Multi-Dimensional Classification via Sparse Label Encoding

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

Abstract

In multi-dimensional classification (MDC), there are multiple class variables in the output space with each of them corresponding to one heterogeneous class space. Due to the heterogeneity of class spaces, it is quite challenging to consider the dependencies among class variables when learning from MDC examples. In this paper, we propose a novel MDC approach named SLEM which learns the predictive model in an encoded label space instead of the original heterogeneous one. Specifically, SLEM works in an *encoding-training-decoding* framework. In the encoding phase, each class vector is mapped into a real-valued one via three cascaded operations including pairwise grouping, one-hot conversion and sparse linear encoding. In the training phase, a multi-output regression model is learned within the encoded label space. In the decoding phase, the predicted class vector is obtained by adapting orthogonal matching pursuit over outputs of the learned multi-output regression model. Experimental results clearly validate the superiority of SLEM against state-of-the-art MDC approaches.

1. Introduction

In traditional supervised learning, the semantics of objects are usually characterized by only one output variable, e.g., multi-class classification. However, in some real-world applications, the semantics of objects need to be characterized along different dimensions. For example, the *e*-commerce websites should categorize laptops from different dimensions (e.g., *brand*, *operating system*, *CPU*, *GPU*, etc.) to make it more convenient for consumers to choose the right laptop for themselves. In fact, similar requirements widely exist in various fields, e.g., text classification (Shatkay et al., 2008), bioinformatics (Rodríguez et al., 2012), resource allocation (Al Muktadir et al., 2019), ecology (Verma et al., 2021), etc. These special applications can be naturally formalized under the *multi-dimensional classification* (MDC) learning framework (Read et al., 2014a; Ma & Chen, 2018; Jia & Zhang, 2020a; Wang et al., 2020). In MDC, each example is represented by a single instance while associated with multiple class variables. Here, each class variable corresponds to one specific class space which characterizes the semantics of objects from one dimension.

Formally speaking, let $\mathcal{X} = \mathbb{R}^d$ be the input (feature) space, and $\mathcal{Y} = C_1 \times C_2 \times \cdots \times C_q$ be the output space which corresponds to the Cartesian product of q class spaces. Here, 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 item in C_j, i.e., $y_{ij} \in C_j$. 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 $\boldsymbol{x}$.

To solve the MDC problem, we can independently deal with each dimension which is actually a multi-class classification problem. Nonetheless, this strategy does not consider potential dependencies among class spaces which would degenerate its generalization ability. Therefore, most existing MDC studies focus on how to model class dependencies more appropriately, e.g., specifying a chaining structure over class variables (Zaragoza et al., 2011; Read et al., 2014b), partitioning class spaces into several groups (Read et al., 2014a), learning a direct acyclic graph structure over class variables (Bielza et al., 2011; Gil-Begue et al., 2021), etc.

Due to the heterogeneity of class spaces, it is quite challenging to directly consider the dependencies among class variables in the original output space as most existing MDC approaches do. In this paper, we attempt to learn the predictive model which solves the MDC problem in its transformed label space. Accordingly, we propose a novel approach named SLEM, i.e., *Sparse Label Encoding for Multi-dimensional classification*, which works in an encoding-

[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 Lab. of Computer Network and Information Integration (Southeast University), Ministry of Education, China. Correspondence to: Min-Ling Zhang <zhangml@seu.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

training-decoding framework by utilizing the sparse property of the transformed label space. In the encoding phase, the output space is transformed into a new one via three cascaded operations, including pairwise grouping, one-hot conversion and sparse linear encoding. To be more specific, pairwise grouping aims at making the results of one-hot conversion sparser and linear encoding further maps the sparse label vector into a real-valued one. In the training phase, SLEM learns a multi-output regression model in the encoded label space to deal with the resulting problem. In the decoding phase, based on the predicted real-valued label vector which is obtained by feeding unseen instance into the learned multi-output regressor, SLEM conducts three inverse operations corresponding to the encoding phase in reverse order to predict the class vector for unseen instance. To be more specific, the inverse operation of sparse linear encoding is implemented by adapting the orthogonal matching pursuit algorithm (Tropp & Gilbert, 2007) and the other two inverse operations can be implemented directly. Experimental results clearly validate the superiority of SLEM against 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 presented. Thirdly, experimental results of comparative studies are reported. Finally, we conclude this paper.

2. Related Work

To solve the MDC problem, we can either decompose it into a number of independent multi-class classification problems, one per class space, or transform it into a single multi-class classification problem, where each distinct class combination in the training set is regarded as one new class. However, the first strategy cannot consider the dependencies among class spaces (i.e., underfitting), while the second strategy cannot return class combinations not appearing in the training set (i.e., overfitting). To alleviate the underfitting problem, the classifier chains model trains a chain of multi-class classifiers, one per class space, where the subsequent classifiers on the chain will augment the feature space with the class spaces which are used to train the preceding classifiers (Zaragoza et al., 2011; Read et al., 2014b; Liu et al., 2017). To alleviate the overfitting problem, the super-class model partitions the class spaces into several groups, where each group will be treated as a new class space (Read et al., 2014a). Besides, due to the powerful modeling abilities of probabilistic graphical model, we can also learn a direct acyclic graph over the class variables to explicitly model the class dependencies (Zhu et al., 2016; Bolt & van der Gaag, 2017; Benjumeda et al., 2018).

The gMML approach solves the MDC problem in a binary-valued label space which is obtained by concatenating the one-vs-rest decomposition of each class space (Ma & Chen, 2018). Because the simple concatenation cannot blend the heterogeneous class spaces into an integrated label space, there is no intrinsic difference between the decomposed label space and the original one. Thus, the responsibilities for considering class dependencies should be taken by the following predictive model induced in the decomposed label space, which is accomplished via a metric approach in gMML. Besides, it is less reasonable to directly align predictive outputs of class labels from different class spaces due to the heterogeneity assumption in MDC. To address or alleviate the aforementioned issues, the proposed approach in this paper attempts to encode the multi-dimensional class spaces into an integrated label space, within which a predictive model is induced. It is worth noting that the label encoding strategy (Tai & Lin, 2012; Shen et al., 2018; Liu & Shen, 2019; Liu et al., 2019) has been successfully applied to solve the multi-label classification (MLC) problem (Zhang & Zhou, 2014; Gibaja & Ventura, 2015), which can be regarded as degenerated version of MDC by restricting each class variable to be binary-valued.

3. The SLEM Approach

The workflow of the proposed SLEM approach is shown in Figure 1 where the whole process can be divided into three parts: encoding, training and decoding. In the following of this section, we will present their technical details.

3.1. Encoding Phase

For each training example $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}$, we generally convert the nominal class vector $\boldsymbol{y}_i \in \mathcal{Y}$ into its one-hot form $\boldsymbol{y}'_i \in \{0,1\}^{\sum_{j=1}^{q} K_j}$ when we need to do some numeric computations for it.[1] We denote the one-hot conversion in output space $\mathcal{Y}$ as $\boldsymbol{y}'_i = \Phi_{\mathcal{Y}}(\boldsymbol{y}_i)$ and its inverse as $\boldsymbol{y}_i = \Phi_{\mathcal{Y}}^{-1}(\boldsymbol{y}'_i)$. It is easy to know that the length-$\sum_{j=1}^{q} K_j$ vector $\boldsymbol{y}'_i$ is always q-sparse,[2] and there is one and only one '1' among the $(1 + \sum_{j=1}^{a-1} K_j)$-th to $(\sum_{j=1}^{a} K_j)$-th entries which we call the a-th *local group* of $\boldsymbol{y}'_i$ $(1 \leq a \leq q)$. In this paper, we refer to this property as *local sparsity*. For the proposed SLEM approach, its decoding process to be introduced in Subsection 3.3 will utilize the sparse property. However, in light of our observations on existing real-world MDC data sets (cf. Table 1), the sparsity of the one-hot vector is usually not sparse enough to ensure the sparsity reconstruction algorithm working properly.

To tackle this issue, motivated by the idea of super-class

[1] Specifically, the one-hot form of the j-th entry y_{ij} in $\boldsymbol{y}_i$ is a length-K_j vector $\boldsymbol{y}'_{ij} \in \{0,1\}^{K_j}$, where the a-th entry in $\boldsymbol{y}'_{ij}$ equals 1 if $y_{ij} = c_a^j$ and 0 otherwise. The vector $\boldsymbol{y}'_i$ corresponds to the concatenation of q different $\boldsymbol{y}'_{ij}$ $(1 \leq j \leq q)$.

[2] A vector is q-sparse if it has at most q non-zero entries.

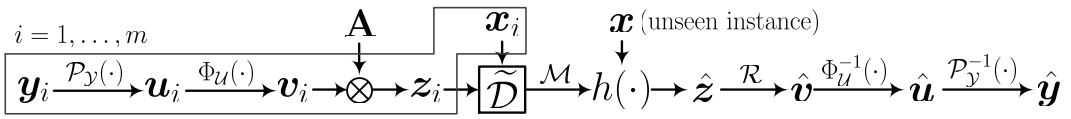


Figure 1. The workflow of the proposed SLEM approach. Here, $\boldsymbol{u}_i = \mathcal{P}_{\mathcal{Y}}(\boldsymbol{y}_i)$, $\boldsymbol{v}_i = \Phi_{\mathcal{U}}(\boldsymbol{u}_i)$, $\boldsymbol{z}_i = \mathbf{A}\boldsymbol{v}_i$ correspond to the three cascaded operations including *pairwise grouping* in output space $\mathcal{Y}$, *one-hot conversion* in output space $\mathcal{U}$, *spare linear encoding* with matrix $\mathbf{A}$, respectively; $\widetilde{\mathcal{D}} = \{(\boldsymbol{x}_i, \boldsymbol{z}_i) \mid 1 \le i \le m\}$ is a multi-output regression data set, $h = \mathcal{M}(\widetilde{\mathcal{D}})$ is the learned multi-output regressor where $\mathcal{M}$ corresponds to the employed multi-output regression algorithm, $\hat{\boldsymbol{z}} = h(\boldsymbol{x})$, $\hat{\boldsymbol{v}} = \mathcal{R}(\hat{\boldsymbol{z}})$ where $\mathcal{R}$ is the sparsity reconstruction algorithm, $\hat{\boldsymbol{u}} = \Phi_{\mathcal{U}}^{-1}(\hat{\boldsymbol{v}})$, $\hat{\boldsymbol{y}} = \mathcal{P}_{\mathcal{Y}}^{-1}(\hat{\boldsymbol{u}})$ where $\Phi_{\mathcal{U}}^{-1}(\cdot)$ and $\mathcal{P}_{\mathcal{Y}}^{-1}(\cdot)$ correspond to the inverse function of $\Phi_{\mathcal{U}}(\cdot)$ and $\mathcal{P}_{\mathcal{Y}}(\cdot)$, respectively.

partition (Read et al., 2014a), we further group the q class spaces C_j ($1 \le j \le q$) into $\lfloor \frac{q}{2} \rfloor$ pairs (plus a singleton class space if q is odd). Let $\mathcal{U} = C'_1 \times C'_2 \times \cdots \times C'_{\lceil \frac{q}{2} \rceil}$ be the newly-obtained output space from $\mathcal{Y} = C_1 \times C_2 \times \cdots \times C_q$, and $[\tau(1), \ldots, \tau(q)]$ be the ascending order according to the number of class labels in each class space, i.e., $C_{\tau(1)}$ and $C_{\tau(q)}$ include the least and most number of class labels respectively. For each class space C'_j ($1 \le j \le \lfloor \frac{q}{2} \rfloor$), it corresponds to the powerset transformation of $C_{\tau(j)}$ and $C_{\tau(\bar{q}-j+1)}$, i.e., each class label in C'_j corresponds to one distinct class combination in the Cartesian product $C_{\tau(j)} \times C_{\tau(\bar{q}-j+1)}$. Here, $\bar{q}$ equals q if q is even and $q-1$ if q is odd. Let s_j be the number of class labels in C'_j, generally we have $s_j = K_{\tau(j)} \times K_{\tau(\bar{q}-j+1)}$. Besides, if q is odd, then $C'_{\lceil \frac{q}{2} \rceil} = C_{\tau(q)}$ and $s_{\lceil \frac{q}{2} \rceil} = K_{\tau(q)}$. we refer to this operation as *pairwise grouping* in this paper.

With the pairwise grouping operation, the output space $\mathcal{Y}$ is transformed into $\mathcal{U}$, and then the length-q class vector $\boldsymbol{y}_i \in \mathcal{Y}$ will be transformed into another length-$\lceil \frac{q}{2} \rceil$ class vector $\boldsymbol{u}_i \in \mathcal{U}$. We denote the pairwise grouping operation in output space $\mathcal{Y}$ as $\boldsymbol{u}_i = \mathcal{P}_{\mathcal{Y}}(\boldsymbol{y}_i)$ and its inverse as $\boldsymbol{y}_i = \mathcal{P}_{\mathcal{Y}}^{-1}(\boldsymbol{u}_i)$. Here, $\mathcal{P}_{\mathcal{Y}}(\cdot)$ just heuristically aims at making the number of class labels in each class space of $\mathcal{U}$ as balanced as possible, and more subtle designs can be explored in the future. In summary, this operation will bring two benefits. The first one is that $\Phi_{\mathcal{U}}(\boldsymbol{u}_i)$ will be sparser than $\Phi_{\mathcal{Y}}(\boldsymbol{y}_i)$ which is also the original intention of this operation.[3] The second one can be regarded as a bonus of this operation that it can consider the dependencies between class spaces grouped into one pair, which can be modeled more reliably than dependencies among many class spaces with limited number of training examples. We denote the one-hot conversion of $\boldsymbol{u}_i$ in output space $\mathcal{U}$ as $\boldsymbol{v}_i$, i.e., $\boldsymbol{v}_i = \Phi_{\mathcal{U}}(\boldsymbol{u}_i)$. It is easy to know that the length-$\sum_{j=1}^{\lceil \frac{q}{2} \rceil} s_j$ vector $\boldsymbol{v}_i$ is always $\lceil \frac{q}{2} \rceil$-sparse, and there is one and only one '1' among the $(1+\sum_{j=1}^{a-1} s_j)$-th to $(\sum_{j=1}^{a} s_j)$-th entries, i.e., the a-th *local group* of $\boldsymbol{v}_i$ ($1 \le a \le \lceil \frac{q}{2} \rceil$).

[3] Because $a \times b$ must be greater than $a + b$ if $a, b > 2$, the length of $\Phi_{\mathcal{U}}(\boldsymbol{u}_i)$ (i.e., $\sum_{j=1}^{\lceil \frac{q}{2} \rceil} s_j$) must be greater than the length of $\Phi_{\mathcal{Y}}(\boldsymbol{y}_i)$ (i.e., $\sum_{j=1}^{q} K_j$), while the sparsity level of $\Phi_{\mathcal{U}}(\boldsymbol{u}_i)$ (i.e., $\lceil \frac{q}{2} \rceil$) is less than the sparsity level of $\Phi_{\mathcal{Y}}(\boldsymbol{y}_i)$ (i.e., q).

For the sake of brevity, we further denote the length of $\boldsymbol{v}_i$ as $s = \sum_{j=1}^{\lceil \frac{q}{2} \rceil} s_j$ and its sparsity level as $k = \lceil \frac{q}{2} \rceil$ in the rest of this paper.

Let $\mathbf{A} \in \mathbb{R}^{s' \times s}$ be an encoding matrix, which can linearly encode any length-s vector $\boldsymbol{v}$ into another length-s' vector $\boldsymbol{z}$, i.e., $\boldsymbol{z} = \mathbf{A}\boldsymbol{v}$. Here, we require $s' \le s$ and hope $s' \ll s$ to be held. In this paper, we focus more on label encoding rather than label compression but the whole proposed framework can also work properly under label compression scenario. According to the theory of compressed sensing (Donoho, 2006), there are valid reconstruction algorithms which can recover the vector $\boldsymbol{v}$ from the compressed observation $\boldsymbol{z}$ if $\boldsymbol{v}$ is k-sparse and $\mathbf{A}$ satisfies k-RIP given in the following Definition 1:

Definition 1. *(Restricted Isometry Property). For matrix $\mathbf{A}$, if there is a constant $\delta_k \in [0, 1)$ which satisfies*

$$(1 - \delta_k) \|\boldsymbol{v}\|_2^2 \le \|\mathbf{A}\boldsymbol{v}\|_2^2 \le (1 + \delta_k) \|\boldsymbol{v}\|_2^2 \qquad (1)$$

where $\boldsymbol{v}$ is any k-sparse vector, then $\mathbf{A}$ is known as satisfying k-order Restricted Isometry Property (k-RIP).

It has been proved that some random matrices satisfy k-RIP with large probability (Baraniuk et al., 2008), e.g., Gaussian matrix and Bernoulli matrix. With the encoding matrix $\mathbf{A}$, each binary-valued vector $\boldsymbol{v}_i \in \{0, 1\}^s$ can be mapped into a real-valued vector $\boldsymbol{z}_i \in \mathbb{R}^{s'}$. Obviously, each entry in $\boldsymbol{z}_i$ is related to all the entries in $\boldsymbol{y}_i$, each of which belongs to one heterogeneous class space C_j ($1 \le j \le q$). Therefore, even if the following induced predictive model deals with each entry of $\boldsymbol{z}_i$ independently, the q class spaces in the original output space $\mathcal{Y}$ will be tackled in a joint manner.

3.2. Training Phase

With the three cascaded operations, i.e., pairwise grouping, one-hot conversion and sparse linear encoding, each class vector $\boldsymbol{y}_i$ can be transformed into one real-valued vector $\boldsymbol{z}_i$, then we can obtain a new data set $\widetilde{\mathcal{D}} = \{(\boldsymbol{x}_i, \boldsymbol{z}_i) \mid 1 \le i \le m\}$ based on $\mathcal{D}$, which actually corresponds to a multi-output regression problem (Borchani et al., 2015; Reeve & Kaban, 2020). To solve the resulting problem, we learn a multi-output regressor $h(\boldsymbol{x}) = \mathbf{W}^\top \boldsymbol{x} + \boldsymbol{b}$ via optimizing

the following formulation:

$$\min_{\mathbf{W},\boldsymbol{b},\hat{\mathbf{V}}} \frac{1}{2}\|\mathbf{W}\|_F^2 + \lambda \sum_{i=1}^{m} \big[\|h(\boldsymbol{x}_i) - \boldsymbol{z}_i\|_2^2 + \gamma_1 \big(\|h(\boldsymbol{x}_i) - \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2 + \gamma_2 \|\hat{\boldsymbol{v}}_i - \boldsymbol{v}_i\|_1 \big) \big] \quad (2)$$

Here, $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{s'}] \in \mathbb{R}^{d\times s'}$ and $\boldsymbol{b} = [b_1, b_2, \ldots, b_{s'}]^\top$ are the model parameters of h to be determined, $\hat{\mathbf{V}} = [\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_m]^\top \in \mathbb{R}^{m\times s}$ with $\hat{\boldsymbol{v}}_i$ corresponding to the recovered sparse vector for $\boldsymbol{v}_i$ based on its prediction $h(\boldsymbol{x}_i)$, and λ, γ_1 and γ_2 are three trade-off parameters. Specifically, the term $\|\mathbf{W}\|_F^2$ penalizes the model's complexity, the term $\|h(\boldsymbol{x}_i) - \boldsymbol{z}_i\|_2^2$ corresponds to the squared error loss, and the two terms $\|h(\boldsymbol{x}_i) - \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2 + \gamma_2 \|\hat{\boldsymbol{v}}_i - \boldsymbol{v}_i\|_1$ require that the regressor h can facilitate the subsequent sparse reconstruction procedure in the decoding phase.

To solve the above problem (2), we derive an alternating method, where the two sets of parameters $\{\mathbf{W}, \boldsymbol{b}\}$ and $\hat{\mathbf{V}}$ are optimized alternately until convergence.

Optimizing W and *b* when $\hat{\mathbf{V}}$ is fixed. When $\hat{\mathbf{V}}$ is fixed, we can reformulate the optimization problem (2) as follows:

$$\min_{\mathbf{W},\boldsymbol{b}} \frac{1}{2}\|\mathbf{W}\|_F^2 + \lambda \sum_{i=1}^{m} \big[\|h(\boldsymbol{x}_i) - \boldsymbol{z}_i\|_2^2 + \gamma_1 \|h(\boldsymbol{x}_i) - \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2 \big] \quad (3)$$

Theorem 1. *The optimization problem (3) can be equivalently reformulated as follows:*

$$\min_{\mathbf{W},\boldsymbol{b}} \frac{1}{2}\|\mathbf{W}\|_F^2 + \frac{\tilde{\lambda}}{2} \sum_{i=1}^{m} \|h(\boldsymbol{x}_i) - \tilde{\boldsymbol{z}}_i\|_2^2 \quad (4)$$

where $\tilde{\lambda} = 2\lambda(1+\gamma_1)$ *and* $\tilde{\boldsymbol{z}}_i = \frac{\boldsymbol{z}_i + \gamma_1 \mathbf{A}\hat{\boldsymbol{v}}_i}{1+\gamma_1}$.

Proof. In Eq.(3), for the second term of the objective function, each summation term can be reformulated as follows:

$$\begin{aligned}
&\|h(\boldsymbol{x}_i) - \boldsymbol{z}_i\|_2^2 + \gamma_1 \|h(\boldsymbol{x}_i) - \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2 \\
=&\|h(\boldsymbol{x}_i)\|_2^2 - 2\langle h(\boldsymbol{x}_i), \boldsymbol{z}_i\rangle + \|\boldsymbol{z}_i\|_2^2 \\
&+ \gamma_1(\|h(\boldsymbol{x}_i)\|_2^2 - 2\langle h(\boldsymbol{x}_i), \mathbf{A}\hat{\boldsymbol{v}}_i\rangle + \|\mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2) \\
=&(1+\gamma_1)\|h(\boldsymbol{x}_i)\|_2^2 - 2\langle h(\boldsymbol{x}_i), \boldsymbol{z}_i + \gamma_1 \mathbf{A}\hat{\boldsymbol{v}}_i\rangle \\
&+ \frac{\|\boldsymbol{z}_i + \gamma_1 \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2}{1+\gamma_1} + C_i \\
=&(1+\gamma_1)\left\| h(\boldsymbol{x}_i) - \frac{\boldsymbol{z}_i + \gamma_1 \mathbf{A}\hat{\boldsymbol{v}}_i}{1+\gamma_1} \right\|_2^2 + C_i \qquad (5)
\end{aligned}$$

where $\langle\cdot,\cdot\rangle$ returns the inner product of two vectors, $C_i = \|\boldsymbol{z}_i\|_2^2 + \gamma_1 \|\mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2 - \frac{\|\boldsymbol{z}_i + \gamma_1 \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2}{1+\gamma_1}$ is a constant which is not dependent on variables $\mathbf{W}$ and $\boldsymbol{b}$. Plugging Eq.(5) into Eq.(3) and this completes the proof. □

According to Theorem 1, the formulation (3) can be optimized by equivalently transforming it into a general regression problem (4), whose closed-form solution can be obtained by solving the following linear equations:

$$\begin{bmatrix} \mathbf{I}_d + \tilde{\lambda}\mathbf{X}^\top\mathbf{X} & \tilde{\lambda}\mathbf{X}^\top\mathbf{1}_m \\ \mathbf{1}_m^\top\mathbf{X} & m \end{bmatrix} \begin{bmatrix} \mathbf{W} \\ \boldsymbol{b}^\top \end{bmatrix} = \begin{bmatrix} \tilde{\lambda}\mathbf{X}^\top\tilde{\mathbf{Z}} \\ \mathbf{1}_m^\top\tilde{\mathbf{Z}} \end{bmatrix}$$

where $\mathbf{I}_d$ is an identity matrix with size $d\times d$, $\mathbf{1}_m$ is a column vector of all ones with length m, $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]^\top \in \mathbb{R}^{m\times d}$ is the instance matrix, $\tilde{\mathbf{Z}} = [\tilde{\boldsymbol{z}}_1, \ldots, \tilde{\boldsymbol{z}}_m]^\top \in \mathbb{R}^{m\times s'}$ is the real-valued label matrix. Moreover, if we transform the d-dimensional feature space to one d'-dimensional feature space with nonlinear mapping function $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, we can learn a nonlinear multi-output regressor, i.e., $h(\boldsymbol{x}) = \mathbf{W}^\top\phi(\boldsymbol{x}) + \boldsymbol{b}$. Here, note that $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{s'}] \in \mathbb{R}^{d'\times s'}$. According to the Representer Theorem (Schölkopf & Smola, 2002), the predictive model can be expressed as a linear combination of the training instances under fairly general conditions, i.e., $\boldsymbol{w}_j = \sum_{i=1}^{m} \theta_{ji}\phi(\boldsymbol{x}_i)$. Let $\boldsymbol{\theta}_j = [\theta_{j1}, \ldots, \theta_{jm}]^\top \in \mathbb{R}^{m\times 1}$, $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{s'}] \in \mathbb{R}^{m\times s'}$, and $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_m)]^\top \in \mathbb{R}^{m\times d'}$, the closed-form solution of kernelized problem (4) can be obtained by solving the following linear equations:

$$\begin{bmatrix} \mathbf{I}_m + \tilde{\lambda}\mathbf{K} & \tilde{\lambda}\mathbf{1}_m \\ \mathbf{1}_m^\top\mathbf{K} & m \end{bmatrix} \begin{bmatrix} \boldsymbol{\Theta} \\ \boldsymbol{b}^\top \end{bmatrix} = \begin{bmatrix} \tilde{\lambda}\tilde{\mathbf{Z}} \\ \mathbf{1}_m^\top\tilde{\mathbf{Z}} \end{bmatrix}$$

where $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \in \mathbb{R}^{m\times m}$ with (i,j)th element $K_{ij} = \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Here, $\kappa(\cdot,\cdot)$ is the kernel function which is utilized to avoid directly computing inner product in d'-dimensional feature space.

Optimizing $\hat{\mathbf{V}}$ when W and *b* are fixed. When $\mathbf{W}$ and $\boldsymbol{b}$ are fixed, we can reformulate the optimization problem (2) as a total of m independent problems as follows:

$$\min_{\hat{\boldsymbol{v}}_i} \|h(\boldsymbol{x}_i) - \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2 + \gamma_2 \|\hat{\boldsymbol{v}}_i - \boldsymbol{v}_i\|_1 \quad (6)$$

In this paper, we solve this problem via accelerated proximal gradient (APG) method (Huang et al., 2016).

Theorem 2. *Let* $g(\hat{\boldsymbol{v}}_i) = \|h(\boldsymbol{x}_i) - \mathbf{A}\hat{\boldsymbol{v}}_i\|_2^2$*, for the derivable function* $g(\hat{\boldsymbol{v}}_i)$*,* $\nabla g(\hat{\boldsymbol{v}}_i)$ *is Lipschitz continuous and the Lipschitz constant is* $L_f = \left\|2\mathbf{A}^\top\mathbf{A}\right\|_F$*, where* ∇ *denotes the gradient operator.*

Proof. For $\nabla g(\hat{\boldsymbol{v}}_i)$, it can be calculated as:

$$\nabla g(\hat{\boldsymbol{v}}_i) = 2\mathbf{A}^\top\mathbf{A}\hat{\boldsymbol{v}}_i - 2\mathbf{A}^\top h(\boldsymbol{x}_i)$$

Given any $\hat{\boldsymbol{v}}_i'$ and $\hat{\boldsymbol{v}}_i$, we have:

$$\begin{aligned}
\frac{\|\nabla g(\hat{\boldsymbol{v}}_i') - \nabla g(\hat{\boldsymbol{v}}_i)\|_2}{\|\hat{\boldsymbol{v}}_i' - \hat{\boldsymbol{v}}_i\|_2} =& \frac{\left\|2\mathbf{A}^\top\mathbf{A}(\hat{\boldsymbol{v}}_i' - \hat{\boldsymbol{v}}_i)\right\|_2}{\|\hat{\boldsymbol{v}}_i' - \hat{\boldsymbol{v}}_i\|_2} \\
\le& \frac{\left\|2\mathbf{A}^\top\mathbf{A}\right\|_F \|\hat{\boldsymbol{v}}_i' - \hat{\boldsymbol{v}}_i\|_2}{\|\hat{\boldsymbol{v}}_i' - \hat{\boldsymbol{v}}_i\|_2} \\
=& \left\|2\mathbf{A}^\top\mathbf{A}\right\|_F
\end{aligned}$$

Algorithm 1 Solving problem (6) via APG

Require: The encoding matrix $\mathbf{A} \in \mathbb{R}^{s' \times s}$, the ground-truth sparse vector $\boldsymbol{v}_i \in \{0,1\}^s$, the trade-off parameter γ_2;
Ensure: The recovered sparse vector $\hat{\boldsymbol{v}}_i$;
1: Calculate the Lipschitz constant $L_f = \left\|2\mathbf{A}^\top \mathbf{A}\right\|_F$;
2: Initialize $\hat{\boldsymbol{v}}_i^{(0)} = \hat{\boldsymbol{v}}_i^{(1)} = \boldsymbol{v}_i$, $r_0 = r_1 = 1$, $t = 1$;
3: **repeat**
4: Obtain $\boldsymbol{\zeta}^{(t)}$ according to Eq.(10);
5: Compute $\boldsymbol{\nu}^{(t)} = \boldsymbol{\zeta}^{(t)} - \frac{1}{L_f}\nabla g(\boldsymbol{\zeta}^{(t)})$;
6: Obtain $\hat{\boldsymbol{v}}_i^{(t+1)}$ according to Eq.(8);
7: Compute $r_{t+1} = \frac{1+\sqrt{1+4r_t^2}}{2}$;
8: $t = t + 1$;
9: **until** convergence
10: Return $\hat{\boldsymbol{v}}_i = \hat{\boldsymbol{v}}_i^{(t)}$.

which completes the proof. □

According to Theorem 2, given any initial value $\hat{\boldsymbol{v}}_i^{(t)}$ of $\hat{\boldsymbol{v}}_i$, let $\Delta\hat{\boldsymbol{v}}_i = \hat{\boldsymbol{v}}_i - \hat{\boldsymbol{v}}_i^{(t)}$, the following inequation always holds:

$$\left\|\nabla g(\hat{\boldsymbol{v}}_i) - \nabla g(\hat{\boldsymbol{v}}_i^{(t)})\right\|_2 \leq L_f \left\|\Delta\hat{\boldsymbol{v}}_i\right\|_2$$

Then, the quadratic approximation of $g(\hat{\boldsymbol{v}}_i)$ around $\hat{\boldsymbol{v}}_i^{(t)}$ can be given as follows:

$$\begin{aligned}\hat{g}(\hat{\boldsymbol{v}}_i) &\simeq g(\hat{\boldsymbol{v}}_i^{(t)}) + \langle\nabla g(\hat{\boldsymbol{v}}_i^{(t)}), \Delta\hat{\boldsymbol{v}}_i\rangle + \frac{L_f}{2}\left\|\Delta\hat{\boldsymbol{v}}_i\right\|_2^2 \\ &= \frac{L_f}{2}\left\|\hat{\boldsymbol{v}}_i - \boldsymbol{\nu}^{(t)}\right\|_2^2 + C_{L_f}\end{aligned}$$

where $C_{L_f} = -\frac{1}{2L_f}\left\|\nabla g(\hat{\boldsymbol{v}}_i^{(t)})\right\|_2^2 + g(\hat{\boldsymbol{v}}_i^{(t)})$ is a constant which is not dependent on variables $\hat{\boldsymbol{v}}_i$, and

$$\boldsymbol{\nu}^{(t)} = \hat{\boldsymbol{v}}_i^{(t)} - \frac{1}{L_f}\nabla g(\hat{\boldsymbol{v}}_i^{(t)}) \tag{7}$$

According to the descent lemma (Bauschke et al., 2017), $\hat{g}(\hat{\boldsymbol{v}}_i)$ is an upper bound of $g(\hat{\boldsymbol{v}}_i)$, i.e., $g(\hat{\boldsymbol{v}}_i) \leq \hat{g}(\hat{\boldsymbol{v}}_i)$ always holds. Therefore, $g(\hat{\boldsymbol{v}}_i)$ can be minimized by iteratively minimizing its approximation $\hat{g}(\hat{\boldsymbol{v}}_i)$. Plugging $\hat{g}(\hat{\boldsymbol{v}}_i)$ into the optimization problem (6), we can obtain the following iterative equation:

$$\begin{aligned}\hat{\boldsymbol{v}}_i^{(t+1)} &= \arg\min_{\hat{\boldsymbol{v}}_i} \frac{L_f}{2}\left\|\hat{\boldsymbol{v}}_i - \boldsymbol{\nu}^{(t)}\right\|_2^2 + \gamma_2\left\|\hat{\boldsymbol{v}}_i - \boldsymbol{v}_i\right\|_1 \\ &= \mathrm{softc}(\boldsymbol{\nu}^{(t)}, \frac{\gamma_2}{L_f}, \boldsymbol{v}_i)\end{aligned} \tag{8}$$

where the (element-wise) function $\mathrm{softc}(\cdot,\cdot,\cdot)$ is defined as follows:

$$\mathrm{softc}(x,\mu,c) = \begin{cases} x-\mu & \text{if } x - c > \mu \\ x+\mu & \text{if } x - c < -\mu \\ c & \text{otherwise.}\end{cases} \tag{9}$$

Algorithm 2 LOMP: $\boldsymbol{v} = \mathcal{R}(\boldsymbol{z}, \mathbf{A}, k, \mathcal{I})$

Input: The encoding matrix $\mathbf{A} \in \mathbb{R}^{s' \times s}$, the real-valued vector $\boldsymbol{z} \in \mathbb{R}^{s'}$, sparsity level k, local sparsity information $\mathcal{I}: s_1, s_2, \ldots, s_k$;
Output: The recovered k-sparse vector $\boldsymbol{v}$;
1: Initialize $\boldsymbol{v}$ as zero vector with length $s = \sum_{j=1}^{k} s_j$;
2: Initialize $\boldsymbol{r}_0 = \boldsymbol{z}$, $J = \varnothing$, $\mathbf{B} = \mathbf{A}$;
3: **for** $i = 1$ to k **do**
4: $j_* = \arg\max_j |\langle \boldsymbol{r}_{i-1}, \mathbf{B}_{:j}\rangle|$;
5: $\boldsymbol{v}(j_*) = 1$;
6: $J = J \cup \{j_*\}$;
7: $\boldsymbol{r}_i = \boldsymbol{z} - \mathbf{A}_{:J}(\mathbf{A}_{:J}^\top \mathbf{A}_{:J})^{-1}\mathbf{A}_{:J}^\top \boldsymbol{z}$;
8: **for** $\kappa = 1$ to k **do**
9: $t_f = \sum_{t=1}^{\kappa} s_t$;
10: **if** $j_* \leq t_f$ **then**
11: $t_b = t_f - s_\kappa$;
12: $T = \{t_b + 1, t_b + 2, \ldots, t_f\}$;
13: $\mathbf{B}_{:T} = 0$;
14: break;
15: **end if**
16: **end for**
17: **end for**
18: Return $\boldsymbol{v}$.

Here, note that softc is different with the well-known soft-thresholding function which is defined as follows:

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

where there is an additional constant term in softc. In a nutshell, the solution in Eq.(8) can be obtained by performing variable substitution method, where the variables $\hat{\boldsymbol{v}}_i$ are substituted with the variables $\boldsymbol{\chi}_i = \hat{\boldsymbol{v}}_i - \boldsymbol{v}_i$.

In (Beck & Teboulle, 2009), it is shown that the convergence rate of Eq.(8) can be improved to $O(t^{-2})$ from $O(t^{-1})$ if we replace $\hat{\boldsymbol{v}}_i^{(t)}$ in Eq.(7) with the following $\boldsymbol{\zeta}^{(t)}$:

$$\boldsymbol{\zeta}^{(t)} = \hat{\boldsymbol{v}}_i^{(t)} + \frac{r_{t-1} - 1}{r_t}(\hat{\boldsymbol{v}}_i^{(t)} - \hat{\boldsymbol{v}}_i^{(t-1)}) \tag{10}$$

where $r_0 = r_1 = 1$ and $r_t = \frac{1+\sqrt{1+4r_{t-1}^2}}{2}$ when $t > 1$. In summary, Algorithm 1 shows the procedure of solving problem (6) via APG.

We alternately optimize the formulations in Eq.(3) and Eq.(6) until convergence, and then we can obtain the optimal model parameters $(\mathbf{W}, \boldsymbol{b})$ of the multi-output regressor.

3.3. Decoding Phase

Given the unseen instance $\boldsymbol{x}$, suppose its ground-truth class vector is $\boldsymbol{y} \in \mathcal{Y}$, then its grouped class vector is $\boldsymbol{u} =$

$\mathcal{P}_{\mathcal{Y}}(\boldsymbol{y}) \in \mathcal{U}$, the one-hot conversion is $\boldsymbol{v} = \Phi_{\mathcal{U}}(\boldsymbol{u}) \in \{0,1\}^s$ and the encoded real-valued vector $\boldsymbol{z} = \mathbf{A}\boldsymbol{v} \in \mathbb{R}^{s'}$. The decoding phase corresponds to the inverse operations of these steps to obtain $\boldsymbol{x}$'s predicted class vector $\hat{\boldsymbol{y}}$.

Specifically, we firstly predict the encoded vector $\hat{\boldsymbol{z}} = h(\boldsymbol{x})$ with the learned multi-output regression model h. Then, we can recover the $\lceil \frac{q}{2} \rceil$-sparse vector $\hat{\boldsymbol{v}} = \mathcal{R}(\hat{\boldsymbol{z}})$ by some reconstruction algorithm $\mathcal{R}$. After that, the corresponding grouped class vector can be returned by $\hat{\boldsymbol{u}} = \Phi_{\mathcal{U}}^{-1}(\hat{\boldsymbol{v}})$. Finally, the predicted class vector can be obtained by $\hat{\boldsymbol{y}} = \mathcal{P}_{\mathcal{Y}}^{-1}(\hat{\boldsymbol{u}})$. Obviously, the first two steps are the key operations which could introduce predictive errors while the last two steps can be done accurately as long as their inputs are accurate. Therefore, we optimize the formulation (2) to accomplish the first multi-output regression learning task. For the sparse reconstruction step, we design a reconstruction algorithm which can consider the local sparsity property by adapting the orthogonal matching pursuit (OMP) algorithm (Tropp & Gilbert, 2007). We name the adapted algorithm as local orthogonal matching pursuit (LOMP). Algorithm 2 shows the pseudo-code of LOMP, where $\boldsymbol{v}(j_*)$ denotes the j_*-th entry of vector $\boldsymbol{v}$, and $\mathbf{A}_{:J}$ denotes the submatrix consisting of the columns of $\mathbf{A}$ indexed by J. In contrast with the standard OMP, the key adaptions in LOMP is to set the related columns to zero which belong to the same local group with the current selected column of $\mathbf{A}$, i.e., steps 8-16. This adaption ensures that there is one and only one '1' in each local group of the recovered $\hat{\boldsymbol{v}}$.

4. Experiments

4.1. Experimental Setup

4.1.1. Benchmark Data Sets

In this paper, the experiments are conducted over a total of 11 benchmark data sets, whose detailed characteristics are summarized in Table 1, including the *number of examples* (#Exam.), the *number of dimensions* (#Dim.), the *number of class labels per dimension* (#Labels/Dim.),[4] and the *number of features* (#Features).

4.1.2. Evaluation Metrics

In this paper, the performance of MDC approaches is measured by three widely-used metrics (Ma & Chen, 2018; Jia & Zhang, 2020a;b;c; Wang et al., 2020), i.e., *Hamming Score* (HS), *Exact Match* (EM) and *Sub-Exact Match* (SEM), whose formal definitions can be given as follows:

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

[4]Here, we record the numbers for all dimensions in turn. If all numbers are the same to each other, then we only record one of them.

Table 1. Characteristics of the benchmark data sets.

Data Set	#Exam.	#Dim.	#Labels/Dim.	#Features[5]
Jura	359	2	4,5	$9n$
Oes10	403	16	3	$298n$
Voice	3136	2	4,2	$19n$
Scm20d	8966	16	4	$61n$
Rf1	8987	8	4,4,3,4,4,3,4,3	$64n$
Scm1d	9803	16	4	$280n$
CoIL2000	9822	5	6,10,10,4,2	$81x$
Flickr	12198	5	3,4,3,4,4	$1536n$
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	$5n$,$5x$

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

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

Here, $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \mid 1 \leq i \leq p\}$ is the test set with p examples, $f : \mathcal{X} \mapsto \mathcal{Y}$ is the MDC model to be evaluated, $r^{(i)} = \sum_{j=1}^{q}\mathbf{1}_{y_{ij}=\hat{y}_{ij}}$ is the number of class labels for which f returns the correct predictions for $\boldsymbol{x}_i$, where y_{ij} and $\hat{y}_{ij}$ correspond to the ground-truth and predicted class label of $\boldsymbol{x}_i$'s j-th class space, and predicate $\mathbf{1}_{\pi}$ returns 1 if π holds and 0 otherwise. Obviously, for all the three metrics, the *larger* the metric value, the *better* the performance. In the experiments, we conduct ten-fold cross validation over each data set for all compared approaches, and both mean metric value and standard deviation are recorded for performance comparison.

4.1.3. Compared Approaches

In this paper, the proposed SLEM approach is compared with five state-of-the-art MDC baselines, including BR, CP, BCC, ESC, gMML. Specifically, to solve the MDC problem, BR trains q independent multi-class classifiers, one per class space, while CP trains a single multi-class classifier by regarding the q class spaces as a compound one, where each distinct class combination corresponds to one new class in the compound class space. BCC trains q chain-structured multi-class classifiers, one per class space, where the subsequent classifiers will augment the feature space with predictions of preceding ones and the chain structure is determined by Bayesian learning techniques (Zaragoza et al., 2011). ESC partitions the class spaces into several groups which are treated as a compound class space, and solves the resulting problem via classifier chains (Read et al., 2014a). gMML decomposes the class spaces into a binary-

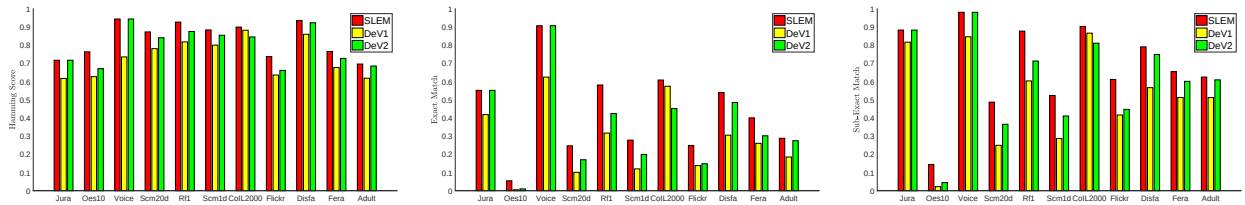
[5]Here, n and x denote numeric and nominal features.

Table 2. Experimental results (mean ± std.) of each MDC approach. In addition, •/○ indicates whether SLEM is statistically superior/inferior to other compared MDC approaches on each data set (pairwise t-test at 0.05 significance level), and the experimental results marked as 'N/A' are unavailable due to "out of memory" error of Libsvm package.

Data Set	*Hamming Score*					
	SLEM	BR	CP	BCC	ESC	gMML
Jura	0.717±0.049	0.586±0.069•	0.570±0.061•	0.568±0.065•	0.558±0.055•	0.606±0.072•
Oes10	0.763±0.014	0.664±0.019•	0.179±0.041•	0.674±0.028•	0.633±0.020•	0.775±0.017○
Voice	0.944±0.010	0.940±0.010	0.916±0.010•	0.938±0.010	0.931±0.009•	0.842±0.009•
Scm20d	0.873±0.004	0.632±0.006•	N/A	0.605±0.008•	N/A	0.600±0.007•
Rf1	0.927±0.002	0.852±0.005•	0.813±0.010•	0.855±0.004•	0.794±0.007•	0.730±0.007•
Scm1d	0.884±0.003	0.725±0.007•	N/A	0.691±0.007•	N/A	0.697±0.007•
CoIL2000	0.899±0.005	0.874±0.005•	0.738±0.006•	0.875±0.005•	0.851±0.008•	0.894±0.004•
Flickr	0.737±0.018	0.715±0.006•	0.658±0.008•	0.715±0.006•	0.651±0.007•	0.779±0.004○
Disfa	0.935±0.002	0.885±0.003•	N/A	0.883±0.003•	0.878±0.003•	0.884±0.003•
Fera	0.765±0.007	0.599±0.008•	N/A	0.593±0.008•	N/A	0.589±0.007•
Adult	0.696±0.006	0.701±0.004○	0.682±0.005•	0.680±0.006•	0.675±0.006•	0.705±0.004○
Data Set	*Exact Match*					
	SLEM	BR	CP	BCC	ESC	gMML
Jura	0.552±0.056	0.329±0.110•	0.326±0.099•	0.304±0.099•	0.298±0.098•	0.368±0.119•
Oes10	0.054±0.036	0.064±0.035	0.077±0.041	0.079±0.045	0.067±0.037	0.079±0.040
Voice	0.907±0.012	0.884±0.017•	0.841±0.016•	0.881±0.017•	0.867±0.016•	0.699±0.017•
Scm20d	0.247±0.015	0.054±0.006•	N/A	0.080±0.009•	N/A	0.052±0.007•
Rf1	0.581±0.008	0.322±0.011•	0.319±0.025•	0.336±0.010•	0.275±0.012•	0.138±0.011•
Scm1d	0.278±0.015	0.115±0.010•	N/A	0.123±0.013•	N/A	0.102±0.009•
CoIL2000	0.608±0.016	0.515±0.012•	0.273±0.012•	0.520±0.010•	0.468±0.019•	0.576±0.015•
Flickr	0.248±0.017	0.187±0.011•	0.125±0.016•	0.187±0.011•	0.114±0.014•	0.287±0.009○
Disfa	0.539±0.015	0.378±0.011•	N/A	0.377±0.011•	0.374±0.011•	0.379±0.011•
Fera	0.400±0.013	0.199±0.013•	N/A	0.196±0.013•	N/A	0.196±0.013•
Adult	0.288±0.011	0.228±0.006•	0.282±0.012	0.272±0.007•	0.269±0.011•	0.230±0.009•
Data Set	*Sub-Exact Match*					
	SLEM	BR	CP	BCC	ESC	gMML
Jura	0.883±0.061	0.844±0.059	0.813±0.040•	0.833±0.056	0.819±0.045•	0.844±0.049
Oes10	0.144±0.051	0.119±0.059	0.107±0.044	0.142±0.055	0.117±0.048	0.176±0.038
Voice	0.981±0.011	0.996±0.005○	0.991±0.005○	0.996±0.005○	0.995±0.005○	0.985±0.011
Scm20d	0.486±0.015	0.105±0.007•	N/A	0.135±0.013•	N/A	0.100±0.009•
Rf1	0.877±0.011	0.655±0.017•	0.580±0.022•	0.669±0.015•	0.542±0.014•	0.375±0.014•
Scm1d	0.523±0.012	0.223±0.016•	N/A	0.206±0.012•	N/A	0.198±0.015•
CoIL2000	0.902±0.013	0.873±0.016•	0.576±0.016•	0.875±0.014•	0.820±0.017•	0.903±0.010
Flickr	0.611±0.035	0.543±0.015•	0.426±0.018•	0.544±0.017•	0.414±0.017•	0.689±0.016○
Disfa	0.791±0.008	0.596±0.011•	N/A	0.588±0.009•	0.575±0.010•	0.590±0.009•
Fera	0.654±0.012	0.387±0.012•	N/A	0.380±0.013•	N/A	0.378±0.013•
Adult	0.624±0.014	0.657±0.010○	0.599±0.008•	0.597±0.012•	0.586±0.011•	0.669±0.008○

valued label space via one-vs-rest strategy and solves the resulting problem via a metric approach (Ma & Chen, 2018).

For BR, CP, BCC, ESC, support vector machine (SVM) is used as the base multi-class classifier to implement each of them. Specifically, the Libsvm package (Chang & Lin, 2011) with default parameter settings is used in experiments. For ESC, the ensemble is constructed with ten base models whose results are combined via majority voting (Read et al., 2014a). For gMML, its parameters λ, t, γ and k are tuned as suggested in (Ma & Chen, 2018). For the proposed SLEM approach, we use random Gaussian matrix to serve as the encoding matrix $\mathbf{A}$ with $s' = s - 1$, and the three trade-off parameters in the formulation (2) are set as $\lambda = 1$, $\gamma_1 = 1$ and $\gamma_2 = 1$.



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

Figure 2. Performance comparision between SLEM and its two variants.

Table 3. Win/tie/loss counts of pairwise t-test (at 0.05 significance level) between SLEM and each MDC approach.

Evaluation metric	SLEM against				
	BR	CP	BCC	ESC	gMML
HS	9/1/1	7/0/0	10/1/0	8/0/0	8/0/3
EM	10/1/0	5/2/0	10/1/0	7/1/0	9/1/1
SEM	7/2/2	5/1/1	8/2/1	6/1/1	5/4/2
In Total	**26/4/3**	**17/3/1**	**28/4/1**	**21/2/1**	**22/5/6**

Table 4. Wilcoxon signed-ranks test for SLEM against its two degenerated versions in terms of each evaluation metric (significance level $\alpha = 0.05$; p-values shown in the brackets).

SLEM **versus**	Evaluation metric		
	HS	EM	SEM
DeV1	**win**[9.77e-04]	**win**[9.77e-04]	**win**[9.77e-04]
DeV2	**win**[3.91e-03]	**win**[3.91e-03]	**win**[3.91e-03]

4.2. Experimental Results

Table 2 shows the experimental results of SLEM and all compared approaches over the benchmark data sets in terms of each evaluation metric.[6] We also conduct pairwise t-test (at 0.05 significance level) to show whether SLEM is statistically superior/inferior to other compared MDC approaches on each data set, and Table 3 summarizes the corresponding win/tie/loss counts.

According to the reported experimental results, the following observations can be made:

- Among all the 144 configurations (11 data set $\times$ 5 compared approaches $\times$ 3 evaluation metrics [excluding the 21 'N/A' cases]), SLEM achieves superior or at least comparable performance against the five compared approaches in 132 cases.
- The two approaches BR and CP represent two extreme MDC baselines which consider none or exhaustive class dependencies, respectively. As shown in Table 3, SLEM achieves 26 and 17 superior cases against BR and CP respectively, which clearly validates the effectiveness of SLEM's dependency modeling strategy.
- The two approaches BCC and ESC specially consider the class dependencies via chain structure or superclass partition in the original label space. The overwhelming advantage of SLEM over BCC and ESC indicates that it is beneficial to learn predictive models in the encoded label space.
- It is shown that SLEM achieves 6 loss cases against gMML, which is relatively larger than other compared approaches. The gMML approach learns predictive model by utilizing the distance metric learning mechanism, which can also be introduced into label encoding in the future.

4.3. Further Analysis

In this subsection, we further compare the performance of SLEM with its two degenerated versions to analyze the effectiveness of SLEM's label encoding strategy. We denote the two variants as DeV1 and DeV2 respectively:

- DeV1: This variant directly obtains $\boldsymbol{v}_i$ by conducting one-hot conversion on $\boldsymbol{y}_i$, i.e., $\boldsymbol{v}_i = \Phi_{\mathcal{Y}}(\boldsymbol{y}_i)$, and the decoding phase is changed accordingly. In other words, DeV1 omits the pairwise grouping operation $\mathcal{P}_{\mathcal{Y}}(\cdot)$.
- DeV2: This variant separately deals with the $\lceil \frac{q}{2} \rceil$ entries in $\boldsymbol{u}_i$ with the same encoding-decoding procedure between $\boldsymbol{v}_i$ and $\hat{\boldsymbol{v}}$ (cf. Figure 1), each of which corresponds to one class space in $\mathcal{U}$. In other words, DeV2 doesn't encode the heterogeneous class spaces into an integrated label space and only considers the dependencies between class spaces grouped into one pair.

The detailed comparative results are shown in Figure 2. Moreover, we also employ *Wilcoxon signed-ranks test* at 0.05 significance level (Demšar, 2006) to test the statistical relationship between SLEM and its two variants, and the

[6] In Table 2, there are a total of 21 cases marked as 'N/A' whose experimental results are unavailable due to "out of memory" error of Libsvm package. The error is caused by the high computational complexity of the CP and ESC over the corresponding data sets.

corresponding test results with p-values shown in brackets are summarized in Table 4.

As shown in Table 4, SLEM achieves statistically superior performance against its two degenerated versions in terms of each evaluation metric which clearly validates the effectiveness of SLEM's label encoding strategy. To be more specific, the superiority of SLEM against DeV1 shows the benefits of the pairwise grouping operation, and the superiority of SLEM against DeV2 shows the benefits of encoding the heterogeneous class spaces into an integrated label space.

5. Conclusion

In this paper, we investigate the label encoding techniques for multi-dimensional classification. Different from most existing MDC approaches, the proposed SLEM approach learns predictive model in the transformed label space instead of the original one. Specifically, SLEM transforms the output space into a new one via three cascaded operations, including pairwise grouping, one-hot conversion and sparse linear encoding. Within the transformed label space, SLEM learns a multi-output regression model based on the training examples, and then obtains a real-valued label vector for unseen instance by feeding it into the learned multi-output regressor. The final predicted class vector is obtained by conducting decoding procedure corresponding to the inverse operations of encoding steps based on the predicted real-valued label vector. We compare the performance of SLEM with five state-of-the-art MDC approaches over eleven benchmark data sets, and the experimental results clearly show the superiority of SLEM against the baselines.

References

Al Muktadir, A. H., Miyazawa, T., Martinez-Julia, P., Harai, H., and Kafle, V. P. Multi-target classification based automatic virtual resource allocation scheme. *IEICE Transactions on Information and Systems*, 102(5):898–909, 2019.

Baraniuk, R., Davenport, M., Devore, R., and Wakin, M. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3): 253–263, 2008.

Bauschke, H. H., Bolte, J., and Teboulle, M. A descent lemma beyond Lipschitz gradient continuity: First-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, 2017.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Benjumeda, M., Bielza, C., and Larrañaga, P. Tractability of most probable explanations in multidimensional Bayesian network classifiers. *International Journal of Approximate Reasoning*, 93:74–87, 2018.

Bielza, C., Li, G., and Larrañaga, P. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011.

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

Borchani, H., Varando, G., Bielza, C., and Larrañaga, P. A survey on multi-output regression. *WIREs Data Mining and Knowledge Discovery*, 5:216–233, 2015.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):Article 27, 2011.

Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7: 1–30, 2006.

Donoho, D. L. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

Gibaja, E. and Ventura, S. A tutorial on multilabel learning. *ACM Computing Surveys*, 47(3):Article 52, 2015.

Gil-Begue, S., Bielza, C., and Larrañaga, P. Multi-dimensional Bayesian network classifiers: A survey. *Artificial Intelligence Review*, 54:519–559, 2021.

Huang, J., Li, G., Huang, Q., and Wu, X. Learning label-specific features and class-dependent labels for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3309–3323, 2016.

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

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

Jia, B.-B. and Zhang, M.-L. Multi-dimensional classification via stacked dependency exploitation. *Science China Information Sciences*, 63(12):Article 222102, 2020c.

Liu, W. and Shen, X. Sparse extreme multi-label learning with oracle property. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 4032–4041, Long Beach, CA, USA, 2019.

Liu, W., Tsang, I. W., and Müller, K. An easy-to-hard learning paradigm for multiple classes and multiple labels. *Journal of Machine Learning Research*, 18(94):1–38, 2017.

Liu, W., Xu, D., Tsang, I. W., and Zhang, W. Metric learning for multi-output tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):408–422, 2019.

Ma, Z. and Chen, S. Multi-dimensional classification via a metric approach. *Neurocomputing*, 275:1121–1131, 2018.

Read, J., Bielza, C., and Larrañaga, P. Multi-dimensional classification with super-classes. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1720–1733, 2014a.

Read, J., Martino, L., and Luengo, D. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535–1546, 2014b.

Reeve, H. and Kaban, A. Optimistic bounds for multi-output learning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8030–8040, Virtual Event, Worldwide, 2020.

Rodríguez, J. D., Pérez, A., Arteta, D., Tejedor, D., and Lozano, J. A. Using multidimensional Bayesian network classifiers to assist the treatment of multiple sclerosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 42(6):1705–1715, 2012.

Schölkopf, B. and Smola, A. J. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, Cambridge, MA, USA, 2002.

Shatkay, H., Pan, F., Rzhetsky, A., and Wilbur, W. J. Multi-dimensional classification of biomedical text: Toward automated, practical provision of high-utility text to diverse users. *Bioinformatics*, 24(18):2086–2093, 2008.

Shen, X., Liu, W., Tseng, I. W., Sun, Q.-S., and Ong, Y.-S. Compact multi-label learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 4066–4073, New Orleans, LA, USA, 2018.

Tai, F. and Lin, H.-T. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9): 2508–2542, 2012.

Tropp, J. A. and Gilbert, A. C. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

Verma, S. P., Uscanga-Junco, O. A., and Díaz-González, L. A statistically coherent robust multidimensional classification scheme for water. *Science of the Total Environment*, 750:Article 141704, 2021.

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

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

Zhang, M.-L. and Zhou, Z.-H. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

Zhu, M., Liu, S., and Jiang, J. A hybrid method for learning multi-dimensional Bayesian network classifiers based on an optimization model. *Applied Intelligence*, 44(1):123–148, 2016.