten-fold cross-validation (at 0.05 significance level). Accordingly, the resulting win/tie/loss counts are summarized in Table III.

Based on the experimental results, the following observations can be made:

- Across all the 129 configurations[5] (10 data sets $\times$ 5 compared approaches $\times$ 3 metrics), MD-KNN achieves

[5]Due to the high computational complexity which leads to "out of memory" error for LIBSVM package, experimental results under 21 configurations are not available in Table 2 for some compared approaches.

TABLE II

EXPERIMENTAL RESULTS (MEAN±STD. DEVIATION) OF EACH MDC APPROACH. IN ADDITION, ●/○ INDICATES WHETHER MD-KNN IS SIGNIFICANTLY SUPERIOR/INFERIOR TO OTHER COMPARED MDC APPROACHES ON EACH DATA SET (PAIRWISE t-TEST AT 0.05 SIGNIFICANCE LEVEL).

Data Set	MD-KNN	BR	CP	ECC	ESC	gMML
	Hamming Score					
Flare1	0.923±0.033	0.922±0.034	0.923±0.033	0.922±0.034	0.923±0.033	0.925±0.034
WaterQuality	0.652±0.013	0.644±0.013●	0.626±0.012●	0.643±0.013●	0.641±0.013●	0.643±0.013●
Scm20d	0.868±0.005	0.666±0.006●	N/A	0.665±0.005●	N/A	0.600±0.007●
Rf1	0.982±0.001	0.891±0.002●	0.928±0.003●	0.888±0.004●	0.919±0.003●	0.730±0.007●
Thyroid	0.967±0.002	0.965±0.002●	0.965±0.002●	0.965±0.002●	0.965±0.002●	0.960±0.002●
Pain	0.971±0.003	0.953±0.003●	0.954±0.003●	0.952±0.004●	0.954±0.003●	0.948±0.004●
Scm1d	0.878±0.002	0.829±0.004●	N/A	0.824±0.003●	N/A	0.697±0.007●
Disfa	0.937±0.002	0.901±0.002●	N/A	0.900±0.002●	0.904±0.003●	0.884±0.003●
Fera	0.764±0.006	0.636±0.008●	N/A	0.631±0.008●	N/A	0.589±0.007●
Adult	0.699±0.005	0.710±0.004○	0.707±0.005○	0.710±0.004○	0.708±0.004○	0.705±0.004○
	Exact Match					
Flare1	0.821±0.073	0.821±0.073	0.821±0.073	0.817±0.078	0.821±0.073	0.821±0.075
WaterQuality	0.009±0.009	0.007±0.008	0.000±0.000●	0.006±0.008	0.006±0.008	0.006±0.008
Scm20d	0.237±0.012	0.065±0.008●	N/A	0.101±0.010●	N/A	0.052±0.007●
Rf1	0.860±0.011	0.428±0.014●	0.612±0.013●	0.438±0.017●	0.580±0.011●	0.138±0.011●
Thyroid	0.788±0.016	0.773±0.015●	0.776±0.014●	0.772±0.014●	0.771±0.014●	0.741±0.015●
Pain	0.833±0.015	0.759±0.015●	0.771±0.016●	0.761±0.016●	0.769±0.015●	0.750±0.018●
Scm1d	0.262±0.014	0.175±0.010●	N/A	0.197±0.013●	N/A	0.102±0.009●
Disfa	0.577±0.012	0.401±0.009●	N/A	0.402±0.010●	0.427±0.011●	0.379±0.011●
Fera	0.400±0.014	0.211±0.013●	N/A	0.211±0.013●	N/A	0.196±0.013●
Adult	0.256±0.009	0.247±0.009●	0.307±0.012○	0.260±0.008	0.310±0.009○	0.230±0.009●
	Sub-Exact Match					
Flare1	0.951±0.036	0.947±0.039	0.951±0.036	0.951±0.036	0.951±0.036	0.957±0.039
WaterQuality	0.060±0.017	0.051±0.024	0.034±0.017●	0.050±0.023	0.046±0.022●	0.049±0.024
Scm20d	0.473±0.019	0.131±0.008●	N/A	0.171±0.007●	N/A	0.100±0.009●
Rf1	0.993±0.002	0.785±0.006●	0.867±0.012●	0.769±0.010●	0.842±0.012●	0.375±0.014●
Thyroid	0.981±0.003	0.982±0.004	0.981±0.005	0.981±0.004	0.982±0.004	0.982±0.005
Pain	0.923±0.008	0.863±0.009●	0.867±0.008●	0.859±0.010●	0.864±0.008●	0.846±0.010●
Scm1d	0.498±0.020	0.348±0.018●	N/A	0.358±0.014●	N/A	0.198±0.015●
Disfa	0.798±0.010	0.652±0.012●	N/A	0.652±0.011●	0.668±0.013●	0.590±0.009●
Fera	0.653±0.011	0.435±0.012●	N/A	0.432±0.013●	N/A	0.378±0.013●
Adult	0.642±0.008	0.669±0.009○	0.637±0.007	0.662±0.009○	0.638±0.008	0.669±0.008○

TABLE III

WIN/TIE/LOSS COUNTS OF PAIRWISE t-TEST (AT 0.05 SIGNIFICANCE LEVEL) BETWEEN MD-KNN AND EACH MDC APPROACH.

Evaluation metric	MD-KNN against				
	BR	CP	ECC	ESC	gMML
HS	8/1/1	4/1/1	8/1/1	5/1/1	8/1/1
EM	8/2/0	4/1/1	7/3/0	4/2/1	8/2/0
SEM	6/3/1	3/3/0	6/3/1	4/3/0	6/3/1
In Total	**22/6/2**	**11/5/2**	**21/7/2**	**13/6/2**	**22/6/2**

superior or at least comparable performance against the five compared approaches in 119 cases.

- CP, ECC, ESC explicitly consider class dependencies when learning from training examples. It is shown that MD-KNN achieves highly competitive performance against these approaches, which clearly validate the effectiveness of MD-KNN's pairwise dependency modeling strategy.
- gMML learns a Mahalanobis distance metric which can make the distance between the predicted class vector of one example and its ground-truth one closer. Although Euclidean distance is simply utilized to measure similarities in this paper, MD-KNN also achieves superior performance against gMML in 22 out of 30 cases.
- All the 10 under-performing cases and 23 out of 30 comparable cases for MD-KNN against other compared approaches occur for *Flare1*, *Thyroid* and *Adult*, which all have nominal features. In this scenario, Hamming distance serves as a complementary metric by MD-KNN to measure similarities, which might not be a good choice and further studies could be explored in the future.

C. Further Analysis

1) The Effectiveness of MD-KNN's Design: There are a total of four noteworthy technical components in MD-KNN's design. Firstly, MD-KNN aims at making *MAP inference* based

TABLE IV
EXPERIMENTAL RESULTS (MEAN±STD. DEVIATION) OF MD-KNN AND ITS FOUR DEGENERATED VERSIONS. IN ADDITION, THE RANK FOR EACH APPROACH PER DATA SET IS ALSO SHOWN IN PARENTHESES.

Data Set	MD-KNN	DeV1	DeV2	DeV3	DeV4
	Hamming Score				
Flare1	0.923±0.033(1)	0.923±0.038(1)	0.923±0.033(1)	0.923±0.033(1)	0.923±0.033(1)
WaterQuality	0.652±0.013(1)	0.644±0.012(5)	0.650±0.013(4)	0.651±0.012(2)	0.651±0.012(2)
Scm20d	0.868±0.005(1)	0.863±0.006(5)	0.866±0.005(4)	0.867±0.005(2)	0.867±0.006(2)
Rf1	0.982±0.001(1)	0.981±0.001(2)	0.981±0.001(2)	0.981±0.001(2)	0.981±0.001(2)
Thyroid	0.967±0.002(1)	0.966±0.003(5)	0.967±0.003(1)	0.967±0.002(1)	0.967±0.002(1)
Pain	0.971±0.003(1)	0.968±0.003(5)	0.970±0.003(2)	0.970±0.003(2)	0.970±0.003(2)
Scm1d	0.878±0.002(1)	0.871±0.003(5)	0.876±0.002(4)	0.877±0.002(2)	0.877±0.002(2)
Disfa	0.937±0.002(1)	0.932±0.002(5)	0.936±0.002(3)	0.937±0.002(1)	0.936±0.002(3)
Fera	0.764±0.006(1)	0.756±0.006(5)	0.762±0.008(3)	0.762±0.008(3)	0.763±0.007(2)
Adult	0.699±0.005(1)	0.693±0.005(3)	0.691±0.004(5)	0.693±0.005(3)	0.698±0.005(2)
	Exact Match				
Flare1	0.821±0.073(1)	0.814±0.085(5)	0.821±0.073(1)	0.817±0.078(4)	0.821±0.073(1)
WaterQuality	0.009±0.009(1)	0.006±0.007(5)	0.008±0.008(3)	0.008±0.008(3)	0.009±0.010(1)
Scm20d	0.237±0.012(1)	0.235±0.013(3)	0.234±0.012(5)	0.235±0.014(3)	0.236±0.013(2)
Rf1	0.860±0.011(1)	0.857±0.010(3)	0.857±0.010(3)	0.858±0.011(2)	0.856±0.008(5)
Thyroid	0.788±0.016(1)	0.778±0.017(5)	0.784±0.016(3)	0.786±0.016(2)	0.784±0.015(3)
Pain	0.833±0.015(1)	0.822±0.015(5)	0.827±0.016(4)	0.830±0.017(2)	0.828±0.015(3)
Scm1d	0.262±0.014(1)	0.251±0.014(5)	0.256±0.016(4)	0.259±0.014(3)	0.260±0.017(2)
Disfa	0.577±0.012(1)	0.552±0.012(5)	0.566±0.012(4)	0.570±0.012(2)	0.568±0.012(3)
Fera	0.400±0.014(1)	0.391±0.013(5)	0.397±0.014(3)	0.397±0.015(3)	0.400±0.014(1)
Adult	0.256±0.009(1)	0.245±0.008(2)	0.191±0.010(5)	0.196±0.013(4)	0.244±0.010(3)
	Sub-Exact Match				
Flare1	0.951±0.036(3)	0.957±0.036(1)	0.951±0.036(3)	0.954±0.037(2)	0.951±0.036(3)
WaterQuality	0.060±0.017(1)	0.044±0.015(5)	0.052±0.018(3)	0.049±0.021(4)	0.057±0.021(2)
Scm20d	0.473±0.019(2)	0.462±0.019(5)	0.467±0.017(4)	0.470±0.017(3)	0.475±0.020(1)
Rf1	0.993±0.002(2)	0.994±0.002(1)	0.993±0.002(2)	0.993±0.002(2)	0.993±0.001(2)
Thyroid	0.981±0.003(4)	0.981±0.003(4)	0.982±0.003(1)	0.982±0.003(1)	0.982±0.003(1)
Pain	0.923±0.008(1)	0.912±0.011(5)	0.920±0.010(4)	0.922±0.011(2)	0.921±0.010(3)
Scm1d	0.498±0.020(2)	0.479±0.016(5)	0.493±0.016(4)	0.496±0.017(3)	0.499±0.015(1)
Disfa	0.798±0.010(2)	0.781±0.011(5)	0.794±0.008(4)	0.799±0.009(1)	0.795±0.009(3)
Fera	0.653±0.011(1)	0.642±0.010(5)	0.649±0.014(4)	0.651±0.014(2)	0.650±0.013(3)
Adult	0.642±0.008(4)	0.633±0.009(5)	0.656±0.006(2)	0.657±0.009(1)	0.649±0.008(3)

on *k*NN counting statistics from a general view. Then, to consider pairwise class dependencies, MD-KNN is specially designed in both *feature* space and *output* space. As shown in Eq.(4), for each pair of class spaces, both the *k*NN counting statistics are used as features and all distinct class combinations are used as new classes. Finally, for each class space, MD-KNN determines the prediction by selecting one from the candidates via *empirical kNN accuracy*. By integrating these four technical components, MD-KNN achieves highly competitive performance against state-of-the-art MDC approaches which has been shown in the previous section. Here, to further validate the effectiveness of MD-KNN's design, we conduct specific comparative studies between MD-KNN and its four degenerated versions, which are denoted by DeV1, DeV2, DeV3, and DeV4 respectively.

Specifically, in DeV1, each class space is independently solved by a standard *k*NN classifier, i.e., none of the technical components above are used here. In DeV2, each class space is independently solved by a classifier trained over the following data set:

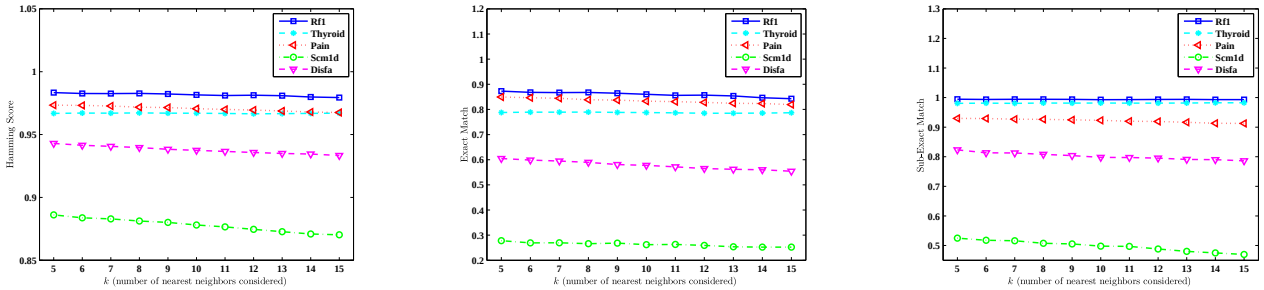
$$\mathcal{D}_j^{\mathrm{MAP}} = \{(\boldsymbol{\delta}_j^{\boldsymbol{x}_i}, y_{ij}) \mid 1 \le i \le m\} \quad (1 \le j \le q)$$

i.e., only MAP rule is used here. In DeV3, the only difference from MD-KNN is that we replace the $\binom{q}{2}$ different $\mathcal{D}_{rs}^{\mathrm{MAP}}$ in Eq.(4) with $q \cdot (q-1)$ different $\mathcal{D}_{rs}^{\mathrm{DeV3}}$ defined as follows:

$$\mathcal{D}_{rs}^{\mathrm{DeV3}} = \{(\boldsymbol{x}_i^{rs}, y_{ir}) \mid 1 \le i \le m\} \quad (1 \le r < s \le q)$$

i.e., the third technical component in MD-KNN is not used here. In DeV4, the only difference from MD-KNN is that we replace *k*NN accuracy selection criterion with majority voting when making an ensemble of $q-1$ predictions in Eq.(5).

Table IV reports the detailed experimental results of these four degenerated versions and the rank for each approach per data set is also shown in parentheses. Furthermore, *Wilcoxon signed-ranks test* [39] is used as the statistical test to show whether MD-KNN performs significantly better than DeV1, DeV2, DeV3, and DeV4 in terms of each evaluation metric respectively. Table V summarizes the statistical test results and



(a) *Hamming Score* (b) *Exact Match* (c) *Sub-Exact Match*

Fig. 1. Performance of MD-KNN changes as k varies from 5 to 15.

TABLE V
WILCOXON SIGNED-RANKS TEST FOR MD-KNN AGAINST ITS FOUR DEGENERATED VERSIONS IN TERMS OF EACH EVALUATION METRIC (SIGNIFICANCE LEVEL $\alpha = 0.05$; p-VALUES SHOWN IN THE BRACKETS).

MD-KNN **versus**	Evaluation metric		
	HS	EM	SEM
DeV1	**win** [3.91e-3]	**win** [1.95e-3]	**win** [1.95e-2]
DeV2	**win** [3.91e-3]	**win** [3.91e-3]	**tie** [2.50e-1]
DeV3	**win** [3.91e-3]	**win** [1.95e-3]	**tie** [6.95e-1]
DeV4	**win** [3.91e-3]	**win** [1.56e-2]	**tie** [8.20e-1]

the p-values for the corresponding tests are also shown in the brackets.

As shown in Table V, MD-KNN achieves statistically better performance than DeV1 in terms of all metrics, and DeV2, DeV3, DeV4 in terms of HS and EM. These results clearly validate the effectiveness of MD-KNN's design.

2) Sensitivity Analysis: As shown in Algorithm 1, there is only one parameter k to be set for MD-KNN, i.e., the number of nearest neighbors considered. Figure 1 shows how the performance of MD-KNN changes as k increases from 5 to 15. It is shown that MD-KNN achieves relatively stable performance when the value of k varies. Therefore, we simply fix k to be the moderate value of 10 in this paper.

V. CONCLUSION

In this paper, a first attempt towards adapting instance-based techniques to solve multi-dimensional classification problem is investigated. Specifically, a novel approach named MD-KNN is proposed which makes use of instance-based techniques in two levels. In the first level, for each pair of class spaces, MAP inference is made based on their corresponding kNN counting statistics. In the second level, predictive label w.r.t. each class space for the unseen instance is determined by selecting one from candidates according to empirical kNN accuracy estimation. Comparative studies on ten real-world MDC data sets clearly validate the effectiveness of the proposed MD-KNN approach.

REFERENCES

[1] T. Theeramunkong and V. Lertnattee, "Multi-dimensional text classification," in *Proceedings of the 19th International Conference on Computational Linguistics*, 2002. [Online]. Available: https://www.aclweb.org/anthology/C02-1155

[2] 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.

[3] J. D. Rodríguez, A. Pérez, 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.

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

[5] H. Borchani, C. Bielza, P. Martinez-Martin, and P. Larrañaga, "Predicting the EQ-5D from the Parkinson's disease questionnaire PDQ-8 using multi-dimensional Bayesian network classifiers," *Biomedical Engineering: Applications, Basis and Communications*, vol. 26, no. 1, pp. 1–11, 2014.

[6] B. Mihaljević, C. Bielza, R. Benavides-Piccione, J. DeFelipe, and P. Larrañaga, "Multi-dimensional classification of GABAergic interneurons with Bayesian network-modeled label uncertainty," *Frontiers in Computational Neuroscience*, vol. 8, 2014, Article 150.

[7] R. Sagarna, A. Mendiburu, I. Inza, and J. A. Lozano, "Assisting in search heuristics selection through multidimensional supervised classification: A case study on software testing," *Information Sciences*, vol. 258, pp. 122–139, 2014.

[8] P. Fernandez-Gonzalez, C. Bielza, and P. Larrañaga, "Multidimensional classifiers for neuroanatomical data," in *ICML Workshop on Statistics, Machine Learning and Neuroscience*, Lille, France, 2015. [Online]. Available: https://sites.google.com/site/stamlins2015/

[9] 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.

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

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

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

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

[14] M. Zhu, S. Liu, and J. Jiang, "A hybrid method for learning multi-dimensional Bayesian network classifiers based on an optimization model," *Applied Intelligence*, vol. 44, no. 1, pp. 123–148, 2016.

[15] 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.

[16] M. Benjumeda, C. Bielza, and P. Larrañaga, "Tractability of most probable explanations in multidimensional Bayesian network classifiers,"

International Journal of Approximate Reasoning, vol. 93, pp. 74–87, 2018.

[17] S. Gil-Begue, P. Larrañaga, and C. Bielza, "Multi-dimensional Bayesian network classifier trees," in *Proceedings of the 19th International Conference on Intelligent Data Engineering and Automated Learning*, Madrid, Spain, 2018, pp. 354–363.

[18] B.-B. Jia and M.-L. Zhang, "Maximum margin multi-dimensional classification," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, New York, NY, 2020, pp. 4312–4319.

[19] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[20] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, no. 2-3, pp. 211–225, 2009.

[21] M.-L. Zhang, "A k-nearest neighbor based multi-instance multi-label learning algorithm," in *Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence*, vol. 2, Arras, France, 2010, pp. 207–212.

[22] D. Wang, J. Wang, F. Hu, L. Li, and X. Zhang, "A locally adaptive multi-label k-nearest neighbor algorithm," in *Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Melbourne, Australia, 2018, pp. 81–93.

[23] A. T. Ruiz, P. Thiam, F. Schwenker, and G. Palm, "A k-nearest neighbor based algorithm for multi-instance multi-label active learning," in *Proceedings of the 8th IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, Siena, Italy, 2018, pp. 139–151.

[24] M. Roseberry, B. Krawczyk, and A. Cano, "Multi-label punitive kNN with self-adjusting memory for drifting data streams," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 6, 2019, Article 60.

[25] 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.

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

[27] 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.

[28] 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.

[29] B.-B. Jia and M.-L. Zhang, "Multi-dimensional classification via stacked dependency exploitation," *Science China Information Sciences*, 2020. [Online]. Available: https://doi.org/10.1007/s11432-019-2905-3

[30] L. C. van der Gaag and P. R. de Waal, "Multi-dimensional Bayesian network classifiers," in *Proceedings of the 3rd European Workshop in Probabilistic Graphical Models*, Prague, Czech Republic, 2006, pp. 107–114.

[31] P. R. de Waal and L. C. van der Gaag, "Inference and learning in multi-dimensional Bayesian network classifiers," in *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Hammamet, Tunisia, 2007, pp. 501–511.

[32] J. D. Rodríguez and J. A. Lozano, "Multi-objective learning of multi-dimensional Bayesian classifiers," in *Proceedings of the 8th International Conference Hybrid Intelligent Systems*, Barcelona, Spain, 2008, pp. 501–506.

[33] J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga, "Bayesian chain classifiers for multidimensional classification," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, vol. 11, 2011, pp. 2192–2197.

[34] 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.

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

[36] R. Walecki, O. Rudovic, V. Pavlovic, and M. Pantic, "Copula ordinal regression for joint estimation of facial action unit intensity," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 4902–4910.

[37] Z. Ma and S. Chen, "A convex formulation for multiple ordinal output classification," *Pattern Recognition*, vol. 86, pp. 73–84, 2019.

[38] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, 2011, Article 27, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[39] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.