Chapter 2
Multiple Criteria Optimization Classification

As the increasingly strong computational power of computers fills the shortage of human brain at calculating, data mining, a major component of data science, has emerged as the times require due to its merit of being capable of extracting novel and useful knowledge which has potential value from large scale of complex data. However, from the mathematical perspective, some data mining methods, such as decision tree, genetic algorithm, and association rules could be considered as heuristic algorithms: which means to select a "better solution" from several alternative solutions as the criterion of classification. These methods lack of exploring how to locate the "best solution" systematically.

Based on [1] and [2], this chapter describes the advanced techniques of applying multi-criteria decision making methods and multi-criteria mathematical programming to conducting data mining process for selecting the "best solution" from multiple alternative solutions, instead of using heuristic algorithms. Section 2.1 is Multi-Criteria Linear Programming (MCLP) for supervised learning, which includes error correction method in classification by using Multiple-Criteria and Multiple-Constraint Levels Linear Programming (MC2LP) [3], Multi-Instance classification based on regularized Multiple Criteria Linear Programming (RMCLP) [4], supportive instances for RMCLP classification [5], and kernel based simple RMCLP for binary classification and regression [6]. Then, Sect. 2.2 describes a group of knowledge-incorporated MCLP classifier [7] and decision rule extraction for RMCLP model [1]. Finally, Sect. 2.3 summarizes three methods of MCDM based data analytics. They are a MCDM approach for estimating the number of clusters [8], parallel RMCLP classification algorithm [9], and an effective intrusion detection framework based on MCLP and support vector machine.

Y. Shi, *Advances in Big Data Analytics*,
https://doi.org/10.1007/978-981-16-3607-3_2

2.1 Multi-criteria Linear Programming for Supervised Learning

2.1.1 Error Correction Method in Classification by Using Multiple-Criteria and Multiple-Constraint Levels Linear Programming

First, the MCLP model for classification is outlined as below [10, 11]:

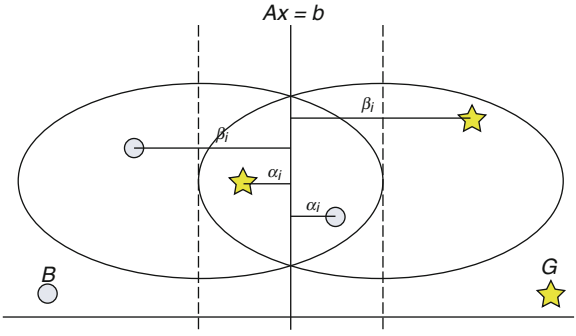
Given a set of n variables about the records $X^T = (x_1, x_2, \ldots, x_l)$, and then let $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$ be one sample of data, where $i = 1, 2, \ldots, l$ and l is the sample size. In linear discriminant analysis, data separation can be achieved by two opposite objectives, that is, minimizing the sum of the deviations (MSD) and maximizing the minimum distances (MMD) of observations from the critical value. That is to say, in order to solve classification problem, we need to minimize the overlapping of data, i.e. α, at the same time, to maximize the distances from the well classified points to the hyperplane, i.e. β.

However, it is difficult for traditional linear programming to optimize MMD and MSD simultaneously. According to the concept of Pareto optimality, we can check all the possible trade-offs between the objective functions by using multiple-criteria linear programming algorithm. The MCLP model can be described by Fig. 2.1.

Moreover, the first Multiple Criteria Linear Programming (MCLP) model can be described as follows:

$$\begin{gathered}
\min \sum_i \alpha_i \\
\min \sum_i \beta_i \\
s.t. A_i X = b + \alpha_i - \beta_i, A_i \in Bad, \\
A_i X = b + \alpha_i - \beta_i, A_i \in Good, \\
\alpha_i, \beta_i \geq 0, i = 1, 2, \ldots, l
\end{gathered}$$

Fig. 2.1 MCLP model



Here, α_i is the overlapping and β_i is the distance from the training sample xi to the discriminator $(w \cdot x_i) = b$ (classification separating hyperplane).

Then, the MC2LP model for classification is introduced in [10].

According to the discussion above, a non-fixed b is very important to our problem. At the same time, for the simplicity and existence of the solution, b should be fixed in some interval.

As a result, for different data, we fix b in different pairs of intervals $[b_l, b_u]$, where b_l and b_u are two fixed numbers. Now our problem is to search the best cutoff between b_l and b_u at every level of their tradeoffs, that is to say, to test every point in the interval $[b_l, b_u]$. We keep the multiple-criteria the same as MCLP, which is, MMD and MSD. And then, the following model is posed:

$$\begin{gathered}
\min \sum_i \alpha_i \\
\min \sum_i \beta_i \\
s.t. A_i X = [b_l, b_u] + \alpha_i - \beta_i, A_i \in Bad, \\
A_i X = [b_l, b_u] + \alpha_i - \beta_i, A_i \in Good, \\
\alpha_i, \beta_i \geq 0, i = 1, 2, \ldots, l
\end{gathered}$$

where A_i, b_l and b_u are given, and X is unrestricted.

In the model, $[b_l, b_u]$ represents a certain tradeoff in the interval. By virtue of the technical of Multiple-criteria and multiple-constraint levels linear programming (MC2LP), we can test each tradeoff between the multiple-criteria and multiple-constraint levels as follows:

$$\begin{gathered}
\min \lambda_1 \sum_i \alpha_i - \lambda_2 \sum_i \beta_i \\
s.t. A_i X = \gamma_1 b_l + \gamma_2 b_u + \alpha_i - \beta_i, A_i \in Bad, \\
A_i X = \gamma_1 b_l + \gamma_2 b_u + \alpha_i - \beta_i, A_i \in Good, \\
\alpha_i, \beta_i \geq 0, i = 1, 2, \ldots, l
\end{gathered}$$

Here, the parameters of $\lambda \times \gamma$ are fixed for each programming problem. Moreover, the advantage of MC2LP is that it can find the potential solutions for all possible trade-offs in the parameter space systematically [12, 13] where the parameter space is

$$\{(\lambda, \gamma) \,|\lambda_1 + \lambda_2 = 1, \gamma_1 + \gamma_2 = 1\}.$$

Of course, in this model, choosing a suitable pair for the goal problem is a key issue and needs domain knowledge. Consequently, a non-parameter choosing MC2LP method should be posed.

For the original MCLP model, one cutoff b is used to predict a new sample's class, that is to say, there is only one hyperplane. The former MC2LP model points out that we can define two cutoffs b_l and b_u instead of the original single cutoff. And then a systematical method can be used to solve this problem. Consequently, all potential solutions at each constrain level tradeoff can be acquired. However, one problem is how to find the cutoffs b_l and b_u.

On one hand, we utilize two cutoffs to discover the solution of higher accuracy; on the other hand, we hope the cutoffs can be obtained from the system directly. Inspired by the idea above, we address our first MC2LP model, which solves the classification problem twice.

For the first step, MCLP model is used to find the vector of external deviations α. It is a function of λ. For simplicity, we set $b = 1$. And then, we fix the parameter of λ to get one potential solution. Now a non-parameter vector of external deviations α is acquired. The component ($\alpha_i > 0$) means the corresponding sample in the training set is misclassified. In other words, Type I and Type II errors occur. According to the idea of MC2LP, we can detect the result of every single MCLP by fixing the parameter of γ at each level in the interval $[b_l, b_u]$. Now, we find the maximal component of α:

$$\alpha_{\max} = \max \{\alpha_i, 1 \leq i \leq l\}. \tag{2.1}$$

Indeed, the smaller the weight of external deviations is, the bigger α_{max} is.

The misclassified samples are all projected into the interval $[1 - \alpha_{max}, 1 + \alpha_{max}]$ according to the weight vector X obtained from the MCLP model. In this way, we define b_l and b_u as $1 - \alpha_{max}$ and $1 + \alpha_{max}$, respectively. It is easy to see, if we want to lessen the number of two types of error, in effect, we just need to inspect the cutoffs by altering the cutoff in the interval

$$[1 - \alpha_{max}, 1 + \alpha_{max}].$$

Moreover, for the second step, a new MC2LP classification model can be stated as follows:

$$\begin{gathered}
\min \lambda_1 \sum_i \alpha_i - \lambda_2 \sum_i \beta_i \\
s.t. A_i X = [1 - \alpha_{\max}, 1 + \alpha_{\max}] + \alpha_i - \beta_i, A_i \in Bad, \\
A_i X = [1 - \alpha_{\max}, 1 + \alpha_{\max}] + \alpha_i - \beta_i, A_i \in Good, \\
\alpha_i, \beta_i \geq 0, i = 1, 2, \ldots, l
\end{gathered}$$

where A_i, α_{max} are given, and X is unrestricted, $[1 - \alpha_{max}, 1 + \alpha_{max}]$ means a certain tradeoff in the interval. At the same time, $\lambda = (\lambda_1, \lambda_2)$ is the parameter chosen in the first step.

The most direct modification of the new MC2LP model is to transfer the single objective function to be a multiple-criteria one. Because the vector of external deviations is a function of λ, it is easy to observe that if the weight between external deviations and internal deviations changes, α changes. Consequently, α_{max} alters. And the ideal α is the one that makes α_{max} not too huge. In other words, we do not hope to check the weight that satisfies λ_1 not too small. Actually, some papers have proved that only if $\lambda_1 > \lambda_2$, then $\alpha \cdot \beta = 0$, which makes the model meaningful [14]. As a result, we only need to check the parameters of objective functions that make α_{max} not too big, in short, not too far away from the original one.

On the other hand, we expect α_{max} not too small. That is to say, we hope the model has some generalization. Hence, two small positive numbers ϵ_1 and ϵ_2 are chosen manually. And then, the interval is built as $[(1 - \alpha_{max} - \epsilon_1, 1 - \alpha_{max} + \epsilon_1), (1 + \alpha_{max} - \epsilon_2, 1 + \alpha_{max} + \epsilon_2)]$. This means that the lower and the upper bound of the interval should be trade-off of some intervals, i.e. the multiple-constrained levels are actually multiple-constrained intervals. Indeed, checking every tradeoff of the intervals is the same as checking every tradeoff of $1 - \alpha_{max} - \epsilon_1$ and $1 + \alpha_{max} + \epsilon_2$. In this case, we can consider the objective function as a multiple-criteria one. It can be stated as follows:

$$\begin{gathered}
\min \sum_i \alpha_i \\
\min \sum_i \beta_i \\
s.t. A_i X = [1 - \alpha_{\max} - \varepsilon_1, 1 + \alpha_{\max} + \varepsilon_2] + \alpha_i - \beta_i, A_i \in Bad, \\
A_i X = [1 - \alpha_{\max} - \varepsilon_1, 1 + \alpha_{\max} + \varepsilon_2] - \alpha_i + \beta_i, A_i \in Bad, \\
\alpha_i, \beta_i \geq 0, i = 1, 2, \ldots, l
\end{gathered} \tag{2.2}$$

where A_i, α_{max}, ϵ_1 and ϵ_2 are given, and X is unrestricted. Here, ϵ_1 and ϵ_2 are two nonnegative numbers.

Lemma 2.1 *For certain trade-off between the objective functions, if b is maintained to be the same sign, then hyperplanes, which are obtained in the MCLP model, keep the same. Furthermore, different signs result in different hyperplanes.*

Proof Assume that the tradeoff between the objective functions is $\lambda = (\lambda_1, \lambda_2)$ and X_1 is the solution obtained by fixing b to be 1. Then, set b_1 as an arbitrary positive number. The MCLP model can be transformed as follows:

$$\begin{gathered}
\min \lambda_1 \sum_i \alpha_i - \lambda_2 \sum_i \beta_i \\
s.t. A_i X = b_1 + \alpha_i - \beta_i, A_i \in Bad, \\
A_i X = b_1 - \alpha_i + \beta_i, A_i \in Good, \\
\alpha_i, \beta_i \geq 0, i = 1, 2, \ldots, l
\end{gathered}$$

The problem above is the same as:

$$\begin{gathered}\min \lambda_1 \frac{\sum\limits_i \alpha_i}{b_1} - \lambda_2 \frac{\sum\limits_i \beta_i}{b_1} \\ s.t. A_i \frac{X}{b_1} = 1 + \frac{\alpha_i}{b_1} - \frac{\beta_i}{b_1}, A_i \in Bad, \\ A_i \frac{X}{b_1} = 1 - \frac{\alpha_i}{b_1} + \frac{\beta_i}{b_1}, A_i \in Good, \\ \alpha_i, \beta_i \geq 0, i = 1, 2, \ldots, l\end{gathered}$$

And then, we let $\alpha i' = \frac{\alpha_i}{b_1}$, $\beta i' = \frac{\beta_i}{b_1}$, $X' = \frac{X}{b_1}$. It is obvious that the solution is $X' = \frac{X}{b_1}$ and the hyperplane $AX' = b_1$ is the same as $AX_1 = 1$.

Similarly, we can prove that when b is a negative number, the solution is the same as the one that is obtained from $b = -1$.

As a result, we just need to compare the solutions (hyperplanes) resulted from $b = 1$ and $b = -1$. For this case, it is easy to see that the signs before α_i and β_i swap when we transform $b = 1$ into $b = 1$. If this happens, then the objective function changes into $-\lambda_1 \sum\limits_i \alpha_i + \lambda_2 \sum\limits_i \beta_i$. This means that the solutions will be different.

According to the lemma, we have the theorem below:

Theorem 2.1 *For our MC2LP model (2.2) above, according to the solutions (hyperplanes), space γ is divided into two non-intersect parts.*

Remark 2.1 When $[1 - \alpha_{max}, 1 + \alpha_{max}]$ is achieved, ϵ_1 and ϵ_2 are chosen to satisfy that 0 is contained by the interval $[1 - \alpha_{max} - \epsilon_1, 1 + \alpha_{max} + \epsilon_2]$. In this case, for any λ, the solutions belong to the trade-offs with same sign will result in the same hyperplane. In other words, there are only two different hyperplanes corresponding to model (2.2). In short, the flexibility of model (2.2) is limited.

In many classification models, including original MCLP model, two types of error is a big issue. In credit card account classification, to correct two types of error can not only improve the accuracy of classification but also help to find some important accounts.

Accordingly, many researchers have focused on this topic. Based on this consideration, more attention should be paid to the samples that locate between two hyperplanes acquired by the original MCLP model, that is, the points in the grey zone [15]. Consequently, we define the external deviations and internal deviations related to two different hyperplanes, the left one and the right one, that is, α^l, α^r, β^l and β^r.

Definition 2.1 The conditions the deviations should satisfy are stated as follows:

$$\alpha_i^l = \begin{cases} 0, & A_i X < 1 - \alpha_{max} and A_i \in Bad; \\ A_i X - (1 - \alpha_{max}), & A_i X \geq 1 - \alpha_{max} and A_i \in Bad; \\ 0, & A_i X \geq 1 - \alpha_{max} and A_i \in Good; \\ (1 - \alpha_{max}) - A_i X, & A_i X < 1 - \alpha_{max} and A_i \in Good. \end{cases}$$

$$\alpha_i^r = \begin{cases} 0, & A_i X < 1 + \alpha_{max} and A_i \in Bad; \\ A_i X - (1 + \alpha_{max}), & A_i X \geq 1 + \alpha_{max} and A_i \in Bad; \\ 0, & A_i X \geq 1 + \alpha_{max} and A_i \in Good; \\ (1 + \alpha_{max}) - A_i X, & A_i X < 1 + \alpha_{max} and A_i \in Good. \end{cases}$$

$$\beta_i^l = \begin{cases} (1 - \alpha_{max}) - A_i X, & A_i X < 1 - \alpha_{max} and A_i \in Bad; \\ 0, & A_i X \geq 1 - \alpha_{max} and A_i \in Bad; \\ A_i X - (1 - \alpha_{max}), & A_i X \geq 1 - \alpha_{max} and A_i \in Good; \\ 0, & A_i X < 1 - \alpha_{max} and A_i \in Good. \end{cases}$$

$$\beta_i^r = \begin{cases} (1 + \alpha_{max}) - A_i X, & A_i X < 1 + \alpha_{max} and A_i \in Bad; \\ 0, & A_i X \geq 1 + \alpha_{max} and A_i \in Bad; \\ A_i X - (1 + \alpha_{max}), & A_i X \geq 1 + \alpha_{max} and A_i \in Good; \\ 0, & A_i X < 1 + \alpha_{max} and A_i \in Good. \end{cases}$$

Figure 2.2 is a sketch for the model. In the graph, the green and the red lines are the left and right hyperplane, b^l and b^r respectively, which are some trade-offs in two intervals, i.e. $[1 - \alpha_{max} - \epsilon_2, 1]$ and $[1, 1 + \alpha_{max} + \epsilon_1]$. And all

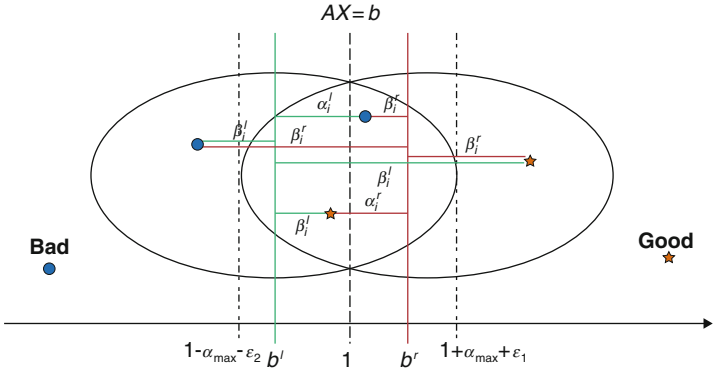


Fig. 2.2 MC2LP model

the deviations are measured according to them in different colors. For instance, if a sample in "Good" class is misclassified as "Bad" class, it means ${\alpha_i}^r > {\beta_i}^l \geq 0$ and ${\alpha_i}^l = {\beta_i}^r = 0$. And then, if a sample in "Bad" class is misclassified as "Good" class, it means ${\alpha_i}^l > {\beta_i}^r \geq 0$ and ${\alpha_i}^r = {\beta_i}^l = 0$. Thus, for the misclassified ones, ${\alpha_i}^r + {\alpha_i}^l - {\beta_i}^r - {\beta_i}^l$ should be minimized.

As a result, a more meticulous model could be stated as follows:

$$
\begin{gathered}
\min \sum_i \left(\alpha_i^r + \alpha_i^l\right) \\
\min \sum_i \left(\alpha_i^l - \beta_i^r\right) \\
\min \sum_i \left(\alpha_i^r - \beta_i^l\right) \\
\max \sum_i \left(\beta_i^r + \beta_i^l\right) \\
s.t. A_i X = 1 + [0, \alpha_{\max} + \varepsilon_1] + \alpha_i^r - \beta_i^r, A_i \in Bad, \\
A_i X = 1 - [0, \alpha_{\max} + \varepsilon_2] + \alpha_i^l - \beta_i^l, A_i \in Bad, \\
A_i X = 1 + [0, \alpha_{\max} + \varepsilon_1] - \alpha_i^r + \beta_i^r, A_i \in Good, \\
A_i X = 1 - [0, \alpha_{\max} + \varepsilon_2] - \alpha_i^l + \beta_i^l, A_i \in Good, \\
\alpha_i^r, \alpha_i^l \beta_i^r, \beta_i^l \geq 0, i = 1, 2, \ldots, l.
\end{gathered}
$$

where A_i, α_{max}, $\epsilon_1 > 0$, $\epsilon_2 > 0$ are given, and X is unrestricted.

In Fig. 2.2, for each point, there are at most two kinds of deviations nonzero. The objective functions appear to deal with the deviations according to the position shown in Fig. 2.2, respectively, whereas they have their own special meaning. That is to say, it measures two types of error in some degree by means of the second and third objective functions. As a result, in this new version of MC2LP, we not only consider the deviations respectively, but also take the relationship of the deviations based on two types of error into account in the objective functions. By virtue of MC2LP method, each tradeoff between $1 - \alpha_{max} - \epsilon_2$ and 1 for the left hyperplane as well as each tradeoff between 1 and $1 + \alpha_{max} + \epsilon_1$ for the right hyperplane can be checked.

After obtaining the weight vector X of the hyperplane, $AX = 1$ is still used to be the classification hyperplane. However, in our new model, we minimize the distance between the left hyperplane and the right one. In other words, we discover the hyperplane that genders the smallest grey area.

Actually, in statistics, Type I and Type II errors are two opposite objectives. That is to say, it is very hard to correct both of them at the same time. As a result, we modify the former model into two different models focusing on two types of error

respectively as follows:

$$\begin{gathered}
\min \sum_i \left(\alpha_i^r + \alpha_i^l\right) \\
\min \sum_i \left(\alpha_i^l - \beta_i^r\right) \\
\max \sum_i \left(\beta_i^r + \beta_i^l\right) \\
s.t. A_i X = 1 + [0, \alpha_{\max} + \varepsilon] + \alpha_i^r - \beta_i^r, A_i \in Bad, \\
A_i X = 1 + \alpha_i^l - \beta_i^l, A_i \in Bad, \\
A_i X = 1 + [0, \alpha_{\max} + \varepsilon] - \alpha_i^r + \beta_i^r, A_i \in Good, \\
A_i X = 1 - \alpha_i^l + \beta_i^l, A_i \in Good, \\
\alpha_i^r, \alpha_i^l \beta_i^r, \beta_i^l \geq 0, i = 1, 2, \ldots, l.
\end{gathered} \tag{2.3}$$

where A_i, α_{max} and $\epsilon > 0$ are given, and X is unrestricted. In this model, $\sum_i \alpha_i^r - \beta_i^l$ is not contained in the objective functions. This model can deal with Type II error, that is, classifying a "Good" point to be a "Bad" one. Now we provide an example to illustrate the effect of model (2.2).

As the result shown above, model (2.3) can correct Type II error in some degree. We conclude this in the proposition below.

Proposition 2.1 *Model (2.3) can correct Type II error by moving the right hyperplane to the right based on the concept of multiple-constraint levels.*

Note that the second objective function in model (2.3) is nonzero for the samples in class "Bad" and getting negative when the right hyperplane moving to the right. That is to say, we tolerate some Type I errors. At the same time, the first objective function in model (2.3) renders Type II errors an increasing punishment with moving the right hyperplane to the right. As a result, it can correct Type II error in some degree.

Similar to model (2.3), (2.4) is posed to deal with Type I error as follows:

$$\begin{gathered}
\min \sum_i \left(\alpha_i^r + \alpha_i^l\right) \\
\min \sum_i \left(\alpha_i^l - \beta_i^r\right) \\
\min \sum_i \left(\beta_i^r + \beta_i^l\right) \\
s.t. \quad A_i X = 1 + \alpha_i^r - \beta_i^r, A_i \in Bad, \\
A_i X = 1 - [0, \alpha_{\max} + \varepsilon_2] + \alpha_i^l - \beta_i^l, A_i \in Bad, \\
A_i X = 1 - \alpha_i^r + \beta_i^r, A_i \in Good, \\
A_i X = 1 - [0, \alpha_{\max} + \varepsilon_2] - \alpha_i^l + \beta_i^l, A_i \in Good, \\
\alpha_i^r, \alpha_i^l \beta_i^r, \beta_i^l \geq 0, i = 1, 2, \ldots, l.
\end{gathered} \tag{2.4}$$

where A_i, α_{max} and $\epsilon > 0$ are given, and X is unrestricted. In this model, $\sum_i \alpha_i^l - \beta_i^r$ is not contained in the objective functions. This model focuses on Type I error, that is, classifying a "Bad" point to be a "Good" one.

The numerical examples to illustrate the theoretical results of this section can be found in [3].

2.1.2 *Multi-instance Classification Based on Regularized Multiple Criteria Linear Programming*

Multi-instance learning (MIL) has received intense interest recently in the field of machine learning. This idea was originally proposed for handwritten digit recognition by [16]. The term multi-instance learning was first introduced by [17] when they were investigating the problem of binding ability of a drug activity prediction. In MIL framework, the training set consists of positive and negative bags of points in the n-dimensional real-space R^n, and each bag contains a number of points (instances). A positive training bag contains at least one positive instance, whereas a negative bag contains only negative instances. The aim of MIL is to construct a learned classifier from the training set for correctly labeling unseen bags. Multi-instance learning has been found useful in diverse domains such as object detection, text categorization, image categorization, image retrieval, web mining, computer-aided medical diagnosis, etc. [12–14, 18].

In this subsection, we propose a novel Multi-instance Learning method based on Regularized Multiple Criteria Linear Programming (called MI-RMCLP), which includes two algorithms for linear and nonlinear cases separately. To our knowledge, MI-RMCLP is the first RMCLP implementation based on MIL, which is a useful extension of RMCLP. The original MI-RMCLP model proposed itself is a nonconvex optimization problem. By an appropriate modification, we will the model to derive two quadratic programming subproblems, which can arrive at the optimal value by an iterative strategy solving these sequential subproblems. All preliminary numerical experiments show that our approach is competitive with other multiple learning formulations.

We first give a brief introduction of RMCLP in the following. For classification about the training data:

$$T = \{(x_1, y_1), \cdots, (x_l, y_l)\} \in \left(R^n \times y\right)^l,$$

where $x_i \in R^n$, $y_i \in = \{1, -1\}$, $i = 1, \cdots, l$, data separation can be achieved by two opposite objectives. The first objective separates the observations by minimizing the sum of the deviations (MSD) among the observations. The second maximizes the minimum distances (MMD) of observations from the critical value [19]. The overlapping of data u should be minimized, while the distance v has to be maximized. However, it is difficult for traditional linear programming to optimize MMD and MSD simultaneously. According to the concept of Pareto optimality, we can seek the best trade-off of the two measurements [2, 20]. So MCLP model can

be described as follows:

$$\min_{u} e^T u \& \max_{v} e^T v, \tag{2.5}$$

$$s.t.\,(w \cdot x_i) + (u_i - v_i) = b, \text{for } \{i \,|\, y_i = 1\}, \tag{2.6}$$

$$(w \cdot x_i) - (u_i - v_i) = b, \text{for } \{i \,|\, y_i = -1\}, \tag{2.7}$$

$$u, v \geq 0, \tag{2.8}$$

where $e \in R^l$ be vector whose all elements are 1, w and b are unrestricted, u_i is the overlapping, and v_i the distance from the training sample x_i to the discriminator $(w \cdot x_i) = b$ (classification separating hyperplane). By introducing penalty parameter $c, d > 0$, MCLP has the following version

$$\min_{u,v} ce^T u - de^T v, \tag{2.9}$$

$$s.t.\,(w \cdot x_i) + (u_i - v_i) = b, \text{for } \{i \,|\, y_i = 1\}, \tag{2.10}$$

$$(w \cdot x_i) - (u_i - v_i) = b, \text{for } \{i \,|\, y_i = -1\}, \tag{2.11}$$

$$u, v \geq 0, \tag{2.12}$$

The geometric meaning of the model is shown in Fig. 2.3.

A lot of empirical studies have shown that MCLP is a powerful tool for classification. However, we cannot ensure this model always has a solution under different kinds of training samples. To ensure the existence of solution, recently, Shi et al. proposed a RMCLP model by adding two regularized items $\frac{1}{2} w^T H w$ and

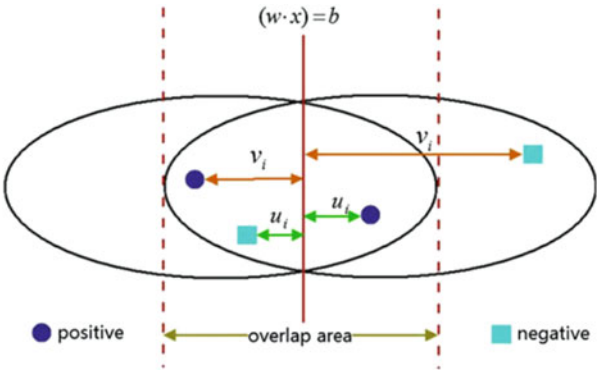


Fig. 2.3 Geometric meaning of MCLP

$\frac{1}{2}u^T Qu$ on MCLP as follows (more theoretical explanation of this model can be found in [2]):

$$\min_z \frac{1}{2}w^T Hw + \frac{1}{2}u^T Qu + de^T u - ce^T v, \tag{2.13}$$

$$s.t.\,(w \cdot x_i) + (u_i - v_i) = b, \text{for } \{i \,|y_i = 1\}, \tag{2.14}$$

$$(w \cdot x_i) - (u_i - v_i) = b, \text{for } \{i \,|y_i = -1\}, \tag{2.15}$$

$$u, v \geq 0, \tag{2.16}$$

where $z = (w^T, u^T, v^T, b)^T \in R^{n+l+l+1}$, $H \in R^n \times n$, $Q \in R^{l \times l}$ are symmetric positive definite matrices. Obviously, the regularized MCLP is a convex quadratic programming.

Compared with traditional SVM, we can find that the RMCLP model is similar to the Support Vector Machine model in terms of the formation by considering minimization of overlapping of the data. However, RMCLP tries to measure all possible distances v from the training samples x_i to separating hyperplane, while SVM fixes the distance as 1 (through bounding planes $(w \cdot x) = b \pm 1$) from the support vectors. Although the interpretation can vary, RMCLP addresses more control parameters than the SVM, which may provide more flexibility for better separation of data under the framework of the mathematical programming. In addition, different with SVM, RMCLP considers all the samples to solve classification problem. These make RMCLP have stronger insensitivity to outliers.

One of the drawbacks of applying the supervised learning model is that it is not always possible for a teacher to provide labeled examples for training. Multiple instance learning (MIL) provides a new way of modeling the teachers' weakness. MIL considers a particular form of weak supervision in which training class labels are associated with sets of patterns, or bags, instead of individual patterns. A negative bag only consists of negative instances, whereas a positive bag comprises both positive and negative instances. The goal of MIL is to find a separate hyperplane, which can decide the label of any new instance.

In the following, we give the formal description of multiple instance learning problem. Given a training set

$$\left\{\mathbf{B}_1^+, \cdots, \mathbf{B}_{m^+}^+, \mathbf{B}_1^-, \cdots, \mathbf{B}_{m^-}^-\right\} \tag{2.17}$$

where a bag $\mathbf{B}_i^+ = \left\{x_{i1}, \cdots, x_{im_i^+}\right\}, x_{ij} \in R^n, j = 1, \cdots, m_i^+, i = 1, \cdots, m^+; \mathbf{B}_i^- = \left\{x_{i1}, \cdots, x_{im_i^-}\right\}, x_{ij} \in R^n, j = 1, \cdots, m_i^-, i = 1, \cdots, m^-;$ $\mathbf{B}^+$ means that the positive bag $\mathbf{B}^+$ contains at least one positive instance x_{ij}; $\mathbf{B}^-$ means that all instance x_{ij} of the negative bag $\mathbf{B}^-$ are negative. The goal is to induce

a real-valued function

$$y = \operatorname{sgn}(\mathbf{g}(x)) \tag{2.18}$$

such that the label of any instance x in R^n space can be predicted.

Now we rewrite the training set (2.17) as

$$\text{Train} = \left\{\mathbf{B}_1^+, \cdots, \mathbf{B}_{m^+}^+, \mathbf{B}_{m^+1}^-, \cdots, \mathbf{B}_{m^+m^-}^-, \right\} = \left\{\mathbf{B}_1^+, \cdots, \mathbf{B}_{m^+}^+, x_{z+1}, \cdots, x_{z+f}\right\} \tag{2.19}$$

where z is the number of the instances in all positive bags and f the number of the instances in negative bags.

The set consisting of subscripts of B_i is expressed as:

$$\Im(i) = \{i \mid x_i \in \mathbf{B}_i\} \tag{2.20}$$

For a separable multi-instance classification problem, if a positive bag can be correctly classified, it should satisfy the following constraint:

$$\max_{j \in \Im(i)} \left(w \cdot x_j\right) - b > 0. \tag{2.21}$$

In RMCLP, v_i means the distance from the training sample x_i to the separating hyperplane and be a nonnegative number. Thus, we can always find an appropriate v_i such that

$$\max_{j \in \Im(i)} \left(w \cdot x_j\right) - b = v_i. \tag{2.22}$$

For nonseparable multi-instance classification, we need to add corresponding slack variable $u_i \geq 0$. Finally, the (2.22) is expressed by

$$\max_{j \in \Im(i)} \left(w \cdot x_j\right) - b = v_i - u_i. \tag{2.23}$$

Similar to [21], it is equivalent to the fact that there exist convex combination coefficients set $\left\{\lambda_j^i \,\middle|\, j \in \Im(i), i = 1, \cdots, m^+\right\}$, such that

$$\left(w \cdot \sum_{j \in \Im(i)} \lambda_j^i x_j\right) + u_i - v_i = b, \tag{2.24}$$

$$\lambda_j^i \geq 0, \sum_{j \in \Im(i)} \lambda_j^i = 1. \tag{2.25}$$

For solving multi-instance classification, so (2.6–2.9) can be converted as:

$$\min_z \frac{1}{2}\|w\|^2 + \frac{1}{2}\|u\|^2 + d\sum_{i=1}^{m^+} u_i + d\sum_{i=z+1}^{z+f} u_i - c\sum_{i=1}^{m^+} v_i - c\sum_{i=z+1}^{z+f} v_i, \tag{2.26}$$

$$s.t.\left(w \cdot \sum_{j\in\Im(i)} \lambda_j^i x_j\right) + (u_i - v_i) = b, i = 1, \cdots, m^+, \tag{2.27}$$

$$(w \cdot x_i) - (u_i - v_i) = b, i = z+1, \cdots, z+f, \tag{2.28}$$

$$\lambda_j^i \geq 0, j \in \Im(i), i = 1, \cdots, m^+, \tag{2.29}$$

$$\sum_{j\in\Im(i)} \lambda_j^i = 1, i = 1, \cdots, m^+, \tag{2.30}$$

$$u, v \geq 0, \tag{2.31}$$

where $z = \left(w^T, u^T, v^T, b, \lambda^T\right)^T, \lambda = \left\{\lambda_j^i \,|j \in \Im(i)\,, i = 1, \cdots, m^+\right\}$, $\Im(i) = \left\{i \,|x_i \in \mathbf{B}_i^+\right\}$.

As both λ_j^i and w are variables, the constraint (2.27) is no longer a linear constraint and (2.26–2.31) becomes a nonlinear optimization problem.

In the following, we give an approximate iterative solution via solving successive quadratic programming problem. Firstly, we fix λ, and solve a quadratic programming with respect to w, u, v, b; then fix w, solve a quadratic programming with respect to u, v, b, λ.

1. For fixed $\lambda_j^i, i = 1, \cdots, m^+, j \in \Im(i)$, we can obtain

$$\hat{x}_i = \sum_{j\in\Im(i)} \lambda_j^i x_j, i = 1, \cdots, m^+, \tag{2.32}$$

So the problem (2.26–2.31) can be written as

$$\min_z \frac{1}{2} w^T H w + \frac{1}{2} u^T Q u + d e^T u - c e^T v, \tag{2.33}$$

$$s.t.\left(w \cdot \hat{x}_i\right) + (u_i - v_i) = b, i = 1, \cdots, m^+, \tag{2.34}$$

$$\left(w \cdot \hat{x}_i\right) - (u_i - v_i) = b, i = z+1, \cdots, z+f, \tag{2.35}$$

$$u, v \geq 0, \tag{2.36}$$

The problem (2.33–2.36) is a standard quadratic programming problem and as same as RMCLP. We choose H and Q to be identity matrix. Its dual problem

can be formulated as

$$\begin{aligned}\max_{\alpha,u} &-\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=1}^{m^+}\left(\left(\hat{x}_i\cdot\hat{x}_j\right)+1\right)\alpha_i\alpha_j\\ &-\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=z+1}^{z+f}\left(\left(\hat{x}_i\cdot\hat{x}_j\right)+1\right)\alpha_i\alpha_j\\ &-\frac{1}{2}\sum_{i=2+1}^{z+f}\sum_{j=1}^{m^+}\left(\left(\hat{x}_i\cdot\hat{x}_j\right)+1\right)\alpha_i\alpha_j\\ &-\frac{1}{2}\sum_{i=2+1}^{z+f}\sum_{j=z+1}^{z+f}\left(\left(\hat{x}_i\cdot\hat{x}_j\right)+1\right)\alpha_i\alpha_j\\ &-\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=1}^{m^+}u_iu_j-\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=z+1}^{z+f}u_iu_j\\ &-\frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=1}^{m^+}u_iu_j\frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=z+1}^{z+f}u_iu_j\end{aligned} \tag{2.37}$$

$$s.t.-u_i-d\le\alpha_i\le-c,i=1,\cdots,m^+, \tag{2.38}$$

$$-u_i-d\le-\alpha_i\le-c,i=z+1,\cdots,z+f, \tag{2.39}$$

where $c, d > 0$. We can compute: $\hat{\alpha}=\left(\hat{\alpha}_1,\cdots,\hat{\alpha}_{m^+},\hat{\alpha}_{z+1},\cdots,\hat{\alpha}_{z+f}\right)^T$ by solving the problem of (2.37–2.39), and (w, b) can be expressed as

$$\hat{w}=-\sum_{i=1}^{m^+}\hat{\alpha}_i\hat{x}_i-\sum_{i=z+1}^{z+f}\hat{\alpha}_i\hat{x}_i, \tag{2.40}$$

$$\hat{b}=\sum_{i=1}^{m^+}\hat{\alpha}_i+\sum_{i=z+1}^{z+f}\hat{\alpha}_i, \tag{2.41}$$

$\hat{w}, \hat{b}$ is the updating value of (w, b).

2. For fixed w, the formula (2.26–2.31) can be substituted as:

$$\begin{aligned}\min_{\lambda,u,v,b}&\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=1}^{m^+}u_iu_j+\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=z+1}^{z+f}u_iu_j+\frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=1}^{m^+}u_iu_j+\frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=z+1}^{z+f}u_iu_j\\ &+d\sum_{i=1}^{m^+}u_i+d\sum_{i=z+1}^{z+f}u_i-c\sum_{i=1}^{m^+}v_i-c\sum_{i=z+1}^{z+f}v_i\end{aligned} \tag{2.42}$$

$$s.t. \left(w \cdot \sum_{j \in \Im(i)} \lambda_j^i x_j \right) + (u_i - v_i) = b, i = 1, \cdots, m^+, \tag{2.43}$$

$$(w \cdot x_i) - (u_i - v_i) = b, i = z + 1, \cdots, z + f, \tag{2.44}$$

$$\lambda_j^i \geq 0, j \in \Im(i), i = 1, \cdots, m^+, \tag{2.45}$$

$$\sum_{j \in \Im(i)} \lambda_j^i = 1, i = 1, \cdots, m^+, \tag{2.46}$$

$$u, v \geq 0, \tag{2.47}$$

thus we are able to establish the following Algorithm 2.1 based on the formulas above.

Algorithm 2.1 Linear MI-RMCLP

Initialize: Given a training set (see (2.19));
Choose appropriate penalty parameters $c, d > 0$;
Choose Q and H to be identity matrixes;
Setting initial values for λ $(k = 1)$, where $\left\{ \lambda_j^i(1) \,\middle|\, j \in \Im(i), i = 1, \cdots, m^+ \right\}$;
Process: **1**. For fixed $\lambda(k) = \left\{ \lambda_j^i(k) \right\}$, the goal is to compute $w(k)$:
1.1. Compute $\left\{ \hat{x}_1, \cdots, \hat{x}_{m^+}, \hat{x}_{r_1}, \cdots, \hat{x}_{z+f} \right\}$ by (2.32);
1.2. Solve quadratic programming (2.38) ~ (2.39),
obtaining the solution $\hat{\alpha} = \left(\hat{\alpha}_1, \cdots, \hat{\alpha}_p, \hat{\alpha}_{z+1}, \cdots, \hat{\alpha}_{z+f} \right)^T$;
1.2. Compute $\hat{w}$ from (2.40);
1.4. Set $w(k) = \hat{w}$.
2. For fixed $w(k)$, the goal is to compute $\lambda(k + 1)$:
2.1. Solve quadratic programming (2.42) ~ (2.47) with the
variables λ, u, v, b, obtaining the solution $\hat{\lambda}$, $\hat{b}$.
2.2. Set $\lambda\ (k + 1) = \hat{\lambda}$, $b\ (k + 1) = \hat{b}$;
2. If $|\lambda(k + 1) - \lambda(k)| < \varepsilon$, goto **Output**:; otherwise,
goto the step 1, setting $k = k + 1$.
Output: Obtain the decision function $f(x) = \text{ sgn } ((w^* \cdot x) + b^*)$,
where $w^* = w(k)$, $b^* = b(k)$.

For nonlinear MI-RMCLP, we firstly introduce the kernel function $K(x, x') = (\Phi(x) \cdot \Phi(x'))$ to replace (x, x'), where $\Phi(x)$ is a mapping from the input space R^n to some Hilbert space $\mathbb{H}$:

$$\Phi : R^n \to \mathbb{H}$$

$$x \to \mathrm{x} = \Phi(x) \tag{2.48}$$

Therefore, the problem (2.26–2.31) can be expressed as

$$\min_{z} \frac{1}{2}\|w\|^2 + \frac{1}{2}\|u\|^2 + d\sum_{i=1}^{m^+} u_i + d\sum_{i=z+1}^{z+f} u_i - c\sum_{i=1}^{m^+} v_i - c\sum_{i=z+1}^{z+f} v_i \tag{2.49}$$

$$s.t. \left(\mathrm{w} \cdot \sum_{j\in\Im(i)} \lambda_j^i \Phi\left(x_j\right)\right) + (u_i - v_i) = b, i = 1, \cdots, m^+, \tag{2.50}$$

$$(\mathrm{w} \cdot \Phi\left(x_i\right)) - (u_i - v_i) = b, i = z+1, \cdots, z+f, \tag{2.51}$$

$$\lambda_j^i \geq 0, j \in \Im(i), i = 1, \cdots, m^+, \tag{2.52}$$

$$\sum_{j\in\Im(i)} \lambda_j^i = 1, i = 1, \cdots, m^+, \tag{2.53}$$

$$u, v \geq 0, \tag{2.54}$$

Similar to Algorithm 2.1, as a given λ, the current problem can be solved by the following quadratic programming problem:

$$\begin{aligned}
\max_{\alpha,u} & -\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=1}^{m^+}\left(\sum_{k\in\Im(i)}\lambda_k^i \sum_{l\in I(j)}\lambda_l^j K\left(x_k \cdot x_l\right) + 1\right)\alpha_i\alpha_j \\
& -\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=z+1}^{z+f}\left(\sum_{k\in\Im(i)}\lambda_k^i K\left(x_k \cdot x_j\right) + 1\right)\alpha_i\alpha_j \\
& -\frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=1}^{m^+}\left(\sum_{l\in I(j)}\lambda_l^j K\left(x_i \cdot x_l\right) + 1\right)\alpha_i\alpha_j \\
& -\frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=z+1}^{z+f}\left(K\left(x_i \cdot x_j\right) + 1\right)\alpha_i\alpha_j \\
& -\frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=1}^{m^+} u_i u_j - \frac{1}{2}\sum_{i=1}^{m^+}\sum_{j=z+1}^{z+f} u_i u_j \\
& -\frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=1}^{m^+} u_i u_j - \frac{1}{2}\sum_{i=z+1}^{z+f}\sum_{j=z+1}^{z+f} u_i u_j
\end{aligned} \tag{2.55, 2.56}$$

$$s.t. -u_i - d \leq \alpha_i \leq -c, i = 1, \cdots, m^+, \tag{2.57}$$

$$-u_i - d \leq -\alpha_i \leq -c, i = z+1, \cdots, z+f, \tag{2.58}$$

We can obtain a solution of $\left(\hat{w}, \hat{b}\right)$ by computing

$$\hat{\mathrm{w}} = -\sum_{i=1}^{m^+} \hat{\alpha}_i \sum_{j \in \Im(i)} \lambda_j^i \Phi\left(x_i\right) - \sum_{i=z+1}^{z+f} \hat{\alpha}_i \Phi\left(x_i\right), \tag{2.59}$$

$$\hat{b} = \sum_{i=1}^{m^+} \hat{\alpha}_i + \sum_{i=z+1}^{z+f} \hat{\alpha}_i, \tag{2.60}$$

where $\hat{\alpha} = \left(\hat{\alpha}_1, \cdots, \hat{\alpha}_p, \hat{\alpha}_{z+1}, \cdots, \hat{\alpha}_{z+f}\right)^T$ is a solution of the problem (2.56)–(2.58).

For fixed w, the problem (2.49–2.54) can be written as

$$\begin{aligned} \min_{\lambda,u,v,b} & \frac{1}{2} \sum_{i=1}^{m^+} \sum_{j=1}^{m^+} u_i u_j + \frac{1}{2} \sum_{i=1}^{m^+} \sum_{j=z+1}^{z+f} u_i u_j + \frac{1}{2} \sum_{i=z+1}^{z+f} \sum_{j=1}^{m^+} u_i u_j + \frac{1}{2} \sum_{i=z+1}^{z+f} \sum_{j=z+1}^{z+f} u_i u_j \\ & + d \sum_{i=1}^{m^+} u_i + d \sum_{i=z+1}^{z+f} u_i - c \sum_{i=1}^{m^+} v_i - c \sum_{i=z+1}^{z+f} v_i \end{aligned} \tag{2.61}$$

$$s.t. - \sum_{j=1}^{m^+} \hat{\alpha}_j \sum_{k \in I(j)} \tilde{\lambda}_k^j \sum_{l \in \Im(i)} \lambda_l^i K\left(x_k, x_l\right) - \sum_{j=z+1}^{z+f} \hat{\alpha}_j \sum_{l \in \Im(i)} \lambda_l^i K\left(x_j, x_l\right) - \left(u_i - v_i\right) = b,$$

$$i = 1, \cdots, m^+, \tag{2.62}$$

$$-\sum_{j=1}^{m^+} \hat{\alpha}_j \sum_{k \in I(j)} \tilde{\lambda}_k^j K\left(x_k, x_i\right) - \sum_{j=z+1}^{z+f} \hat{\alpha}_j K\left(x_j, x_i\right) + \left(u_i - v_i\right) = b, i = z+1, \cdots, z+f \tag{2.63}$$

$$\lambda_j^i \geq 0, j \in \Im(i), i = 1, \cdots, m^+, \tag{2.64}$$

$$\sum_{j \in \Im(i)} \lambda_j^i = 1, i = 1, \cdots, m^+, \tag{2.65}$$

$$u, v \geq 0, \tag{2.66}$$

where $\tilde{\lambda} = \left(\tilde{\lambda}_i^i \,|\, j \in \Im(i),\ i = 1, \ldots, m^+\right)$ and $\hat{\alpha} = \left(\hat{\alpha}_1, \cdots, \hat{\alpha}_{z+1}, \cdots, \hat{\alpha}_{z+f}\right)^T$ are known.

The ultimate separating hypersurface can be expressed as

$$g(x) = -\sum_{j=1}^{m^+} \hat{\alpha}_j \sum_{k \in I(j)} \tilde{\lambda}_k^j K\left(x_k, x\right) - \sum_{j=z+1}^{z+f} \hat{\alpha}_j K\left(x_j, x\right) + \hat{b}, \tag{2.67}$$

In the following, we give out Algorithm 2.2 for nonlinear MI-RMCLP.

Algorithm 2.2 Nonlinear MI-RMCLP

Initialize: Given a training set (see (2.19));
Choose appropriate penalty parameters $c, d > 0$;
Choose Q and H to be identity matrixes;
Choose appropriate
Setting initial values for λ $(k = 1)$, where $\left\{\lambda_j^i(1) \,\middle|\, j \in \Im(i), i = 1, \cdots, m^+\right\}$;
Process: **1**. For fixed $\lambda(k) = \left\{\lambda_j^i(k)\right\}$, the goal is to compute $w(k)$:
1.1. Solve quadratic programming (2.56) ~ (2.58), obtaining the solution.
$\hat{\alpha} = \left(\hat{\alpha}_1, \cdots, \hat{\alpha}_p, \hat{\alpha}_{z+1}, \cdots, \hat{\alpha}_{z+f}\right)^T$;
1.2. Set $\tilde{\lambda} = \lambda(k)$;
2. For fixed $\hat{\alpha}, \tilde{\lambda}$, the goal is to compute $\hat{\lambda} = \left\{\lambda_j^i\right\}$:
2.1. Solve quadratic programming (2.61) ~ (2.66) with the variables (λ, u, v, b), obtaining the solution $\hat{\lambda} = \left\{\lambda_j^i\right\}$.
2.2. Set $\lambda\,(k+1) = \hat{\lambda}, b\,(k+1) = \hat{b}$;
2. If $|\lambda(k+1) - \lambda(k)| < \varepsilon$, goto **Output**:; otherwise, goto the step 1, setting $k = k + 1$.
Output: Obtain the decision function $f(x) = \text{sgn}\,(g(x))$, where $g(x)$ by (2.18).

To demonstrate the capabilities of our algorithm, we report results on 12 data sets, 2 from the UCI machine learning repository [22], and 10 from [23]. "Elephant," "Fox" and "Tiger" data sets are from an image annotation task in which the goal is to determine whether or not a given animal is present in an image. The other seven data sets are from the OHSUMED data, and the task is to learn binary concepts associated with the Medical Subject Headings of MEDLINE documents. The "Musk1" and "Musk2" data sets from the UCI machine learning repository are used to test our nonlinear multi-instance RMCLP, which involves bags of molecules and their activity levels and is commonly used in multi-instance classification. Detailed information about these data sets can be found in [21].

Our algorithm code was written in MATLAB 2010. The experiment environment is Intel Core i5 CPU, 2 GB memory. The "quadprog" function with MATLAB is employed to solve quadratic programming problem related to this section. The testing accuracies for our method are computed using standard tenfold cross-validation [24]. The RBF kernel parameter σ is selected from the set $\{2^i | i = -7, \cdots, 7\}$ by tenfold cross-validation on the tuning set comprising of random 10% of the training data. Once the parameters are selected, the tuning set was returned to the training set to learn the final decision function. The (c, d) are set 1. If the difference between 2 is less than 10^{-4} or the iterations $K > 100$, our algorithms will be stopped.

We compare our results with MICA [21], mi-SVM [23], MI-SVM [25], EM-DD [25] and SVM-CC [26]. MI-RMCLP is our method in Table 2.1 and Fig. 2.4. The results of tenfold cross-validation accuracy are listed in Table 2.1 and Fig. 2.4. The results for mi-SVM, MI-SVM and EM-DD are taken from [21].

Table 2.1 Results of all methods in the case of rbf kernel

Data Sets	MICA (%)	mi-SVM (%)	MI-SVM (%)	EM-DD (%)	SVM-CC (%)	MI-RMCLP (%)
Elephant	80.5	**82.2**	81.4	78.3	81.5	79.3
Fox	**58.7**	58.2	57.8	56.1	57.3	57.6
TST1	94.5	92.6	92.9	85.8	**95.0**	91.2
TST2	85.0	78.2	84.5	84.0	82.7	**86.0**
TST3	86.0	**87.0**	82.2	69.0	86.4	85.1
TST4	**87.7**	82.8	82.4	80.5	82.1	81.4
TST7	78.9	81.3	78.0	75.4	77.4	**82.7**
TST9	61.4	**67.5**	60.2	65.5	62.0	62.9
TST10	**82.3**	79.6	79.5	78.5	81.5	77.6
Musk-1	84.4	87.4	77.9	84.8	**88.9**	85.8
Musk-2	90.5	82.6	84.3	84.9	89.6	**91.7**

Note: Best accuracy is in bold

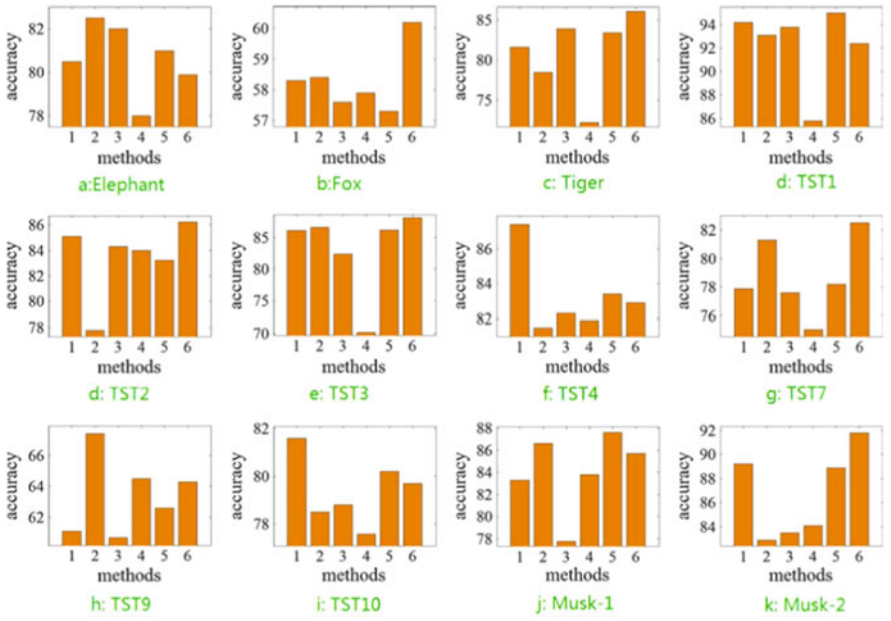


Fig. 2.4 Results of all methods in the case of linear kernel. X-axis represents different methods: 1: MICA; 2: mi-SVM; 3: MI-SVM; 4: EM-DD; 5: SVM-CC; 6: Mi-RMCLP. Y-axis represents the accuracy

2.1.3 Supportive Instances for Regularized Multiple Criteria Linear Programming Classification

Although RMCLP performs excellently in classifying lots of benchmark datasets, its shortage is also obvious. By taking account of every training instances into consideration, RMCLP is sensitive to noisy and imbalanced training samples. In

other words, the classification boundary may shift significantly even if there is merely a slight change of training samples. This difficulty can be described in Fig. 2.5, assume there is a two groups classification problem, the first group is denoted by "." and the second group is denoted by "☆" . We can observe that it is a linear-separable dataset and the classification boundary is denoted by a line "/". Figure 2.5a shows that on an ideal training sample, RMCLP successfully classify all the instances. In Fig. 2.5b, when we add some noisy instances into the first group, the classification boundary shifts towards the first group, making more instances in the first group misclassified. In Fig. 2.5c, we can observe that when we add instances into the second group to make the number of instances in two groups imbalanced, the classification boundary also changes significantly, causing a great number of misclassifications. In Fig. 2.5d, we can see that if we choose some representative instances (also called supportive instances) for RMCLP, which locate inside the blue circle, then although more noisy and imbalanced instances are added into the training sample, the classification boundary always keeps unchanged and will have a good ability to do prediction. That is to say, building RMCLP model only on supportive instances can improve its accuracy and stability.

According to the above observation, in this subsection, we propose a clustering-based sample selection method, which chooses the instances in the clustering center as the supportive samples (just as SVM [27] chooses the support vectors to draw a classification boundary). Experimental results on synthetic and real-life datasets show that our new method not only can significantly improve the prediction accuracy, but also can dramatically reduce the number of training instances.

Lots of empirical studies have shown that MCLP is a powerful tool for classification. However, there is no theoretical work on whether MCLP always can

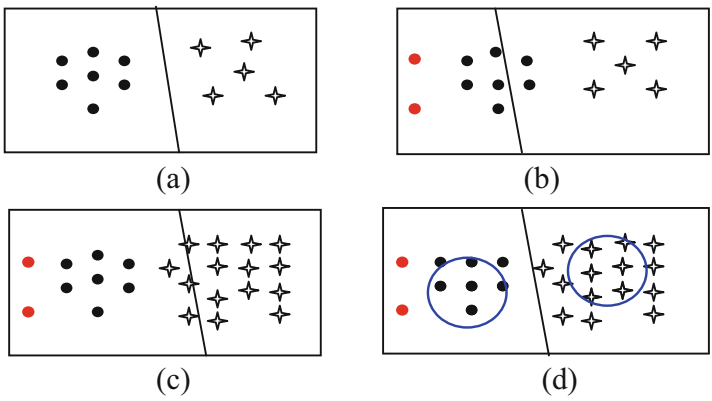


Fig. 2.5 (**a**) The original RMCLP model built on an ideal training sample; (**b**) when adding two noisy instances in the left side, the classification boundary shifts towards the left side; (**c**) when the training sample is imbalanced, the boundary also shifts significantly; (**d**) if we select representative training instances which locate around the distribution centers (inside the circle), the classification boundary becomes satisfactory

find an optimal solution under different kinds of training samples. To go over this difficulty, recently, [2] proposed a RMCLP model by adding two regularized items $\frac{1}{2}x^T Hx$ and $\frac{1}{2}\alpha^T Q\alpha$ on MCLP as follows:

$$\begin{aligned} \textbf{Minimize} & \frac{1}{2}x^T Hx + \frac{1}{2}\alpha^T Q\alpha + d^T\alpha - c^T\beta \\ \textbf{\textit{Subject to}}: & \begin{array}{c} A_i x - \alpha_i + \beta_i = b, \forall A_i \in G_1; \\ A_i x + \alpha_i - \beta_i = b, \forall A_i \in G_2; \\ \alpha_i, \beta_i \geq 0. \end{array} \end{aligned} \tag{2.68}$$

where $H \in R^{r*r}$, $Q \in R^{n*n}$ are symmetric positive definite matrices. d^T, $c^T \in R^n$. The RMCLP model is a convex quadratic program. Theoretically studies [2] have shown that RMCLP can always find a global optimal solution.

Besides two groups classification problem, a recent work [28] also introduced a multiple groups RMCLP model. As far as three groups classification problem be considered, we first find a projection direction x and a group of hyper planes (b_1, b_2), to an arbitrary training instance A_i, if $A_i x < b_1$, then $A_i \in G_1$; if $b_1 \leq A_i x < b_2$ then $A_i \in G_2$; and if $A_i x \geq b_2$, then $A_i \in G_3$. Extending this method to n group classification, we can also find a direction x and $n - 1$ dimension vector $b = [b_1, b_2, \ldots, b_{n-1}] \in R^{n-1}$, to make sure that to any training instance A_i:

$$\begin{array}{l} A_i x < b_1, \forall A_i \in G_1; \\ b_{j-1} \leq A_i x < b_j, \forall A_i \in G_i, 1 < i < n; \\ A_i x \geq b_{n-1}, \forall A_i \in G_n; \end{array} \tag{2.69}$$

We first define $c_i = \frac{b_{i-1}+b_i}{2}$ as the midline in group $i(1 < i < n)$. Then, to the misclassified records, we define α_i^+ as the distance from c_i to $A_i x$, which equals $(c_i - A_i x)$, when misclassify a group i's record into group j $(j < i)$, and we define α_i^- as the distance from $A_i x$ to c_i, which equals $(c_i - A_i x)$, when misclassify a group i's record into group j $(j > i)$. Similarly, to the correct classified records, we define β_i^- when A_i is in the left side of c_i, and we define β_i^+ when A_i is in the right side of c_i. When we have a n groups training sample with size m, we have $\alpha = \left\{\alpha_i^+, \alpha_i^-\right\} \in R^{m*2}$, $\beta = \left\{\beta_i^+, \beta_i^-\right\} \in R^{m*2}$, and we can build a multiple groups Regularized Multi-Criteria Linear Programming (SRMCLP) as follows:

$$\textbf{Minimize} \quad \frac{1}{2}x^T Hx + \frac{1}{2}\alpha^T Q\alpha + d^T\alpha + c^T\beta$$

$$\textbf{\textit{Subject to}}: \begin{array}{c} A_i x - \alpha_i^- - \beta_i^- + \beta_i^+ = \frac{1}{2}b_1, \forall A_i \in G_1; \\ A_i x - \alpha_i^- + \alpha_i^+ - \beta_i^- + \beta_i^+ = \frac{1}{2}\left(b_{i-1} + b_i\right), \forall A_i \in G_i, 1 < i < n; \\ A_i x + \alpha_i^+ - \beta_i^- + \beta_i^+ = 2b_{n-1}, \forall A_i \in G_n; \\ \alpha_i^-, \alpha_i^+, \beta_i^-, \beta_i^+ \geq 0. \end{array} \tag{2.70}$$

Input: training sample *Tr*, testing sample *Ts*, parameter ε, exclusion percentage *s*

Output: selected sample *Tr'*

Begin

1.Set *Tr'=Tr*

2. While (|PrevClusteringCenter-CurrClusteringCenter| < ε)

{

2.1. Calculate current clustering center; $cent = \frac{1}{|Tr'|}\sum_i x_i$

2.2. For each instances $i \in Tr'$ do

{

2.2.1 Calculate the Euclidean distance of the clustering center,

$$dis_i = \sqrt{\sum_{r \in r} \left| cent_r - Tr_r^i \right|^2}$$

2.2.2 get *s%* of the instances which are farthest from the clustering center, denoted as the subset *{P}*

2.2.3 exclude *{P}* from the training, *Tr'=Tr\{P}*.

}

}

3. Return the selected sample $T_{r'}$

End

Fig. 2.6 Clustering method to get the supportive method

Since this multiple groups RMCLP model is mainly designed to solve the ordinal separable dataset, we also call it Ordinal RMCLP model [28].

Figure 2.6 gives the whole procedure of the sample selection algorithm. The main idea of our algorithm is that it iteratively discards training instances in each group which are far away from the clustering center until the clustering center for each group is stable (with the given threshold ε), then the remained instances will be taken as the supportive instances (just as the support vectors to SVM) and used to build a classifier. From Fig. 2.6, We can observe that this algorithm is similar to the well-known k-means algorithm. However, the main difference between them is that, our algorithm is based on supervised learning framework, while k-means is an unsupervised learning algorithm. In our algorithm, although the clustering centers shift in each iteration, each instance keeps a constant class label. But in k-means, the class label of each instance may change frequently. An important issue of k-means clustering is how to choose the initial points, if we choose a good initial point, we can get a global optimal solution; otherwise, we may only get a local optimal solution. On the contrast, our sample selection method can avoid this problem. It always leads to a global minimal solution.

There are some important parameters in our algorithm. The first important parameter is ε, which determinates when the algorithm stops. The second parameter is the exclusion percentage s, which indicates how many instances that are far away from the clustering center should be discarded in each iteration. This parameter, in fact, determines the convergence speed. The larger value of s, the faster algorithm converges. To analyze the computation complexity of our new algorithm, we take an extremely bad situation into consideration. Assume there are n instances in the training sample, we assign the values $s = 1$ and $\varepsilon = 0$. Then, the algorithm will discard only one instances in each iteration. To the worst case, after n times iterations, the algorithm converges to the clustering center. In the i^{th} iteration, it needs to calculate the $(n - i)$ instances to get the clustering center, so we can roughly infer that the computation complexity is about $O(n^2)$.

To investigate whether our new algorithm works, we use two synthetic datasets and a well-known US bank's real-life credit card dataset for testing. In our experiments, the RMCLP is implemented by Visual Fortran 6.5.

The 6000 credit card records are randomly selected from 25,000 real-life credit card records of a major US bank. Each record has 113 variables, with 38 original variables and 65 derived variables. The 38 original variables are balance, purchase, payment, cash advance, and related variables, with the former 5 items each have six variables that represent raw data of six consecutive months and the last item includes interest charges, data of last payment, times of cash advance, account open data and so on. The 65 derived variables (CHAR01–CHAR65) are derived from original 38 variables using simple arithmetic methods to reinforce the comprehension of cardholders' behaviors. In this section, we use the derived 65 variables. We then define five classes for this dataset using a label variable: The Number of Over-limits. The five classes are defined as Bankrupt charge-off accounts (THE NUMBER OF OVER-LIMITS$\geq$13), Non-bankrupt charge-off accounts (7$\leq$THE NUMBER OF OVER-LIMITS$\leq$12), Delinquent accounts (3$\leq$THE NUMBER OF OVER-LIMITS$\leq$6), Current accounts (1$\leq$THE NUMBER OF OVER-LIMITS$\leq$2), and Outstanding accounts (no over limit). Bankrupt charge-off accounts are accounts that have been written off by credit card issuers due to reasons other than bankrupt claims. The charge-off policy may vary among authorized institutions. Delinquent accounts are accounts that haven't paid the minimum balances for more than 90 days. Current accounts are accounts that have paid the minimum balances. The outstanding accounts are accounts that have not balances. In our randomly selected 6000 records, there are 72 Bankrupt charge-off accounts, 205 Non-bankrupt charge-off accounts, 454 Delinquent accounts, 575 Current accounts and 4694 outstanding accounts.

Two groups credit card dataset To acquire a two groups training sample, we combine the Bankrupt charge-off accounts, Non-bankrupt charge-off accounts and Delinquent accounts together to form a "bad" group. And then we combine the current accounts and the outstanding accounts into a "good" group. According to the previous research work on this dataset, we first randomly select a benchmark training size of 700 bad records and 700 good records, and the remained 4600

Table 2.2 Comparison of different percentage of training instances

Percent. of training	Training sample		Testing sample (4600 instances)	
	Right instances	Accuracy (%)	Right instances	Accuracy (%)
100 (1400)	1096	78.29	3394	72.78
90 (1260)	998	79.20	3295	71.63
80 (1120)	912	81.43	3292	71.57
70 (980)	789	80.51	3571	77.63
60 (840)	667	79.40	3761	81.76
50 (700)	559	79.86	3881	84.37
40 (560)	449	80.18	3964	86.17
30 (420)	331	78.81	4050	88.04
20 (280)	232	82.86	4073	88.54
10 (140)	116	82.86	1971	42.85

records are combined to test the performance. Now what we need to do is to examine three assumptions: first, is the randomly selected 1400 points are suitable to build model? second, are there any noisy instances in this randomly selected dataset? third, can we reduce the 1400 points in a much smaller size and improve the accuracy synchronously? Experimental results in Table 2.2 tell us the answers. The first column of Table 2.2 is the current training sample's size, from the 1400 instances to 140 instances, the second and the third columns list the performance on different training samples and the fourth and the fifth columns exhibit the performance on the same 4600 testing instances. The experiment is conducted as follows: firstly, we build a RMCLP model on all the 1400 training instances, and we get a benchmark accuracy as 72.78%. Then we call our sample selection algorithm with parameter $s = 1$ and $\varepsilon = 0.1$. We do experiments on night special datasets, 10%, 20%, ..., 90% of the original 1400 training sample. We finally list the performance of RMCLP in Table 2.2. Intuitionally, we though the larger the training sample, the more information we could get, and thus the model would be more accurate when do prediction. However, Table 2.2, we can see that the 1400 randomly selected instances is not the best training set for RMCLP model, there exist noisy and useless instances which deteriorate its performance. Our new sample selection method reduces the training samples continuously. When get 20% of the original training sample (that is 280 instances), we can build a RMCLP with the highest accuracy of 88.54% on the testing set.

Multiple Groups credit card dataset Besides two groups RMCLP model, in this part, we also study the performance of our new algorithm on multiple groups RMCLP model. For three groups classification, we choose the Bankrupt charge-off accounts as the first group, the Non-bankrupt charge-off as the second group and the Delinquent as the third group. Based upon the three groups dataset, we construct the four groups dataset by adding the Current account as the fourth group. At last, we construct a five groups dataset by adding the Outstanding accounts as the fifth group.

Table 2.3 Comparison on three groups credit card dataset

3 Groups (22 + 155 + 404)	Original RMCLP		RMCLP After sample selection	
	Corrected Rec.	Accuracy (%)	Corrected Rec.	Accuracy (%)
Group1	12	54.5	19	86.36
Group2	12	7.7	89	57.42
Group3	402	99.5	390	96.53
Average	426	72.32	481	85.71

Table 2.4 Comparison on four groups credit card dataset

4 Groups (22 + 155 + 404 + 525)	Original RMCLP		RMCLP After sample selection	
	Corrected Rec.	Accuracy (%)	Corrected Rec.	Accuracy (%)
Group1	16	72.7	19	86.36
Group2	52	32.5	122	78.71
Group3	38	9.4	267	66.09
Group4	525	100.0	510	97.14
Average	631	57.05	918	82.00

Table 2.5 Comparison on five groups credit card dataset

5 Groups (22 + 155 + 404 + 525 + 4644)	Original RMCLP		RMCLP After sample selection	
	Corrected Rec.	Accuracy (%)	Corrected Rec.	Accuracy (%)
Group1	13	59.1	16	72.73
Group2	130	82.9	130	82.87
Group3	273	67.6	365	90.35
Group4	161	30.7	438	82.43
Group5	4644	100.0	4520	97.33
Average	5221	90.80	5469	95.11

Tables 2.3, 2.4 and 2.5 list the results of comparisons. The second and the third columns list the results of the original RMCLP method, the fourth and the fifth columns list the results of RMCLP after selecting the supportive instances. We can observe that in three groups classification, the original RMCLP's average accuracy is 72.32%, while that of the supportive instances is 85.71%. The improvement of accuracy is as large as 12.39%. In four groups classification, the average accuracy of the original RMCLP is 57.05%, on the contrast, after selecting the supportive instances, the accuracy improves to 82.00%, as high as 25.95% improvement. To the five groups classification, the improvement after selecting supportive instances is 4.31%. From these compressive results, we can validate our former conclusion that selecting supportive instances for RMCLP can significantly improve its accuracy.

2.1.4 *Kernel Based Simple Regularized Multiple Criteria Linear Programming for Binary Classification and Regression*

In this section, a novel kernel based regularized multiple criteria linear program are proposed for both classification and regression scenarios.

Given an observed dataset $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$ with l instances. Each instance x_i belongs to the category y_i. $x_i \in \chi \subseteq R^n$ and $y_i \in y$ are the n attributes values and corresponding label for the instance i. The goal of classification problem is to predict the corresponding label $y_i \in y$ when new instance $x_j \in \chi$ arrives. When $Card(y) = 2$, the issue is binary classification problem. In order to facilitate description, here we let $y = \{-1, 1\}$ for following introduction. Under this binary classification problem, supposed we have positive instances number is l1, negative instances number is l_2, where $l_1 + l_2 = l$. $\xi_A = 0$, $\xi_B = 0$ which are not marked in the picture.

In contrast to points A and B, points C and D are improperly predicted. Hence their distance could be constructed as $\beta_C = 0$, $\beta_D = 0$ and $\xi_C > 0$, $\xi_D > 0$. In summary, following the idea described above the basic MCLP model [29] for classification could be written as this:

$$\begin{aligned} &\min_{w,b,\xi,\beta} \sum_{i=1}^{l} \xi_i \\ &\max_{w,b,\xi,\beta} \sum_{i=1}^{l} \beta_i \\ s.t. &\, y_i \left(x_i^T w + b \right) = \beta_i - \xi_i, \\ &\xi_i \geq 0, \beta_i \geq 0, i = 1, \cdots, l; \end{aligned} \tag{2.71}$$

Here w and b could be seem as the slope and intercept of the discriminant hyperplane. One of the objectives $\sum \xi_i$ could be considered as the measure of misclassification, thus we minimized it to avoid the inappropriate model construction.

And the other goal $\sum \beta_i$ is to maximize the generalization capability of the chosen classification function. As we introduced before, there exist no single solution that could make the both these two goals in conflict optimal at the same time. In [30, 31], compromise solution is introduced and analyzed for this multiple objective model Eq. (2.71). However, the algorithm that obtained compromise solution were usually time consuming and not suitable for real world application.

As a result, many methods convert model Eq. (2.71) into single objective linear program:

$$\begin{aligned} &\min_{w,b,\xi,\beta} \sum_{i=1}^{l} \xi_i - \gamma \sum_{i=1}^{l} \beta_i \\ &s.t. y_i \left(x_i^T w + b \right) = \beta_i - \xi_i, \\ &\xi_i \geq 0, \beta_i \geq 0, i = 1, \cdots, l; \end{aligned} \tag{2.72}$$

Unfortunately, naive model Eq. (2.72) confronts the unsolvable defect because of the nature of linear programming. More sophisticated approaches need to be investigated. Therefore, an improved model would be illustrated in next section.

Although model Eq. (2.72) avoided the computational cost of multiple objectives, it had a fatal solvability problem. Therefore, we added new quadratic term to the objective function and proposed a new simple regularized MCLP model showed as below:

$$\begin{aligned} &\min_{w,b,\xi,\beta} \sum_{i=1}^{l} \xi_i - \gamma \sum_{i=1}^{l} \beta_i + \frac{1}{2} \tau \beta^T H \beta \\ &s.t. y_i \left(x_i^T w + b \right) = \beta_i - \xi_i, \\ &\xi_i \geq 0, \beta_i \geq 0, i = 1, \cdots, l; \\ &b \in \{-1, 1\}. \end{aligned} \tag{2.73}$$

Furthermore, users want to guarantee the slope of the hyperplane not too large. Then, we made the regularization term $w^T K w$ as a part of the goal and obtained the following model:

$$\begin{aligned} &\min_{w,b,\xi,\beta} \sum_{i=1}^{l} \xi_i - \gamma \sum_{i=1}^{l} \beta_i + \frac{1}{2} \tau \beta^T H \beta + \frac{1}{2} \kappa w^T K w \\ &s.t. y_i \left(x_i^T w + b \right) = \beta_i - \xi_i, \\ &\xi_i \geq 0, \beta_i \geq 0, i = 1, \cdots, l; \\ &b \in \{-1, 1\}; \end{aligned} \tag{2.74}$$

In order to write the formulas in matrix form, we let

$$A = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_l^T \end{bmatrix}_{l*n}, Y = \begin{bmatrix} y_1 & 0 & \cdots & 0 \\ 0 & y_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & y_l \end{bmatrix}_{l*l} \tag{2.75}$$

So model Eq. (2.74) could be rewritten as this:

$$\begin{gathered} \min_{w,\beta,\xi} \frac{1}{2} w^T H w + \frac{1}{2} \lambda_1 \beta^T K \beta - \lambda_2 e^T \beta + \lambda_3 e^T \xi \\ s.t. Y\left(Aw + be\right) - \beta + \xi = 0, \\ b \in \{-1, 1\}, \beta \geq 0, \xi \geq 0 \end{gathered} \tag{2.76}$$

Where $w \in R^n$, $\beta \in R^l$, $\xi \in R^l$, $e = [1, \cdots, 1]_l^T$ is the vector of all ones. K and H are $n \times n$ and $l \times l$ positive matrix, respectively. We simply set positive matrix H, K in model Eq. (2.76) as identity matrix. And to solve the problem with inequality type constraints, we have to find the saddle point of the Lagrangian function for model Eq. (2.76)

$$\begin{aligned} L\left(w, \beta, \xi, \alpha_{equ}, \alpha_\beta, \alpha_\xi\right) = & \left(\frac{1}{2} w^T w + \frac{1}{2} \lambda_1 \beta^T \beta - \lambda_2 e^T \beta + \lambda_3 e^T \xi\right) \\ & + \alpha_{equ}^T \left(Y\left(Aw + be\right) - \beta + \xi\right) - \alpha_\beta^T \beta - \alpha_\xi^T \xi \end{aligned} \tag{2.77}$$

where α_{equ} is free, $\alpha_\beta \geq 0$, $\alpha_\xi \geq 0$ are Lagrangian multipliers. Minimization with respect to w, β, ξ implies the following

$$\nabla_w L\left(w, \beta, \xi, \alpha_{equ}, \alpha_\beta, \alpha_\xi\right) = w + A^T Y \alpha_{equ} = 0 \tag{2.78}$$

$$\nabla_\beta L\left(w, \beta, \xi, \alpha_{equ}, \alpha_\beta, \alpha_\xi\right) = \lambda_1 \beta - \lambda_2 e - \alpha_{equ} - \alpha_\beta = 0 \tag{2.79}$$

$$\nabla_\xi L\left(w, \beta, \xi, \alpha_{equ}, \alpha_\beta, \alpha_\xi\right) = \lambda_3 e + \alpha_{equ} - \alpha_\xi = 0 \tag{2.80}$$

Sustaining Eq. (2.78) into function Eq. (2.77), we get

$$L\left(w, \beta, \xi, \alpha_{equ}, \alpha_\beta, \alpha_\xi\right) = -\frac{1}{2} \alpha_{equ}^T Y A A^T Y \alpha_{equ} - \frac{1}{2} \lambda_1 \beta^T \beta + b e^T Y \alpha_{equ}$$

Therefore, the dual problem for model Eq. (2.76) is obtained as

$$\begin{gathered}\max -\frac{1}{2}\alpha_{equ}^{T} Y A A^{T} Y \alpha_{equ}-\frac{1}{2}\lambda_{1}\beta^{T}\beta+be^{T} Y \alpha_{equ} \\ s.t. \lambda_{1}\beta-\lambda_{2}e-\alpha_{equ}\geq 0, \\ \lambda_{3}e+\alpha_{equ}\geq 0, \\ \beta\geq 0, \\ b\in\{-1,1\}\end{gathered} \tag{2.81}$$

According to the Eq. (2.78), the decision function is

$$f(x)=\operatorname{sign}(w\cdot x+b)=\operatorname{sign}\left(-YA^{T}\alpha_{equ}x+b\right).$$

When introduce kernel functions

$$\begin{gathered}R^{n}\rightarrow H \\ x\rightarrow \Phi(x)\end{gathered} \tag{2.82}$$

We have $K(x_i, x_j)=\Phi(x_i)\cdot\Phi(x_j)$. Therefore, the dual problem Eq. (2.81) could be rewritten as

$$\begin{gathered}\min \frac{1}{2}\alpha_{equ}^{T} Y K(A, A) Y \alpha_{equ}+\frac{1}{2}\lambda_{1}\beta^{T}\beta-be^{T} Y \alpha_{equ} \\ s.t. \lambda_{1}\beta-\lambda_{2}e-\alpha_{equ}\geq 0, \\ \lambda_{3}e+\alpha_{equ}\geq 0, \\ \beta\geq 0, \\ b\in\{-1,1\}\end{gathered} \tag{2.83}$$

Furthermore, the decision boundary turns into

$$f(x)=\operatorname{sign}(w\cdot\Phi(x)+b)=\operatorname{sign}\left(-YK(A, x)\alpha_{equ}+b\right).$$

Theorem 2.2 *Given the solution of the dual problem Eq. (2.83) as* $\left(\alpha_{equ}^{*}, \beta^{*}\right)$, *the solution of its corresponding primal problem w.r.t. H space can be obtained as*

below:

$$w^* = -Y\Phi(A)^T\alpha^*_{equ} \tag{2.84}$$

Proof From dual problem Eq. (2.83), we can get its Lagrangian function as:

$$\begin{aligned} L\left(\alpha_{equ}, \beta, \alpha_1, \alpha_2\right) &= \frac{1}{2}\alpha^T_{equ} YK\left(A, A\right) Y\alpha_{equ} + \frac{1}{2}\lambda_1\beta^T\beta - be^T Y\alpha_{equ} \\ &-\alpha_1^T\left(\lambda_1\beta - \lambda_2 e - \alpha_{equ}\right) - \alpha_2^T\left(\lambda_3 e + \alpha_{equ}\right) - \alpha_3^T\beta \end{aligned} \tag{2.85}$$

Where $\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$. From the KTT condition, we have the equations below:

$$\lambda_1\beta - \lambda_2 e - \alpha_{equ} \geq 0 \tag{2.86}$$

$$\lambda_3 e + \alpha_{equ} \geq 0 \tag{2.87}$$

$$\beta \geq 0 \tag{2.88}$$

$$\left(\lambda_1\beta - \lambda_2 e - \alpha_{equ}\right)^T\alpha_1 = 0 \tag{2.89}$$

$$\left(\lambda_3 e + \alpha_{equ}\right)^T\alpha_2 = 0 \tag{2.90}$$

$$\beta^T\alpha_3 = 0 \tag{2.91}$$

$$\nabla_{\alpha_{equ}} L\left(\alpha_{equ}, \beta, \alpha_1, \alpha_2\right) = YK\left(A, A\right) Y\alpha_{equ} - bYe + \alpha_1 - \alpha_2 = 0 \tag{2.92}$$

$$\nabla_\beta L\left(\alpha_{equ}, \beta, \alpha_1, \alpha_2\right) = \lambda_1\beta - \lambda_1\alpha_1 - \alpha_3 = 0 \tag{2.93}$$

Sustaining Eq. (2.84) into Eq. (2.92), so

$$\begin{aligned} \nabla_{\alpha_{equ}} L\left(\alpha_{equ}, \beta, \alpha_1, \alpha_2\right) &= YK\left(A, A\right) Y\alpha_{equ} - bYe + \alpha_1 - \alpha_2 \\ &= -Y\left(w^* \cdot \Phi(A) + be\right) + \alpha_1^* - \alpha_2^* = 0 \end{aligned} \tag{2.94}$$

This satisfies the constraint of problem Eq. (2.76) when $\beta = \alpha_1^*$, $\xi = \alpha_2^*$. Therefore, $\left(w^*, \alpha_1^*, \alpha_2^*\right)$ is the feasible solution of primal problem Eq. (2.76) w.r.t. H space. Furthermore, introducing Eqs. (2.89), (2.90) and (2.92), the objective function of primal problem Eq. (2.76) turns into:

$$\begin{aligned} &\frac{1}{2}w^{*T}w^* + \frac{1}{2}\lambda_1\beta^{*T}\beta^* - \lambda_2 e^T\beta^* + \lambda_3 e^T\xi^* \\ &= -\frac{1}{2}\alpha^{*T}_{equ} YK\left(A, A\right) Y\alpha^*_{equ} - \frac{1}{2}\lambda_1\beta^{*T}\beta^* + be^T Y\alpha^*_{equ} \end{aligned} \tag{2.95}$$

As a result, the object value of the primal problem at points (w^*, β^*, ξ^*) is the optimal value of its dual problem at points (α_{equ}, β^*) w.r.t. H space.

Base on the Theorem 2.2, we introduced Algorithm 2.3 using kernel based simple regular multiple constraint linear program (KSRMCLP) for binary classification problem.

Given a training set $\{(x_1, y_1), \cdots, (x_l, y_l)\}$, being different from classification problem, regression is not to give a new arrival instance x_i a category label but a real number value, $y_i \in R$. That is mean the possible set of y_i has been changed from finite labels set y to infinite R. Following the idea of $\epsilon - tube$, a model for regression problem could be constructed from a binary classification model [32]. Given a real number ϵ, two different category points could be generated when we add and minus ϵ on the regression output y_i. When we have l instances $\{(x_1, y_1), \cdots, (x_l, y_l)\}$ for regression, $2 \times l$ instances $\{(x_1, y_1 + \epsilon)_{pos}, \cdots, (x_l, y_l + \epsilon)_{pos}, (x_1, y_1 - \epsilon)_{neg}, \cdots, (x_\ell, y_\ell - \epsilon)_{neg}\}$ could be constructed. According to the binary classification model we propose in the last section, a model for regression problem could be given as:

$\min \frac{1}{2} w^T H w + \frac{1}{2}\lambda_1 \beta^T K \beta - \lambda_2 e^T \beta + \lambda_3 e^T \xi$

$$s.t. Y\left(A_{reg} w + be\right) = \beta - \xi, \tag{2.96}$$

$$\beta \geq 0, \xi \geq 0$$

Algorithm 2.3 KSRMCLP Algorithm for Binary Classification

Input:

Training dataset $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_l, y_l)\}$ with l instances, $x_i \in R^n$ and$y_i \in \{-1, 1\}$, kernel function $K_\theta(x_i, x_j)$ and its parameters θ, model parameters $\lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0$.

Output:

Binary classification discriminate function $f(x)$.

1: **Begin**

2: Construct data matrix A, label matrix Y according to Eq. (2.75).

$$A = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_l^T \end{bmatrix}_{l*n}, Y = \begin{bmatrix} y_1 & 0 & \cdots & 0 \\ 0 & y_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & y_l \end{bmatrix}_{l*l}$$

3: Construct and solve the optimization problem according to model Eq. (2.83).

$\min \frac{1}{2}\alpha_{equ}^T Y K_\theta (A, A) Y \alpha_{equ} + \frac{1}{2}\lambda_1 \beta^T \beta - be^T Y \alpha_{equ}$,

$s.t. \lambda_1 \beta - \lambda_2 e - \alpha_{equ} \geq 0$,

$\lambda_3 e + \alpha_{equ} \geq 0$,

$\beta \geq 0$,

$$b \in \{-1, 1\}$$

4: Obtain the decision function $f(x) = sign\ (-YK_\theta(A, x)\alpha_{equ} + b)$.
5: **End**

where $w \in R^{n+1}$, $\beta, \xi \in R^{2l}$, and

$$A_{reg} = \begin{bmatrix} x_1^T, y_1 + \epsilon \\ \vdots \\ x_l^T, y_l + \epsilon \\ x_1^T, y_1 - \epsilon \\ \vdots \\ x_l^T, y_l - \epsilon \end{bmatrix}_{2l \times (n+1)}, Y = \begin{bmatrix} I_{l \times l} & O \\ O & -I_{l \times l} \end{bmatrix}_{2l \times 2l} \tag{2.97}$$

The constraint of Eq. (2.96) could be divided into two parts, the positive and the negative. For positive points, the corresponding target value is $y_i + \epsilon$, for negative points is $y_i - \epsilon$. Thus matrix Y is useless and variables β, ξ, also change into Boos, $\beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg}$. Then, model Eq. (2.96) could be written as,

$\min \frac{1}{2} w^T H w + \frac{1}{2}\lambda_1 \beta_{pos}^T K \beta_{pos} + \frac{1}{2}\lambda_1 \beta_{neg}^T K \beta_{neg} - \lambda_2 e^T \left(\beta_{pos} + \beta_{neg}\right) + \lambda_3 e^T \left(\xi_{pos} + \xi_{neg}\right)$

$$\begin{aligned} s.t. Aw + be + \eta\,(y + \epsilon e) &= \beta_{pos} - \xi_{pos} \\ Aw + be + \eta\,(y - \epsilon e) &= -\left(\beta_{neg} - \xi_{neg}\right) \\ \beta_{pos} \geq 0, \beta_{neg} &\geq 0, \xi_{pos} \geq 0, \xi_{neg} \geq 0 \end{aligned} \tag{2.98}$$

where $w \in R^n$, $\beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg} \in R^l$, $b \in R$ are variables.

$$A = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_l^T \end{bmatrix}_{l \times n}, y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_l \end{bmatrix}_{l \times 1} \tag{2.99}$$

We know $\eta \neq 0$, w, b, β_{pos}, β_{neg}, ξ_{pos}, ξ_{neg} are all variables, so η could be removed from the expression. Model Eq. (2.98) turns into:

$\min \frac{1}{2} w^T H w + \frac{1}{2}\lambda_1 \beta_{pos}^T K \beta_{pos} + \frac{1}{2}\lambda_1 \beta_{neg}^T K \beta_{neg} - \lambda_2 e^T \left(\beta_{pos} + \beta_{neg}\right) + \lambda_3 e^T \left(\xi_{pos} + \xi_{neg}\right)$

$$\begin{aligned} s.t. Aw + be + (y + \epsilon e) &= \beta_{pos} - \xi_{pos} \\ Aw + be + (y - \epsilon e) &= -\left(\beta_{neg} - \xi_{neg}\right), \\ \beta_{pos} \geq 0, \beta_{neg} &\geq 0, \xi_{pos} \geq 0, \xi_{neg} \geq 0 \end{aligned} \tag{2.100}$$

where $w \in R^n$, $\beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg} \in R^l$, $b \in R$ are variables. And $\epsilon, \lambda_1, \lambda_2, \lambda_3 \in R$, positive matrices H, K are given in advance. Similar to the procedure last part, we set K, H as identity matrix, the Lagrangian function of model Eq. (2.100) is derived as

$$L\left(w, \beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg}\right) = -\frac{1}{2}\left(\alpha_{pos} + \alpha_{neg}\right)^T AA^T \left(\alpha_{pos} + \alpha_{neg}\right) - \frac{1}{2}\lambda_1 \beta_{pos}^T \beta_{pos}$$
$$-\frac{1}{2}\lambda_1 \beta_{neg}^T \beta_{neg} + (be + y)^T \left(\alpha_{pos} + \alpha_{neg}\right) + \epsilon e^T \left(\alpha_{pos} - \alpha_{neg}\right) \tag{2.101}$$

where α_{pos}, α_{neg} are free variables, $\alpha_{\beta_{pos}} \geq 0, \alpha_{\beta_{neg}} \geq 0, \alpha_{\xi_{pos}} \geq 0, \alpha_{\xi_{neg}} \geq 0$ are corresponding Lagrangian multipliers. Also, from KKT condition, we have

$$\nabla_w L\left(w, \beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg}\right) = w + A^T \left(\alpha_{pos} + \alpha_{neg}\right) = 0 \tag{2.102}$$

$$\nabla_{\beta_{pos}} L\left(w, \beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg}\right) = \lambda_1 \beta_{pos} - \lambda_2 e - \alpha_{pos} - \alpha_{\beta_{pos}} = 0 \tag{2.103}$$

$$\nabla_{\beta_{neg}} L\left(w, \beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg}\right) = \lambda_1 \beta_{neg} - \lambda_2 e + \alpha_{neg} - \alpha_{\beta_{neg}} = 0 \tag{2.104}$$

$$\nabla_{\xi_{pos}} L\left(w, \beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg}\right) = \lambda_3 e + \alpha_{pos} - \alpha_{\xi_{pos}} = 0 \tag{2.105}$$

$$\nabla_{\xi_{neg}} L\left(w, \beta_{pos}, \beta_{neg}, \xi_{pos}, \xi_{neg}\right) = \lambda_3 e - \alpha_{neg} - \alpha_{\xi_{neg}} = 0 \tag{2.106}$$

Therefore, the dual problem for model Eq. (2.100) is obtained:

$$\begin{gathered}
\max -\frac{1}{2}\left(\alpha_{pos} + \alpha_{neg}\right)^T AA^T \left(\alpha_{pos} + \alpha_{neg}\right) - \frac{1}{2}\lambda_1 \left(\beta_{pos}^T \beta_{pos} + \beta_{neg}^T \beta_{neg}\right) + \\
(be + y)^T \left(\alpha_{pos} + \alpha_{neg}\right) + \epsilon e^T \left(\alpha_{pos} - \alpha_{neg}\right) \\
s.t. \lambda_1 \beta_{pos} - \lambda_2 e - \alpha_{pos} \geq 0, \\
\lambda_1 \beta_{neg} - \lambda_2 e + \alpha_{neg} \geq 0, \\
\lambda_3 e + \alpha_{pos} \geq 0, \\
\lambda_3 e - \alpha_{neg} \geq 0, \\
\beta_{pos} \geq 0, \\
\beta_{neg} \geq 0, \\
b \in \{-1, 1\}
\end{gathered} \tag{2.107}$$

where $\alpha_{pos}, \alpha_{neg}, \beta_{pos}, \beta_{neg} \in R^l$, $b \in R$ are variables. And $\epsilon \geq 0$, $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, $\lambda_3 \geq 0$ are given in advance.

When introducing kernel function Eq. (2.82), model Eq. (2.107) turns into

$$\begin{aligned} \min \tfrac{1}{2}\left(\alpha_{pos} + \alpha_{neg}\right)^T K\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) + \tfrac{1}{2}\lambda_1\left(\beta_{pos}^T\beta_{pos} + \beta_{neg}^T\beta_{neg}\right) \\ -(be + y)^T\left(\alpha_{pos} + \alpha_{neg}\right) - \epsilon e^T\left(\alpha_{pos} - \alpha_{neg}\right) \\ s.t. \lambda_1\beta_{pos} - \lambda_2 e - \alpha_{pos} \geq 0, \\ \lambda_3 e + \alpha_{pos} \geq 0, \\ \lambda_3 e - \alpha_{neg} \geq 0, \\ \beta_{pos} \geq 0, \\ \beta_{neg} \geq 0, \\ b \in \{-1, 1\} \end{aligned} \tag{2.108}$$

From the decision hyperplane $w \cdot x + b + y = 0$, the regression function could be obtained as

$$f(x) = -\left(w \cdot x + b\right) = A^T\left(\alpha_{pos} + \alpha_{neg}\right) \cdot x - b$$

With kernel function, regression function could be derived from

$$f(x) = \Phi(A)^T\left(\alpha_{pos} + \alpha_{neg}\right)\Phi(x) - b = K\left(A, x\right)\left(\alpha_{pos} + \alpha_{neg}\right) - b$$

Theorem 2.3 *Given the solution of Dual Problem Eq. (2.108)* $\left(\alpha_{pos}^*, \alpha_{neg}^*, \beta_{pos}^*, \beta_{neg}^*\right)$*, the solution of its corresponding primal problem w.r.t. H space can be obtained as below:*

$$w^* = -\Phi\left(A^T\right)\left(\alpha_{pos}^* + \alpha_{neg}^*\right) \tag{2.109}$$

Proof From dual problem Eq. (2.108), we can get its Lagrangian function as:

$$\begin{aligned} L\left(\alpha_{pos}, \alpha_{neg}, \beta_{pos}, \beta_{neg}\right) = \frac{1}{2}\left(\alpha_{pos} + \alpha_{neg}\right)^T K\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) \\ +\frac{1}{2}\lambda_1\left(\beta_{pos}^T\beta_{pos} + \beta_{neg}^T\beta_{neg}\right) \\ -(be + y)^T\left(\alpha_{pos} + \alpha_{neg}\right) - \epsilon e^T\left(\alpha_{pos} - \alpha_{neg}\right) \\ -\alpha_1^T\left(\lambda_1\beta_{pos} - \lambda_2 e - \alpha_{pos}\right) \end{aligned} \tag{2.110}$$

$$-\alpha_2^T \left(\lambda_1 \beta_{neg} - \lambda_2 e + \alpha_{neg}\right)$$

$$-\alpha_3^T \left(\lambda_3 e + \alpha_{pos}\right)$$

$$-\alpha_4^T \left(\lambda_3 e - \alpha_{neg}\right)$$

$$-\alpha_5^T \beta_{pos}$$

$$-\alpha_6^T \beta_{neg}$$

where $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, $\alpha_3 \geq 0$, $\alpha_4 \geq 0$, $\alpha_5 \geq 0$, $\alpha_6 \geq 0$. from the KTT condition, we have the equation below:

$$\lambda_1 \beta_{pos} - \lambda_2 e - \alpha_{pos} \geq 0 \tag{2.111}$$

$$\lambda_1 \beta_{neg} - \lambda_2 e + \alpha_{neg} \geq 0 \tag{2.112}$$

$$\lambda_3 e + \alpha_{pos} \geq 0 \tag{2.113}$$

$$\lambda_3 e - \alpha_{neg} \geq 0 \tag{2.114}$$

$$\beta_{pos} \geq 0 \tag{2.115}$$

$$\beta_{neg} \geq 0 \tag{2.116}$$

$$\alpha_1^T \left(\lambda_1 \beta_{pos} - \lambda_2 e - \alpha_{pos}\right) = 0 \tag{2.117}$$

$$\alpha_2^T \left(\lambda_1 \beta_{neg} - \lambda_2 e + \alpha_{neg}\right) = 0 \tag{2.118}$$

$$\alpha_3^T \left(\lambda_3 e + \alpha_{pos}\right) = 0 \tag{2.119}$$

$$\alpha_4^T \left(\lambda_3 e - \alpha_{neg}\right) = 0 \tag{2.120}$$

$$\alpha_5^T \beta_{pos} = 0 \tag{2.121}$$

$$\alpha_6^T \beta_{neg} = 0 \tag{2.122}$$

$$\nabla_{\alpha_{pos}} L = K\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) - (be + y) - \epsilon e + \alpha_1 - \alpha_3 = 0 \tag{2.123}$$

$$\nabla_{\alpha_{neg}} L = K\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) - (be + y) + \epsilon e - \alpha_2 + \alpha_4 = 0 \tag{2.124}$$

$$\nabla_{\beta_{pos}} L = \lambda_1 \beta_{pos} - \lambda_1 \alpha_1 - \alpha_5 = 0 \tag{2.125}$$

$$\nabla_{\beta_{neg}} L = \lambda_1 \beta_{neg} - \lambda_1 \alpha_2 - \alpha_6 = 0 \tag{2.126}$$

Sustaining Eq. (2.109) into Eqs. (2.123) and (2.124), we have

$$\begin{aligned}
&\nabla_{\alpha_{pos}} L\left(\alpha_{pos}, \alpha_{neg}, \beta_{pos}, \beta_{neg}\right) \\
&= K\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) - (be + y) - \epsilon e + \alpha_1 - \alpha_3 \qquad (2.127) \\
&= -\left(w^* \cdot \Phi(A) + be + (y + \epsilon e) - \alpha_1 + \alpha_3\right) = 0 \\
&\nabla_{\alpha_{pos}} L\left(\alpha_{pos}, \alpha_{neg}, \beta_{pos}, \beta_{neg}\right) \\
&= K\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) - (be + y) + \epsilon e - \alpha_1 + \alpha_3 \qquad (2.128) \\
&= -\left(w^* \cdot \Phi(A) + be + (y - \epsilon e) + \alpha_1 - \alpha_3\right) = 0
\end{aligned}$$

This satisfies the constraint of primal problem Eq. (2.100), so $\left(w^*, \alpha_1^*, \alpha_2^*, \alpha_3^*, \alpha_4^*\right)$ is the feasible solution of primal problem Eq. (2.100) w.r.t. H space. Furthermore, introducing Eqs. (2.117)–(2.122), the objective function of primal problem Eq. (2.100) turns into:

$$\begin{aligned}
&\frac{1}{2}w^T w + \frac{1}{2}\lambda_1\left(\beta_{pos}^T \beta_{pos} + \beta_{neg}^T \beta_{neg}\right) - \lambda_2 e^T\left(\beta_{pos} + \beta_{neg}\right) + \lambda_3 e^T\left(\xi_{pos} + \xi_{neg}\right) \\
&= -\frac{1}{2}\left(\alpha_{pos} + \alpha_{neg}\right)^T K\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) - \frac{1}{2}\lambda_1\left(\beta_{pos}^T \beta_{pos} + \beta_{neg}^T \beta_{neg}\right) \\
&\quad + (be + y)^T\left(\alpha_{pos} + \alpha_{neg}\right) + \epsilon e^T\left(\alpha_{pos} - \alpha_{neg}\right)
\end{aligned}$$

As a result, the object value of the primal problem at points $\left(w^*, \beta_{pos}^*, \beta_{neg}^*, \xi_{pos}^*, \xi_{neg}^*\right)$ is the optimal value of its dual problem at points $\left(\alpha_{pos}^*, \alpha_{neg}^*, \beta_{pos}^*, \beta_{neq}^*\right)$.

Base on the Theorem 2.3, we introduced Algorithm 2.4 from kernel based simple regular multiple constraint linear programming (KSRMCLP) for regression problem.

Algorithm 2.4 KSRMCLP Algorithm for Regression
Input:

Training dataset $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_l, y_l)\}$, $x_i \in R^n$ and $y_i \in R$. Kernel function $K_\theta(x_i, x_j)$ and its parameters θ, model parameters $\epsilon \geq 0$, $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, $\lambda_3 \geq 0$.

Output:

Regression estimated function $f(x)$.

1: **Begin**

2: Construct data matrix A, target value vector y according to equation formula

below:

$$A = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_l^T \end{bmatrix}_{l*n}, \; y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_l \end{bmatrix}_{l\times 1}$$

3: Construct and solve the optimization problem according to Eq. (2.108).

$$\min \tfrac{1}{2}\left(\alpha_{pos} + \alpha_{neg}\right)^T K_\theta\left(A, A\right)\left(\alpha_{pos} + \alpha_{neg}\right) + \tfrac{1}{2}\lambda_1\left(\beta_{pos}^T\beta_{pos} + \beta_{neg}^T\beta_{neg}\right)$$
$$-(be + y)^T(\alpha_{pos} + \alpha_{neg}) - \epsilon e^T(\alpha_{pos} - \alpha_{neg}),$$
$$s.\,t.\ \lambda_1\beta_{pos} - \lambda_2 e - \alpha_{pos} \geq 0,$$
$$\lambda_1\beta_{neg} - \lambda_2 e + \alpha_{neg} \geq 0,$$
$$\lambda_3 e + \alpha_{pos} \geq 0,$$
$$\lambda_3 e - \alpha_{neg} \geq 0,$$
$$\beta_{pos} \geq 0,$$
$$\beta_{neg} \geq 0,$$
$$b \in \{-1, 1\}$$

4: Obtain the decision function $f(x) = K_\theta(A, x)(\alpha_{pos} + \alpha_{neg}) - b$.

5: **End**

2.2 Multiple Criteria Linear Programming with Expert and Rule Based Knowledge

2.2.1 A Group of Knowledge-Incorporated Multiple Criteria Linear Programming Classifier

Prior knowledge in some classifiers usually consists of a set of rules, such as, if A then $x \in G$ (or $x \in B$), where condition A is relevant to the attributes of the input data. One example of such form of knowledge can be seen in the breast cancer recurrence or nonrecurrence prediction. Usually, doctors can judge if the cancer recur or not in terms of some measured attributes of the patients. The prior knowledge used by doctors in the breast cancer dataset includes two rules which depend on two features of the total 32 attributes: tumor size (T) and lymph node status (L). The rules are [33]:

If $L \geq 5$ and $T \geq 4$ Then RECUR and If $L = 0$ and $T \leq 1.9$ Then NONRECUR

The conditions *$L \geq 5$ and $T \geq 4$ ($L = 0$ and $T \leq 1.9$)* in the above rules can be written into such inequality as $Cx \leq c$, where C is a matrix driven from the condition, x represents each individual sample, c is a vector. For example, if each sample x is expressed by a vector $[x_1, \ldots, x_L, \ldots, x_T, \ldots, x_r]^T$, for the rule: *if $L \geq 5$ and $T \geq 4$ then RECUR*, it also means: *if $x_L \geq 5$ and $x_T \geq 4$, then $x \in$ RECUR*, where x_L and x_T are the corresponding values of attributes L and T of a certain sample data, r is the number of attributes. Then its corresponding inequality $Cx \leq c$ can be

written as:

$$\begin{bmatrix} 0 \ldots -1 \ldots \ 0 \ \ \ldots 0 \\ 0 \ldots \ 0 \ \ \ldots -1 \ldots 0 \end{bmatrix} x \leq \begin{bmatrix} -5 \\ -4 \end{bmatrix}.$$

where x is the vector with r attributes include two features relevant to prior knowledge.

Similarly, the condition $L = 0$ *and* $T \leq 1.9$ can also be reformulated to be inequalities. With regard to the condition $L = 0$, in order to express it into the formulation of $Cx \leq c$, we must replace it with the condition $L \geq 0$ *and* $L \leq 0$. Then the condition $L = 0$ *and* $T \leq 1.9$ can be represented by two inequalities: $C^1 x \leq c^1$ and $C^2 x \leq c^2$, as follows:

$$\begin{bmatrix} 0 \ldots -1 \ldots 0 \ldots 0 \\ 0 \ldots \ 0 \ \ \ldots 1 \ldots 0 \end{bmatrix} x \leq \begin{bmatrix} 0 \\ 1.9 \end{bmatrix} \text{and} \begin{bmatrix} 0 \ldots 1 \ldots 0 \ldots 0 \\ 0 \ldots 0 \ldots 1 \ldots 0 \end{bmatrix} x \leq \begin{bmatrix} 0 \\ 1.9 \end{bmatrix}$$

We notice the fact that the set $\{x | \, Cx \leq c\}$ can be regarded as polyhedral convex set. In Fig. 2.7, the triangle and rectangle are such sets.

In two-class classification problem, the result RECUR or NONRECUR is equal to the expression $x \in B$ or $x \in G$. So according to the above rules, we have:

$$Cx \leq c \Rightarrow x \in G \quad (or \quad x \in B) \tag{2.129}$$

In MCLP classifier, if the classes are linearly separable, then $x \in G$ is equal to $x^T w \geq b$, similarly, $x \in B$ is equal to $x^T w \leq b$. That is, the following implication must hold:

$$Cx \leq c \Rightarrow x^T w \geq b \quad \left(or \quad x^T w \leq b\right) \tag{2.130}$$

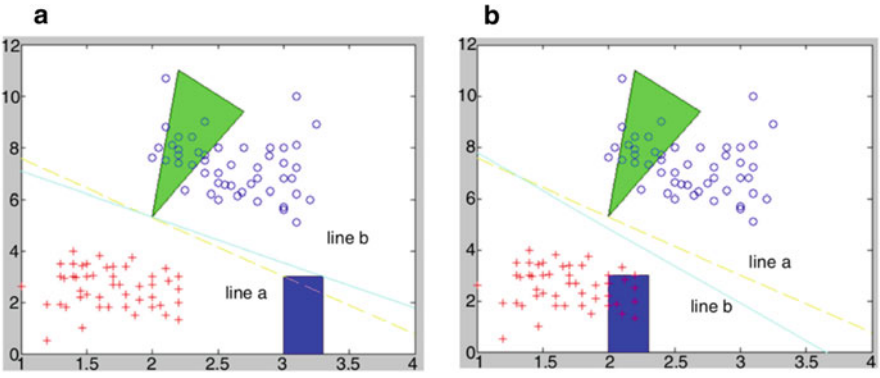


Fig. 2.7 The classification result by MCLP (line a) and knowledge-incorporated MCLP (line b)

For a given (w, b), the implication $Cx \leq c \Rightarrow x^T w \geq b$ holds, this also means that $Cx \leq c$, $x^T w < b$ has no solution x. According to nonhomogeneous Farkas theorem, we can conclude that $C^T u + w = 0$, $c^T u + b \leq 0$, $u \geq 0$, has a solution (u, w) [33].

The above statement is able to be added to constraints of an optimization problem. In this way, the prior knowledge in the form of some equalities and inequalities in constraints is embedded to the original multiple linear programming (MCLP) model. The knowledge-incorporated MCLP model is described in the following.

Knowledge-incorporated MCLP model Now, we are to explain the knowledge-incorporated MCLP model. This model is to deal with linear knowledge and linear separable data. The combination of the two kinds of input can help to improve the performances of both methods.

Suppose there are a series of knowledge sets as follows:

$$\begin{array}{l} \text{If } \ C^i x \leq c^i, i = 1, \ldots, k \quad \text{Then } x \in G \\ \text{If } D^j x \leq d^j, \ j = 1, \ldots, l \quad \text{Then } x \in B \end{array}$$

This knowledge also means the convex sets $\{x | \, C^i x \leq c^i\}$, $i = 1, \ldots, k$ lie on the G side of the bounding plane, the convex sets $\{x | \, D^j x \leq d^j\}$, $j = 1, \ldots, l$ on the B side.

Based on the above theory in the last section, we converted the knowledge to the following constraints:

There exist $u^i, \ i = 1, \ldots, k, \ v^j, j = 1, \ldots, l$, such that:

$$\begin{array}{llll} C^{iT} u^i + w = 0, & c^{iT} u^i + b \leq 0, & u^i \geq 0, & i = 1, \ldots, k \\ D^{jT} v^j - w = 0, & d^{jT} v^j - b \leq 0, & v^j \geq 0, & j = 1, \ldots, l \end{array} \tag{2.131}$$

However, there is no guarantee that such bounding planes precisely separate all the points. Therefore, some error variables need to be added to the above formulas. The constraints are further revised to be:

There exist $u^i, r^i, \rho^i, \ i = 1, \ldots, k \quad and \quad v^j, s^j, \sigma^j, j = 1, \ldots, l$, such that:

$$\begin{array}{llll} -r^i \leq C^{iT} u^i + w \leq r^i, & c^{iT} u^i + b \leq \rho^i, & u^i \geq 0, & i = 1, \ldots, k \\ -s^j \leq D^{jT} v^j - w \leq s^j, & d^{jT} v^j - b \leq \sigma^j, & v^j \geq 0, & j = 1, \ldots, l \end{array} \tag{2.132}$$

After that, we embed the above constraints to the MCLP classifier, and obtained the knowledge-incorporated MCLP classifier:

$$
\begin{aligned}
&\text{Minimize} \quad d_\alpha^+ + d_\alpha^- + d_\beta^+ + d_\beta^- + C\left(\sum\left(r_{\mathrm{i}} + \rho^{\mathrm{i}}\right) + \sum\left(s^{\mathrm{j}} + \sigma^{\mathrm{j}}\right)\right)\\
&\text{Subject to :}\\
&\qquad \alpha^* + \sum_{i=1}^{n} \alpha_i = d_\alpha^- - d_\alpha^+\\
&\qquad \beta^* - \sum_{i=1}^{n} \beta_i = d_\beta^- - d_\beta^+\\
&\qquad\quad x_{11}w_1 + \cdots + x_{1\mathrm{r}}w_{\mathrm{r}} = \mathrm{b} + \alpha_1 - \beta_1, \qquad \text{for } \mathrm{A}_1 \in \mathrm{B},\\
&\qquad\qquad\quad \vdots\\
&\qquad\quad x_{\mathrm{n}1}w_1 + \cdots + x_{\mathrm{nr}}w_{\mathrm{r}} = \mathrm{b} - \alpha_{\mathrm{n}} + \beta_{\mathrm{n}}, \quad \text{for } \mathrm{A}_{\mathrm{n}} \in \mathrm{G},\\
&\qquad -r^{\mathrm{i}} \le \mathrm{C}^{\mathrm{i}'}u^{\mathrm{i}} + w \le r^{\mathrm{i}}, \qquad \mathrm{i} = 1, \ldots, \mathrm{k}\\
&\qquad \mathrm{c}^{\mathrm{i}'}u^{\mathrm{i}} + b \le \rho^{\mathrm{i}}\\
&\qquad -s^{\mathrm{j}} \le \mathrm{D}^{\mathrm{j}'}v^{\mathrm{j}} - w \le s^{\mathrm{j}}, \qquad \mathrm{j} = 1, \ldots, l\\
&\qquad \mathrm{d}^{\mathrm{j}'}v^{\mathrm{j}} - b \le \sigma^{\mathrm{j}}\\
&\qquad \alpha_1, \ldots, \alpha_{\mathrm{n}} \ge 0, \qquad \beta_1, \ldots, \beta_{\mathrm{n}} \ge 0, \qquad \left(u^{\mathrm{i}}, v^{\mathrm{j}}, r^{\mathrm{i}}, \rho^{\mathrm{i}}, s^{\mathrm{j}}, \sigma^{\mathrm{j}}\right) \ge 0
\end{aligned}
\tag{2.133}
$$

In this model, all the inequality constraints are derived from the prior knowledge. The last objective $\mathrm{C}(\sum(r_{\mathrm{i}} + \rho^{\mathrm{i}}) + \sum(s^{\mathrm{j}} + \sigma^{\mathrm{j}}))$ is about the slack error variables added to the original knowledge equality constraints. The last objective attempts to drive the error variables to zero. We want to get the best bounding plane (w, b) by solving this model to separate the two classes.

We notice the fact that if we set the value of parameter C to be zero, this means to take no account of knowledge. Then this model will be equal to the original MCLP model. Theoretically, the larger the value of C, the greater impact on the classification result of the knowledge sets.

Knowledge-incorporated KMCLP Model If the data set is nonlinear separable, the above model will be inapplicable. We need to figure out how to embed prior knowledge into the KMCLP model, which can solve nonlinear separable problem.

As is shown in the above part, in generating KMCLP model, we suppose:

$$
w = \sum_{i=1}^{n} \lambda_i y_i X_i \tag{2.134}
$$

If expressed by matrix, the above formulation will be:

$$w = X^T Y\lambda \tag{2.135}$$

where Y is $n*n$ diagonal matrix, the value of each diagonal element depends on the class label of the corresponding sample data, which can be $+1$ or -1. X is the $n*r$ input matrix with n samples, r attributes. λ is a n-dimensional vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^T$.

$$Y = \begin{bmatrix} y_1 & 0 & \ldots & 0 \\ 0 & y_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & y_n \end{bmatrix}, \qquad X = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1r} \\ x_{21} & x_{22} & \ldots & x_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nr} \end{bmatrix}$$

Therefore, w in the original MCLP model is replaced by $X^T Y\lambda$, thus forming the KMCLP model. And in this new model, the value of each λ_i is to be worked out by the optimization model.

In order to incorporate prior knowledge into KMCLP model, the inequalities about the knowledge must be transformed to be the form with λ_i instead of w. Enlightened by the KMCLP model, we also introduce kernel to the expressions of knowledge. Firstly, the equalities in (2.131) are multiplied by input matrix X [34]. Then replacing w with $X^T Y\lambda$, (2.131) will be:

$$\begin{aligned} &XC^{iT}u^i + XX^T Y\lambda = 0, \quad c^{iT}u^i + b \le 0, \quad u^i \ge 0, \quad i = 1, \ldots, k \\ &XD^{jT}v^j - XX^T Y\lambda = 0, \quad d^{jT}v^j - b \le 0, \quad v^j \ge 0, \quad j = 1, \ldots, l \end{aligned} \tag{2.136}$$

Kernel function is introduced here to replace XC^{iT} and XX^T. Also slack errors are added to the expressions, then such kind of constraints are formulated:

$$\begin{aligned} &-r^{\mathrm{i}} \le K\left(X, \mathrm{C}^{\mathrm{i}T}\right)u^{\mathrm{i}} + K\left(X, X^T\right)Y\lambda \le r^{\mathrm{i}}, \qquad \mathrm{i} = 1, \ldots, \mathrm{k} \\ &\mathrm{c}^{\mathrm{i}T}u^{\mathrm{i}} + b \le \rho^{\mathrm{i}} \\ &-s^{\mathrm{j}} \le K\left(X, \mathrm{D}^{\mathrm{j}T}\right)v^{\mathrm{j}} - K\left(X, X^T\right)Y\lambda \le s^{\mathrm{j}}, \qquad \mathrm{j} = 1, \ldots, l \\ &\mathrm{d}^{\mathrm{j}T}v^{\mathrm{j}} - b \le \sigma^{\mathrm{j}} \end{aligned} \tag{2.137}$$

These constraints can be easily embedded to KMCLP model as the constraints acquired from prior knowledge.

Knowledge-incorporated KMCLP classifier:

$$
\begin{aligned}
&\text{Min}\left(d_\alpha^+ + d_\alpha^- + d_\beta^+ + d_\beta^-\right) \ + C\left(\sum_{i=1}^{k}\left(r_\mathrm{i}+\rho^\mathrm{i}\right)+\sum_{j=1}^{l}\left(s^\mathrm{j}+\sigma^\mathrm{j}\right)\right)\\
\text{s.t.}\quad &\lambda_1 y_1 K\left(X_1, X_1\right)+\cdots+\lambda_n y_n K\left(X_n, X_1\right)=b+\alpha_1-\beta_1, , \qquad \text{for } X_1 \in \mathrm{B},\\
&\qquad\qquad .\\
&\qquad\qquad .\\
&\qquad\qquad .\\
&\lambda_1 y_1 K\left(X_1, X_n\right)+\cdots+\lambda_n y_n K\left(X_n, X_n\right)=b-\alpha_n+\beta_n, \quad \text{for } X_\mathrm{n} \in \mathrm{G},\\
&\alpha^*+\sum_{i=1}^{n}\alpha_i=d_\alpha^- - d_\alpha^+,\\
&\beta^*-\sum_{i=1}^{n}\beta_i=d_\beta^- - d_\beta^+,\\
&-r^\mathrm{i} \le K\left(X, \mathrm{C}^{\mathrm{i}T}\right)u^\mathrm{i}+K\left(X, X^T\right)Y\lambda \le r^\mathrm{i}, \qquad \mathrm{i}=1,\ldots,\mathrm{k}\\
&\mathrm{c}^{\mathrm{i}T}u^\mathrm{i}+b \le \rho^\mathrm{i}\\
&-s^\mathrm{j} \le K\left(X, \mathrm{D}^{\mathrm{j}T}\right)v^\mathrm{j}-K\left(X, X^T\right)Y\lambda \le s^\mathrm{j}, \qquad \mathrm{j}=1,\ldots,l\\
&\mathrm{d}^{\mathrm{j}T}v^\mathrm{j}-b \le \sigma^\mathrm{j}\\
&\alpha_1,\ldots,\alpha_\mathrm{n} \ge 0, \qquad \beta_1,\ldots,\beta_\mathrm{n} \ge 0, \qquad \lambda_1,\ldots,\lambda_\mathrm{n} \ge 0,\\
&\left(u^\mathrm{i}, v^\mathrm{j}, r^\mathrm{i}, \rho^\mathrm{i}, s^\mathrm{j}, \sigma^\mathrm{j}\right) \ge 0\\
&d_\alpha^-, d_\alpha^+, d_\beta^-, d_\beta^+ \ge 0
\end{aligned}
\tag{2.138}
$$

In this model, all the inequality constraints are derived from prior knowledge. u^i, $v^i \in R^p$, where p is the number of conditions in one knowledge. For example, in the knowledge *if* $x_L \ge 5$ *and* $x_T \ge 4$, *then* $x \in$ *RECUR*, the value of p is 2. r^i, ρ^i, s^j and σ^j are all real numbers. And the last objective Min $\sum (r_\mathrm{i} + \rho^\mathrm{i}) + \sum (s^\mathrm{j} + \sigma^\mathrm{j})$ is about the slack error variables added to the original knowledge equality constraints. As we talked in last section, the larger the value of C, the greater impact on the classification result of the knowledge sets.

In this model, several parameters need to be set before optimization process. Apart from C we talked about above, the others are parameter of kernel function q (if we choose RBF kernel) and the ideal compromise solution α^* and β^*. We want to get the best bounding plane (λ, b) by solving this model to separate the two classes. And the discrimination function of the two classes is:

$$
\begin{aligned}
&\lambda_1 y_1 K\left(X_1, z\right)+\cdots+\lambda_n y_n K\left(X_n, z\right) \le b, \qquad \textit{then} \quad z \in B\\
&\lambda_1 y_1 K\left(X_1, z\right)+\cdots+\lambda_n y_n K\left(X_n, z\right) \ge b, \qquad \textit{then} \quad z \in G
\end{aligned}
\tag{2.139}
$$

where z is the input data which is the evaluated target with r attributes. X_i represents each training sample. y_i is the class label of *ith* sample.

In the above models, the prior knowledge we deal with is linear. That means the conditions in the above rules can be written into such inequality as $Cx \le c$, where C is a matrix driven from the condition, x represents each individual sample, c is a vector. The set $\{x|\ Cx \le c\}$ can be viewed as polyhedral convex set, which is a linear

geometry in input space. But, if the shape of the region which consists of knowledge is nonlinear, for example, $\{x| \ ||x||^2 \leq c\}$, how to deal with such kind of knowledge?

Suppose the region is nonlinear convex set, we describe the region by $g(x) \leq 0$. If the data is in this region, it must belong to class B. Then, such kind of nonlinear knowledge may take the form of:

$$\begin{array}{lll} g(x) \leq 0 & \Rightarrow & x \in B \\ h(x) \leq 0 & \Rightarrow & x \in G \end{array} \tag{2.140}$$

Here $g(x)$: $R^r \to R^p$ $(x \in \Gamma)$ and $h(x)$: $R^r \to R^q$ $(x \in \Delta)$ are functions defined on a subset Γ and Δ of R^r which determine the regions in the input space. All the data satisfied $g(x) \leq 0$ must belong to the class B and $h(x) \leq 0$ to the class G.

With KMCLP classifier, this knowledge equals to:

$$\begin{array}{lll} g(x) \leq 0 & \Rightarrow & \lambda_1 y_1 K(X_1, x) + \cdots + \lambda_n y_n K(X_n, x) \leq b, \quad (x \in \Gamma) \\ h(x) \leq 0 & \Rightarrow & \lambda_1 y_1 K(X_1, x) + \cdots + \lambda_n y_n K(X_n, x) \geq b, \quad (x \in \Delta) \end{array} \tag{2.141}$$

This implication can be written in the following equivalent logical form [35]:

$$\begin{array}{lll} g(x) \leq 0 & , & \lambda_1 y_1 K(X_1, x) + \cdots + \lambda_n y_n K(X_n, x) - b > 0, \text{has no solution } x \in \Gamma. \\ h(x) \leq 0 & , & \lambda_1 y_1 K(X_1, x) + \cdots + \lambda_n y_n K(X_n, x) - b < 0, \text{has no solution } x \in \Delta. \end{array} \tag{2.142}$$

The above expressions hold, then there exist $v \in R^p$, $r \in R^q$, v,$r \geq 0$ such that:

$$\begin{array}{l} -\lambda_1 y_1 K(X_1, x) - \cdots - \lambda_n y_n K(X_n, x) + b + v^T g(x) \geq 0, \quad (x \in \Gamma) \\ \lambda_1 y_1 K(X_1, x) + \cdots + \lambda_n y_n K(X_n, x) - b + r^T h(x) \geq 0, \quad (x \in \Delta) \end{array} \tag{2.143}$$

Add some slack variables on the above two inequalities, then they are converted to:

$$\begin{array}{l} -\lambda_1 y_1 K(X_1, x) - \cdots - \lambda_n y_n K(X_n, x) + b + v^T g(x) + s \geq 0, \quad (x \in \Gamma) \\ \lambda_1 y_1 K(X_1, x) + \cdots + \lambda_n y_n K(X_n, x) - b + r^T h(x) + t \geq 0, \quad (x \in \Delta) \end{array} \tag{2.144}$$

The above statement is able to be added to constraints of an optimization problem.

Suppose there are a series of knowledge sets as follows:

$$\text{If } g_i(x) \leq 0, \text{Then } x \in B \quad \left(g_i(x) : R^r \rightarrow R^p{}_i \left(x \in \Gamma_i\right), i = 1, \ldots, k\right)$$

$$\text{If } h_j(x) \leq 0, \text{Then } x \in G \quad \left(h_j(x) : R^r \rightarrow R^q{}_j \left(x \in \Delta_j\right), j = 1, \ldots, l\right)$$

Based on the above theory in last section, we converted the knowledge to the following constraints:

There exist $v_i \in R^p{}_i$, $i = 1, \ldots, k$, $r_j \in R^q{}_j$, $j = 1, \ldots, l$, v_i,$r_j \geq 0$ such that:

$$\begin{aligned} &-\lambda_1 y_1 K\left(X_1, x\right) - \cdots - \lambda_n y_n K\left(X_n, x\right) + b + {v_i}^T g_i(x) + s_i \geq 0, \quad (x \in \Gamma) \\ &\lambda_1 y_1 K\left(X_1, x\right) + \cdots + \lambda_n y_n K\left(X_n, x\right) - b + {r_j}^T h_j(x) + t_j \geq 0, \quad (x \in \Delta) \end{aligned} \tag{2.145}$$

These constraints can be easily imposed to KMCLP model as the constraints acquired from prior knowledge.

Nonlinear knowledge in KMCLP classifier [36]:

$$\begin{aligned} &\text{Min}\left(d_\alpha^+ + d_\alpha^- + d_\beta^+ + d_\beta^-\right) \quad + C\left(\sum_{i=1}^{k} s_i + \sum_{j=1}^{l} t^j\right) \\ \text{s.t.} \quad &\lambda_1 y_1 K\left(X_1, X_1\right) + \cdots + \lambda_n y_n K\left(X_n, X_1\right) = b + \alpha_1 - \beta_1, , \qquad \text{for } X_1 \in \text{B}, \\ &\qquad\qquad \vdots \\ &\lambda_1 y_1 K\left(X_1, X_n\right) + \cdots + \lambda_n y_n K\left(X_n, X_n\right) = b - \alpha_n + \beta_n, \quad \text{for } X_\text{n} \in \text{G}, \\ &\alpha^* + \sum_{i=1}^{n} \alpha_i = d_\alpha^- - d_\alpha^+, \\ &\beta^* - \sum_{i=1}^{n} \beta_i = d_\beta^- - d_\beta^+, \\ &-\lambda_1 y_1 K\left(X_1, x\right) - \cdots - \lambda_n y_n K\left(X_n, x\right) + b + {v_i}^T g_i(x) + s_i \geq 0, \qquad \text{i} = 1, \ldots, \text{k} \\ &s_i \geq 0, \qquad \text{i} = 1, \ldots, \text{k} \\ &\lambda_1 y_1 K\left(X_1, x\right) + \cdots + \lambda_n y_n K\left(X_n, x\right) - b + {r_j}^T h_j(x) + t_j \geq 0, \qquad \text{j} = 1, \ldots, l \\ &t_j \geq 0, \qquad \text{j} = 1, \ldots, l \\ &\alpha_1, \ldots, \alpha_\text{n} \geq 0, \qquad \beta_1, \ldots, \beta_\text{n} \geq 0, \qquad \lambda_1, \ldots, \lambda_\text{n} \geq 0, \\ &\left(v_i, r_j\right) \geq 0 \\ &d_\alpha^-, d_\alpha^+, d_\beta^-, d_\beta^+ \geq 0 \end{aligned} \tag{2.146}$$

In this model, all the inequality constraints are derived from the prior knowledge. The last objective $C\left(\sum_{i=1}^{k} s_i + \sum_{j=1}^{l} t^j\right)$ is about the slack error. Theoretically, the larger the value of C, the greater impact on the classification result of the knowledge sets.

The parameters need to be set before optimization process are C, q (if we choose RBF kernel), α^* and β^*. The best bounding plane of this model decided by (λ, b) of the two classes is the same with formula (2.139).

2.2.2 *Decision Rule Extraction for Regularized Multiple Criteria Linear Programming Model*

In this section, we present a clustering-based rule extraction method to generate decision rules from the black box RCMLP model. Our method can improve the interpretability of the RMCLP model by using explicit and explainable decision rules. To achieve this goal, a clustering algorithm will first be used to generate *prototypes* (which are the clustering centers) for each group of examples identified by the RMCLP model. Then, hyper cubes (whose edges are parallel to the axes) will be extracted around each prototype. This procedure will be repeated until all the training examples are covered by a hyper cube. Finally, the hyper cubes will be translated to a set of *if-then* decision rules. Experiments on both synthetic and real-world data sets have demonstrate the effectiveness of our rule extraction method.

For ease of description, we introduce some notations first. Assume a r-dimensional space, the coordinate of the clustering center p is $p = (p_1, \ldots, p_r)$, and the classification hyper plane is $\sum_{i=1}^{r} a_i x_i = b$ (where x_i is the direction of the hyper plane). For each class, we prefer hyper cubes which cover as many examples as possible. Intuitively, if we pick a point u on the classification boundary and then draw cubes based on both clustering center p and u, then the generated hyper cube will cover the largest area with respect to the current prototype p. The distance from p to the hyper plane can be calculated by Eq. (2.147) as follows:

$$d = Distance\,(f, p_i) = \frac{\sum_{i=1}^{r} p_i x_i - b}{\sqrt{x_i^2}} \tag{2.147}$$

After computing d, Step 2.3 draws hyper cubes $H = DrawHC(d, P_i)$ by using the prototype point P_i as the central point, and each edge has a length of $\sqrt{2}d$ meanwhile parallel with the axis. By so doing, we can get *if-then* rules which are easily understood. For example, for a specific example $a_1 \in G_1$, a decision rule can be described in the following form:

$$\begin{aligned} &if \quad (l_1 \le a_{11} \le u_1)\ and\ (l_2 \le a_{12} \le u_2) \ldots\ldots and\ (l_r \le a_{1r} \le u_r) \\ &then\ a_1\ belongs\ to\ class\ 1 \end{aligned} \tag{2.148}$$

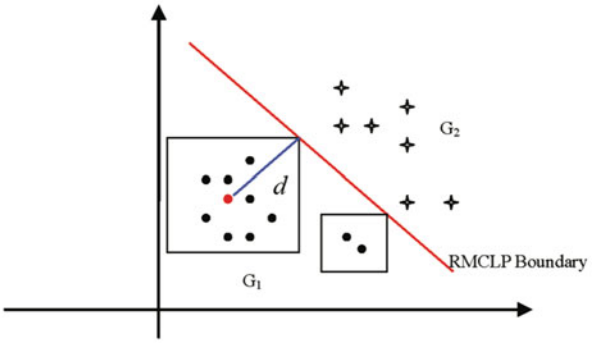


Fig. 2.8 An illustration of Algorithm 2.5 which generates hyper cubes from RMCLP models. Based on the RMCLP model's decision boundary (the red line), Algorithm 2.5 first calculates several clustering centers for each class (e.g., the red circle in Group 1), then it calculates the distance d from the classification boundary to the clustering center (the blue line). After that, it generates a series of hyper cubes. Each hyper cube's edge is parallel to the axes and the length is $\sqrt{2}d$. Finally, the hyper cubes can be easily translated into rules that are explainable and understandable

Figure 2.8 illustrates an example with two dimensions. Examples in G_1 ($a_i \in G_1$) are covered by hyper cubes with a central point as its clustering center and a vertex on the hyper plane $\sum_{i=1}^{r} a_i x_i = b$.

The main computational cost of Algorithm 2.5 is from *Steps 2.1~2.3*, where a K-Means clustering model and two distance functions are calculated. Assume there are l iterations of K-Means. In each iteration, there are k clusters. Therefore, the total time complexity of K-Means will be O($lknr$), where n is the number of training examples, r is the number of dimensions.

On the other hand, calculating distance d for each clustering center by (2.147) will take a linear time complexity, so the computational cost of *Step 2.2* will be $O(k)$ for k clustering centers. Finally, the time cost of extracting hyper cubes in *Step 2.3* will be $O(kr)$ for k clustering centers in r dimensional space. To sum up, the total computational complexity of Algorithm 2.5 can be denoted by (2.149),

$$O(lknr) + O(k) + O(kr) = O(lknr) \tag{2.149}$$

The above analysis indicates that the hyper cube extracting method in *Steps 2.2* and *2.3* is dominated by the K-Means clustering model in *Step 2.1*. It is in linear time complexity with respect to training example size.

Algorithm 2.5 Extract Rules from MCLP Models

Input: The data set $A = \{a_1, a_2, \ldots, a_n\}$, RMCLP model f
Output: Rule Set $\{w\}$
Begin
Step 1. Classify all the examples in A using model f;
Step 2. Define Covered set $C = \Phi$, Uncovered set $U = A$;
Step 3. **While** (*U is not empty*) do
Step 3.1 **For** each group G_i,
Calculate the clustering center $P_i = K\text{-}means(G_i \cap U)$;
End for
Step 3.2 Calculate distances between each P_i and boundary $d = Distance(f, P_i)$;
Step 3.3 Draw a new hypercube $H = DrawHC(d, P_i)$;
Step 3.4 **For** all the examples$a_i \in U$,
If a_i is covered by H
$U = U \backslash a_i$, $C = C \cup a_i$;
End If
End For
End While
Step 4 Translate each hypercube H into rule;
Step 5 Return the rule set $\{w\}$
End

To demonstrate the effectiveness of the proposed rules extraction method, we will test our method on both synthetic and real-world data sets. The whole testing system is implemented in a Java environment by integrating WEKA data mining tools [37]. The clustering method used in our experiments is the *simple K-Means* package in WEKA.

As shown in Fig. 2.9a, we generate a 2-dimensional 2-class data set containing 60 examples, with 30 examples for each class. In each class, we use 50% of the examples to train a RMCLP model. That is, 30 training examples in total are used to train the RMCLP model. All examples comply with Gaussian distribution $x \sim N(\mu, \Sigma)$, where μ is mean vector and Σ is covariate matrix. The first group is generated by a mean vector $\mu_1 = [1,1]$ with a covariance matrix $\Sigma_1 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. The second group is generated by a mean vector $\mu_2 = [2,2]$ with a covariance matrix $\Sigma_2 = \Sigma_1$.

Here we only discuss the two-group classification problem. It is not difficult to extend to multiple-group classification applications. It is expected to extract knowledge from the RMCLP model in the form of:

$$\boldsymbol{if}\ (a \le x1 \le b, c \le x2 \le d)\ \ \boldsymbol{then}\ \ Definition\ 1 \tag{2.150}$$

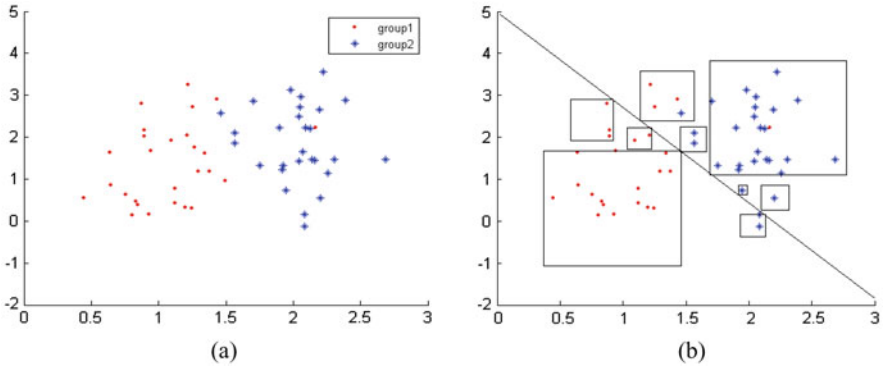


Fig. 2.9 (**a**) The synthetic dataset; (**b**) Experimental results. The straight line is the RMCLP model's classification boundary, and the squares are hyper cubes generated by using Algorithm 2.5. All the examples are covered by the squares whose edges are parallel to the axes

The result is shown in Fig. 2.9b; we can observe that for the total of 60 examples, three examples in group 1, and one example in group 2 are misclassified by the RMCLP model. That is to say, the accuracy of RMCLP on this synthetic dataset is 56/60 = 92.3%. By using our rule extraction algorithm, we can generate nine squares, four squares for group 1, and five squares for group 2. All the squares can be translated to explainable rules in the form of (6) as follows:

K_1: if $0.6 \leq x_1 \leq 0.8$ and $2 \leq x_2 \leq 2.8$, then $x \in G_1$;
K_2: if $1.1 \leq x_1 \leq 1.3$ and $1.8 \leq x_2 \leq 2.1$, then $x \in G_1$;
K_3: if $0.4 \leq x_1 \leq 1.5$ and $-1 \leq x_2 \leq 1.6$, then $x \in G_1$;
K_4: if $0.9 \leq x_1 \leq 2.2$ and $-0.8 \leq x_2 \leq 0$, then $x \in G_1$;
K_5: if $1.2 \leq x_1 \leq 1.6$ and $2.2 \leq x_2 \leq 3.2$, then $x \in G_2$;
K_6: if $1.4 \leq x_1 \leq 1.6$ and $1.8 \leq x_2 \leq 2.0$, then $x \in G_2$;
K_7: if $1.7 \leq x_1 \leq 2.8$ and $1.0 \leq x_2 \leq 4.0$, then $x \in G_2$;
K_8: if $1.9 \leq x_1 \leq 2.0$ and $0.7 \leq x_2 \leq 0.8$, then $x \in G_2$;
K_9: if $2.1 \leq x_1 \leq 2.4$ and $0.1 \leq x_2 \leq 0.5$, then $x \in G_2$;

where k_i $(i = 1, \ldots, 9)$ denotes the i^{th} rule. From the results on this synthetic data set, we can observe that by using the proposed rule extraction method, we can not only obtain prediction results from RMCLP, but also comprehensible rule.

As one of the basic services offered by the Internet, E-Mail usage is becoming increasingly widely adopted. Along with constant global network expansion and network technology improvement, people's expectations of an E-Mail service are increasingly demanding. E-Mail is no longer merely a communication tool for people to share their ideas and information; its wide acceptance and technological advancement has given it the characteristics of a business service [38], and it is being commercialized as a technological product.

At the same time, many business and specialized personal users of E-Mail want an E-Mail account that is safe, reliable, and equipped with a first-class customer

support service. Therefore, many websites have developed their own user-pays E-mail service to satisfy this market demand. According to statistics, the Chinese network has advanced so much in the past few years that, by 2005, the total market size of Chinese VIP E-mail services reached 6.4 hundred million RMB. This enormous market demand and market prospect also means increasing competition between the suppliers. How to analyze the pattern of lost customer accounts and decrease the customer loss rate have become a focal point of competition in today's market [39, 40].

Our partner company's VIP E-Mail data are mainly stored in two kinds of repository systems; one is customer databases, the other is log files. They are mainly composed of automated machine recorded customer activity journals and large amount of manually recorded tables; these data are distributed among servers located in different departments of our partnering companies, coving more than 30 kinds of transaction data charts and journal documents, with over 600 attributes.

If we were to directly analysis these data, it would lead to a "course of dimensionality", that is to say, a drastic rise in computational complexity and classification error with data of large dimensions. Hence, the dimensionality of the feature space must be reduced before classification is undertaken. According to the accumulated experience functions, we eventually selected 230 attributes from the original 600 attributes.

Figure 2.10 displays the procedure of feature selection of the VIP E-Mail dataset. We selected a part of the data charts and journal documents from the VIP E-Mail System. The left upper part of Fig. 2.10 displays the three logging journal documents and two email transaction journal documents; when the user logs into the pop3 server, the machine will record the user's login into the log file *pop3login*; similarly when the user logs into the smtp server, the machine will record this into the log file *smtplogin*; when the user logs into the E-Mail system through http protocol, the machine will record it into the log file *weblogin*; when the user successfully sends an E-Mail by smtp protocol, the system will record it into the log file *smtprcptlog*; when receiving a letter, it will be recorded into the log file *mx_rcptlog*.

We extracted 37 attributes from these five log files, that is, 184 attributes in total, to describe user logins and transactions. From the databases, shown in the left lower section of Fig. 2.8, we extracted six features about "customer complaint about the VIP E-Mail Service", 24 features about "customer payment" and 16 features about "customer's personal information" (for example, age, gender, occupation, income etc.) to form the operational table. Thus, 185 features from log files and 65 features from databases eventually formed the Large Table, and the 230 attributes depicted the features of the customers. The accumulated experience functions used in the feature selection are confidential, and further discussion of them exceeds the range of this section.

Considering the integrality of the customer records, we eventually extracted two groups from a huge number of data: the current and the lost. Ten thousand nine hundred and ninety-six customers, 5498 for each class, were chosen from the dataset. Combining the 10,996 SSN with the 230 features, we eventually acquired the Large Table with 5498 current records and 5498 lost records, which became the dataset for data mining.

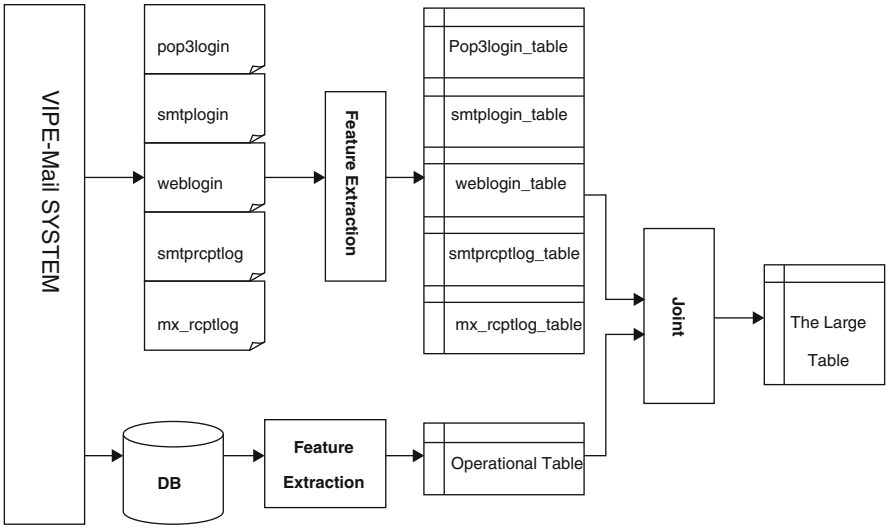


Fig. 2.10 The roadmap of the VIP Email Dataset

Table 2.6 Ten Folder Cross Validation on VIP Email Dataset

Cross validation	Training set (500 Bad data + 500 Good data)				Testing set (4998 Bad data + 4998 Good data)			
	LOST	Accuracy (%)	CURRENT	Accuracy (%)	LOST	Accuracy (%)	CURRENT	Accuracy (%)
DataSet 1	444	88.80	455	91.00	4048	80.99	4311	86.25
DataSet 2	447	89.40	459	91.80	4081	81.65	4355	87.13
DataSet 3	449	89.80	465	92.00	4079	81.61	4362	87.27
DataSet 4	440	88.00	467	92.40	4006	80.15	4286	85.75
DataSet 5	435	87.00	460	92.00	4010	80.23	4420	88.44
DataSet 6	436	87.20	460	92.00	3995	79.93	4340	86.83
DataSet 7	445	89.00	464	92.80	4008	80.19	4403	88.10
DataSet 8	443	88.60	455	91.00	4052	81.07	4292	85.87
DataSet 9	429	85.80	457	91.40	3955	79.13	4436	88.76
DataSet10	440	88.00	456	91.20	4087	81.77	4355	87.13

Table 2.6 lists the ten-folder cross validation results of the RMCLP model's performance on the VIP Email Dataset. The columns "LOST" and "CURRENT" refer to the number of records that were correctly classified as "lost" and "current" respectively. The column "Accuracy" was calculated using correctly classified records divided by the total records in that class. From Table 2.6, we can observe that the average prediction accuracy of the RMLCP on this data set is 80.67% on the first class and 87.15% on the second class. That is, on the whole 10,996 test examples, the average accuracy of RMCLP is 82.91%.

Table 2.7 Comparisons between RMCLP's Rule and Decision Tree's Rule

RMCLP's Rule	Decision Tree's Rule
RULE 1: if 0 <= The number of emails <= 3 *and* 0 <= the number of POP3 login on Tuesday <= 6 *and* 0 <= the number of HTTP login <= 1 *and* 0 <= Free Email Service <= 1 *and* 0 <= The percentage of Charge Type 7 <= 0.3 *and* 0 <= The total Charge Fee <= 45 ... then *class LOST* [0.816]	**RULE 1′**: if The number of emails <= 1 *and* the number of POP3 login on Tuesday <= 3 *and* number of HTTP login <= 1 *and* Free Email Service = 1 *and* The percentage of Charge Type 7 <= 0.25 *and* The total Charge Fee <= 50 ... then *class LOST* [0.746]
RULE 6: if 0 <= The number of HTTP Login <= 5 *and* 0 <= Free Email Service Status <= 1 *and* 0.2 <= The percentage of Charge Type 11 <= 0.5 *and* 0 <= The total Charge Fee <= 4 *and* 0 <= The number of emails <= 3 *and* 0 <= CONTACT_NUMBER <= 1 *and* 0 <= IDNUM <= 1 ... then *class CURRENT* [0.802]	**RULE 6′**: if The number of HTTP Login <= 3 *and* Free Email Service Status = 0 *and* The percentage of Charge Type 11 > 0.294 *and* The total Charge Fee <= 5 *and* The number of Emails <= 1 *and* CONTACT_NUMBER = 1 *and* IDNUM = 0 ... then *class CURRENT* [0.739]
Average Accuracy: 80.90%	Average Accuracy: 74.25%

As discussed above, a decision tree is widely used to extract rules from training examples. In the following experiments, we will compare our method with a decision tree (which is implemented by the WEKA *J48* package).

Table 2.7 shows the comparison results between our method and the decision tree. By using our rule extraction method, we obtain more than 20 hyper cubes. Due to space limitation, we only list the two most representative rules (i.e., Rule 1 for class "LOST" and Rule 6 for class "CURRENT") in the left side of Table 2.7. Then we find the corresponding rules from the decision tree (i.e., Rule $1'$ for class "LOST" and Rule $6'$ for class "CURRENT"), and list them in the right side of Table 2.7.

From these results, we can observe that our rule extraction method acquires much more accurate rules than the decision tree method. For example, when comparing Rule 1 with Rule $1'$, we can safely say that Rule 1 is supported by 81.6% examples in the "LOST" class; by contrast, rules from decision tree only get 74.6% supportive examples. Similarly, when comparing Rule 6 with Rule $6'$, our method also achieves better support than the decision tree.

At the bottom of Table 2.7, we list the average accuracy of the two methods. It is obvious that the average accuracy of rules extracted from RMCLP is 80.90%. This is better than the decision tree's accuracy of 74.25%. Moreover, compared to the RMCLP's performance in Table 2.6 (which equals 82.91%), we can say that the average accuracy of the extracted rules (i.e., 80.90%) suffers only a little loss in performance. Therefore, our rule extraction method from the RMCLP model can effectively extract comprehensible rules from the RMCLP model.

2.3 Multiple-Criteria Decision Making Based Data Analysis

2.3.1 A Multicriteria Decision Making Approach for Estimating the Number of Clusters

Estimating the number of clusters for a given data set is closely related to the validity measures and the data set structures. Many validity measures have been proposed and can be classified into three categories: external, internal, and relative [41]. External measures use predefined class labels to examine the clustering results. Because external validation uses the true class labels in the comparison, it is an objective indicator of the true error rate of a clustering algorithm. Internal measures evaluate clustering algorithms by measuring intra- and inter-cluster similarity. An algorithm is regarded as good if the resulting clusters have high intra-class similarities and low inter-class similarities. Relative measures try to find the best clustering structure generated by a clustering algorithm using different parameter values. Extensive reviews of cluster validation techniques can be found in [41] and [42, 43].

Although external measures perform well in predicting the clustering error in previous studies, they require a priori structure of a data set and can only be applied to data sets with class labels. Since this study concentrates on data sets without class labels, it utilizes relative validity measures. The proposed approach can be applied to a wide variety of clustering algorithms. For simplicity, this study chooses the well-known k-means clustering algorithm. Figure 2.11 describes the MCDM-based approach for determining the number of clusters in a data set. For a given data set, different numbers of clusters are considered as alternatives and the performances of k-means clustering algorithm on the relative measures with different numbers of clusters represent criteria by MCDM methods. The output is a ranking of numbers of clusters, which evaluates the appropriateness of different numbers of clusters for a given data set based on their overall performances for multiple criteria (i.e., selected relative measures).

2.3.1.1 MCDM Methods

This study chooses three MCDM methods for estimating the number of clusters for a data set. This section introduces the selected MCDM methods (i.e., WSM, PROMETHEE, and TOPSIS) and explains how they are used to estimate the optimal number of clusters for a given data set.

MCDM Method 1: Weighted Sum Method (WSM)

The weighted sum method (WSM) was introduced by Zadeh [44]. It is the most straightforward and widely-used MCDM method for evaluating alternatives. When

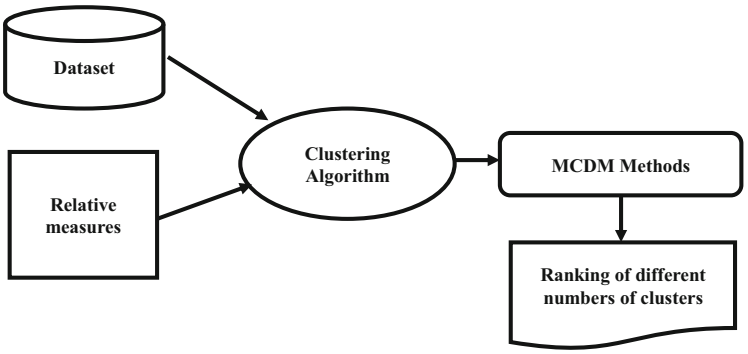


Fig. 2.11 A MCDM-based approach for determining the number of clusters in a dataset

an MCDM problem involves both benefit and cost criteria, two approaches can be used to deal with conflicting criteria. One is the benefit to cost ratio and the other is the benefit minus cost [45]. For the estimation of optimal number of clusters for a data set, the relative indices Dunn, silhouette, and PBM are benefit criteria and have to be maximized, while Hubert, normalized Hubert, Davies-Bouldin index, SD, S_Dbw, CS, and C-index are cost criteria and have to be minimized. This study chooses the benefit minus cost approach and applies the following formulations to rank different numbers of clusters.

Suppose there are m alternatives, k benefit criteria, and n cost criteria. The total benefit of alternative $A_i^{benefit}$ is defined as follows:

$$A_i^{benefit} = \sum_{j=1}^{k} w_j a_{ij},\ for\ \ i = 1, 2, 3, \ldots, m$$

where a_{ij} represents the performance measure of the jth criterion for alternative Ai. Similarly, the total cost of alternative $A_i^{\cos t}$ is defined as follows:

$$A_i^{\cos t} = \sum_{j=1}^{n} w_j a_{ij},\ for\ \ i = 1, 2, 3, \ldots, m$$

$where \sum_{j=1}^{k} w_j + \sum_{j=1}^{n} w_j = 1; 0 < w_j \leq 1$. Then the importance of alternative $A_i^{WSM-score}$ is defined as follows:

$$A_i^{WSM-score} = A_i^{benefit} - A_i^{\cos t},\ for\ \ i = 1, 2, 3, \ldots, m$$

The best alternative is the one has the largest WSM score [45].

MCDM Method 2: Preference Ranking Organization Method for Enrichment of Evaluations (PROMETHEE)

Brans proposed the PROMETHEE I and PROMETHEE II, which use pairwise comparisons and outranking relationships to choose the best alternative [46]. The final selection is based on the positive and negative preference flows of each alternative. The positive preference flow indicates how an alternative is outranking all the other alternatives and the negative preference flow indicates how an alternative is outranked by all the other alternatives [47]. While PROMETHEE I obtains partial ranking because it does not compare conflicting actions [48], PROMETHEE II ranks alternatives according to the net flow which equals to the balance of the positive and the negative preference flows. An alternative with a higher net flow is better [47]. Since the goal of this study is to provide a complete ranking of different numbers of clusters, PROMETHEE II is utilized. The following procedure presented by Brans and Mareschal [47] is used in the experimental study:

Step 1. Define aggregated preference indices.
Let a,b ∈A, and let

$$\begin{cases} \pi\,(a,b) = \sum_{j=1}^{k} p_j\,(a,b)\,w_j, \\ \pi\,(a,b) = \sum_{j=1}^{k} p_j\,(b,a)\,w_j. \end{cases}$$

where A is a finite set of possible alternatives $\{a_1, a_2, \ldots, a_n\}$, k represents the number of evaluation criteria, and w_j is the weight of each criterion. For estimating the number of clusters for a given data set, the alternatives are different numbers of clusters and the criteria are relative indices. Arbitrary numbers for the weights can be assigned by decision-makers. The weights are then normalized to ensure that $\sum_{j=1}^{k} w_j = 1$. $\pi(a,b)$ indicates how a is preferred to b over all the criteria and $\pi(b,a)$ indicates how b is preferred to a over all the criteria. $P_j(a,b)$ and $P_j(b,a)$ are the preference functions for alternatives a and b. The relative indices Dunn, silhouette, and PBM have to be maximized, and Hubert, normalized Hubert, DB, SD, S_Dbw, CS, and C-index have to be minimized.

Step 2. Calculate $\pi(a,b)$ and $\pi(b,a)$ for each pair of alternatives of A. There are six types of preference functions and the decision-maker needs to choose one type of the preference functions for each criterion and the values of the corresponding parameters [49]. The usual preference function, which requires no input parameter, is used for all criteria in the experiment.

Step 3. Define the positive and the negative outranking flow as follows:
The positive outranking flow:

$$\phi^{+}(a) = \frac{1}{n-1} \sum_{x \in A} \pi\,(a, x)\,,$$

The negative outranking flow:

$$\phi^{-}(a) = \frac{1}{n-1} \sum_{x \in A} \pi\,(x, a)\,,$$

Step 4. Compute the net outranking flow for each alternative as follows:

$$\phi(a) = \phi^{+}(a) - \phi^{-}(a).$$

When $\phi(a) > 0$, a is more outranking all the alternatives on all the evaluation criteria. When $\phi(a)<0$, a is more outranked.

MCDM Method 3: Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)

The Technique for order preference by similarity to ideal solution (TOPSIS) method was proposed by Hwang and Yoon [50] to rank alternatives over multiple criteria. It finds the best alternatives by minimizing the distance to the ideal solution and maximizing the distance to the nadir or negative-ideal solution [37]. This section uses the following TOPSIS procedure, which was adopted from [51] and [37], in the empirical study:

Step 1. Calculate the normalized decision matrix. The normalized value r_{ij} is calculated as

$$r_{ij} = x_{ij} / \sqrt{\sum_{i=1}^{J} x_{ij}^2},\ j = 1, \ldots, J;\ i = 1, .., n.$$

Step 2. Develop a set of weights wi for each criterion and calculate the weighted normalized decision matrix. The weighted normalized value vij is calculated as:

$$v_{ij} = w_i r_{ij},\ j = 1, .., J;\ i = 1, .., n.$$

Weight of the ith criterion, and $\sum_{i=1}^{n} w_i = 1$.

Step 3. Find the ideal alternative solution S+, which is calculated as:

$$S^{+} = \left\{v_1^{+}, \ldots, v_n^{+}\right\} = \left\{\left(\max_j v_{ij} | i \in I^{'}\right), \left(\min_j v_{ij} | i \in I^{''}\right)\right\}$$

where I′ is associated with benefit criteria and I′′ is associated with cost criteria. In this study, benefit and cost criteria of TOPSIS are defined the same as the benefit and cost criteria in WSM.

Step 4. Find the negative-ideal alternative solution S2, which is calculated as:

$$S^{-} = \left\{v_1^{-}, \ldots, v_n^{-}\right\} = \left\{\left(\min_j v_{ij} | i \in I^{'}\right), \left(\max_j v_{ij} | i \in I^{''}\right)\right\}$$

Step 5. Calculate the separation measures, using the n-dimensional Euclidean distance. The separation of each alternative from the ideal solution is calculated as:

$$D_j^{+} = \sqrt{\sum_{i=1}^{n} \left(v_{ij} - v_i^{+}\right)^2}, j = 1, \ldots, J.$$

The separation of each alternative from the negative-ideal solution is calculated as:

$$D_j^{-} = \sqrt{\sum_{i=1}^{n} \left(v_{ij} - v_i^{-}\right)^2}, j = 1, \ldots, J.$$

Step 6. Calculate a ratio R_j^{+} that measures the relative closeness to the ideal solution and is calculated as:

$$R_j^{+} = D_j^{-} / \left(D_j^{+} + D_j^{-}\right), j = 1, \ldots, J.$$

Step 7. Rank alternatives by maximizing the ratio R_j^{+}.

2.3.1.2 Clustering Algorithm

The k-means algorithm, the most well-known partitioning method, is an iterative distance-based technique [32]. The input parameter k predefines the number of clusters. First, k objects are randomly chosen to be the centers of these clusters. All objects are then partitioned into k clusters based on the minimum squared-error criterion, which measures the distance between an object and the cluster center. The new mean of each cluster is calculated and the whole process iterates until the cluster centers remain the same [11, 52]. Let X = $\{x_i\}$ be the n objects to be

clustered, C = {$C_1, C_2, \ldots, C_k$} is the set of clusters. Let mi be the mean of cluster C_i. The squared-error between μ_i and the objects in cluster C_i is defined as.

$$WCSS\,(C_i) = \sum_{x_j \in C_i} \left\| x_j - \mu_i \right\|^2$$

Then the aim of k-means algorithm is to minimize the sum of the squared error over all k clusters, that is

$$\min \left(WCSS(C) = \arg\min_{C} \sum_{x_j \in C_i} \left\| x_j - \mu_i \right\|^2 \right.$$

where WCSS denotes the sum of the squared error in the inner-cluster.

Two critical steps of k-means algorithm have impact on the sum of squared error. First, generate a new partition by assigning each observed point to its closest cluster center, the formula is as follows:

$$C_i{}^{(t)} = \left\{ x_j : \left\| x_j - m_i{}^{(t)} \right\| \leq \left\| x_j - m_{i*}{}^{(t)} \right\| \; for all i* = 1, .., k \right\}$$

where $m_i{}^{(t)}$ denotes the mean of the i^{th} cluster in t^{th} times clustering, while $C_i{}^{(t)}$ represents all sets contained in the i^{th} cluster in t^{th} times clustering. Second, compute new cluster mean centers using the following formula.

$$m_i{}^{(t+1)} = \frac{1}{\mid C_i{}^{(t+1)} \mid} \sum_{x_j \in C_i^{(t)}} x_j$$

where $m_i{}^{(t+1)}$ denotes the mean of the i^{th} cluster in $(t+1)^{th}$ times clustering while $C_i{}^{(t+1)}$ represents all sets contained in the i^{th} cluster in $(t+1)^{th}$ times clustering. The algorithm is implemented using WEKA (Waikato Environment for Knowledge Analysis), a free machine learning software [53].

2.3.1.3 Clustering Validity Measures

Ten relative measures are selected for the experiment, namely, the Hubert Γ statistic, the normalized Hubert Γ, the Dunn's index, the Davies-Bouldin index, the CS measure, the SD index, the S_Dbw index, the silhouette index, PBM, and the C-index. Relative measures can also be used to identify the optimal number of clusters in a data set and some of them, such as the C-index and silhouette, have exhibited good performance in previous studies. The following paragraphs define these relative measures.

- **Hubert Γ statistic** [54]:

$$\Gamma = (1/M)\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P(i,j)\cdot Q(i,j)$$

where n is the number of objects in a data set, M = n(n − 1)/2, P is the proximity matrix of the data set, and Q is an n*n matrix whose (i, j) element is equal to the distance between the representative points (v_{ci},v_{cj}) of the clusters where the objects xi and x_j belong [42]. C indicates the agreement between P and Q.

- **Normalized Hubert** Γ:

$$\hat{\Gamma} = \frac{\left[(1/M)\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(P(i,j)-\mu_P)\cdot(Q(i,j)-\mu_Q)\right]}{\sigma_P\sigma_Q}$$

where μ_P, μ_Q, σ_p, and σ_Q represent the respective means and variances of P and Q matrices [43].

Dunn's index [55] evaluates the quality of clusters by measuring inter cluster distance and intra cluster diameter.

$$D = \min_{i=1,\ldots,K}\left\{\min_{j=i+1,\ldots,K}\left[\frac{d(C_i,C_j)}{\max_{l=1,\ldots,K} diam(C_l)}\right]\right\}$$

where K is the number of clusters, C_i is the i^{th} cluster, d(C_i,C_j) is the distance between cluster C_i and C_j, and $diam(C_l)$ is the diameter of the lth cluster. Larger values of D suggest good clusters, and a D larger than 1 indicates compact separated clusters.

- **Davies-Bouldin index is defined as** [56]:

$$DB_K = \frac{1}{K}\sum_{i=1}^{K} R_i,\ R_i \max_{i=1,\ldots,K,i\neq j} R_{ij},\ R_{ij} = \frac{s_i + s_j}{d_{ij}},\ i = 1,\ldots,K$$

where K is the number of clusters, s_i and sj represent the respective dispersion of clusters i and j, dij measures the dissimilarity between two clusters, and R_{ij} measures the similarity between two clusters [42, 43]. It is the average similarity between each cluster and its most similar one.

- **The CS measure is proposed to evaluate clusters with different densities and/or sizes** [57]. It is computed as:

$$CS = \frac{\sum_{i=1}^{K}\left\{\frac{1}{N_i}\sum_{x_j\in C_i}\max_{x_k\in C_i}\left\{d\left(x_j, x_k\right)\right\}\right\}}{\sum_{i=1}^{K}\left\{\min_{j\in\{1,2,\ldots,K\}, j\neq i}\left\{d\left(v_i, v_j\right)\right\}\right\}}, v_i = \frac{1}{N_i}\sum_{x_j\in C_i} x_j$$

where N_i is the number of objects in cluster i and d is a distance function. The smallest CS measure indicates a valid optimal clustering.

- **SD index combines the measurements of average scattering for clusters and total separation between clusters** [42]:

$$SD(K) = Dis\left(c_{\max}\right) \times Scat(K) + Dis(K)$$

where c_{max} is the maximum number of input clusters,

$$Scat(K) = \frac{1}{K}\sum_{i=1}^{K}\|\sigma\left(v_i\right)\| / \|\sigma(X)\| \quad \text{and}$$

$$Dis(K) = \frac{D_{\max}}{D_{\min}}\sum_{k=1}^{K}\left(\sum_{z=1}^{K}\|v_k - v_z\|\right)^{-1}$$

D_{max} is the maximum distance between cluster centers and the D_{min} is the minimum distance between cluster centers.

S_Dbw index is similar to SD index and is defined as [42]:

$$SDbw(K) = Scat(K) + Densbw(K),$$

$$Densbw(K) = \frac{1}{K\cdot(K-1)}\sum_{i=1}^{K}\left(\sum_{\substack{j=1\\ j\neq i}}^{K}\frac{density\left(u_{ij}\right)}{\max\left\{density\left(v_i\right), density\left(v_j\right)\right\}}\right),$$

$$density(u) = \sum_{l=1}^{N_{ij}} f\left(x_l, u\right)$$

where N_{ij} is the number of objects that belong to the cluster C_i and C_j, and function f(x,u) is defined as:

$$f(x,u) = \begin{cases} 0, if \;\; d(x,u) > stdev \\ 1, \; otherwise \end{cases}, stdev = \frac{1}{K}\sqrt{\sum_{i=1}^{K} \|\sigma(v_i)\|}$$

Silhouette is an internal graphic display for clustering methods evaluation. It represents each cluster by a silhouette, which shows how well objects lie within their clusters. It is defined as [58]:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where i represents any object in the data set, a(i) is the average dissimilarity of i to all other objects in the same cluster A, and b(i) is the average dissimilarity of i to all objects in the neighboring cluster B, which is defined as the cluster that has the smallest average dissimilarity of i to all objects in it. Note that A ≠B and the dissimilarity is computed using distance measures. Since a(i) measures how dissimilar i is to its own cluster and b(i) measures how dissimilar i is to its neighboring cluster, an s(i) close to one indicates a good clustering method. The average s(i) of the whole data set measures the quality of clusters.

- **PBM is developed by [40] and it is based on the intra-cluster and inter-cluster distances**:

$$PBM = \left(\frac{1}{K}\frac{E_1}{E_K}D_K\right)^2$$
$$\text{where } E_1 = \sum_{i=1}^{N} \|x_i - \overline{x}\|, E_K = \sum_{l=1}^{N}\sum_{x_i \in C_l} \|x_i - \overline{x}_l\|,$$
$$D_K = \max_{l,m=1,\ldots,K} \|\overline{x}_l - \overline{x}_m\|$$

The C-index [59] is based on intra-cluster distances and their maximum and minimum possible values [60]:

$$CI = \frac{\theta - \min\theta}{\max\theta - \min\theta}, \theta = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{i,j} \left\|x_i - x_j\right\|$$

2.3.2 *Parallel Regularized Multiple Criteria Linear Programming Classification Algorithm*

In this section, the focus is on the RMCLP, and the designed and proposed Parallel version of RMCLP algorithm (PRMCLP). In order to overcome the compute and

storage requirements that increase rapidly with the number of training sample, the second strategy is adopted, inspire by some findings in [61].

Let us give a brief introduction of MCLP as follows. For classification of the training data:

$$T = \{(x_1, y_1), \ldots, (x_l, y_l)\} \in \left(\Re^n \times y\right)^l \tag{2.151}$$

where $x_i \in \Re^n$, $y_i \in y = \{1, -1\}$, $i = 1, \ldots, l$, data separation can be achieved by two opposite objectives. The first objective separates the observations by minimizing the sum of the deviations (MSD) among the observations. The second maximizes the minimum distances (MMD) of observations from the critical value [62]. The overlapping of data $\xi^{(1)}$ should be minimized while the distance $\xi^{(2)}$ has to be maximized. However, it is difficult for traditional linear programming to optimize MMD and MSD simultaneously. According to the concept of Pareto optimality, we can seek the best trade-off between the two measurements [10, 63]. So MCLP model can be described as follows:

$$\min e^{\mathrm{T}}\xi^{(1)} \,\&\, \max e^{\mathrm{T}}\xi^{(2)} \tag{2.152}$$

$$s.t.\,(w \cdot x_i) + \left(\xi_i{}^{(1)} - \xi_i{}^{(2)}\right) = b,\ for\ \{i|y_i = 1\}, \tag{2.153}$$

$$(w \cdot x_i) - \left(\xi_i{}^{(1)} - \xi_i{}^{(2)}\right) = b,\ for\ \{i|y_i = -1\}, \tag{2.154}$$

$$\xi^{(1)}, \xi^{(2)} \geq 0 \tag{2.155}$$

where $e \in \mathrm{R}^l$ be vector whose all elements are 1, w and b are unrestricted, $\xi_i^{(1)}$ is the overlapping and $\xi_i^{(2)}$ the distance from the training sample x_i to the discriminator $(w \cdot x_i) = b$ (classification separating hyperplane). By introducing penalty parameter C,D > 0, MCLP has the following version

$$\min_{\xi_i{}^{(1)}, \xi_i{}^{(2)}} Ce^{\mathrm{T}}\xi^{(1)} - De^{\mathrm{T}}\xi^{(2)}, <?pag?> \tag{2.156}$$

$$s.t.\,(w \cdot x_i) + \left(\xi_i{}^{(1)} - \xi_i{}^{(2)}\right) = b,\ for\ \{i|y_i = 1\}, \tag{2.157}$$

$$(w \cdot x_i) - \left(\xi_i{}^{(1)} - \xi_i{}^{(2)}\right) = b,\ for\ \{i|y_i = -1\}, \tag{2.158}$$

$$\xi^{(1)}, \xi^{(2)} \geq 0 \tag{2.159}$$

A lot of empirical studies have shown that MCLP is a powerful tool for classification. However, we cannot ensure that this model always has a solution under different kinds of training samples. To ensure the existence of solution, recently, Shi et al. proposed a RMCLP model by adding two regularized items $\frac{1}{2}\omega^{\mathrm{T}} H\omega$ and $\frac{1}{2}\xi^{(1)\mathrm{T}} Q\xi^{(1)}$ in MCLP as follows (more theoretical explanation of this

model can be found in [63]):

$$\min_{z} \frac{1}{2} w^{\mathrm{T}} H w + \frac{1}{2} \xi^{(1)\mathrm{T}} Q \xi^{(1)} + \frac{1}{2} b^2 + C e^{\mathrm{T}} \xi^{(1)} - D e^{\mathrm{T}} \xi^{(2)}, \tag{2.160}$$

$$s.t. \, (w \cdot x_i) + \left(\xi_i{}^{(1)} - \xi_i{}^{(2)} \right) = b, \; for \, \{ i | y_i = 1 \}, \tag{2.161}$$

$$(w \cdot x_i) - \left(\xi_i{}^{(1)} - \xi_i{}^{(2)} \right) = b, \; for \, \{ i | y_i = -1 \}, \tag{2.162}$$

$$\xi^{(1)}, \xi^{(2)} \geq 0 \tag{2.163}$$

where $z = (w^{\mathrm{T}}, \xi^{(1)T}, \xi^{(2)T}, b)^{\mathrm{T}} \in R^{n+l+l+1}$, $H \in R^{n \times n}$ is symmetric positive definite matrices. Obviously, the regularized MCLP is a convex quadratic programming. According to the dual theorem, (2.160)–(2.163) can be formulated as:

$$\min_{\alpha, \xi^{(1)}} \frac{1}{2} \alpha^{\mathrm{T}} \left(K \left(A, A^{\mathrm{T}} \right) + e e^{\mathrm{T}} \right) \alpha + \frac{1}{2} \xi^{(1)\mathrm{T}} Q \xi^{(1)}, \tag{2.164}$$

$$s.t. - Q \xi^{(1)} - C e \leq E \alpha \leq -D e, \tag{2.165}$$

where $A = \left[x_1^{\mathrm{T}}, \ldots, x_l^{\mathrm{T}} \right]^{\mathrm{T}} \in R^{l \times n}$, $E = \mathrm{diag} \, \{ y_1, \ldots, y_l \}$
and
$K \left(A, A^{\mathrm{T}} \right) = \Phi(A) \Phi(A)^{\mathrm{T}} = \left(\Phi(A) \cdot \Phi(A)^{\mathrm{T}} \right)_{l \times l}$

and Φ is a mapping from the input space Rn to some Hilbert space H [64].

In order to realize the parallelization of RMCLP, we firstly translate RMCLP into a unconstrained optimization problem. To simplify, (2.164) can be rewritten as

$$\begin{aligned} &\min_{\pi} \tfrac{1}{2} \pi^{\mathrm{T}} \Lambda \pi, \\ &s.t. G \pi - C e \leq 0, \\ &H \pi + D e \leq 0, \end{aligned} \tag{2.166}$$

where $\pi = [\alpha^{\mathrm{T}}, \xi^{(1)\mathrm{T}}]^{\mathrm{T}}$, and $G = [-Q, -E]$, $H = [E, O]$, $O \in R^{l \times l}$ is a null matrix, Λ is written as

$$\begin{pmatrix} K \left(A, A^{\mathrm{T}} \right) + e e^{\mathrm{T}} & 0 \\ 0 & Q \end{pmatrix} \tag{2.167}$$

Next, we represent the objective (2.164) as the following unconstrained optimization problem

$$\min_{\pi} f(\pi) = \frac{1}{2} \pi^{\mathrm{T}} \Lambda \pi + \lambda^{\mathrm{T}} \max \{ G \pi - C e, 0 \}^2 + \mu \max \{ H \pi + D e, 0 \}^2 \tag{2.168}$$

where C,D ∈R are the artificial parameters, and $\lambda = \{\lambda_1, \ldots, \lambda_l\}$, $\mu = \{\mu_1, \ldots, \mu_l\}$.

Define d is the search direction of the optimization problem (2.168), here, we choose the negative gradient direction as the feasible direction:

$$d = -\nabla f(\pi) / \|\nabla f(\pi)\| \tag{2.169}$$

where

$$\nabla f(\pi) = \Lambda\pi + 2\lambda^{\mathrm{T}}\mathrm{diag}\left(G^{\mathrm{T}}\max\{G\pi - Ce, 0\}\right) + 2\mu^{\mathrm{T}}\mathrm{diag}\left(H^{\mathrm{T}}\max\{H\pi + De, 0\}\right) \tag{2.170}$$

Now, we use PVD idea to split our model [61]. Suppose we can use p processors, the variable of the unconstrained optimization problem (2.168) can be divided into p chunks: {1, …, p}, where the dimension of the i^{th} chunk is mi

$$\pi = \{\pi_1, \ldots, \pi_m\}, \pi_i \in R^{m_i}, i = 1, \ldots, p, \sum_{i=1}^{p} m_i = 2l \tag{2.171}$$

In the next step, we allocate the p-th variable to p-th processor, and decompose the problem (2.168) into the subproblem with mi dimensions. Each processor solves one corresponding subproblem, which update other variables on the basis of some rules except for computing the mi variables itself. After each processor finishes updating, a quick synchronous step is performed: searching the results obtained by each processor and computing the current solution. Repeating then this, our algorithm can be described as

Theorem 2.4 *The sequence generated by* $\{\pi^k\}$ *of Algorithm 2.4 either terminates at a stationary point* $\{\pi^k\}$*, or is an infinite sequence, whose accumulation point is stationary and* $\lim_{k\to\infty} \nabla f\left(\pi^k\right) = 0$.

Proof

For $\forall \pi, \pi^{'} \in R^{2l}$, we have

$$\nabla f(\pi) = \Lambda\pi + 2\lambda^{\mathrm{T}}\mathrm{diag}\left(G^{\mathrm{T}}\max\{G\pi - Ce, 0\}\right) + 2\mu^{\mathrm{T}}\mathrm{diag}\left(H^{\mathrm{T}}\max\{H\pi + De, 0\}\right)$$

So

(2.172)

$$\begin{aligned}&\left\|\nabla f(\pi) - \nabla f\left(\pi^{'}\right)\right\| = \left\|\Lambda\left(\pi - \pi^{'}\right) + 2\lambda^{\mathrm{T}}\mathrm{diag}\left(G^{\mathrm{T}}(\max\{G\pi - Ce, 0\} - \max\{G\pi - Ce, 0\})\right)\right.\\ &\left. + 2\mu^{\mathrm{T}}\mathrm{diag}\left(H^{\mathrm{T}}\max\{H\pi + De, 0\} - \max\{H\pi + De, 0\}\right)\right\|\\ &\le \|\Lambda\|\left\|\pi - \pi^{'}\right\| + 2\left\|\lambda^{\mathrm{T}}\right\|\left\|\mathrm{diag}\left(G^{\mathrm{T}}(\max\{G\pi - Ce, 0\} - \max\{G\pi - Ce, 0\})\right)\right\|\\ &+ 2\left\|\mu^{\mathrm{T}}\right\|\left\|\mathrm{diag}\left(H^{\mathrm{T}}\max\{H\pi + De, 0\} - \max\{H\pi + De, 0\}\right)\right)\right\|\end{aligned} \tag{2.173}$$

i) For any $G\pi_i, G\pi'_i \le Ce, where i = 1, \ldots, m$, we have

$$\left\|\text{diag}\left(G^{\mathrm{T}}\left(\max\{G\pi - Ce, 0\} - \max\{G\pi - Ce, 0\}\right)\right)\right\| = 0 \le \left\|G^{\mathrm{T}}G\left(\pi_i - \pi_i^{'}\right)\right\| \tag{2.174}$$

ii) For any $G\pi_i, G\pi'_i > Ce, where i = 1, \ldots, m$, we have

$$\left\|\text{diag}\left(G^{\mathrm{T}}\left(\max\{G\pi - Ce, 0\} - \max\{G\pi - Ce, 0\}\right)\right)\right\| = \left\|G^{\mathrm{T}}G\left(\pi_i - \pi_i^{'}\right)\right\| \tag{2.175}$$

Taken together, we can obtain

$$\begin{aligned}&\left\|\text{diag}\left(G^{\mathrm{T}}\left(\max\{G\pi - Ce, 0\} - \max\left\{G\pi^{'} - Ce, 0\right\}\right)\right)\right\| \\ &\le \left\|G^{\mathrm{T}}G\left(\pi - \pi^{'}\right)\right\| \le \left\|G^{\mathrm{T}}\right\| \|G\| \left(\pi - \pi^{'}\right)\right\|\end{aligned} \tag{2.176}$$

Similarly, we have

$$\begin{aligned}&\left\|\text{diag}\left(H^{\mathrm{T}}\left(\max\{H\pi - De, 0\} - \max\left\{H\pi^{'} - De, 0\right\}\right)\right)\right\| \\ &\le \left\|H^{\mathrm{T}}\left(\pi - \pi^{'}\right)\right\| \le \left\|H^{\mathrm{T}}\right\| \|H\| \left(\pi - \pi^{'}\right)\right\|\end{aligned} \tag{2.177}$$

As the result, let $\|\Lambda\| + 2\|\Lambda\| \left\|G^{\mathrm{T}}\right\| \|G\| + 2\|\mu\| \left\|H^{\mathrm{T}}\right\| \|\mathrm{H}\| = K$,
we can obtain

$$\left\|\nabla f(\pi) - \nabla f\left(\pi^{'}\right)\right\| \le \left\|\pi - \pi^{'}\right\| \tag{2.178}$$

According to the Theorem 2.2 in [19], $\{\pi^k\}$ either terminates at a stationary point $\left\{\pi^{\overline{k}}\right\}$, or is an infinite sequence, whose accumulation point is stationary and $\lim_{k\to\infty} \nabla f\left(\pi^k\right) = 0$.

Theorem 2.5 *If A of Algorithm 2.4 is positive definite, then the sequence of iterates* $\{\pi^k\}$ *generated by the subproblem of (2.168) converges linearly to the unique solution* $\overline{\pi}$*, and the rate of convergence is*

$$\left\|\pi^k - \overline{\pi}\right\| \le \left(\frac{2}{\gamma}\left(f\left(\pi^k\right) - f(\overline{\pi})\right)\right)^{\frac{1}{2}} \left(1 - \frac{1}{p}\left(\frac{\gamma}{K}\right)^2\right)^{\frac{1}{2}}, \tag{2.179}$$

where $\gamma, K > 0$ *are constants.*

Proof For

$$\begin{aligned}&\forall \pi, \pi^{'} \in R^{2l}\\&\left(\nabla f(\pi)-\nabla f\left(\pi^{'}\right)\right)\left(\pi-\pi^{'}\right)=(\pi-\pi')^{\mathrm{T}} \Lambda\left(\pi-\pi^{'}\right)+\left(2 \lambda^{\mathrm{T}} \operatorname{diag}\left(G^{\mathrm{T}}\left(\max\{G \pi-C e, 0\}\right.\right.\right.\\&\left.-\max\{G \pi-C e, 0\}\right))+2 \mu^{\mathrm{T}} \operatorname{diag}\left(H^{\mathrm{T}} \max\{H \pi+D e, 0\}\right.\\&\left.\left.\left.-\max\{H \pi+D e, 0\}\right)\right)\right)\left(\pi-\pi^{'}\right)\end{aligned} \tag{2.180}$$

It is known that

$$\begin{aligned}&\operatorname{diag}\left(G^{\mathrm{T}}(\max\{G \pi-C e, 0\}-\max\{G \pi-C e, 0\})\right)\left(\pi-\pi^{'}\right) \geq 0,\\&\operatorname{diag}\left(G^{\mathrm{T}}(\max\{G \pi-C e, 0\}-\max\{G \pi-C e, 0\})\right)\left(\pi-\pi^{'}\right) \geq 0\end{aligned} \tag{2.181}$$

Since Λ is a positive definite matrix, we have

$$\begin{aligned}&\left(\nabla f(\pi)-\nabla f\left(\pi^{'}\right)\right)\left(\pi-\pi^{'}\right) \geq(\pi-\pi')^{\mathrm{T}} \Lambda\left(\pi-\pi^{'}\right) \geq \frac{\gamma}{2}\|\pi-\pi'\|^{2},\\&\forall \pi \in R^{2l}\end{aligned} \tag{2.182}$$

where γ is a constant. As a result, subproblem of (2.168) converges linearly to the unique solution $\overline{\pi}$, and the rate of convergence is

$$\left\|\pi^{k}-\overline{\pi}\right\| \leq\left(\frac{2}{\gamma}\left(f\left(\pi^{k}\right)-f(\overline{\pi})\right)\right)^{\frac{1}{2}}\left(1-\frac{1}{p}\left(\frac{\gamma}{K}\right)^{2}\right)^{\frac{1}{2}} \tag{2.183}$$

2.3.3 *An Effective Intrusion Detection Framework Based on Multiple Criteria Linear Programming and Support Vector Machine*

The main contributions of this section include the following:

(a) Modifications to the chaos particle swarm optimization have been proposed by adopting the time-varying inertia weight factor (TVIW) and time-varying acceleration coefficients (TVAC), namely TVCPSO, to make it faster in searching for the optimum and avoid the search being trapped into local optimum.
(b) A weighted objective function that simultaneously takes into account trade-off between the maximizing the detection rate and minimizing the false alarm rate, along with considering the number of features is proposed to eliminate the redundant and irrelevant features, as long as increase the attacks' detection rate.

(c) An extended version of multiple criteria linear programming, namely PMCLP, has been adopted to increase the performance of this classifier in dealing with the unbalance intrusion detection dataset.
(d) The proposed TVCPSO has been adopted to provide an effective IDS framework by determining parameters and selecting a subset of features for multiple criteria linear programming and support vector machines.

In the recent years, biology inspired approaches has been used to solve complex problems in a variety of domains such as computer science, medicine, finance and engineering [65]. Swarm intelligence considered as an artificial intelligence techniques which inspired from a flock of birds, a school of fish swims or a colony of ants and their unique capability to solve complex problems [65]. Briefly, swarm intelligence (SI) considered as some methodologies, techniques and algorithms inspired by study of collective behaviors in decentralized systems [66]. Particle swarm optimization is one of these techniques, which introduced by Eberhart and Kennedy in 1995 [67]. Particle swarm optimization is a population based meta-heuristic optimization technique that simulates the social behavior of individuals, namely, particles. This technique, compare with the other algorithms in this group has several advantages such as simple to implement, scalability, robustness, quick in finding approximately optimal solutions and flexibility [39].

In particle swarm optimization, each individual of a population that considered as a representative of the potential solution move through an n-dimensional search space. After the initialization of the population, at each iteration particle seeks the optimal solution by changing its direction which consists of its velocity and position according to two factors, its own best previous experience (*pbest*) and the best experience of all particles (*gbest*). Equations (2.184) and (2.185), respectively represents updating the velocity and position of each percale at iteration $[t + 1]$. At the end of each iteration the performance of all particles will be evaluated by predefined fitness functions.

$$\begin{aligned} v^{id}\,[t+1] &= w.v^{id}\;[t] + c_1\, r_1 \left(p^{id,best}\,[t] - x^{id}\,[t]\right) \\ &\quad + c_2\, r_2 \left(p^{gd,best}\,[t] - x^{id}\,[t]\right) \quad \mathrm{d} = 1, 2, \ldots, \mathrm{D} \end{aligned} \tag{2.184}$$

$$x^{id}\,[t+1] = p^{id}\,[t] + v^{id}\,[t+1] \quad \mathrm{d} = 1, 2, \ldots, \mathrm{D} \tag{2.185}$$

Where, $i = 1, 2, \ldots, N$, N is the number of swarm population. In D-dimensional search space, $x^i[t] = \{x^{i1}[t], x^{i2}[t], \ldots, x^{iD}[t]\}$ represent the current position of the i^{th} particle at iteration [t]. Likewise, the velocity vector of each particle at iteration [t] represented by $v^i[t] = \{v^{i1}[t], v^{i2}[t], \ldots, v^{iD}[t]\}$. $p^{i,\,best}[t] = \{p^{i1}[t], p^{i2}[t], \ldots, p^{iD}[t]\}$ represent the best position that particle i has obtained until iteration t, and $p^{g,\,best}[t] = \{p^{g1}[t], p^{g2}[t], \ldots, p^{gD}[t]\}$ represent the previous best position of whole particle until iteration t.

To control the pressure of local and global search, the concept of an inertia weight w was introduced in the PSO algorithm by [68]. r_1 *and* r_2 are two D-dimensional vectors with random number between 0 and 1. c_1 *and* c_2 are positive acceleration coefficients which respectively called cognitive parameter and social parameter. In

fact, these two parameters control the importance of particles' self-learning versus learning from all the swarm's population.

In this research, in order to balance the global exploration and local exploitation, time-varying acceleration coefficients (TVAC) [68, 69] and time-varying inertia weight (TVIW) [69, 70] is adopted to justify the acceleration coefficients and inertia weight, respectively. Both of these concepts help PSO algorithm to have better performance to find the region of global optimum and do not trap in local minima [68, 69, 71].

In TVAC, the acceleration coefficients adjusted by decreasing the value of c_1 from initial value of c_{1i} to c_{1f}, while the value of c_2 is increasing from its initial value of c_{2i} to c_{2f} as shown in Eqs. (2.186) and (2.187). Moreover, in TVIW, the inertia weight w is updated according to the Eq. (2.188), which means a large inertia weight makes PSO has more global search ability at the beginning of the run and by a linearly decreasing the inertia weight makes PSO has better local search.

$$c_1 = c_{1i} + \frac{t}{t_{max}} \left(c_{1f} - c_{1i}\right) \tag{2.186}$$

$$c_2 = c_{2i} + \frac{t}{t_{max}} \left(c_{2f} - c_{2i}\right) \tag{2.187}$$

$$w = w_{max} - \frac{t}{t_{max}} \left(w_{max} - w_{min}\right) \tag{2.188}$$

Here, t represents the current iteration and t_{max} means the maximum number of iterations, c_{1i}, c_{1f},c_{2i}, c_{2f} are the constant values and w_{max}, w_{min} are the predefined maximum and minimum inertia weight.

2.3.3.1 Discrete Binary PSO

Although the original PSO was proposed to act in continuing space, Kennedy and Eberhart [67] proposed the discrete binary version of PSO. In this model particle moves in a state space restricted to zero and one on each dimension, in terms of the changes in probabilities that a bit will be in one state or the other. The formula proposed in Eq. (2.8) remains unchanged except that $x^{id}[t]$, $p^{gd,\,best}[t]$ *and* $p^{id,\,best}[t] \in \{0, 1\}$ and v^{id} restricted to the [0.0, 1.0] [15, 65]. By introducing the sigmoid function, the velocity mapped from a continuous space to probability space as following:

$$sig\left(v^{id}\right) = \frac{1}{1 + e^{(-v^{id})}} \quad \mathrm{d} = 1, 2, \ldots, \mathrm{D} \tag{2.189}$$

The new particle position calculated by using the following rule:

$$x^{id}\,[t+1] = \begin{cases} 1, & if\ rnd\,() < sig\left(v^{id}\right) \\ 0 & if\ rnd\,() \geq sig\left(v^{id}\right) \end{cases}, \mathrm{d} = 1, 2, \ldots, \mathrm{D} \tag{2.190}$$

Where, $sig(v^{id})$ is a sigmoid function and *rnd( )* is a random number in range [0.0, 1.0].

Although traditional PSO gains considerable results in different fields, however, the performance of the PSO depends on the preset parameters and it often suffers the problem of being trapped in local optima. In order to further enhance the search ability of swarm in PSO and avoids the search being trapped in local optimum, chaotic concept has been introduced by [68, 69, 71]. Here, chaos is characterized as ergodicity, randomicity and regularity.

In this section, Logistic equation which is a typical chaotic system adopted to make the chaotic local search as represented in the following:

$$z_{j+1} = \mu z_j \left(1 - z_j\right) \quad j = 1, 2, \ldots m \tag{2.191}$$

Here, by considering n-dimensional vector $z_j = (z_{j1}, z_{j2}, \ldots, z_{jn})$, each component of this system is a random value in the range [0, 1], μ is the control parameter and the system of Eq. (2.15) has been proved to be completely chaotic when $0 \leq z_0 \leq 1$ and $\mu = 4$. Chaos queues $z_1, z_2, z_3, \ldots, z_m$ are generated by iteration of Logistic equation.

In fact, the basic ideas of chaotic are adopted in this section are described as follows:

Chaos initialization: In spite of standard PSO, which particle's position in the search space initialized randomly, here chaos initialization is adopted to better initialize the position of each particle and to increase the diversity of the population.

Chaotic local search (CLS): By using the chaos queues, it helps PSO to does not trapped in a local optimum besides it can cause to search the optimum quickly. It will happen by generating the chaos queues based on the optimal position ($p^{g,\,best}$), and then replace the position of one particle of the population with the best position of the chaos queues.

Although different performance metrics has been proposed to evaluate the effectiveness of IDSs, the most two popular of these metrics are detection rate (DR) and false alarm rate (FAR). By comparing the actual nature of a given record which here "Positive" means an "attack classes" and "Negative" means a "normal record" to the prediction ones, it's possible to consider four outcomes for this situation as shown in Table 2.8, which known as the confusion matrix.

Table 2.8 Confusion matrix

	Test Result Positive (Predicted as an attack)	Test Result Negative (Predicted as a normal record)
Actual Positive Class (Attack record)	True positive (TP)	False negative (FN)
Actual Negative Class (Normal record)	False positive (FP)	True negative (TN)

Here, true positive and true negative means correctly labeled the records as an attack and normal, respectively, that is, IDSs predict the labels perfectly. False positive (FP), refer to normal record is considered as an attack and False negative (FN) means those attack records falsely considered as a normal one.

A well performed IDS should has a high detection rate (DR) as well as low false positive rate. In intrusion detection domain false positive rate typically named false alarm rate (FAR). Thus, the particles with higher detection rate, lower false positive rate and the small number of selected features can produce a high objective function value. Hence, in this research a weighted objective function that simultaneously takes into account trade-off between the maximizing the detection rate and minimizing the false alarm rate, along with considering the number of features is proposed according to the following equation:

$$\begin{aligned}&\text{Objective function } \left(\mathrm{F}_{fit}\right) = \\ &\quad w_{DR}.\left[\frac{TP}{(\mathrm{TP}+\mathrm{FN})}\right] + w_{FAR}.\left[1-\frac{FP}{(\mathrm{FP}+\mathrm{TN})}\right] + w_F.\left[1-\frac{\sum_{i=1}^{nF}\mathrm{f}_i}{n_F}\right]\end{aligned} \tag{2.192}$$

Since any of these three elements of objective function have different effect on the performance of IDS, we convert this multiple criteria problem to a single weighted fitness function that combines the three goals linearly into one. Where w_{DR}, w_{FAR} and w_F represents the importance of detection rate, false alarm rate and number of selected features in the objective function. Detection rate or sensitivity in biomedical informatics terms, known as a true positive rate (TPR), which means the ratio of true positive recognition to the total actual positive class; $\frac{TP}{(\mathrm{TP}+\mathrm{FN})}$. False alarm rate (FAR) or false positive rate (FPR) defined as: $\frac{FP}{(\mathrm{FP}+\mathrm{TP})}$. f_i represents the value of feature mask ("1" represents that feature i is selected and "0" represents that feature i is not selected), and n_F indicates the number of features.

The specific steps of TVCPSO–MCLP and TVCPSO–SVM are described as follows:

Step 1: Chaotic initialization for n + 2 particle, for the MCLP algorithm, the first two parameters are α^* and β^* and for SVM algorithm the first two parameters are c and γ. The rest of n particle is binary features mask of feature sets which here is 41 features of NSL-KDD cup 99 datasets. Here in binary features mask, 1 and 0 adopted to present as selected features and discarded features, respectively.

(a) Initialize a vector $z_0 = (z_{01}, z_{02}, \ldots, z_{0n})$, each component of it is set as a random value in the range [0, 1], and by iteration of Logistic equation a chaos queue $z_1, z_2, \ldots, z_n$ is obtained.

(b) In order to transfer the chaos queue z_j into the parameter's range the following equation is used:

$$\hat{Z}_{jk} = a_k + (b_k - a_k).z_{jk} \qquad (k = 1, 2, \ldots, n) \tag{2.193}$$

Where the value range of each particle defined by $[a_k, b_k]$.

Step 2: Compute the fitness value of the initial vector $\hat{Z}_j$ $(j = 1, 2, \ldots, m)$ and then choose the best M solutions as the initial positions of M particles.

Step 3: Randomly initialize the velocity of M particles, here, $v_j = (v_{j1}, v_{j2}, \ldots., v_{jn})\ j = (1, 2, \ldots, M.)$

Step 4: Update the velocity and position of each classifier's parameters (α^*, β^* in MCLP and c, γ in SVM) according to Eqs. (2.184) and (2.185), and in order to update the velocity and position of the features in each particle Eqs. (2.184) and (2.190) have been used, respectively.

Step 5: Evaluate the fitness of each particle according to Eq. (2.192) and then compare the evaluated fitness value of each particle (personal optimal fitness (pfit)) to its personal best position ($p^{i,\,best}$):

(a) If the pfit is better than $p^{i,\,best}$ then update the $p^{i,\,best}$ as the current position, otherwise keep the previous ones in memory.
(b) If the pfit is better than $p^{g,\,best}$ then update the $p^{g,\,best}$ as the current position, otherwise keep the previous $p^{g,\,best}$.

Step 6: Optimize $p^{g,\,best}$ by chaos local search according to the following steps:

(a) Consider $T = 0$, scale the $p^{gk,\,best}$ into the range of [0,1] by $z_k^T = \frac{p^{gk,best} - a_k}{b_k - a_k}\ (k = 1, 2, \ldots, n)$.
(b) Generate the chaos queues $Z_j^T\ (T = 1, 2, \ldots, m)$ by iteration of Logistic equation.
(c) Obtain the solution set $p = (p^1, p^2, \ldots, p^m)$ by scale the chaotic variables Z_j^T into the decision variable according to the $p_k^T = a_k + (b_k - a_k)\,.z_k^T$.
(d) Evaluate the fitness value of each feasible solution $p = (p^1, p^2, \ldots, p^m)$, and get the best solution $\hat{p}^{g,best}$.

Step 7: If the stopping criteria are satisfied, then stop the algorithms and get the global optimum that are the optimal value of (α^*, β^* in MCLP and c, γ in SVM) and the most appropriate subset of features. Otherwise, go to step 5.

References

1. Sun, D., Liu, L., Zhang, P., Zhu, X., Shi, Y.: Decision rule extraction for regularized multiple criteria linear programming model. Int. J. Data Warehousing Mining. **7**(3), 88–101 (2011)
2. Shi, Y., Tian, Y., Chen, X., Zhang, P.: Regularized multiple criteria linear programs for classification. Sci. China Ser. F Inf. Sci. **52**(10), 1812–1820 (2009)
3. Wang, B., Shi, Y.: Error correction method in classification by using multiple-criteria and multiple-constraint levels linear programming. Int. J. Comput. Commun. Contr. **7**(5), 976–989 (2014)
4. Qi, Z., Tian, Y., Shi, Y.: Multi-instance classification based on regularized multiple criteria linear programming. Neural Comput. Applic. **23**(3), 857–863 (2013)
5. Zhang, P., Tian, Y., Zhang, Z., Shi, Y., Li, X.: Supportive instances for regularized multiple criteria linear programming classification. Int. J. Oper. Quant. Manag. **14**(4), 249–263 (2008)

6. Zhao, X., Shi, Y., Niu, L.: Kernel based simple regularized multiple criteria linear program for binary classification and regression. Intellig. Data Anal. **19**(3), 505–527 (2015)
7. Zhang, D., Tian, Y., Shi, Y.: A group of knowledge-incorporated multiple criteria linear programming classifiers. J. Comput. Appl. Math. **235**(13), 3705–3717 (2011)
8. Peng, Y., Zhang, Y., Kou, G., Shi, Y.: A multicriteria decision making approach for estimating the number of clusters in a data set. PLoS One. **7**(7), e41713 (2012)
9. Qi, Z., Tian, Y., Shi, Y., Alexandrov, V.: Parallel rmclp classification algorithm and its application on the medical data. IEEE Trans. Cloud Comput. (2015). https://doi.org/10.1109/TCC.2015.2481381
10. Shi, Y., Wise, W., Lou, M., et al.: Multiple criteria decision making in credit card portfolio management. In: Multiple Criteria Decision Making in New Millennium, pp. 427–436 (2001)
11. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Practical Machine Learning Tools and Techniques, p. 578. Morgan Kaufmann, Burlington, MA (2005)
12. Qi, Z., Xu, Y., Wang, L., Song, Y.: Online multiple instance boosting for object detection. Neurocomputing. **74**(10), 1769–1775 (2011)
13. Shao, Y., Yang, Z., Wang, X., Deng, N.: Multiple instance twin support vector machines. Lect. Note Oper. Res. **12**, 433–442 (2010)
14. Zhou, Z.: Multi-instance learning: a survey. Department of Computer Science & Technology, Nanjing University, Tech. Rep 2 (2004)
15. Chen, Y., Zhang, L., Shi, Y.: Post mining of multiple criteria linear programming classification model for actionable knowledge in credit card churning management. In: 2011 IEEE 11th International Conference on Data Mining Workshops, pp. 204–211. IEEE, New York (2011)
16. Keeler, J.D., Rumelhart, D.E., Leow, W.K.: Integrated segmentation and recognition of hand-printed numerals. In: Proceedings of the NIPS, pp. 557–563 (1990)
17. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artif. Intell. **89**(1–2), 31–71 (1997)
18. Viola, P., Platt, J., Zhang, C.: Multiple instances boosting for object detection. In: Proceedings of the NIPS, pp. 1417–1424 (2006)
19. Ferris, M.C., Mangasarian, O.L.: Parallel variable distribution. SIAM J. Optim. **4**(4), 815–832 (1994)
20. Shi, Y., Liu, R., Yan, N., Chen, Z.: Multiple criteria mathematical programming and data mining. In: International Conference on Computational Science, pp. 7–17. Springer, New York (2008)
21. Mangasarian, O.L., Wild, E.W.: Multiple instance classification via successive linear programming. J. Optim. Theory Appl. **137**(3), 555–568 (2008)
22. Murphy, P.M., Aha, D.W.: UCI machine learning repository (1992)
23. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple instance learning. In: NIPS, vol. 2, pp. 561–568 (2002)
24. Deng, N., Tian, Y.: Support vector machines: theory, algorithms and extensions. Science Press, Beijing (2009)
25. Zhang, Q., Goldman, S.A.: Em-dd: an improved multiple-instance learning technique. In: Advances in Neural Information Processing Systems, pp. 1073–1080 (2001)
26. Yang, Z.X., Deng, N.: Multi-instance support vector machine based on convex combination. In: The Eighth International Symposium on Operations Research and Its Applications, vol. 481, p. 487. Citeseer (2009)
27. Vapnik, V.N.: The Nature of Statistical Learning Theory, 2nd edn. Springer, New York (2000)
28. Zhang, J., Shi, Y., Zhang, P.: Several multi-criteria programming methods for classification. Comput. Oper. Res. **36**(3), 823–836 (2009)
29. He, J., Shi, Y., Xu, W.: Classifications of credit cardholder behavior by using multiple criteria non-linear programming. In: CASDMKM, pp. 154–163 (2004)
30. Kou, G., Liu, X., Peng, Y., Shi, Y., Wise, M., Xu, W.: Multiple criteria linear programming approach to data mining: models, algorithm designs and software development. Optim. Methods Softw. **18**(4), 453–473 (2003)
31. Yu, P.L.: A class of solutions for group decision problems. Manag. Sci. **19**(8), 936–946 (1973)

32. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, vol. 1, pp. 281–297 (1967)
33. Fung, G., Mangasarian, O.L., Shavlik, J.W.: Knowledge-based support vector machine classifiers. In: NIPS, pp. 521–528. Citeseer (2002)
34. Fung, G., Mangasarian, O.L., Shavlik, J.W.: Knowledge-based nonlinear kernel classifiers. In: Learning Theory and Kernel Machines, pp. 102–113. Springer, New York (2003)
35. Mangasarian, O.L., Wild, E.W.: Nonlinear knowledge in kernel machines. In: Data Mining and Mathematical Programming. Centre de Recherches Mathématiques Montréal Proceedings and & Lecture Notes, pp. 181–198 (2008)
36. Zhang, D., Tian, Y., Shi, Y.: Nonlinear knowledge in kernel-based multiple criteria linear programming classifier. In: Proceedings of the MCDM, pp. 622–629 (2009)
37. Olson, D.L.: Comparison of weights in topsis models. Math. Comput. Model. **40**(7–8), 721–727 (2004)
38. Thomsen, C., Pedersen, T.B.: A survey of open source tools for business intelligence. Int. J. Data Warehousing Mining. **5**(3), 56–75 (2009)
39. Olariu, S., Zomaya, A.Y.: Handbook of Bioinspired Algorithms and Applications. CRC Press, Boca Raton, FL (2005)
40. Pakhira, M.K., Bandyopadhyay, S., Maulik, U.: Validity index for crisp and fuzzy clusters. Pattern Recogn. **37**(3), 487–501 (2004)
41. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. (CSUR). **31**(3), 264–323 (1999)
42. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Cluster validity methods: Part I. ACM SIGMOD Rec. **31**(2), 40–45 (2002)
43. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Clustering validity checking methods: Part II. ACM SIGMOD Rec. **31**(3), 19–27 (2002)
44. Zadeh, L.: Optimality and non-scalar-valued performance criteria. IEEE Trans. Autom. Control. **8**(1), 59–60 (1963)
45. Triantaphyllou, E.: Multi-criteria Decision Making: A Comparative Study. Kluwer Academic Publishers, Dordrecht, The Netherlands (2000)
46. Brans, J.P.: L'ingénierie de la décision: l'élaboration d'instruments d'aide a la décision. Université Laval, Faculté des sciences de l'administration (1982)
47. Brans, J.P., Mareschal, B.: Promethee methods. In: Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 163–186. Springer, New York (2005)
48. Brans, J.: How to decide with promethee. http://www.visualdecision.com/Pdf/How%20to%20use%20PROMETHEE.pdf (1994)
49. Brans, J.P., Vincke, P.: Note—a preference ranking organisation method: (the promethee method for multiple criteria decision-making). Manag. Sci. **31**(6), 647–656 (1985)
50. Hwang, C.L., Yoon, K.: Multiple attribute decision making methods and applications. Springer, Berlin (1981)
51. Opricovic, S., Tzeng, G.H.: Compromise solution by mcdm methods: a comparative analysis of vikor and topsis. Eur. J. Oper. Res. **156**(2), 445–455 (2004)
52. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann, Burlington, MA (2006)
53. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
54. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 4th edn. Academic Press, Cambridge (2008)
55. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybernetics. **3**, 32–57 (1973)
56. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979)
57. Chou, C., Su, M., Lai, E.: A new cluster validity measure and its application to image compression. Pattern. Anal. Applic. **7**(2), 205–220 (2004)

58. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
59. Hubert, L.J., Levin, J.R.: A general statistical framework for assessing categorical clustering in free recall. Psychol. Bull. **83**(6), 1072–1080 (1976)
60. Vendramin, L., Campello, R.J., Hruschka, E.R.: Relative clustering validity criteria: a comparative overview. Statist. Anal. Data Mining. **3**(4), 209–235 (2010)
61. Mangasarian, L.: Parallel gradient distribution in unconstrained optimization. SIAM J. Control. Optim. **33**(6), 1916–1925 (1995)
62. Freed, N., Glover, F.: Evaluating alternative linear programming models to solve the two-group discriminant problem. Decis. Sci. **17**(2), 151–162 (1986)
63. Shi, Y., Peng, Y., Xu, W., Tang, X.: Data mining via multiple criteria linear programming: applications in credit card portfolio management. Int. J. Inf. Technol. Decis. Making. **1**(01), 131–151 (2002)
64. Chen, W., Tian, Y.: Kernel regularized multiple criteria linear programming. In: 3rd International Symposium on Optimization and Systems Biology, pp. 345–352. Citeseer (2009)
65. Kolias, C., Kambourakis, G., Maragoudakis, M.: Swarm intelligence in intrusion detection: a survey. Comput. Secur. **30**(8), 625–642 (2011)
66. Wu, S.X., Banzhaf, W.: The use of computational intelligence in intrusion detection systems: a review. Appl. Soft Comput. **10**(1), 1–35 (2010)
67. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE, New York (1995)
68. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), pp. 69–73. IEEE, New York (1998)
69. Chen, H., Yang, B., Wang, S., Wang, G., Liu, D., Li, H., Liu, W.: Towards an optimal support vector machine classifier using a parallel particle swarm optimization strategy. Appl. Math. Comput. **239**, 180–197 (2014)
70. Huang, C.L., Dun, J.F.: A distributed pso–svm hybrid system with feature selection and parameter optimization. Appl. Soft Comput. **8**(4), 1381–1391 (2008)
71. Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evol. Comput. **8**(3), 240–255 (2004)