Efficient Decomposition Selection for Multi-class Classification

Yawen Chen[†], Zeyi Wen[*], Bingsheng He[§‡], Jian Chen[†]
[†]South China University of Technology, [*]The University of Western Australia,
[§]School of Computing, National University of Singapore; [‡]NUS Center for Trust Internet and Community
ywchenscut@gmail.com, zeyi.wen@uwa.edu.au, hebs@comp.nus.edu.sg, ellachen@scut.edu.cn

Abstract—Choosing a decomposition method for multi-class classification is an important trade-off between efficiency and predictive accuracy. Trying all the decomposition methods to find the best one is too time-consuming for many applications, while choosing the wrong one may result in large loss on predictive accuracy. In this paper, we propose an automatic decomposition method selection approach called "D-Chooser", which is lightweight and can choose the best decomposition method accurately. D-Chooser is equipped with our proposed difficulty index which consists of sub-metrics including distribution divergence, overlapping regions, unevenness degree and relative size of the solution space. The difficulty index has two intriguing properties: 1) fast to compute and 2) measuring multi-class problems comprehensively. Extensive experiments on real-world multi-class problems show that D-Chooser achieves an accuracy of 80.56% in choosing the best decomposition method. It can choose the best method in just a few seconds, while existing approaches verify the effectiveness of a decomposition method often takes a few hours. We also provide case studies on Kaggle competitions and the results confirm that D-Chooser is able to choose a better decomposition method than the winning solutions.

Index Terms—Machine Learning, Multi-class classification, Decomposition method.

1 Introduction

Multi-class classification is used in many real-world applications such as medical data analysis [1], [2] and object detection [3]. One fundamental way to tackle the multi-class classification problem is through decomposition into multiple binary classification problems. Among the various decomposition methods [4], one-vs-all, one-vs-one and error-correcting output codes (ECOC) are the mainstream decomposition methods for multi-class classification. There are studies aiming to learn the optimal codebook for ECOC [5], [6] using optimization, which theoretically considers all of the three decomposition methods. However, a study indicates that learning the codebook is infeasible due to the high computation cost [6]. Hence, only the mainstream decomposition methods (i.e., one-vs-all, one-vs-one and ECOC) have been implemented in popular libraries such as scikit-learn [7], WEKA [8], LibSVM [9] and ThunderSVM [10]. As a result, most of the machine learning and data mining practitioners only use the three mainstream methods either implicitly or explicitly.

Which decomposition method is more effective is problem dependent [11], as illustrated in Figure 1. More detailed experimental setups for Figure 1 are shown in Section 4. Some pre-eminent experts may be able to tell which decomposition method to use for a problem magically, and that decomposition method appears to be the best for the problem. However, for many practitioners, to choose the best decomposition method for building an effective classifier, they need to try one-vs-all, one-vs-one and ECOC for their problems at hand. This process is very time-consuming, since the classification

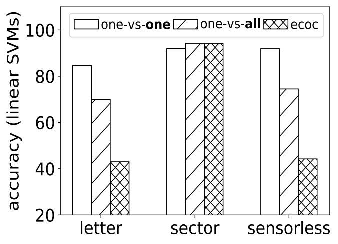


Fig. 1: Best decomposition is problem dependent.

algorithms often have existing hyper-parameters to tune (e.g., C and γ in SVMs with RBF kernel), and the decomposition method selection adds one more hyper-parameter to the classification algorithms. Let us consider an example of using SVMs with the radial basis function (RBF) kernel, and consider a 10 by 10 grid with 10 candidate values for γ of RBF and 10 candidate values for the regularization constant C. Then, we need to train and test 100 multi-class SVM classifiers in total for each decomposition method. As we have three common decomposition methods to test, totally we need to train and test 300 multi-class SVM classifiers. This cost of verifying the effectiveness for all the decomposition methods is prohibitively high in many applications. In contrast, if we do not try all the decomposition methods, choosing a wrong decomposition method may result in great penalty on the effectiveness of the multi-class classifier.

It is important but challenging to choose the best decomposition method for a multi-class problem. To help practitioners choose the best decomposition method efficiently and accurately, we propose an automatic decomposition method selection approach called "D-Chooser". D-Chooser

- *Correspondence to Zeyi Wen at zeyi.wen@uwa.edu.au* is equipped with a novel difficulty index which is easy to compute and can measure the multi-class learning problem comprehensively using our proposed data-aware sub-metrics. Thus, the effectiveness of one-vs-all, one-vs-one and ECOC decomposition can be compared correctly and efficiently via the difficulty indices.

D-Chooser has the following key steps. First, we represent each class in a binary problem with representative data, and then a probability distribution is constructed to capture the key properties of the class. Second, we compute the average divergence of all the binary problems in each decomposition with novel techniques to make the average divergences of the three decomposition methods more comparable. Moreover, overlapping regions, unevenness degree and relative size of the solution space are computed based on the given data set. Third, we select the decomposition method with the smallest difficulty index which is calculated based on the sub-metrics computed in the second step. All the steps of D-Chooser together aim to comprehensively measure the difficulty of the binary problems, so that the best decomposition method can be chosen automatically by D-Chooser. To summarize, the major contributions of this work are listed below.

- We propose D-Chooser for decomposition method selection. To comprehensively measure multi-class problems in D-Chooser, we develop a novel difficulty index which considers a series of data-aware sub-metrics including distribution divergence, overlapping regions, unevenness degree and relative size of the solution space.
- To compute the sub-metrics efficiently, we develop clustering based techniques to represent the binary problems, such that the distributions can be quickly constructed based on the clusters using our derived equations. The time complexity of our D-chooser is $\mathcal{O}(\mathtt{K}|\mathtt{D}|n)$ where $\mathtt{K}$, $|\mathtt{D}|$ and n are the number of classes, the number of the total training instances and the data dimensionality, respectively. In comparison, the complexity of verifying a decomposition method is much larger, e.g., $\mathcal{O}(\Gamma \cdot \mathtt{K}|\mathtt{D}|^2 n)$ [12] where Γ is the number of iterations in parameter-tuning. Thus, D-Chooser only needs a few seconds to select the best decomposition method in the tested data sets, while verifying a decomposition method often takes a few hours.
- We conduct extensive experiments and the experimental results show that D-Chooser achieves an accuracy of 80.56% in choosing the best decomposition method. We also provide case studies on Kaggle competitions and demonstrate that D-Chooser is able to choose a better decomposition method than the winning solutions.

The remainder of this paper is structured as follows. We first present the preliminaries in Section 2. Then, we elaborate the details of D-Chooser in Section 3 and we show our experimental results in Section 4. Finally, we discuss the related work in Section 5 and draw a conclusion of the paper in Section 6.

2 Preliminaries

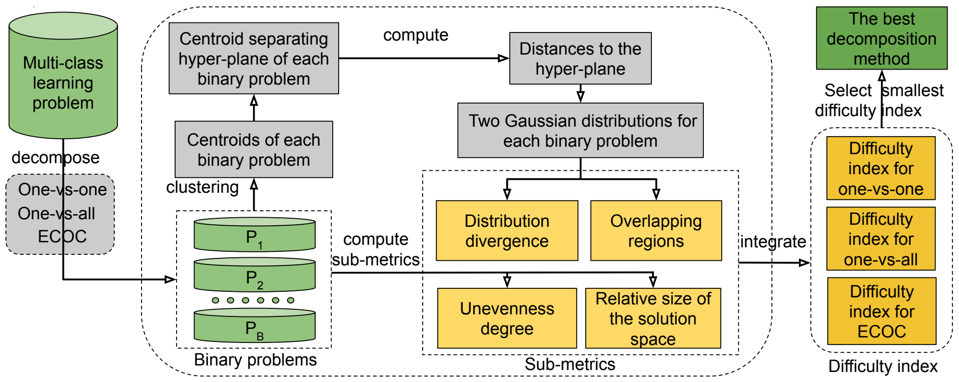
A multi-class classification problem can be decomposed into a number of binary problems. We present three mainstream decomposition methods next.

One-vs-all decomposition: The one-vs-all decomposition learns a binary classifier for each class, where the class is distinguished from all the other classes. Formally, given a multi-class classification problem with $\mathtt{K}$ classes, the total number of binary classifiers is $\mathtt{K}$. There are many ways to combine the results of the $\mathtt{K}$ binary classifiers. In this paper, we use a widely used approach where the final prediction result is the class with the largest decision value among the $\mathtt{K}$ binary classifiers [13]. Formally, the predicted class is computed by $class = \operatorname{argmax}_{i \in \{1,2,\ldots,\mathtt{K}\}} v_i$, where v_i is the decision value computed from the i-th binary classifier.

One-vs-one decomposition: The one-vs-one decomposition consists of dividing the multi-class classification problem into as many binary classification problems as all the possible combinations between pairs of classes, so one binary classifier is learned to discriminate between each pair of classes. Then, the outputs of these binary classifiers are combined in order to predict the final class of an instance. Formally, given a multi-class classification problem with $\mathtt{K}$ classes, the total number of binary classifiers is $\frac{\mathtt{K}(\mathtt{K}-1)}{2}$. We use a common approach, namely majority voting [14], [15], to decide the final prediction result. Specifically, the class with the maximum vote among the $\mathtt{K}$ classes serves as the predicted class and is defined by $class = \operatorname{argmax}_{i \in \{1,2,\ldots,\mathtt{K}\}} \sum_{1 \le i \neq j \le \mathtt{K}} s_{ij}$, where s_{ij} is 1 if the predicted class is i in the binary classification problem containing instances of classes i and j; s_{ij} is 0 otherwise.

Error-correcting output codes decomposition: The error-correcting output codes (ECOC) decomposition method replaces the original class label by a "codeword" [16]. A codeword is a vector of $\mathtt{E}$ dimensions where each dimension is either 1 or -1 and $\mathtt{E}$ is a user defined hyper-parameter. For better generality, the codeword is generated randomly in practice. The codewords of the $\mathtt{K}$ classes together form a $\mathtt{K} \times \mathtt{E}$ matrix denoted by $\mathtt{M}$ which is also called "code book". Each column of the matrix corresponds to a binary classifier. Table 1 shows an example codeword matrix for a data set with four classes and $\mathtt{E}$ is equal to six which means six binary classifiers need to be trained. An instance $\boldsymbol{x}$ of class i is a positive instance for the j^{th} binary classifier if and only if $\mathtt{M}_{ij}$ is 1; it is a negative instance otherwise. For example, the classifier B_3 in Table 1 treats the instances of class 1 and 2 as positive instances, and likewise treats the instances of class 3 and 4 as negative instances.

When predicting the label of an unseen instance, each binary classifier outputs a "-1" or "1", and the results of all the binary classifiers form a vector of $\mathtt{E}$ dimensions. Then, this output vector is compared against the codewords in the matrix $\mathtt{M}$. The class with the most similar codeword to the output vector is chosen as the predicted label for this unseen instance. The common methods to measure the similarity between the vector and a codeword include Euclidean distance and Hamming distance [17]. Compared with one-vs-all and one-vs-one, the advantage of ECOC is that the number of binary classifiers is controllable by the users through the hyper-parameter $\mathtt{E}$.



Multi-class learning problem
decompose
One-vs-one
One-vs-all
ECOC
Centroid separating hyper-plane of each binary problem
compute
Distances to the hyper-plane
Centroids of each binary problem
clustering
Two Gaussian distributions for each binary problem
P1
P2
PB
Binary problems
compute sub-metrics
Distribution divergence
Overlapping regions
Unevenness degree
Relative size of the solution space
Sub-metrics
integrate
The best decomposition method
Select smallest difficulty index
Difficulty index for one-vs-one
Difficulty index for one-vs-all
Difficulty index for ECOC
Difficulty index

Fig. 2: Process of the best decomposition method selection in D-Chooser.

TABLE 1: An example ECOC codeword matrix.

label \ classifier	B_1	B_2	B_3	B_4	B_5	B_6
1	1	-1	1	-1	-1	1
2	-1	-1	1	-1	1	1
3	1	1	-1	1	-1	-1
4	-1	1	-1	-1	1	1

3 Our Proposed D-Chooser

Here, we elaborate our proposed Decomposition method Chooser (D-Chooser). Designing the decomposition chooser has two key challenges. First, the decomposition chooser should be lightweight, since an inefficient decomposition chooser downgrades its advantage over verifying the decomposition methods. Second, the decomposition chooser should select the best method correctly, as choosing the wrong one leads to a prominent loss on the predictive accuracy.

3.1 Design rationale

We develop *difficulty index* and integrate it into D-Chooser, such that the effectiveness among the decomposition methods can be compared correctly and efficiently. In order to achieve a lightweight decomposition chooser, we develop clustering based techniques to construct the distributions, such that the distribution based sub-metrics including distribution divergence and overlapping regions can be computed efficiently. With the cluster centroids, the distances to the separating hyper-plane of the clusters are measured for the training instances, and a probability distribution is constructed based on these distances. We measure the distribution divergence and overlapping regions of all the binary problems, with mechanisms to make the distribution divergence and overlapping regions of different decomposition methods comparable.

To allow the difficulty index to measure a problem from different perspectives, the unevenness degree and relative size of the solution space sub-metrics are combined into the index. These two sub-metrics can be efficiently computed based on the meta information (e.g., cardinality) of the problems. The overview of the whole process of the best decomposition method selection is shown in Figure 2. The proposed difficulty index (shown at the right of the figure) consists of four data-aware sub-metrics for comprehensively measuring multi-class problems. The four sub-metrics are distribution divergence sub-metric, overlapping regions sub-metric, unevenness degree sub-metric and relative size of solution space sub-metric. Next, we formulate the problem and present the details of the sub-metrics.

3.2 Problem formulation

The decomposition method selection problem is to find a decomposition method which leads to the highest predictive accuracy. Formally, the problem is to find the decomposition method with the minimum difficulty index shown as follows.

$$\arg\min_{m\in\mathcal{M}} I_m(\mathcal{P}) = \boldsymbol{\omega}^\intercal \boldsymbol{z}^m \tag{1}$$

where $I_m(\mathcal{P})$ is the difficulty index of the multi-class problem $\mathcal{P}$ under the decomposition method m, $\mathcal{M}$ is a set of decomposition methods which is $\{\text{ova}, \text{ovo}, \text{ecoc}\}$, $\boldsymbol{\omega}$ is a weight vector and $\boldsymbol{z}^m$ is a vector of sub-metrics for measuring the multi-class problem $\mathcal{P}$. In this work, we propose two metrics based on hyper-plane construction which can work on linearly separable problems. For non-linearly separable problems, our proposed techniques support kernel functions to map the problems from their original data spaces to high dimensional spaces induced by the kernel functions, where the problems may become linearly separable [18].

3.2.1 Sub-metrics of the difficulty index

As we have formulated in Problem (1), the difficulty index $I(\mathcal{P})$ is equal to $\boldsymbol{\omega}^\intercal \boldsymbol{z}$, and $\boldsymbol{z}$ consists of a set of sub-metrics[1]. In this paper, the sub-metrics considered include distribution divergence, overlapping regions, unevenness degree and relative size of the solution space. Formally, the vector of sub-metrics is defined as $\boldsymbol{z} = (z_1, z_2, z_3, z_4)$ where z_1, z_2, z_3 and z_4 measure distribution divergence, overlapping regions,

1. For ease of presentation, m is omitted in $I_m(\mathcal{P})$ and $\boldsymbol{z}^m$.

unevenness degree and relative size of the solution space, respectively. The four sub-metrics are defined as follows.

$$\begin{aligned}
z_1 &= \frac{1}{|\mathcal{B}|}\sum_{(o,o')\in\mathcal{B}} \int g_o(u)\log\frac{g_o(u)}{g_{o'}(u)}du \\
z_2 &= \frac{1}{|\mathcal{B}|}\sum_{(o,o')\in\mathcal{B}} \int \min(g_o(u), g_{o'}(u))du \\
z_3 &= \frac{1}{|\mathcal{B}|\cdot 2}\sum_{(o,o')\in\mathcal{B}} \left(\frac{|\mathtt{D}_o|}{|\mathtt{D}_{o'}|}+\frac{|\mathtt{D}_{o'}|}{|\mathtt{D}_o|}\right) \\
z_4 &= \frac{1}{|\mathcal{B}|\cdot n}\sum_{(o,o')\in\mathcal{B}} (|\mathtt{D}_o|+|\mathtt{D}_{o'}|)
\end{aligned} \tag{2}$$

where $\mathcal{B}$ is the set of binary problems, $g_o(u)$ is a distance distribution for the o class of the binary problem, and $g_{o'}(u)$ is that for the o' class of the binary problem; $|\mathtt{D}_o|$ and $|\mathtt{D}_{o'}|$ are the number of instances in the o class and the o' class, respectively; n is the data dimensionality.

3.3 Sub-metrics computed based on distributions

Here we present two sub-metrics (i.e., z_1 and z_2) based on distributions in the difficulty index: one for measuring distribution divergence between the two classes of a binary problem, and the other for measuring the overlapping regions. In the following, we first present the details of constructing distributions. Then, we present the distribution divergence sub-metric and the overlapping region sub-metric.

3.3.1 *Constructing the distributions*

Construction of the distributions of training data has three key steps: finding centroids, computing the hyper-plane and estimating the distribution. The overview of the steps for constructing the distribution is shown in Figure 3. We elaborate more details of the three steps in the following.

Step 1: Finding centroids

A straightforward way of capturing the essential information of a binary problem is that each class of the training data set is represented by its centroid. The triangles on the upper right of Figure 3 represent the centroids of two classes respectively. The empty circles indicate the instances from the positive class and the solid circles represent instances belonging to the negative class. To achieve good efficiency, we can use the formula to compute the centroid directly, instead of applying the clustering algorithm. Moreover, directly applying existing algorithms to find the separating hyper-plane for the centroids cannot reduce the time complexity. Hence, we need to mathematically derive a fast way to compute the hyper-plane and the distance of a training instance to the hyper-plane.

By exploiting this property of k-means and with the formulas we derive, the time complexity of the k-means clustering algorithm to represent class i is only $\mathcal{O}(|\mathtt{D}_i|\cdot n)$, which is remarkably lower than $\mathcal{O}(|\mathtt{D}_i|^2\cdot n)$. Moreover, the hyper-plane separating the centroids of the binary problem can be computed in a time complexity of $\mathcal{O}(n)$ as we will show later of this section, which is again dramatically cheaper than $\mathcal{O}(n^2k^2)$. Next, we present the details of computing the centroid of their original data space and of the data space induced by the kernel functions.

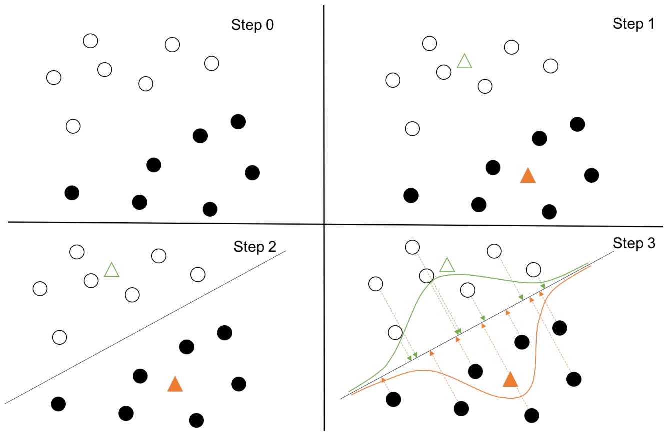


Fig. 3: Constructing the distribution.

Computing the centroid in their original data space: Many multi-class classification problems can be solved in their original data space by classifiers such as linear SVMs and logistic regression. Here, we present computing the centroid in the original data space. The centroid $\bar{\boldsymbol{x}}_i$ of a class i in a binary classification problem denoted by $\mathtt{P}_{ij}$, where j represents the other class of the binary classification problem. Let $\mathtt{D}_i$ denote the set of training instances of class i in $\mathtt{P}_{ij}$. We compute the centroid using the following equation for $k=1$ by $\bar{\boldsymbol{x}}_i = \frac{1}{|\mathtt{D}_i|}\sum_{t=1}^{|\mathtt{D}_i|}(\boldsymbol{x}_{t1}, \boldsymbol{x}_{t2}, ..., \boldsymbol{x}_{tn})$, where n is the number of dimensions of the training instances, $\boldsymbol{x}_{te}$ denotes the value of the e-th dimension of the training instance $\boldsymbol{x}_t$, and $|\mathtt{D}_i|$ is the number of instances in $\mathtt{D}_i$. The above equation means the e-th dimension of the centroid $\bar{\boldsymbol{x}}_i$ is the average value of the e-th dimension on $\mathtt{D}_i$.

Computing the centroid when kernel functions are used: Some classifiers such as SVMs and logistic regression apply kernel functions to map the problem to higher dimensional data spaces. We have two options to find the centriods in the space induced by the kernel function: (i) compute the centroid in the space induced by the kernel function; (ii) map the centroid computed in the original data space with the kernel function. The time complexity of the first option is $\mathcal{O}(|\mathtt{D}_i|^2\cdot n)$ for class i, because the whole Hessian matrix needs to be computed [19]. The second option has a lower time complexity of $\mathcal{O}(|\mathtt{D}_i|\cdot n)$ which is the cost of computing the centroids. Therefore, we use the second option to compute the centroid directly using $\mathtt{D}_i$, and map the centroid to the high dimensional space with the kernel function.

Step 2: Computing the hyper-plane

In the following, we show how the hyper-plane can be efficiently constructed. Without loss of generality, we suppose each class has one centroid, as a centroid can represent multiple centroids through compounding [20]. Hence, only two centroids are used to represent the binary classification problem. Therefore, the hyper-plane is the perpendicular bisecting hyper-plane of a line segment connecting the two centroids, and is perpendicular to the line segment at its midpoint. The hyper-plane is depicted as the solid line in Figure 3. Suppose $\bar{\boldsymbol{x}}_o$ and $\bar{\boldsymbol{x}}_{o'}$ are two centroids of the binary classification problem $\mathcal{P}_B$, $\bar{\boldsymbol{x}}_o$ is the centroid of the positive class, and $\bar{\boldsymbol{x}}_{o'}$ is the centroid of the negative class. Let $\mathtt{L}_{\bar{\boldsymbol{x}}_o\bar{\boldsymbol{x}}_{o'}}$ denote the line passing through $\bar{\boldsymbol{x}}_o$ and $\bar{\boldsymbol{x}}_{o'}$. The perpendicular bisecting hyper-plane $\mathtt{H}$ of the line segment connecting $\bar{\boldsymbol{x}}_o$ and $\bar{\boldsymbol{x}}_{o'}$ can be presented as $\boldsymbol{w}^\intercal\boldsymbol{x}+b=0$. Since $\mathtt{L}_{\bar{\boldsymbol{x}}_o\bar{\boldsymbol{x}}_{o'}}$ is perpendicular to $\mathtt{H}$ and $\boldsymbol{w}^\intercal$ is also perpendicular to $\mathtt{H}$ (i.e., $\boldsymbol{w}^\intercal$ is the normal vector to $\mathtt{H}$), $\boldsymbol{w}^\intercal$ can be written as $\boldsymbol{w}^\intercal = \beta\cdot(\bar{\boldsymbol{x}}_o-\bar{\boldsymbol{x}}_{o'})^\intercal$, where β is an unknown real value (i.e., $\beta\in\mathbb{R}$). Let $\boldsymbol{x}_m$ be the midpoint of the line segment of $\bar{\boldsymbol{x}}_o$ to $\bar{\boldsymbol{x}}_{o'}$, and $\boldsymbol{x}_m=\frac{\bar{\boldsymbol{x}}_o+\bar{\boldsymbol{x}}_{o'}}{2}$. We know that point $\boldsymbol{x}_m$ is on $\mathtt{H}$, and from the above equation, we can obtain the following equation.

$$b=-\beta\cdot(\bar{\boldsymbol{x}}_o-\bar{\boldsymbol{x}}_{o'})^\intercal\boldsymbol{x}_m=\beta\cdot(\bar{\boldsymbol{x}}_{o'}-\bar{\boldsymbol{x}}_o)^\intercal(\frac{\bar{\boldsymbol{x}}_o+\bar{\boldsymbol{x}}_{o'}}{2})$$

Then, the perpendicular bisecting hyper-plane $\mathtt{H}$ can be presented as follows by substituting b and $\boldsymbol{w}$.

$$\beta\cdot(\bar{\boldsymbol{x}}_o-\bar{\boldsymbol{x}}_{o'})^\intercal\boldsymbol{x}+\beta\cdot(\bar{\boldsymbol{x}}_{o'}-\bar{\boldsymbol{x}}_o)^\intercal(\frac{\bar{\boldsymbol{x}}_o+\bar{\boldsymbol{x}}_{o'}}{2})=0 \quad (3)$$

The perpendicular bisecting hyper-plane is a simple and computationally efficient technique. Moreover, our experimental results show that our method is able to select the best decomposition method with accuracy higher than the more complex hyper-plane found by optimization. It is worthy pointing out that this hyper-plane $\mathtt{H}$ is unknown, even given the centroids $\bar{\boldsymbol{x}}_o$ and $\bar{\boldsymbol{x}}_{o'}$, due to the unknown constant β. The constant β is used to compute the distance as presented in Step 3. The sign of β controls the relative position of an instance to the hyper-plane (e.g., $+1$ represents the instance on the left side and -1 represents the instance on the right side).

Our method can work on multiple centroids in each class. To support that, we can find the centroids of each class of the binary problem using k-means. Then, we have $2k$ centroids in total (i.e., each class has k centroids). Finding the separating hyper-plane for these $2k$ centroids can be done through training a binary SVM classifier. However, using multiple centroids needs higher computation cost than using one centroid which can be computed with a formula.

Step 3: Estimating a probability distribution

Although other distributions are applicable in our proposed difficulty index, we focus our discussion on constructing a Gaussian distribution. To construct the distribution, we first compute the distance between a training instance of $\mathtt{D}_o$ to the hyper-plane $\mathtt{H}$ using the formula below.

$$d_t=\frac{\boldsymbol{w}^\intercal\boldsymbol{x}_t+b}{\|\boldsymbol{w}\|}=\frac{\beta}{|\beta|}\cdot\frac{(\bar{\boldsymbol{x}}_o-\bar{\boldsymbol{x}}_{o'})^\intercal(\boldsymbol{x}_t-\frac{\bar{\boldsymbol{x}}_o+\bar{\boldsymbol{x}}_{o'}}{2})}{\|\bar{\boldsymbol{x}}_o-\bar{\boldsymbol{x}}_{o'}\|} \quad (4)$$

We present more details on distance computation (e.g., distances in spaces induced by kernels) later in this section. It is important to note that the distance d_t can be either positive or negative depending on β, and the denominator of the above equation is a constant. Also note that $\beta\in\mathbb{R}$ and $\frac{\beta}{|\beta|}$ is either -1 or 1, and $\frac{\beta}{|\beta|}$ can be cancelled out when computing the distribution divergence and overlapping regions.

All the distances of the instances in $\mathtt{D}_o$ are denoted by $\{d_1,d_2,...,d_{|\mathtt{D}_o|}\}$. We assume these distances follow a Gaussian probability distribution (i.e., $g_o(u)\sim(\mu_o,\sigma_o^2)$), and denote the probability density function by $p(d)=\frac{1}{\sqrt{2\pi}\sigma_o}\exp(-\frac{\|d-\mu_o\|^2}{2\sigma_o^2})$, where d is a variable in $\mathbb{R}$. The mean μ_o is estimated using $|\mathtt{D}_o|$ as follows.

$$\hat{\mu}_o=\frac{1}{|\mathtt{D}_o|}\sum_{t=1}^{|\mathtt{D}_o|}d_t=\frac{\beta}{|\mathtt{D}_o|\cdot|\beta|}\sum_{t=1}^{|\mathtt{D}_o|}f(\boldsymbol{x}_t) \quad (5)$$

where $f(\boldsymbol{x}_t)=\frac{(\bar{\boldsymbol{x}}_o-\bar{\boldsymbol{x}}_{o'})^\intercal(\boldsymbol{x}_t-\frac{\bar{\boldsymbol{x}}_o+\bar{\boldsymbol{x}}_{o'}}{2})}{\|\bar{\boldsymbol{x}}_o-\bar{\boldsymbol{x}}_{o'}\|}$. It is straightforward to extend $f(\boldsymbol{x}_t)$ to the kernel space. Similarly, we can estimate σ_o^2 by the equation below.

$$\hat{\sigma}_o^2=\frac{1}{|\mathtt{D}_o|-1}\sum_{t=1}^{|\mathtt{D}_o|}\left(f(\boldsymbol{x}_t)-\frac{1}{|\mathtt{D}_o|}\sum_{l=1}^{|\mathtt{D}_o|}f(\boldsymbol{x}_l)\right)^2 \quad (6)$$

By using Equations (5) and (6), we can construct the Gaussian probability distribution for the training instances of other classes (e.g., $\mathtt{D}_{o'}$). We illustrate the distributions on the lower right of Figure 3. It is important to point out that Equation (6) does not have the unknown constant β, and therefore $\hat{\sigma}_o^2$ can be computed given the data set $\mathtt{D}_o$. In contrast, $\hat{\mu}_o$ cannot be computed given $\mathtt{D}_o$ due to the existence of the unknown constant β in Equation (5). We will show how to cancel out the unknown constant when we need to use $\hat{\mu}_o$ in computing divergence.

Algorithm 1 summarizes the process of constructing the Gaussian probability distribution of the data set. Given the training data set and hyper-plane, we can compute the distances of the training instances to the hyper-plane (cf. Line 3 of Algorithm 1). Using those distances, we can compute the statistics to construct the distance distribution (cf. Line 4 of Algorithm 1).

Algorithm 1: Constructing the Gaussian distribution

```
Input: Data set $\mathtt{D}_i$ and hyper-plane $\mathtt{H}$
Output: Mean $\hat{\mu}_i$ and variance $\hat{\sigma}_i$
1 $d_1 \leftarrow 0, d_2 \leftarrow 0, ..., d_{|\mathtt{D}_i|} \leftarrow 0$
2 for $t=1$ to $|\mathtt{D}|_i$ and $\boldsymbol{x}_t \in \mathtt{D}_i$ do
3     $d_t \leftarrow$ ComputeDistance($\boldsymbol{x}_t$, $\mathtt{H}$)        //Eq. (4)
4 $\hat{\mu}_i, \hat{\sigma}_i^2 \leftarrow$ ComputeStats($d_1$, $d_2$, ..., $d_{|\mathtt{D}_i|}$)  //Eqs. (5) & (6)
```

3.3.2 *Computing the distribution divergence*

Here we first present details of computing the distribution divergence of a binary classification problem. Then we discuss computing the aggregated distribution divergence of one-vs-all, one-vs-one and ECOC, such that the divergence can be compared among the three decomposition methods. Given a binary classification problem $\mathcal{P}_B$, we can construct two Gaussian distributions $g_o(u)\sim(\mu_o,\sigma_o^2)$ and $g_{o'}(u)\sim(\mu_{o'},\sigma_{o'}^2)$. In this paper, we focus our presentation on using KL divergence to measure the difference of two distributions, although other statistical distances, such as Hellinger distance [21] and Jensen-Shannon divergence [22], can be easily integrated into D-Chooser. We show the equations of computing the KL divergence from $g_{o'}(u)$ to $g_o(u)$. By the definition of KL divergence, we have

$$\mathrm{Dvg}(g_o(u)\|g_{o'}(u))=\log\frac{\hat{\sigma}_{o'}}{\hat{\sigma}_o}+\frac{\hat{\sigma}_o^2+(\hat{\mu}_o-\hat{\mu}_{o'})^2}{2\hat{\sigma}_{o'}^2}-\frac{1}{2}. \quad (7)$$

Note that we use $\boldsymbol{z}_1$ in Equation (2) for consistency with the other sub-metrics. Here we use $\mathrm{Dvg}(g_o(u)\|g_{o'}(u))$ for a more informative and common representation of KL divergence. The detailed derivation is in Appendix A. The unknown β in the term $(\hat{\mu}_o - \hat{\mu}_{o'})^2$ of Equation (7) can be cancelled out using the definition of μ as follows.

$$\begin{aligned}(\hat{\mu}_o - \hat{\mu}_{o'})^2 &= \Big(\frac{\beta}{|\mathtt{D}_o|\cdot|\beta|}\sum_{t=1}^{|\mathtt{D}_o|} f(\boldsymbol{x}_t) - \frac{\beta}{|\mathtt{D}_{o'}|\cdot|\beta|}\sum_{l=1}^{|\mathtt{D}_{o'}|} f(\boldsymbol{x}_l)\Big)^2 \\ &= \Big(\frac{1}{|\mathtt{D}_o|}\sum_{t=1}^{|\mathtt{D}_o|} f(\boldsymbol{x}_t) - \frac{1}{|\mathtt{D}_{o'}|}\sum_{l=1}^{|\mathtt{D}_{o'}|} f(\boldsymbol{x}_l)\Big)^2\end{aligned}$$

By using the above result of $(\hat{\mu}_o - \hat{\mu}_{o'})^2$ and Equation (6), we can rewrite Equation (7) as follows.

$$\begin{aligned}&\mathrm{Dvg}(g_o(u)\|g_{o'}(u)) \\ &= \frac{1}{2}\log\frac{|\mathtt{D}_o|-1}{|\mathtt{D}_{o'}|-1} - \frac{1}{2} \\ &+ \frac{1}{2}\log\frac{\sum_{l=1}^{|\mathtt{D}_{o'}|}\big(f(\boldsymbol{x}_l) - \frac{1}{|\mathtt{D}_{o'}|}\sum_{t=1}^{|\mathtt{D}_{o'}|} f(\boldsymbol{x}_t)\big)^2}{\sum_{t=1}^{|\mathtt{D}_o|}\big(f(\boldsymbol{x}_t) - \frac{1}{|\mathtt{D}_o|}\sum_{l=1}^{|\mathtt{D}_o|} f(\boldsymbol{x}_l)\big)^2} \\ &+ \frac{(|\mathtt{D}_{o'}|-1)\sum_{t=1}^{|\mathtt{D}_o|}\big(f(\boldsymbol{x}_t) - \frac{1}{|\mathtt{D}_o|}\sum_{l=1}^{|\mathtt{D}_o|} f(\boldsymbol{x}_l)\big)^2}{2(|\mathtt{D}_o|-1)\sum_{l=1}^{|\mathtt{D}_{o'}|}\big(f(\boldsymbol{x}_l) - \frac{1}{|\mathtt{D}_{o'}|}\sum_{t=1}^{|\mathtt{D}_{o'}|} f(\boldsymbol{x}_t)\big)^2} \\ &+ \frac{(|\mathtt{D}_{o'}|-1)\big(\frac{1}{|\mathtt{D}_o|}\sum_{t=1}^{|\mathtt{D}_o|} f(\boldsymbol{x}_t) - \frac{1}{|\mathtt{D}_{o'}|}\sum_{l=1}^{|\mathtt{D}_{o'}|} f(\boldsymbol{x}_l)\big)^2}{2\sum_{l=1}^{|\mathtt{D}_{o'}|}\big(f(\boldsymbol{x}_l) - \frac{1}{|\mathtt{D}_{o'}|}\sum_{t=1}^{|\mathtt{D}_{o'}|} f(\boldsymbol{x}_t)\big)^2}\end{aligned} \tag{8}$$

Computing the aggregated divergence

In one-vs-all, one-vs-one and ECOC decomposition, we have multiple binary classifiers. The KL divergence is asymmetric, that is, $\mathrm{Dvg}(g_o(u)\|g_{o'}(u))$ may not equal $\mathrm{Dvg}(g_{o'}(u)\|g_o(u))$. The straightforward way to compute the aggregated KL divergence is to compute two KL divergences of each binary classification problem, and then to compute the average of all the KL divergences in a multi-class classification problem. However, in the one-vs-all decomposition, a better approach is to compute only $\mathrm{Dvg}(g_o(u)\|g_{o'}(u))$ where o is the class for "one" and o' is the class for "all". This is because $\mathtt{D}_o$ which forms the distribution of interest (i.e., $g_o(u)$) appears in binary problems of one-vs-all, one-vs-one and ECOC decomposition. In contrast, $\mathtt{D}_{o'}$ may not appear in any binary problems of one-vs-one or ECOC decomposition. For example, suppose we have four classes C_1, C_2, C_3, C_4 and $B(C_i, C_j)$ denotes a binary classifier composed of the data in classes C_i and C_j. Let the class o stand for C_1. In one-vs-one, we can generate six binary classifiers including $B_1(C_1, C_2)$, $B_2(C_1, C_3)$, $B_3(C_1, C_4)$, $B_4(C_2, C_3)$, $B_5(C_2, C_4)$, $B_6(C_3, C_4)$ and C_1 appears in the first three binary classifiers. In one-vs-all, C_1 appears in binary classifiers such as $B_1(C_1, (C_2, C_3, C_4))$. In ECOC, C_1 may appear in several binary classifiers according to the codewords. For example, a codeword $[1, -1, -1, 1]$ can form a binary classifier $B_1((C_2, C_3), (C_1, C_4))$ which includes C_1. However, the class o' which is (C_2, C_3, C_4) in one-vs-all may not appear in any binary classifiers of one-vs-one or ECOC. Therefore $g_o(u)$ is more important than $g_{o'}(u)$ and $\mathrm{Dvg}(g_{o'}(u)\|g_o(u))$ is not considered for one-vs-all in D-Chooser (i.e., one-side divergence). Hence, the one-side KL divergence of one-vs-all and the average KL divergences of one-vs-one and ECOC are more comparable. Finally, the decomposition method with a larger average/one-side KL divergence is chosen by D-Chooser.

Algorithm 2: Computing KL divergence

```
Input: Data sets $\mathtt{D}_i$ and $\mathtt{D}_j$ of problem $\mathtt{P}_{ij}$
Output: $KL_{ij}$ and $KL_{ji}$
1 $\bar{\boldsymbol{x}}_i^1, ..., \bar{\boldsymbol{x}}_i^k \leftarrow$ GetCentroid($\mathtt{D}_i$)
2 $\bar{\boldsymbol{x}}_j^1, ..., \bar{\boldsymbol{x}}_j^k \leftarrow$ GetCentroid($\mathtt{D}_j$)
3 $\mathtt{H} \leftarrow$ FindHyperPlane($\bar{\boldsymbol{x}}_i^1, ..., \bar{\boldsymbol{x}}_i^k, \bar{\boldsymbol{x}}_j^1, ..., \bar{\boldsymbol{x}}_j^k$)
4 $\hat{\mu}_i, \hat{\sigma}_i^2 \leftarrow$ ConstructGaussian($\mathtt{D}_i$, $\mathtt{H}$)   //Alg. 1
5 $\hat{\mu}_j, \hat{\sigma_j}^2 \leftarrow$ ConstructGaussian($\mathtt{D}_j$, $\mathtt{H}$)   //Alg. 1
6 $KL_{ij} \leftarrow$ ComputeKLDiverg.($\hat{\mu}_i, \hat{\sigma}_i, \hat{\mu}_j, \hat{\sigma}_j$) //Eq. (7)
7 $KL_{ji} \leftarrow$ ComputeKLDiverg.($\hat{\mu}_j, \hat{\sigma}_j, \hat{\mu}_i, \hat{\sigma}_i$) //Eq. (7)
```

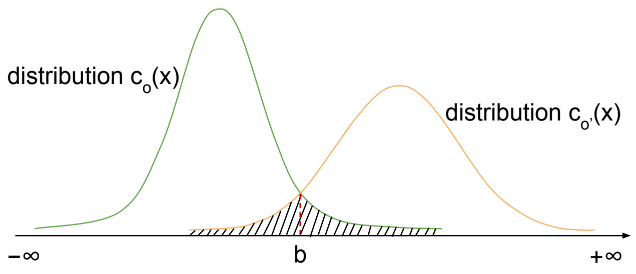


Fig. 4: The overlapping region of two distributions.

Algorithm 2 summarizes the process of computing the KL divergence. We first construct the distance distributions of the binary problem, using the techniques discussed earlier in this section (cf. Lines 1 to 5 of Algorithm 2). Then, we compute the KL divergence of the distributions using Equation (13) and (14) as shown in Lines 7 to 8 in Algorithm 2.

3.3.3 *Computing the overlapping regions*

Another sub-metric for measuring the difficulty of a binary classification problem $\mathcal{P}_B$ is the overlapping region of the two distributions. Figure 4 shows an example of the overlapping region of two distributions. The two distributions intersect at point b and the shadow area illustrates the overlapping region. The smaller the overlapping region, the easier the classification problem is. Formally, the overlapping region, denoted by $\boldsymbol{z}_2$, of the two distributions $g_o(u)$ and $g_{o'}(u)$ can be written as follows.

$$\mathrm{Rgn}(\mathcal{P}_B) = \int_{-\infty}^{b} g_o(u)du + \int_{b}^{+\infty} g_{o'}(u)du$$

where b is the value that two distributions intersect, $g_o(u)$ is to the left of $g_{o'}(u)$. The equation can be written in a more compact form as follows.

$$\mathrm{Rgn}(\mathcal{P}_B) = \int \min(g_o(u), g_{o'}(u))du \tag{9}$$

The overlapping regions of the $|\mathcal{B}|$ binary problems are averaged, and the average value is used as a sub-metric to measure the multi-class problem under the decomposition method.

3.3.4 Distance computation with kernel functions

With the hyper-plane $\mathtt{H}$, we can compute the distance d_t from a training instance $\boldsymbol{x}_t$ to the hyper-plane $\mathtt{H}$, such that a Gaussian probability distribution of these distances can be constructed. If the separating hyper-plane for the centroids is obtained by training SVMs, we can use the decision function of SVMs to obtain the distance from an instance $\boldsymbol{x}_t$ to the hyper-plane. The decision function is as follows: $d_t = \sum_{l=1}^{n} y_l \alpha_l K(\boldsymbol{x}_l, \boldsymbol{x}_t) + b$ where y_l is the label (either -1 or +1) of the training instance $\boldsymbol{x}_l$, α_l is the parameter of the hyper-plane, $K(\cdot)$ is the kernel function for mapping data from their original data space to a higher dimensional data space [23], and b is the bias of the hyper-plane of the trained SVM.

Alternatively, the distances can be computed directly using Equation (4). It is known that some problems can be solved in their original data spaces, while the others may be solved in higher dimensional data spaces induced by kernel functions. Computing distances to the separating hyper-plane for the centroids is not trivial as discussed earlier in this section. In the following, we present details of computing the distance when kernel functions are used, which is more complex than computing distances in the original data space. Then, we provide a uniform form for the distance computation equations.

Kernel functions can be used in some classifiers such as SVMs and logistic regression. Here, we discuss how to compute the distance in the data space induced by the kernel functions. Let $\mathtt{K}(\boldsymbol{x}_t, \bar{\boldsymbol{x}}_i)$ denote a kernel function, where $\mathtt{K}(\boldsymbol{x}_t, \bar{\boldsymbol{x}}_i) = \varphi(\boldsymbol{x}_t)^\intercal \varphi(\bar{\boldsymbol{x}}_i)$ and $\varphi(\cdot)$ is a mapping function which maps the instances into a higher dimensional data space. Then, by extending Equation (4), the distance in the space induced by the kernel function can be computed as follows.

$$\begin{aligned} d_t &= \frac{\beta}{|\beta|} \cdot \frac{[\varphi(\bar{\boldsymbol{x}}_i) - \varphi(\bar{\boldsymbol{x}}_j)]^\intercal [\varphi(\boldsymbol{x}_t) - \frac{\varphi(\bar{\boldsymbol{x}}_i) + \varphi(\bar{\boldsymbol{x}}_j)}{2}]}{\|\varphi(\bar{\boldsymbol{x}}_i) - \varphi(\bar{\boldsymbol{x}}_j)\|} \\ &= \frac{\varphi(\bar{\boldsymbol{x}}_i)^\intercal \varphi(\boldsymbol{x}_t) - \varphi(\bar{\boldsymbol{x}}_j)^\intercal \varphi(\boldsymbol{x}_t) - \frac{\varphi(\bar{\boldsymbol{x}}_i)^\intercal \varphi(\bar{\boldsymbol{x}}_i) - \varphi(\bar{\boldsymbol{x}}_j)^\intercal \varphi(\bar{\boldsymbol{x}}_j)}{2}}{\|\varphi(\bar{\boldsymbol{x}}_i) - \varphi(\bar{\boldsymbol{x}}_j)\|} \\ &= \frac{\beta}{|\beta|} \cdot \frac{\mathtt{K}(\bar{\boldsymbol{x}}_i, \boldsymbol{x}_t) - \mathtt{K}(\bar{\boldsymbol{x}}_j, \boldsymbol{x}_t) - \frac{\mathtt{K}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i) - \mathtt{K}(\bar{\boldsymbol{x}}_j, \bar{\boldsymbol{x}}_j)}{2}}{\sqrt{\mathtt{K}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i) + \mathtt{K}(\bar{\boldsymbol{x}}_j, \bar{\boldsymbol{x}}_j) - 2\mathtt{K}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j)}} \end{aligned} \tag{10}$$

We can rewrite the distance computation functions for the original space (i.e., Equation 4) and for the space induced by kernels in a uniform way as follows.

$$d_t = \frac{\beta}{|\beta|} \cdot f(\boldsymbol{x}_t) \tag{11}$$

where $f(\boldsymbol{x}_t) = \frac{(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_j)^\intercal (\boldsymbol{x}_t - \frac{\bar{\boldsymbol{x}}_i + \bar{\boldsymbol{x}}_j}{2})}{\|\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_j\|}$ when the distance is computed in the original data space, and $f(\boldsymbol{x}_t) = \frac{\mathtt{K}(\bar{\boldsymbol{x}}_i, \boldsymbol{x}_t) - \mathtt{K}(\bar{\boldsymbol{x}}_j, \boldsymbol{x}_t) - \frac{\mathtt{K}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i) - \mathtt{K}(\bar{\boldsymbol{x}}_j, \bar{\boldsymbol{x}}_j)}{2}}{\sqrt{\mathtt{K}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_i) + \mathtt{K}(\bar{\boldsymbol{x}}_j, \bar{\boldsymbol{x}}_j) - 2\mathtt{K}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j)}}$ when the distance is computed in the data space induced by the kernel function $\mathtt{K}(\cdot, \cdot)$. Note that $f(\boldsymbol{x}_t)$ can be computed purely based on the training instances of problem $\mathtt{P}_{ij}$, i.e., $\mathtt{D}_i \cup \mathtt{D}_j$, and the kernel function.

3.4 Sub-metrics computed directly from data

Here, we present two sub-metrics computed directly from the meta information of the training data to measure the difficulty of a multi-class problem.

3.4.1 Unevenness degree

The intuition of this sub-metric is that the problem with more balanced data tends to be easier to solve. Formally, the unevenness degree is defined as $\mathrm{Unv}(\mathcal{P}_B) = \frac{|\mathtt{D}_o|}{2 \cdot |\mathtt{D}_{o'}|} + \frac{|\mathtt{D}_{o'}|}{2 \cdot |\mathtt{D}_o|}$. If the binary problem $\mathcal{P}_B$ is even, i.e., two classes have the same number of instances, $\mathrm{Unv}(\mathcal{P}_B)$ equals 1. The more uneven the problem is, the larger the value of $\mathrm{Unv}(\mathcal{P}_B)$. The unevenness degrees of the $|\mathcal{B}|$ binary problems produced by the decomposition method are averaged, and the average value is used as the unevenness degree sub-metric to measure the multi-class problem under the decomposition method.

3.4.2 Relative size of the solution space

Suppose we have $|\mathtt{D}_o|$ and $|\mathtt{D}_{o'}|$ training instances in the two classes of the binary problem, respectively, and all the training instances are linearly independent. Then, the training instances form a linear system with $(|\mathtt{D}_o| + |\mathtt{D}_{o'}|)$ equations, and each equation has n variables where n is the data dimensionality. Then, the dimension of the solution space [24] is $\max(0, n - |\mathtt{D}_o| - |\mathtt{D}_{o'}|)$. Here, we use the relative size of the solution space defined as $\frac{|\mathtt{D}_o| + |\mathtt{D}_{o'}|}{n}$. The larger the relative size of the solution space, the easier the binary problem. The key intuition is that the larger the relative size of solution space means the smaller feasible regions, and hence it is easier to find a solution for the binary problem, due to the smaller search space. For small-scale data, the relative size of solution space is negligible, and thus we exclude this sub-metric for small-scale data sets. More explanation can be found in the first paragraph of Section 4.

3.4.3 Summary of the sub-metrics

The relationship of the difficulty index and the sub-metrics is summarized below. (i) The larger the distributions diverge, the easier the problem is. Hence, the difficulty index is smaller. (ii) The larger the overlapping region, the harder the problem is. So, the overlapping region has positive influence on the difficulty index. (iii) If the problem is more even, the problem is easier and the difficulty index of the problem is smaller. (iv) The smaller the relative size of the solution space, the harder the problem is. Hence, the difficulty index is larger.

The whole process of D-Chooser making a decision is summarized in Algorithm 3. For each of the decomposition methods, we first use it to decompose the multi-class problem into a number of binary problems (Line 3). Then, for each binary problem, we construct distance distributions and compute the four sub-metrics (Lines 6 to 11). Finally, we compute the difficulty index and return the decomposition method with the minimum difficulty index (Lines 12 and 13). The time complexity of D-chooser is $\mathcal{O}(\mathtt{K}|\mathtt{D}|n)$. When dealing with large-scale and high-dimensional data, due to the sparsity of data, we can use techniques for processing sparse data such as sparse matrix compression and principle component analysis to further accelerate D-chooser. High

Algorithm 3: Choose the best decomposition method

```
Input: problem $\mathcal{P}$, decomp. methods $\mathcal{M}$, weight
       vector $\boldsymbol{\omega}$
Output: the best decomposition method
1  $I \leftarrow \mathbf{0}$ //initialize difficulty index for each method
2  foreach $m$ in $\mathcal{M}$ do
3      $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{|\mathcal{B}|}$ ←Decompose($\mathcal{P}$, $m$)
4      $\boldsymbol{z}^m \leftarrow \mathbf{0}$ //initialize difficulty index
5      for $i \leftarrow 1$ to $|\mathcal{B}|$ do
6          $\mathtt{D}_o, \mathtt{D}_{o'} \leftarrow$ GroupInstance($\mathcal{P}_i$)
7          $g_o(u), g_{o'}(u) \leftarrow$ ConstructDist($\mathtt{D}_o$, $\mathtt{D}_{o'}$)
           //Alg. 1
8          $\boldsymbol{z}_1^m \leftarrow \boldsymbol{z}_1^m$+CompDvg($g_o(u), g_{o'}(u)$) //Alg. 2
9          $\boldsymbol{z}_2^m \leftarrow \boldsymbol{z}_2^m$+ CompRegn($g_o(u), g_{o'}(u)$)    //Eq.
           (9)
10         $\boldsymbol{z}_3^m \leftarrow \boldsymbol{z}_3^m$+ CompUnevenness($\mathtt{D}_o$,$\mathtt{D}_{o'}$)
           //§ 3.4.1
11         $\boldsymbol{z}_4^m \leftarrow \boldsymbol{z}_4^m$+ CompSoltnSpace($\mathtt{D}_o$,$\mathtt{D}_{o'}$)
                                                      //§ 3.4.2
12     $\boldsymbol{z}^m \leftarrow \frac{\boldsymbol{z}^m}{|\mathcal{B}|}, I_m \leftarrow \boldsymbol{\omega}^\intercal \boldsymbol{z}^m$
13 return $\arg\min_{m \in \mathcal{M}}(I_m)$
```

TABLE 2: Data set information.

data set	# classes	# training ins.	# test ins.	dim.
acoustic	3	78,823	10,000	48
connect-4	3	54,045	13,512	126
letter	26	15,000	5,000	16
pendigits	10	7,494	3,498	16
poker	10	25,010	1M	10
sector	105	6,412	3,207	55,197
sensorless	11	48,509	10,000	48
svmguide	6	300	312	10
usps	10	7,291	2,007	256

performance computing can be easily implemented in D-chooser as the difficulty index is computed based on numbers of binary problems.

4 Experimental Studies

In this section, we study the efficiency and predictive accuracy of our proposed D-Chooser equipped with the difficulty index. The experiments were conducted on a workstation running Linux with 2 Xeon E5-2640v4 10 core CPUs, 256GB main memory and a Titan X Pascal GPU of 12GB memory. We used nine data sets from the LibSVM website. Both of the training and test sets are publicly available on the website. The information of the data sets is shown in Table 2. The data sets include different types of data. For example, *acoustic* is applied to vehicle classification which includes the acoustic signals sensed by the sensors; *connect-4* has all legal 8-ply positions in Connect Four game and is used to predict if the first player wins, loses or gets a draw in the game; *letter* consists images of 26 English letters; *usps* and *pendigits* are used in handwriting recognition; *poker* is used to classify the poker hands; *sector* collects the texts on corporate web pages and is applied for text classification; *sensorless* extracts features from electric current drive signals; data in *svmguide* are from an application on traffic light signals.

TABLE 3: Whether D-Chooser chooses best decomposition.

data set	Chooses decomposition correctly?			
	SVMs with RBF	linear SVMs	logistic regression	naive Bayes
acoustic	80.00% (✗)	70.07% (✓)	70.27% (✗)	66.97% (✓)
connect-4	83.78% (✓)	75.67% (✓)	75.78% (✓)	63.12% (✗)
letter	98.06% (✓)	84.55% (✓)	84.26% (✓)	64.12% (✗)
pendigits	98.03% (✓)	94.40% (✓)	95.60% (✓)	82.25% (✓)
poker	57.54% (✓)	50.30% (✗)	50.12% (✓)	50.12% (✓)
sector	94.61% (✗)	94.23% (✓)	94.11% (✓)	84.10% (✓)
sensorless	99.88% (✓)	91.89% (✓)	92.34% (✓)	73.36% (✓)
svmguide	66.35% (✓)	59.94% (✓)	72.44% (✓)	63.14% (✓)
usps	96.12% (✗)	93.03% (✓)	93.57% (✓)	77.13% (✓)

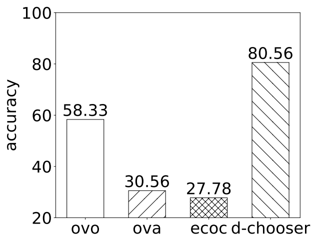
All types of data were transformed into LIBSVM format and more details can be found on the LIBSVM website. D-Chooser was implemented in C++. We used common multi-class classifiers including linear and non-linear SVMs, logistic regression and Gaussian naive Bayes from scikit-learn [7]. We used the scikit-learn implementation of ECOC where the codebook is randomly generated. Since only the mainstream decomposition methods (i.e., one-vs-all, one-vs-one and ECOC) have been implemented in popular libraries such as scikit-learn, WEKA and LibSVM, in this paper, we focus on these three mainstream decomposition methods and use the same strategy for the ECOC codeword. The non-linear SVMs ran on GPUs due to its higher computation cost, and the other algorithms ran on CPUs. As SVMs and logistic regression have hyper-parameters, we applied Bayesian optimization [25] to find the best hyper-parameters under each decomposition method. Bayesian optimization is a strategy for global optimization of functions. It aims to optimize expensive-to-evaluate functions. The regularization constant C of the classifiers was chosen from 2^{-3} to 2^{12}, and γ of the RBF kernel was chosen from 2^{-15} to 2^6. The sub-metrics are normalized using the sigmoid function, such that their values are always between 0 and 1. The normalization is used to avoid one sub-metric dominating the overall difficulty index. For the small-scale data set, we exclude the relative size of the solution space because this sub-metric should be small for such data set while the sigmoid normalization produces a relatively large value (e.g., a value that is extremely closed to 0.50). The weight vector $\boldsymbol{\omega}$ is set to (-1, 1, 1, -1). The intuitions of this weight vector is that the larger the distributions diverge, the easier the problem; the larger the overlapping region, the harder the problem; the problem is more even, the problem is easier; the smaller the relative size of the solution space, the harder the problem.

4.1 Predictive accuracy of D-Chooser

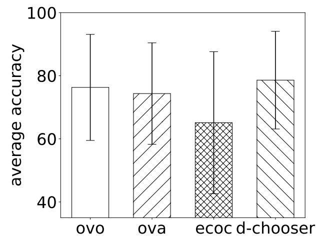
Table 3 shows whether D-Chooser chooses the best decomposition method for a data set. The value in each cell of the table is the best accuracy among all the decomposition methods, (✓) indicates that D-Chooser selects the best decomposition correctly and (✗) indicates it makes a mistake. As we can see from the results, in the 9×4 classification problems (i.e., 9 data sets on 4 classification algorithms totalling to 36), D-Chooser achieves 80.56% accuracy (i.e., 29 out of 36) in choosing the best decomposition method. D-Chooser even achieves 8/9 accuracy when the classifier is logistic regression. In a further investigation on the data sets where D-Chooser makes mistakes, we find that the effectiveness between the best decomposition method and the one chosen by D-Chooser is almost the same. For example, D-Chooser predicts ECOC to be the best decomposition method for non-linear SVMs on *acoustic*, and the corresponding classifier has an accuracy of 78.99% which is very close to the best decomposition method (i.e., one-vs-one) resulting in an accuracy of 80.00%. For the data sets the accuracies prominently differ, D-Chooser can select the best decomposition method accurately (cf. Figure 1). This indicates that D-Chooser is robust and can help practitioners choose near optimal decomposition methods.

To provide full information to the results shown in Table 3, we also report the best accuracy of the classifier with the best hyper-parameter(s) corresponding to the decomposition method in Tables 4 and 6. Table 4 shows the results of the best accuracy of one-vs-all, one-vs-one and ECOC on each classifier of each data set. Table 6 shows the best hyper-parameter(s) for each classifier on each data set. The results confirm that no decomposition method is more effective than the other decomposition methods, and the difference on accuracy is distinct in some data sets. Table 5 summarizes the values of difficulty index of each decomposition method. Note that the sub-metrics distribution divergence and overlapping region are dependent on the distribution which is constructed in the problem space. As SVMs with RBF, linear classifiers and Gaussian naive Bayes solve the problems in different data spaces, the difficulty indices of these classifiers are different from each other (cf. Section 3.3.4). Linear classifiers including linear SVMs and logistic regression solve the problems in the original data space. SVMs with RBF and Gaussian naive Bayes solve the problems by mapping the data to the feature space. The derivation for the problem space of Gaussian naive Bayes is elaborated in Appendix B. From Table 4 and 5, the ratio of cases where the second smallest difficulty index corresponds to the second best performance is 50.00% and the ratio of cases where the largest difficulty index corresponds to the worst performance is 55.56%. The ratios 50.00% and 55.56% are higher than the ratio of randomly choosing the second or third best decomposition which is 33.33%. This indicates that the performance rank is correlated to the difficulty index rank. The drops of predictive accuracy compared with 80.56% may result from the sigmoid normalization. When the original values are large, the normalized values are similar. Therefore, the subtle difference between the normalized sub-metrics of the second and third best decomposition methods may lead to the accuracy difference. For example, in Table 5, when linear classifiers were trained with data set *sensorless*, the difficulty index for the best decomposition method is -0.41. The difficulty indices of the second and third best decomposition method are -0.25 and -0.21 which are closed to each other. We aim to find a more appropriate normalization method to improve the difficulty index in the future work.

Existing libraries have default decomposition methods associated with them. For example, LibSVM uses the one-vs-one decomposition by default. Here, we study the overall effectiveness if an user always insists using a particular decomposition method (e.g., the default method of a library). From Table 4, the probability of OVO, OVA and ECOC to be the best decomposition method is 58.33% (21/36), 30.56% (11/36) and 27.78% (10/36) respectively. None of them is



(a) Accuracy of selecting the best decomposition method



(b) Average classification accuracy

Fig. 5: Comparison of OVO, OVA, ECOC and D-chooser

higher than the accuracy of D-chooser which is 80.56% in selecting the best decomposition method. We illustrate the accuracy in Figure 5(a). Figure 5(b) shows the average classification accuracy—the average accuracy over the 36 classification problems. In the figure, OVO, OVA and ECOC stand for one-vs-one, one-vs-all and error-correcting output codes, respectively. The classification assisted by D-Chooser achieves the highest average accuracy (i.e., 78.61%), while classifications using a fixed decomposition method have a lower average accuracy. OVO, OVA and ECOC achieve 76.31%, 74.35%, 65.13% average classification accuracy, respectively. A classifier with D-chooser can achieve 2.30% higher classification accuracy than OVO in all tested data sets *on average*, which is quite remarkable in practice.

4.2 Efficiency of D-Chooser

Next, we report the elapsed time of D-Chooser, in comparison with the average elapsed time of verifying the effectiveness of a decomposition method. Table 7 shows the elapsed time of D-Chooser for choosing the best decomposition method, in comparison with the average time of verifying a decomposition method.

Here we summarize the key findings from the experiments first. The time of D-Chooser is much less than the time of searching for the best decomposition method. For example, the searching for the best decomposition of SVMs with the RBF kernel for the *letter* data set took about 23 hours on OVA and ECOC (which can be avoided, as OVA or ECOC is not the best), while the time taken by D-Chooser was only 0.71 seconds. Instead of examining all the decomposition methods, D-Chooser performs relatively lightweight data-driven sub-metric calculation. The best decomposition method is found by D-Chooser using various sub-metrics such as distribution divergence and overlapping regions (cf. Section 3). In summary, D-Chooser is much faster than verifying a decomposition method. This demonstrates that D-Chooser is light-weight.

More specifically, for SVMs with the RBF kernel, the time of D-Chooser for making a decision is negligible compared with that of searching. With the help of D-Chooser, users can avoid performing hyper-parameter searches on the unpromising decomposition methods, and the time saved for the users is remarkable. For linear SVMs, logistic regression and Gaussian naive Bayes, the elapsed time of D-Chooser is often a few times smaller than that of the searching for the best decomposition method. This demonstrates D-Chooser is efficient in various settings. In summary, D-Chooser is several

TABLE 4: Best test accuracy of each decomposition method.

data set	the best test accuracy of each decomposition method (%)											
	SVMs with RBF			linear SVMs			logistic regression			naive Bayes		
	OVA	OVO	ECOC	OVA	OVO	ECOC	OVA	OVO	ECOC	OVA	OVO	ECOC
acoustic	79.91	80.00	78.99	69.85	69.82	70.07	70.27	70.22	67.02	54.55	47.35	66.97
connect-4	83.78	65.26	65.26	75.67	75.60	75.25	75.78	75.70	75.30	57.62	52.26	63.12
letter	98.02	98.06	97.70	69.96	84.55	43.00	72.26	84.26	43.24	64.12	64.04	43.12
pendigits	97.94	97.94	98.03	87.66	94.40	73.10	89.97	95.60	79.22	82.08	82.25	61.32
poker	57.54	57.16	56.00	50.14	50.30	50.10	50.12	50.12	50.12	50.12	50.12	50.12
sector	94.61	91.83	94.14	94.23	91.92	94.23	93.70	91.43	94.11	51.26	78.20	84.10
sensorless	99.88	99.88	99.80	74.49	91.89	44.24	75.98	92.34	42.49	72.57	73.36	25.26
svmguide	65.39	66.35	40.71	56.42	59.94	18.59	65.71	72.44	44.55	62.18	63.14	48.72
usps	95.32	95.42	96.12	91.68	93.03	88.84	91.78	93.57	88.84	54.06	77.13	61.29

TABLE 5: Difficulty index of each decomposition method.

data set	Difficulty index of each decomposition method								
	SVMs with RBF			linear classifiers			naive Bayes		
	OVA	OVO	ECOC	OVA	OVO	ECOC	OVA	OVO	ECOC
acoustic	-0.18	-0.11	-0.25	-0.18	-0.11	-0.25	-0.10	-0.03	-0.15
connect-4	0.16	0.23	0.26	0.16	0.23	0.26	0.18	0.25	0.28
letter	-0.33	-0.47	-0.15	-0.31	-0.47	-0.15	-0.33	-0.47	-0.15
pendigits	-0.50	-0.50	-0.72	-0.40	-0.49	-0.47	-0.39	-0.49	-0.43
poker	-0.21	0.03	0.07	0.08	0.22	0.23	0.08	0.12	0.23
sector	-0.50	-0.47	-0.58	-0.49	-0.47	-0.55	-0.49	-0.47	-0.55
sensorless	-0.21	-0.41	-0.25	-0.21	-0.41	-0.25	-0.17	-0.41	-0.20
svmguide	1.12	0.91	1.02	1.12	0.91	1.02	1.12	0.91	1.02
usps	-0.46	-0.48	-0.42	-0.44	-0.48	-0.42	-0.45	-0.48	-0.41

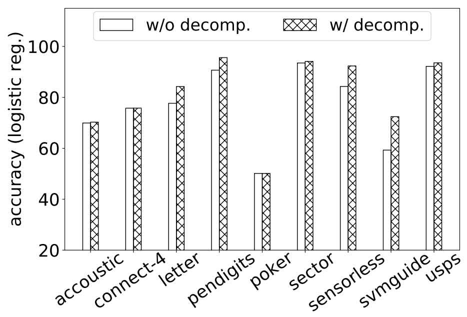


Fig. 6: Effect of decomposition.

orders of magnitude faster than verifying a decomposition method. This demonstrates that D-Chooser is light-weight.

It is important to point out that when considering three decomposition methods (i.e., OVA, OVO and ECOC), D-Chooser can help avoid performing hyper-parameter search operations for two decomposition methods. Concretely, suppose OVO is the best decomposition for the *letter* data set. Then, the hyper-parameter search operations on OVA and ECOC are not necessary, because we do not use OVA and ECOC and their corresponding best hyper-parameter(s) to train a model anyway. In comparison, the hyper-parameter search for OVO is useful, since we will use OVO and the corresponding best hyper-parameter(s) for training the final classifier to tackle the *letter* multi-class problem.

Table 9 shows the results of the elapsed time of each hyper-parameter search operation, and the sum of the time of three hyper-parameter search operations forms the time of searching for the best decomposition method. For example, in order to select the best decomposition method for SVMs with the RBF kernel in the *letter* data set, we need to perform a hyper-parameter search to obtain the best accuracy of each decomposition method. As we can see from the table, performing three hyper-parameter search operations for the OVA, OVO, and ECOC decomposition take 1658.86, 4095.23, and 81248.08 seconds, respectively. Therefore, the total time is 1658.86+4095.23+81248.08=87002.17 seconds for searching the best decomposition method for SVMs with the RBF kernel in the *letter* data set. As we have shown in Table 7, the elapsed time for verifying a decomposition method is much longer than the prediction time of D-Chooser. I.e., D-Chooser can help practitioners avoid performing grid searches on OVA and ECOC in the *letter* data set (saving about 82906.94 seconds), since OVO is the best decomposition and there is no need to find the best hyper-parameters for OVA and ECOC. Note that the time reported for D-Chooser in Table 7 includes the time for finding centroids. The avoidance of examining all the decomposition methods makes D-Chooser more efficient.

4.3 Benefit of using decomposition

In the literature [26], logistic regression and naive Bayes can be directly used for multi-class classification without using the decomposition. Our experimental results in Table 8 show that decomposition almost always achieves higher accuracies than non-decomposition in the data sets. In Table 8, directly using logistic regression and naive Bayes for multi-class classification is denoted as "none" (short for non-decomposition). We compare the classification accuracies between decomposition methods and the non-decomposition method. For example, in the *letter* data set, using one-vs-one (OVO) decomposition for logistic regression has 6.58% higher accuracy than logistic regression with non-decomposition (cf. Table 8 and Figure 6). Note that in Figure 6 "w/o decomp." denotes solving multi-class classification without decomposition into binary classification problems, and "w/ decomp." denotes multi-class classification with decomposition into binary classification problems. This demonstrates that decomposition methods are very important for multi-class classification, even for the classifiers that naturally support multi-class classification. We observed similar trend on naive Bayes as well. Note that the difference in *acoustic* and *poker* is subtle in the figure, because using decomposition or not produces similar models for the two data sets and hence similar predictive accuracy.

4.4 Individual sub-metric

We also measured the accuracy of D-Chooser with difficulty index using only one sub-metric. The accuracy of D-Chooser with difficulty index using only relative size of the solution

TABLE 6: Classifier best hyper-parameters of one-vs-all, one-vs-one and ECOC.

data set	SVMs with RBF (C, γ)			linear SVMs (C)			logistic regression (C)		
	OVA	OVO	ECOC	OVA	OVO	ECOC	OVA	OVO	ECOC
acoustic	(476.18, 0.19)	(111.14, 0.26)	(226.62, 0.29)	2.62	3.78	11.10	1205.88	2589.75	9.15
connect-4	(1.00, 0.50)	(0.13, 3.05E-05)	(0.13, 3.05E-05)	1.00	0.13	0.13	2958.86	2172.15	2234.33
letter	(3221.65, 3.80)	(51.13, 3.67)	(1361.02, 2.63)	20.57	5.93	4.52	17.57	1108.39	2.03
pendigits	(2877.93, 3.05E-05)	(141.52, 3.05E-05)	(2705.04, 3.05E-05)	1.22	1.58	1909.13	705.16	828.04	2499.13
poker	(130.61, 0.002)	(1537.17, 0.27)	(1119.66, 0.29)	1.27	14.62	0.90	3137.03	3566.97	4001.20
sector	(256.00, 3.91E-03)	(128.00, 0.02)	(153.60, 0.02)	3.36	16.10	2.44	4087.39	3046.21	31.04
sensorless	(14.89, 13.26)	(13.58,13.39)	(17.50, 15.53)	6.48	24.51	66.17	1110.71	3179.51	3386.81
svmguide	(3685.38, 0.36)	(3986.27, 0.45)	(4089.23, 12.54)	20.86	124.40	2947.41	3525.16	4067.19	3836.24
usps	(2164.60, 0.02)	(3948.60, 0.02)	(252.23, 0.01)	1.98	2.002	2.40	0.54	0.13	28.56

TABLE 7: Efficiency of D-Chooser and verifying a decomposition method (sec).

data set	D-Chooser vs verifying a decomposition method		
	SVMs with RBF	linear SVMs	logistic regression
acoustic	0.63 vs $1.70\text{x}10^6$	1.88 vs $7.24\text{x}10^3$	1.88 vs 823.54
connect-4	0.39 vs $3.70\text{x}10^5$	0.59 vs $1.72\text{x}10^3$	0.59 vs 178.13
letter	0.71 vs $2.90\text{x}10^4$	1.45 vs $2.60\text{x}10^3$	1.45 vs 480.19
pendigits	0.10 vs $5.02\text{x}10^3$	0.24 vs 567.51	0.24 vs 325.11
poker	0.33 vs $5.91\text{x}10^5$	0.58 vs $4.34\text{x}10^3$	0.58 vs 537.10
sector	$1.07\text{x}10^3$ vs $6.59\text{x}10^4$	27.55 vs $3.79\text{x}10^4$	27.55 vs $3.46\text{x}10^4$
sensorless	1.50 vs $3.03\text{x}10^4$	4.96 vs $1.42\text{x}10^4$	4.96 vs $8.11\text{x}10^3$
svmguide	0.01 vs 279.40	0.01 vs 206.55	0.01 vs 104.27
usps	0.92 vs $1.30\text{x}10^4$	3.86 vs $2.99\text{x}10^3$	3.86 vs $7.97\text{x}10^3$

TABLE 8: Accuracy comparison between decomposition methods and non-decomposition method (%).

data set	logistic regression				naive Bayes			
	OVA	OVO	ECOC	none	OVA	OVO	ECOC	none
acoustic	70.27	70.22	67.02	69.93	54.55	47.35	66.97	47.35
connect-4	75.78	75.70	75.30	75.76	57.62	52.26	63.12	52.26
letter	72.26	84.26	43.24	77.68	64.12	64.04	43.12	64.04
pendigits	89.97	95.60	79.22	90.68	82.08	82.25	61.32	82.25
poker	50.12	50.12	50.12	50.13	50.12	50.12	50.12	50.12
sector	93.70	91.43	94.11	93.51	51.26	78.20	84.10	78.30
sensorless	75.98	92.34	42.49	84.29	72.57	73.36	25.26	73.36
svmguide	65.71	72.44	44.55	59.29	62.18	63.14	48.72	63.14
usps	91.78	93.57	88.84	92.18	54.06	77.13	61.29	77.13

space, only unevenness degree, only overlapping regions and only distribution divergence are 50.00%, 66.67%, 66.67% and 72.22%, respectively. In comparison, D-Chooser with the difficulty index using all the four sub-metrics achieves an accuracy of 80.56% as shown in Table 3.

4.5 Effect of the number of centroids k

We varied k from 1 to 20, and measured whether D-Chooser can make accurate prediction on all the tested data sets. The prediction accuracy of D-chooser, which is equal to the number of data sets with correctly chosen decomposition divided by the total number of data sets tested, is illustrated in Figure 7. The results for linear SVMs and logistic regression are combined into one curve because the difficulty index computed by D-Chooser is the same for both of them. As we can see from Figure 7, increasing the number of centroids to represent the training data set has little impact on the overall accuracy of D-Chooser. Therefore, we recommend using one centroid per class, since the overall accuracy is excellent.

4.6 Case study

We study two real multi-class classification problems in the Kaggle platform. We reran the best multi-class solutions on Kaggle for the problems and examined whether they can be improved with better decomposition methods.

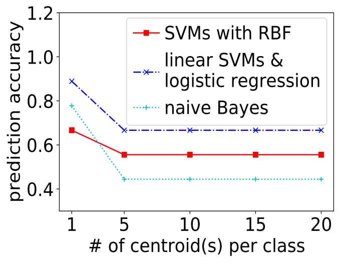


Fig. 7: Effect of the number of clusters.

Toxic Comment Classification Challenge

In this problem, practitioners are given a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity include *toxic*, *severe_toxic*, *obscene*, *threat*, *insult* and *identity_hate*. The task is to predict the type of toxicity for each comment in the test set. The best solution[2] that uses multi-class SVM classifier applies the one-vs-all decomposition method. The accuracy we reproduced is 96.06% based on the kernel provided. In comparison, D-Chooser recommends the one-vs-one decomposition for this problem. The multi-class SVM classifier we trained achieves accuracy of 97.55%, which is notably higher than the winning solution.

Personalized Medicine: Redefining Cancer Treatment

In this problem, data mining practitioners are asked to classify genetic mutations based on clinical evidence described in text. There are nine different classes a genetic mutation can be classified on. We reproduced the results of the solution based on the kernel provided. The multi-class solution[3] using linear SVMs achieves accuracy of 62.57% with the one-vs-all decomposition. In comparison, the multi-class SVM classifier we trained has accuracy of 63.45% with the error-correcting output codes decomposition.

5 Related Work

Multi-class classification problems occur in many real-world applications [27]. Some algorithms are specifically designed to solve multi-class problems [28], [29], while other algorithms (e.g., k-NN and logistic regression) naturally can solve multi-class problems [30]. The multi-class problems are commonly tackled by extending the binary classification algorithms. The benefit of this extension is that the extension

2. https://bit.ly/2rVLvFz
3. https://bit.ly/35kI6xC

TABLE 9: Elapsed time of the hyper-parameter search on each decomposition method for each classifier (sec).

data set	elapsed time of hyper-parameter search											
	SVMs with RBF			linear SVMs			logistic regression			naive Bayes		
	OVA	OVO	ECOC	OVA	OVO	ECOC	OVA	OVO	ECOC	OVA	OVO	ECOC
acoustic	7.73x10^5	1.18x10^6	3.02x10^6	9242.85	5905.76	6569.48	640.18	452.50	1377.95	0.32	0.33	0.31
connect-4	4.09x10^5	2.31x10^5	4.63x10^5	2005.89	1075.06	2067.93	210.83	129.12	194.43	0.40	0.38	0.47
letter	1658.86	4095.23	8.12x10^4	391.21	584.11	6823.24	501.17	492.07	447.34	0.22	2.18	0.32
pendigits	1348.84	3015.07	1.07x10^4	441.99	272.14	988.41	327.21	258.24	389.88	0.05	0.24	0.07
poker	8.60x10^4	7.64x10^5	9.23x10^5	3948.03	2969.66	6113.44	419.13	724.94	467.23	3.95	66.46	5.23
sector	5.47x10^4	1.59x10^4	1.27x10^5	1498.22	8.92x10^4	2.30x10^4	3945.35	8.21x10^4	1.78x10^4	1015.11	9.50x10^4	1289.35
sensorless	3719.12	6010.71	8.13x10^4	1.33x10^4	1714.68	2.76x10^4	7577.73	1260.42	1.55x10^4	0.48	1.34	0.60
svmguide	300.59	217.49	320.12	206.50	203.05	210.09	104.70	101.76	106.36	0.01	0.03	0.01
usps	3088.83	3904.28	3.19x10^4	1342.13	493.97	7141.22	2242.31	759.41	2.09x10^4	0.29	0.89	0.40

module can be used as a meta algorithm and any binary classifier can be plugged into the module to solve the multi-class problems. This benefit is exploited by the popular machine learning libraries such as scikit-learn [7] and WEKA [8]. The default decomposition methods in the libraries are used by many applications in practice [27].

Extending the binary classification algorithms for multi-class classification is also known as "decomposition" of the multi-class classification problems. Some surveys discussed common decomposition methods—including one-vs-all, one-vs-one, error-correcting output-coding and hierarchical strategies—for the multi-class classification [26], [4]. Hsu and Lin [13] studied the performance of the three main decomposition methods under the context of SVMs. A study aims to learn the codebook for ECOC [6] using optimization. The study indicates that learning the codebook is infeasible due to the high computation cost. There are also variations of ECOC in the literature [31]. In this paper, we focus on three mainstream decomposition methods, and aim to automatically select the best decomposition method given a problem. The three mainstream decomposition methods including one-vs-one, one-vs-all and ECOC have been used either in existing libraries such as LibSVM [9] and ThunderSVM [10], or in applications such as bird categorization [32], human activity recognition [33] and visual recognition [34]. We have described more technical details of the three mainstream methods in Section 2.

Combining the outputs of the binary classifiers to form the final result is extensively studied [4]. Experimental study on combining the prediction result of the binary classifiers on one-vs-all and one-vs-one schemes is presented in this reference [11]. They found that voting based methods are more robust and the choice of the best method is classifier dependent. We use the majority voting based method for one-vs-one decomposition and use the maximum decision value for the one-vs-all decomposition [35].

To the best of our knowledge, little work has been done to automatic selection of decomposition methods. Some studies aim to automatically select the best combination of binary classes using data characteristics [31], [36]. For example, DECOC [31] first computes the separability of two classes based on the inter-class distance. Then the confidence score can be derived using the separability. The columns in codebook with k highest confidence scores are selected. ECOC-MDC [36] produces a codebook based on six measurements of data complexity. Chen et. al. [37] use the similarity and balance degree to estimate the unclassifiable rate of a multi-class problem. One-vs-one or one-vs-all is selected for SVM training according to the unclassifiable rate. Zhou et. al. [29] proposed to learn an N-class problem rather than a binary problem at a time. Some work construct an optimization problem to obtain the best coding matrix [6], [38]. Zhong et al. [6] proposed joint learning ECOC to minimize the objective function of the linear classifier and maximize the distance between two codes of each class. In LightMC [38], authors use softmax as the decoding strategy and dynamically optimize the codebook through minimizing the cross entropy loss.

6 Conclusion and Future Work

In this paper, we have proposed D-Chooser, a lightweight approach for decomposition method selection. D-Chooser exploits data-aware sub-metrics including distribution divergence, overlapping areas, unevenness degree and relative size of the solution space to comprehensively measure the problem. We have conducted extensive experiments to study D-Chooser. Experimental results show that D-Chooser achieves competitive accuracy and only needs a few seconds to choose a decomposition method while verifying the effectiveness of a decomposition method is much slower. Case studies on Kaggle competitions have shown that D-Chooser can choose a better decomposition method than the winning solutions.

In the future work, we aim to study setting the values for the weight vector of sub-metrics and investigate the effectiveness of D-Chooser on imbalanced data. The study of the importance of different metrics needs much more labelled data, which will be one of the challenges. The effect of each sub-metric is problem dependent. Considering the interaction of the four sub-metrics, simply tuning each weight may not be sufficient to discover the relationships among the sub-metrics. Moreover, we plan to study the ranking quality of different decomposition methods based on the difficulty index. To achieve a high quality ranking of decomposition methods, we will explore more classifier-dependent sub-metrics. For instance, sub-metrics that can distinguish the problems solved in the same space but using different classifiers will be considered. We will also study the feature spaces of more classifiers to facilitate our D-chooser, as they needs careful theoretical analysis. When two sub-metric values normalized by sigmoid function approach 1, the difference between these sub-metrics is subtle. Therefore to make the sub-metrics more comparable, we aim to evaluate more normalization methods such as min-max normalization. The theoretical analysis of excluding the relative size of the solution space for small-scale data sets will be studied.

Acknowledgments

Prof. Chen is supported by the National Natural Science Foundation of China (Grant No. 62072186), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019B1515130001), the Guangzhou Science and Technology Planning Project (Grant No. 201904010197) and the Opening Project of Guangdong Key Laboratory of Big Data Analysis and Processing. This work is in part supported by a research project grant under NUS Centre for Trusted Internet and Community.

References

[1] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov *et al.*, "Multiclass cancer diagnosis using tumor gene expression signatures," *Proceedings of the National Academy of Sciences*, vol. 98, no. 26, pp. 15 149–15 154, 2001.

[2] H. Greenspan, B. Van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016.

[3] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 221–228.

[4] A. C. Lorena, A. C. De Carvalho, and J. M. Gama, "A review on the combination of binary classifiers in multiclass problems," *Artificial Intelligence Review*, vol. 30, no. 1-4, p. 19, 2008.

[5] G. Rätsch, S. Mika, and A. J. Smola, "Adapting codes and embeddings for polychotomies," in *Advances in Neural Information Processing Systems*, 2003, pp. 529–536.

[6] G. Zhong, K. Huang, and C.-L. Liu, "Joint learning of error-correcting output codes and dichotomizers from data," *Neural Computing and Applications*, vol. 21, no. 4, pp. 715–724, 2012.

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

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research (JMLR)*, vol. 9, no. 8, pp. 1871–1874, 2008.

[10] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, "ThunderSVM: A fast SVM library on GPUs and CPUs," *Journal of Machine Learning Research (JMLR)*, 2018. [Online]. Available: https://github.com/zeyiwen/thundersvm

[11] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[12] S. Vempati, A. Vedaldi, A. Zisserman, and C. Jawahar, "Generalized rbf feature maps for efficient detection." in *BMVC*, 2010, pp. 1–11.

[13] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[14] L. S. Penrose, "The elementary statistics of majority voting," *Journal of the Royal Statistical Society*, vol. 109, no. 1, pp. 53–57, 1946.

[15] P. Germain, A. Lacasse, F. Laviolette, M. Marchand, and J.-F. Roy, "Risk bounds for the majority vote: From a PAC-Bayesian analysis to a learning algorithm," *Journal of Machine Learning Research (JMLR)*, vol. 16, no. 1, pp. 787–860, 2015.

[16] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research (JAIR)*, vol. 2, pp. 263–286, 1994.

[17] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1061–1069.

[18] Z. Wen, R. Zhang, K. Ramamohanarao, J. Qi, and K. Taylor, "Mascot: fast and highly scalable SVM cross-validation using GPUs and SSDs," in *International Conference on Data Mining*. IEEE, 2014, pp. 580–589.

[19] J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. H. Nguyen, "Generalized Hessian matrix and second-order optimality conditions for problems with $c^{1,1}$ data," *Applied mathematics and optimization*, vol. 11, no. 1, pp. 43–56, 1984.

[20] A. L. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 27, no. 6, pp. 835–850, 2005.

[21] R. Beran *et al.*, "Minimum hellinger distance estimates for parametric models," *The annals of Statistics*, vol. 5, no. 3, pp. 445–463, 1977.

[22] B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," in *Proceedings of the International Symposium on Information Theory*. IEEE, 2004, p. 31.

[23] Z. Wang, C. Li, S. Jegelka, and P. Kohli, "Batched high-dimensional bayesian optimization via structural kernel learning," in *International Conference on Machine Learning*, 2017, pp. 3656–3664.

[24] F. H. Don, "On the symmetric solutions of a linear matrix equation," *Linear Algebra and its Applications*, vol. 93, pp. 1–7, 1987.

[25] A. Klein, S. Falkner, N. Mansur, and F. Hutter, "Robo: A flexible and robust bayesian optimization framework in python," in *NIPS 2017 Bayesian Optimization Workshop*, 2017.

[26] M. Aly, "Survey on multiclass classification methods," *Neural Networks*, vol. 19, pp. 1–9, 2005.

[27] U. Feige, Y. Mansour, and R. E. Schapire, "Robust inference for multiclass classification," in *Algorithmic Learning Theory*, 2018, pp. 368–386.

[28] A. Cotter, N. Srebro, and J. Keshet, "A GPU-tailored approach for training kernelized SVMs," in *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2011, pp. 805–813.

[29] J. T. Zhou, I. W. Tsang, S.-S. Ho, and K.-R. Müller, "N-ary decomposition for multi-class classification," *Machine Learning*, pp. 1–22, 2019.

[30] I. Rish, "An empirical study of the naive Bayes classifier," in *IJCAI Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, no. 22. IBM, 2001, pp. 41–46.

[31] J. Zhou, H. Peng, and C. Y. Suen, "Data-driven decomposition for multi-class classification," *Pattern Recognition*, vol. 41, no. 1, pp. 67–76, 2008.

[32] T. Berg, J. Liu, S. W. Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur, "Birdsnap: Large-scale fine-grained visual categorization of birds," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 2019–2026.

[33] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *International Workshop on Ambient Assisted Living*. Springer, 2012, pp. 216–223.

[34] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2072–2079.

[35] V. Feofanov, E. Devijver, and M.-R. Amini, "Transductive bounds for the multi-class majority vote classifier," in *AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3566–3573.

[36] M. Sun, K. Liu, Q. Wu, Q. Hong, B. Wang, and H. Zhang, "A novel ecoc algorithm for multiclass microarray data classification based on data complexity analysis," *Pattern Recognition*, vol. 90, pp. 346–362, 2019.

[37] Y. Chen, Z. Wen, J. Chen, and J. Huang, "Selecting proper multi-class svm training methods," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[38] Z. Liu, G. Ke, J. Bian, and T. Liu, "Lightmc: A dynamic and efficient multiclass decomposition algorithm," *arXiv preprint arXiv:1908.09362*, 2019.

[39] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 4. IEEE, 2007, pp. IV–317.

[40] F. Nielsen and V. Garcia, "Statistical exponential families: A digest with flash cards," *arXiv preprint arXiv:0911.4863*, 2009.

Appendix A
Derivation of Equation (7)

Next, we present the detailed steps of deriving Equation (7) using the definition of the KL divergence from $\mathtt{N}_j$ to $\mathtt{N}_i$.

$$KL(\mathtt{N}_i \| \mathtt{N}_j) = \int z_i(x) \log \frac{z_i(x)}{z_j(x)} dx \tag{12}$$

where x is a continuous variable, $z_i(x)$ and $z_j(x)$ are the probability densities at x in the distribution $\mathtt{N}_i$ and $\mathtt{N}_j$, respectively. The KL divergence has two important properties: not negative [39] and not symmetric, i.e., $KL(\mathtt{N}_i \| \mathtt{N}_j)$ may not equal to $KL(\mathtt{N}_j \| \mathtt{N}_i)$. As $\mathtt{N}_i$ and $\mathtt{N}_j$ follow the Gaussian distribution, then the probability density function of $\mathtt{N}_i$ is $z_i(x) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp(-\frac{\|x-\mu_i\|^2}{2\sigma_i^2})$ and similarly the probability density function of $\mathtt{N}_j$ is $z_j(x) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp(-\frac{\|x-\mu_j\|^2}{2\sigma_j^2})$. The KL divergence from $\mathtt{N}_j$ to $\mathtt{N}_i$ can be written as below.

$$\begin{aligned} KL(\mathtt{N}_i \| \mathtt{N}_j) &= \int z_i(x) \log \frac{z_i(x)}{z_j(x)} dx \\ &= \log \frac{\sigma_j}{\sigma_i} + \frac{\sigma_i^2 + (\mu_i - \mu_j)^2}{2\sigma_j^2} - \frac{1}{2} \end{aligned} \tag{13}$$

and the KL divergence from $\mathtt{N}_i$ to $\mathtt{N}_j$ can be written as follows.

$$\begin{aligned} KL(\mathtt{N}_j \| \mathtt{N}_i) &= \int z_j(x) \frac{\log z_j(x)}{\log z_i(x)} dx \\ &= \log \frac{\sigma_i}{\sigma_j} + \frac{\sigma_j^2 + (\mu_j - \mu_i)^2}{2\sigma_i^2} - \frac{1}{2} \end{aligned} \tag{14}$$

Derivation of Equations (13) *and* (14)

Here, we derive the KL divergence of two Gaussian distributions $\mathtt{N}_i$ and $\mathtt{N}_j$. We first derive the equation for$KL(\mathtt{N}_i \| \mathtt{N}_j)$. Equation (12) can be rewritten as follows, when $\mathtt{N}_i$ and $\mathtt{N}_j$ follow the Gaussian distribution.

$$\begin{aligned} KL(\mathtt{N}_i \| \mathtt{N}_j) &= \int z_i(x) \log z_i(x) dx - \int z_i(x) \log z_j(x) dx \\ &= -\ln \sqrt{2\pi}\sigma_i - \frac{1}{2} - (-\ln \sqrt{2\pi}\sigma_j - \\ &\quad \frac{\int z_i(x) x^2 dx - \int z_i(x) 2x\mu_j dx + \int z_i(x)\mu_j^2 dx}{2\sigma_j^2}) \\ &= -\ln \sqrt{2\pi}\sigma_i - \frac{1}{2} + \ln \sqrt{2\pi}\sigma_j \\ &\quad + \frac{\sigma_i^2 + \mu_i^2 - 2\mu_i\mu_j + \mu_j^2}{2\sigma_j^2} \\ &= \ln \frac{\sigma_j}{\sigma_i} + \frac{\sigma_i^2 + (\mu_i - \mu_j)^2}{2\sigma_j^2} - \frac{1}{2} \end{aligned} \tag{15}$$

where μ and σ are the mean and variance, respectively. Similarly, we can derive the equation for $KL(\mathtt{N}_j \| \mathtt{N}_i)$.

$$\begin{aligned} KL(\mathtt{N}_j \| \mathtt{N}_i) &= \int z_j(x) \log z_j(x) dx - \int z_j(x) \log z_i(x) dx \\ &= -\ln \sqrt{2\pi}\sigma_j - \frac{1}{2} - (-\ln \sqrt{2\pi}\sigma_i - \\ &\quad \frac{\int z_j(x) x^2 dx - \int z_j(x) 2x\mu_i dx + \int z_i(x)\mu_i^2 dx}{2\sigma_i^2}) \\ &= -\ln \sqrt{2\pi}\sigma_j - \frac{1}{2} + \ln \sqrt{2\pi}\sigma_i \\ &\quad + \frac{\sigma_j^2 + \mu_j^2 - 2\mu_i\mu_j + \mu_i^2}{2\sigma_i^2} \\ &= \ln \frac{\sigma_i}{\sigma_j} + \frac{\sigma_j^2 + (\mu_i - \mu_j)^2}{2\sigma_i^2} - \frac{1}{2} \end{aligned} \tag{16}$$

Appendix B
Problem space of Gaussian naive Bayes

Since this paper focuses on decomposition method, we use a binary problem as an example to demonstrate the problem space of Gaussian naive Bayes. Naive Bayes constructs the classifier based on probabilities. Assume that we have an instance $\boldsymbol{x} \in \mathbb{R}^n$. Let p be the probability and the conditional probability that $\boldsymbol{x}$ belongs to the class C_j is denoted as $p(C = C_j|\boldsymbol{x})$ where $C_j \in \{C_1,\ C_2\}$. Naive Bayes assigns the class with the highest conditional probability to instance $\boldsymbol{x}$. Then based on the assumption that each feature of $\boldsymbol{x}$ is independent, the conditional probability can be computed using the Bayes' theorem as follows.

$$\begin{aligned} p(C = C_1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|C = C_1)p(C = C_1)}{\sum_{C_j \in \{C_1,\ C_2\}} p(\boldsymbol{x}|C = C_j)p(C = C_j)} \\ &= \mathrm{S}(\sum_{i=1}^{n} \ln \frac{p(x_i|C = C_1)}{p(x_i|C = C_2)} + \ln \frac{p(C = C_1)}{p(C = C_2)}) \end{aligned} \tag{17}$$

where $\mathrm{S}(x) = 1/(1 + e^{-x})$ is the sigmoid function and x_i is the i-th element in vector $\boldsymbol{x}$. Gaussian naive Bayes assumes that each feature of the instances that belong to the same class follows Gaussian distribution. Thus the probability of x_i given class C_j (i.e., $p(x_i|C = C_j)$) can be computed using the probability density function for Gaussian distribution. Based on the definition of exponential family [40], we use the canonical decomposition of Gaussian distribution to present $p(x_i|C = C_j)$ as below.

$$p(x_i|C = C_j) = \exp(\boldsymbol{\theta}_{i,C_j} \cdot \boldsymbol{\varphi}_i(x_i) - F(\boldsymbol{\theta}_{i,C_j})) \tag{18}$$

where $\boldsymbol{\theta}_{i,C_j} = (\theta_{i,C_j,1},\ \theta_{i,C_j,2}) = (\frac{\mu_{i,C_j}}{\sigma^2_{i,C_j}},\ -\frac{1}{2\sigma^2_{i,C_j}})$ and $F(\boldsymbol{\theta}_{i,C_j}) = -\frac{\theta^2_{i,C_j,1}}{4\theta^2_{i,C_j,2}} + \frac{1}{2}\log(-\frac{\pi}{\theta_{i,C_j,2}})$; μ_{i,C_j} and σ^2_{i,C_j} are the mean and variance of the i-th features in training instances labeled with class C_j, respectively. The sufficient statistic $\boldsymbol{\varphi}_i(x_i)$ equals $\boldsymbol{\varphi}_i(x_i) = (x_i,\ x_i^2)$ [40]. Substituting Equation (18) into Equation (17), we have

$$p(C = C_1|\boldsymbol{x}) = \mathrm{S}(\sum_{i=1}^{n}(\boldsymbol{\omega}_i \cdot \boldsymbol{\varphi}_i(x_i) + b_i)) \tag{19}$$

where $\boldsymbol{\omega}_i = \boldsymbol{\theta}_{i,C_1} - \boldsymbol{\theta}_{i,C_2}$ and $b_i = [F(\boldsymbol{\theta}_{i,C_2}) - F(\boldsymbol{\theta}_{i,C_1})] + \ln \frac{p(C=C_1)}{p(C=C_2)}$. Equation (19) implies that when feature x_i is mapped into another space where the mapped feature is

$\varphi_i(x_i) = (x_i,\ x_i^2)$, the problem can be solved as a linear problem using Gaussian naive Bayes. Hence the distance can be computed by substituting the mapped instance $\varphi(\boldsymbol{x})$ into Equation (10).

Yawen Chen is currently a PhD candidate at School of Software Engineering, South China University of Technology, China. Her research interests include machine learning, and high-performance computing.

Zeyi Wen is a Lecturer at The University of Western Australia (UWA). Before joining UWA, Zeyi worked as a Research Fellow in National University of Singapore from 2017 and 2019, after receiving his PhD degree from and working as a Research Fellow at The University of Melbourne. Zeyi's areas of research include parallel computing, machine learning and data mining.

Bingsheng He received the bachelor degree in computer science from Shanghai Jiao Tong University (1999-2003), and the PhD degree in computer science in Hong Kong University of Science and Technology (2003-2008). He is an Associate Professor in School of Computing of National University of Singapore. His research interests are high performance computing, distributed and parallel systems, and database systems.

Jian Chen is currently a Professor of the School of Software Engineering at South China University of Technology where she started as an Assistant Professor in 2005. She received her B.S. and Ph.D. degrees, both in Computer Science, from Sun Yat-Sen University, China, in 2000 and 2005 respectively. Her research interests can be summarized as developing effective and efficient data analysis techniques for complex data and the related applications.