symmetry

MDPI

Article

Kernel-Free Quadratic Surface Minimax Probability Machine for a Binary Classification Problem

Yulan Wang, Zhixia Yang * and Xiaomei Yang

College of Mathematics and Systems Science, Xinjiang University, Urumuqi 830046, China; 107551900393@stu.xju.edu.cn (Y.W.); xiaomeiy@xju.edu.cn (X.Y.)
* Correspondence: yangzhx@xju.edu.cn; Tel.: +86-189-9913-4166

Abstract: In this paper, we propose a novel binary classification method called the kernel-free quadratic surface minimax probability machine (QSMPM), that makes use of the kernel-free techniques of the quadratic surface support vector machine (QSSVM) and inherits the advantage of the minimax probability machine (MPM) without any parameters. Specifically, it attempts to find a quadratic hypersurface that separates two classes of samples with maximum probability. However, the optimization problem derived directly was too difficult to solve. Therefore, a nonlinear transformation was introduced to change the quadratic function involved into a linear function. Through such processing, our optimization problem finally became a second-order cone programming problem, which was solved efficiently by an alternate iteration method. It should be pointed out that our method is both kernel-free and parameter-free, making it easy to use. In addition, the quadratic hypersurface obtained by our method was allowed to be any general form of quadratic hypersurface. It has better interpretability than the methods with the kernel function. Finally, in order to demonstrate the geometric interpretation of our QSMPM, five artificial datasets were implemented, including showing the ability to obtain a linear separating hyperplane. Furthermore, numerical experiments on benchmark datasets confirmed that the proposed method had better accuracy and less CPU time than corresponding methods.

Keywords: classification; quadratic surface support vector machine; kernel-free; minimax probability machine; second-order cone programming problem

Citation: Wang, Y.; Yang, Z.; Yang, X. Kernel-Free Quadratic Surface Minimax Probability Machine for a Binary Classification Problem. *Symmetry* **2021**, *13*, 1378. https://doi.org/10.3390/sym13081378

Academic Editor: Jan Awrejcewicz

Received: 1 July 2021
Accepted: 26 July 2021
Published: 28 July 2021

1. Introduction

Machine learning is an important branch in the field of artificial intelligence, which has a wide range of applications in various fields of contemporary science [1]. With the development of machine learning, the classification problem has been widely concerned and studied in the fields of pattern recognition [2], text classification [3], image processing [4], financial time series prediction [5], skin disease [6], intrusion detection systems [7], etc. The classification problem is a vital task in supervised learning that learns a classification rule from a training set with known labels and then uses it to assign a new sample to a class.

At present, there are many famous classification methods. Among these existing methods, Lanckriet et al. [8,9] proposed an excellent classifier, called the minimax probability machine (MPM). For a given binary classification problem, the MPM not only deals with it in the linear case, but also in the nonlinear case by the kernel trick. It is worth noting that the MPM does not have any parameters, which is an important advantage. Therefore, it has been widely used in computer vision [10], engineering technology [11,12], agriculture [13], and novelty detection [14]. Moreover, many researchers have proposed a variety of improved versions of the MPM from different perspectives [14–25]. The representative works can be briefly reviewed as follows. In [15], Thomas and Gregory proposed MPM regression (MPMR), which transformed the regression problem into a classification problem, and then used the classifier MPM to obtain a regression function. To further exploit the structural

information of the training set, Gu et al. [17] proposed the structural MPM (SMPM) by combining the finite mixture models with the MPM. In addition, Yoshiyama et al. [21] proposed the Laplacian MPM (Lap-MPM), which improved the performance of the MPM in semisupervised learning. However, the nonlinear MPM using kernel techniques lacks interpretability and usually depends heavily on the choice of a proper kernel function and the corresponding kernel parameters. Furthermore, choosing the appropriate kernel function and adjusting its parameters may require much computational time and effort. Therefore, it naturally occurs to us that the study of a kernel-free nonlinear MPM is of great significance.

For the first time, Dagher [26] proposed a kernel-free nonlinear classifier, namely the quadratic surface support vector machine (QSSVM), in 2008. It was based on the maximum margin idea, and the training points were separated by a quadratic hypersurface without a kernel function, avoiding the time-consuming process of selecting the appropriate kernel function and its corresponding parameters. Furthermore, in order to improve the classification accuracy and robustness, Luo et al. [27] proposed the soft-margin quadratic surface support vector machine (SQSSVC). After that, Bai et al. [28] proposed the quadratic kernel-free least-squares support vector machine for target diseases' classification. Following these leading works, some scholars performed further studies, e.g., see [29–34] for the classification problem, [35] for the regression problem, and [36] for the cluster problem. The good performance of these methods demonstrates that the quadratic hypersurface is an effective method to flexibly capture the nonlinear structure of data. Thus, it can be seen that it is very interesting to study the kernel-free nonlinear MPM using the above kernel-free technique.

In this paper, for the binary classification problem, a new kernel-free nonlinear method is proposed, which is called the kernel-free quadratic surface minimax probability machine (QSMPM). It was constructed on the basics of the MPM by using the kernel-free techniques of the QSSVM. Specifically, it tries to seek a quadratic hypersurface that separates two classes of samples with maximum probability. However, the optimization problem derived directly was too difficult to solve. Therefore, a nonlinear transformation was introduced to change the quadratic function involved into a linear function. Through such processing, our optimization problem finally became a second-order cone programming problem, which was solved efficiently by an alternate iteration method. It is important to point out that our QSMPM addresses the following key issues. First, our method directly generates a nonlinear (quadratic) hypersurface without the kernel function, so there is no need to select the appropriate kernel. Second, our method does not need to choose any parameters. Third, the quadratic hypersurface obtained by our method has better interpretability than the one by the methods with the kernel function. Fourth, it is rather flexible because the quadratic hypersurface obtained by our method can be any general form of the quadratic hypersurface. In our experiment, the results of five artificial datasets showed that the proposed method can find the general form of the quadratic surface and has also the ability to obtain the linear separating hyperplane. Numerical experiments on 14 benchmark datasets verified that the proposed method was superior to corresponding methods in both accuracy and CPU time. What is more gratifying is that when the number of samples or the dimension is relatively large, our method can obtain good classification performance quickly. In addition, the results of the Friedman test and Nemenyi post-hoc test indicated that our QSMPM was statistically the best one compared to other methods.

The rest of this paper is organized as follows. Section 2 briefly reviews the related works, the QSSVM, and the MPM. Section 3 presents our method QSMPM, gives its algorithm, and analyzes the computational complexity of the QSMPM. In Section 4, we show the interpretability of our method. In Section 5, the results of the numerical experiments on the artificial datasets and benchmark datasets are presented, and a further statistical analysis is performed. Finally, Section 6 gives the conclusion and future work of this paper.

Throughout this paper, we use lower case letters to represent scalars, lower case bold letters to represent vectors, and upper case bold letters to represent matrices. $\mathbb{R}$ denotes

the set of real numbers. $\mathbb{R}^d$ denotes the space of d-dimensional vectors. $\mathbb{R}^{d\times d}$ denotes the space of $d \times d$ matrices. $\mathbb{S}^d$ denotes the set of $d \times d$ symmetric matrices. $\mathbb{S}^d_+$ denotes the set of $d \times d$ symmetric positive semidefinite matrices. $\boldsymbol{I}_d$ denotes the $d \times d$ identity matrix. $\|\boldsymbol{x}\|_2$ denotes the two-norm of the vector $\boldsymbol{x}$.

2. Related Work

In this section, we briefly introduce the QSSVM and the MPM. For a binary classification problem, the training set is given as:

$$T = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_{m_+ + m_-}, y_{m_+ + m_-})\}, \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the i-th sample and $y_i \in \{+1, -1\}$ corresponds to the class label, $i = 1, 2, \ldots, m_+ + m_-$. The number of samples in class +1 and class -1 is m_+ and m_-, respectively. For the training set (1), we want to find a hyperplane or quadratic hypersurface:

$$g(\boldsymbol{x}) = 0, \tag{2}$$

and then use a decision function:

$$f(\boldsymbol{x}) = \text{sign}(g(\boldsymbol{x})) \tag{3}$$

to determine whether a new sample $\boldsymbol{x} \in \mathbb{R}^d$ is assigned to class +1 or class -1.

2.1. Quadratic Surface Support Vector Machine

We first shortly outline the quadratic surface support vector machine (QSSVM) [26]. For the given training set (1), the goal of the QSSVM is to seek a quadratic separating hypersurface:

$$g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} + c = 0, \tag{4}$$

where $\boldsymbol{A} \in \mathbb{S}^d$, $\boldsymbol{b} \in \mathbb{R}^d$, $c \in \mathbb{R}$, which separates the samples into two classes with the largest margin. In order to obtain the quadratic hypersurface (4), the QSSVM establishes the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{b},c} \quad & \sum_{i=1}^{m_+ + m_-} \|\boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}\|_2^2 \\ \text{s.t.} \quad & y_i\left(\frac{1}{2}\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}_i + c\right) \geq 1, i = 1, \ldots, m_+ + m_-. \end{aligned} \tag{5}$$

The optimization problem (5) is a convex quadratic programming problem.

After obtaining the optimal solution $\boldsymbol{A}_*$, $\boldsymbol{b}_*$, and c_* to the optimization problem (5), for a given new sample $\boldsymbol{x} \in \mathbb{R}^d$, its label is assigned to either class +1 or class -1 by the decision function:

$$f(\boldsymbol{x}) = \text{sgn}\left(\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_*\boldsymbol{x} + \boldsymbol{b}_*^{\mathrm{T}}\boldsymbol{x} + c_*\right). \tag{6}$$

To allow some samples in the training set (1) to be misclassified, Luo et al. further proposed the soft-margin quadratic surface support vector machine (SQSSVM); please refer to [27].

2.2. Minimax Probability Machine

Now, we briefly review the minimax probability machine (MPM) [8,9]. Let us leave the training set (1) aside for a moment and suppose that these samples have some distribution. Specifically, assume that the samples in class +1 are drawn from a distribution with the mean vector $\boldsymbol{\mu}_+ \in \mathbb{R}^d$ and the covariance matrix $\boldsymbol{\Sigma}_+ \in \mathbb{S}^d_+$, without making other specific distributional assumptions. A similar assumption is also given for the samples in class -1 with the mean vector $\boldsymbol{\mu}_- \in \mathbb{R}^d$ and the covariance matrix $\boldsymbol{\Sigma}_- \in \mathbb{S}^d_+$. Denote the two

distributions as $\boldsymbol{x}_+ \sim (\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+)$ and $\boldsymbol{x}_- \sim (\boldsymbol{\mu}_-, \boldsymbol{\Sigma}_-)$, respectively. Based on the above assumptions, the MPM attempts to obtain a separating hyperplane:

$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - b = 0, \tag{7}$$

where $\boldsymbol{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$, which separates the two classes of samples with maximal probability with respect to all distributions having these mean vectors and covariance matrices. This is expressed as:

$$\begin{aligned} \max_{\boldsymbol{w},b,\alpha} \quad & \alpha \\ \text{s.t.} \quad & \inf_{\boldsymbol{x}_+ \sim (\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+)} \mathrm{pr}\{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_+ - b \geq 0\} \geq \alpha, \\ & \inf_{\boldsymbol{x}_- \sim (\boldsymbol{\mu}_-, \boldsymbol{\Sigma}_-)} \mathrm{pr}\{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_- - b \leq 0\} \geq \alpha, \end{aligned} \tag{8}$$

where $\alpha \in (0, 1)$ represents the lower bound of the accuracy for future data, namely the worst-case accuracy. The infimum "inf" is taken over all distributions having these mean vectors $\boldsymbol{\mu}_\pm \in \mathbb{R}^d$ and covariance matrices $\boldsymbol{\Sigma}_\pm \in \mathbb{S}_+^d$.

The constraint condition of the above optimization problem (8) is the probabilistic constraint, which is difficult to solve. In order to convert the probabilistic constraints to easy, tractable constraints, the following lemma [9] is given:

Lemma 1 ([9]). *Let $\boldsymbol{x}$ be a d-dimensional random vector with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma} \in \mathbb{S}_+^d$. Given $\boldsymbol{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$, such that $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} \leq b$ and $\alpha \in (0, 1)$, the condition:*

$$\inf_{\boldsymbol{x} \sim (\boldsymbol{\mu}, \boldsymbol{\Sigma})} \mathrm{pr}\{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - b \leq 0\} \geq \alpha \tag{9}$$

holds if and only if:

$$b - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu} \geq \kappa(\alpha)\sqrt{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{w}}, \tag{10}$$

where $\kappa(\alpha) = \sqrt{\frac{\alpha}{1-\alpha}}$.

Using the above Lemma 1, the optimization problem (8) is equivalent to:

$$\begin{aligned} \max_{\boldsymbol{w},b,\alpha} \quad & \alpha \\ \text{s.t.} \quad & -b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_+ \geq \kappa(\alpha)\sqrt{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_+\boldsymbol{w}}, \\ & b - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_- \geq \kappa(\alpha)\sqrt{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_-\boldsymbol{w}}. \end{aligned} \tag{11}$$

Then, through a series of algebraic operations (see Theorem 2 in [9], for the details), the above optimization problem (11) leads to:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \left\| \boldsymbol{\Sigma}_+^{\frac{1}{2}}\boldsymbol{w} \right\|_2 + \left\| \boldsymbol{\Sigma}_-^{\frac{1}{2}}\boldsymbol{w} \right\|_2 \\ \text{s.t.} \quad & \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) = 1. \end{aligned} \tag{12}$$

When its optimal solution $\boldsymbol{w}_*$ is obtained, for the optimization problem (11), the optimal solution with respect to b is given by:

$$b_* = \boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{\mu}_- + \frac{\left\| \boldsymbol{\Sigma}_-^{\frac{1}{2}}\boldsymbol{w}_* \right\|_2}{\left\| \boldsymbol{\Sigma}_+^{\frac{1}{2}}\boldsymbol{w}_* \right\|_2 + \left\| \boldsymbol{\Sigma}_-^{\frac{1}{2}}\boldsymbol{w}_* \right\|_2}, \tag{13}$$

or:

$$b_* = \boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{\mu}_+ - \frac{\left\|\boldsymbol{\Sigma}_+^{\frac{1}{2}}\boldsymbol{w}_*\right\|_2}{\left\|\boldsymbol{\Sigma}_+^{\frac{1}{2}}\boldsymbol{w}_*\right\|_2 + \left\|\boldsymbol{\Sigma}_-^{\frac{1}{2}}\boldsymbol{w}_*\right\|_2}. \tag{14}$$

Now, let us return to the training set (1). It is easy to see that these required mean vectors $\boldsymbol{\mu}_\pm \in \mathbb{R}^d$ and covariance matrices $\boldsymbol{\Sigma}_\pm \in \mathbb{S}_+^d$ are able to be estimated by the training set (1) as follows:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_\pm &= \frac{1}{m_\pm}\sum_{i=1}^{m_\pm}\boldsymbol{x}_i \in \mathbb{R}^d, \\ \hat{\boldsymbol{\Sigma}}_\pm &= \frac{1}{m_\pm}\sum_{i=1}^{m_\pm}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_\pm)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_\pm)^{\mathrm{T}} \in \mathbb{S}_+^d. \end{aligned} \tag{15}$$

Therefore, in practice, these mean vectors $\boldsymbol{\mu}_\pm$ and covariance matrices $\boldsymbol{\Sigma}_\pm$ in (12)–(14) should be replaced by $\hat{\boldsymbol{\mu}}_\pm$ and $\hat{\boldsymbol{\Sigma}}_\pm$, and the optimal solutions of $\boldsymbol{w}$ and b thus obtained are denoted as $\hat{\boldsymbol{w}}_*$ and $\hat{b}_*$. Then, for a given new sample $\boldsymbol{x} \in \mathbb{R}^d$, its label is assigned to either class +1 or class −1 by the decision function:

$$f(\boldsymbol{x}) = \operatorname{sgn}(\hat{\boldsymbol{w}}_*^{\mathrm{T}}\boldsymbol{x} - \hat{b}_*). \tag{16}$$

In addition, for nonlinear cases and more details, please refer to [8,9].

3. Kernel-Free Quadratic Surface Minimax Probability Machine

In this section, we first formulate the kernel-free quadratic surface minimax probability machine (QSMPM). Then, its algorithm is given.

3.1. Optimization Problem

For the binary classification problem with the training set (1), we attempt to find a quadratic separating hypersurface:

$$g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} - c = 0, \tag{17}$$

where $\boldsymbol{A} \in \mathbb{S}^d, \boldsymbol{b} \in \mathbb{R}^d, c \in \mathbb{R}$, which separates the two classes of the samples. Inspired by the MPM, we construct the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{A},\boldsymbol{b},c,\alpha} \quad & \alpha \\ \text{s.t.} \quad & \inf_{\boldsymbol{x}_+\sim(\boldsymbol{\mu}_+,\boldsymbol{\Sigma}_+)} \operatorname{pr}\{\frac{1}{2}\boldsymbol{x}_+^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}_+ + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}_+ - c \geq 0\} \geq \alpha, \\ & \inf_{\boldsymbol{x}_-\sim(\boldsymbol{\mu}_-,\boldsymbol{\Sigma}_-)} \operatorname{pr}\{\frac{1}{2}\boldsymbol{x}_-^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}_- + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}_- - c \leq 0\} \geq \alpha, \end{aligned} \tag{18}$$

where $\alpha \in (0,1)$ represents the lower bound of the accuracy for future data, namely the worst-case accuracy. The notation $\boldsymbol{x}_+ \sim (\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+)$ refers to the class distribution that has the prescribed mean vector $\boldsymbol{\mu}_+ \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma}_+ \in \mathbb{S}_+^d$, but otherwise arbitrary, and likewise for $\boldsymbol{x}_-$.

The above optimization problem (18) corresponds to the optimization problem (8), which is used to derive the optimization problem (11). Analogically, the optimization problem (18) should be used to derive the required optimization problem. Unfortunately, it does not have a counterpart when the functions in curly braces in the optimization problem (18) are quadratic because of the lack of the corresponding Lemma 1. In order

to overcome this difficulty, we change the quadratic functions as a linear function by introducing a nonlinear transformation from $\boldsymbol{x} = ([\boldsymbol{x}]_1, [\boldsymbol{x}]_2, \ldots, [\boldsymbol{x}]_d)^{\mathrm{T}} \in \mathbb{R}^d$ to:

$$\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{x}) = (\frac{1}{2}[\boldsymbol{x}]_1^2, [\boldsymbol{x}]_1[\boldsymbol{x}]_2, \ldots, [\boldsymbol{x}]_1[\boldsymbol{x}]_d, \frac{1}{2}[\boldsymbol{x}]_2^2, \ldots, [\boldsymbol{x}]_2[\boldsymbol{x}]_d, \ldots, \frac{1}{2}[\boldsymbol{x}]_d^2, [\boldsymbol{x}]_1, [\boldsymbol{x}]_2, \ldots, [\boldsymbol{x}]_d)^{\mathrm{T}} \in \mathbb{R}^{\frac{d^2+3d}{2}}. \tag{19}$$

By representing the upper triangular entries of the symmetric matrix:

$$\boldsymbol{A} = \boldsymbol{A}^{\mathrm{T}} = (a_{ij})_{d\times d} \in \mathbb{S}^d \tag{20}$$

as a vector:

$$\boldsymbol{a} = (a_{11}, a_{12}, \ldots, a_{1d}, a_{22}, \ldots, a_{2d}, \ldots, a_{dd})^{\mathrm{T}} \in \mathbb{R}^{\frac{d^2+d}{2}}, \tag{21}$$

and defining:

$$\boldsymbol{w} = (\boldsymbol{a}^{\mathrm{T}}, \boldsymbol{b}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{\frac{d^2+3d}{2}}, \tag{22}$$

the quadratic function (17) of $\boldsymbol{x}$ in d-dimensional space yields the linear function of $\boldsymbol{z}$ in $\frac{d^2+3d}{2}$-dimensional space as follows:

$$g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} - c = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{z} - c. \tag{23}$$

Following the transformation (19), the training set (1) in the d-dimensional space correspondingly becomes:

$$\tilde{T} = \{(\boldsymbol{z}_1, y_1), (\boldsymbol{z}_2, y_2), \cdots, (\boldsymbol{z}_{m_+ + m_-}, y_{m_+ + m_-})\}, \tag{24}$$

where $\boldsymbol{z}_i = \boldsymbol{z}(\boldsymbol{x}_i)$; in other words, $\boldsymbol{z}_i \in \mathbb{R}^{\frac{d^2+3d}{2}}$ is obtained by replacing $\boldsymbol{x}$ in the formula (19) with $\boldsymbol{x}_i = ([\boldsymbol{x}_i]_1, [\boldsymbol{x}_i]_2, \ldots, [\boldsymbol{x}_i]_d)^{\mathrm{T}} \in \mathbb{R}^d$, $i = 1, 2, \cdots, m_+ + m_-$. For the training set (24), it is naturally assumed that the samples of the two classes are sampled from $\boldsymbol{z}_+ \sim (\boldsymbol{\mu}_{\boldsymbol{z}_+}, \boldsymbol{\Sigma}_{\boldsymbol{z}_+})$ and $\boldsymbol{z}_- \sim (\boldsymbol{\mu}_{\boldsymbol{z}_-}, \boldsymbol{\Sigma}_{\boldsymbol{z}_-})$, respectively, where these mean vectors $\boldsymbol{\mu}_{\boldsymbol{z}_\pm} \in \mathbb{R}^{\frac{d^2+3d}{2}}$ and covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{z}_\pm} \in \mathbb{S}_+^{\frac{d^2+3d}{2}}$ can be estimated as:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_\pm} &= \frac{1}{m_\pm}\sum_{i=1}^{m_\pm} \boldsymbol{z}_i \in \mathbb{R}^{\frac{d^2+3d}{2}}, \\ \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_\pm} &= \frac{1}{m_\pm}\sum_{i=1}^{m_\pm} (\boldsymbol{z}_i - \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_\pm})(\boldsymbol{z}_i - \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_\pm})^{\mathrm{T}} \in \mathbb{S}_+^{\frac{d^2+3d}{2}}. \end{aligned} \tag{25}$$

Based on the transformation (19), the optimization problem (18) is replaced by:

$$\begin{aligned} \max_{\boldsymbol{w},c,\alpha} \quad & \alpha \\ \text{s.t.} \quad & \inf_{\boldsymbol{z}_+ \sim (\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_+}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_+})} \mathrm{pr}\{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{z}_+ - c \geq 0\} \geq \alpha, \\ & \inf_{\boldsymbol{z}_- \sim (\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_-}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_-})} \mathrm{pr}\{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{z}_- - c \leq 0\} \geq \alpha. \end{aligned} \tag{26}$$

Now, Lemma 1 [9] is applicable to the optimization problem (26). Thus, we have:

$$\begin{aligned} \max_{\boldsymbol{w},c,\alpha} \quad & \alpha \\ \text{s.t.} \quad & -c + \boldsymbol{w}^{\mathrm{T}}\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_+} \geq \kappa(\alpha)\sqrt{\boldsymbol{w}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_+}\boldsymbol{w}}, \\ & c - \boldsymbol{w}^{\mathrm{T}}\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_-} \geq \kappa(\alpha)\sqrt{\boldsymbol{w}^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_-}\boldsymbol{w}}, \end{aligned} \tag{27}$$

where $\kappa(\alpha) = \sqrt{\frac{\alpha}{1-\alpha}}$. Moreover, a series of algebraic operation shows that the above optimization problem (27) is equivalent to the following second-order cone programming problem:

$$\begin{aligned} \min_{w} \quad & \left\| \hat{\Sigma}_{z_+}^{\frac{1}{2}} w \right\|_2 + \left\| \hat{\Sigma}_{z_-}^{\frac{1}{2}} w \right\|_2 \\ \text{s.t.} \quad & w^{\mathrm{T}} (\hat{\mu}_{z_+} - \hat{\mu}_{z_-}) = 1. \end{aligned} \tag{28}$$

When its optimal solution w_* is obtained, for the optimization problem (27), the optimal solution with respect to c is given by:

$$c_* = w_*^{\mathrm{T}} \hat{\mu}_{z_-} + \frac{\left\| \hat{\Sigma}_{z_-}^{\frac{1}{2}} w_* \right\|_2}{\left\| \hat{\Sigma}_{z_+}^{\frac{1}{2}} w_* \right\|_2 + \left\| \hat{\Sigma}_{z_-}^{\frac{1}{2}} w_* \right\|_2}, \tag{29}$$

or:

$$c_* = w_*^{\mathrm{T}} \hat{\mu}_{z_+} - \frac{\left\| \hat{\Sigma}_{z_+}^{\frac{1}{2}} w_* \right\|_2}{\left\| \hat{\Sigma}_{z_+}^{\frac{1}{2}} w_* \right\|_2 + \left\| \hat{\Sigma}_{z_-}^{\frac{1}{2}} w_* \right\|_2}. \tag{30}$$

In the next subsection, we show how to solve the optimization problem (28).

3.2. Algorithm

Now, we present the solving process of the optimization problem (28), which is achieved by referring to [9]. By constructing an orthogonal matrix $F \in \mathbb{R}^{\frac{d^2+3d}{2} \times \frac{d^2+3d-2}{2}}$ whose columns span the subspace of vectors orthogonal to $\hat{\mu}_{z_+} - \hat{\mu}_{z_-} \in \mathbb{R}^{\frac{d^2+3d}{2}}$, the unknown variable $w \in \mathbb{R}^{\frac{d^2+3d}{2}}$ is converted into $u \in \mathbb{R}^{\frac{d^2+3d-2}{2}}$. Specifically, let $w = w_0 + Fu$, where $w_0 = \frac{(\hat{\mu}_{z_+} - \hat{\mu}_{z_-})}{\left\| \hat{\mu}_{z_+} - \hat{\mu}_{z_-} \right\|_2^2} \in \mathbb{R}^{\frac{d^2+3d}{2}}$; the optimization problem (28) is transferred to the unconstrained optimization problem:

$$\min_{u} \quad \left\| \hat{\Sigma}_{z_+}^{\frac{1}{2}} (w_0 + Fu) \right\|_2 + \left\| \hat{\Sigma}_{z_-}^{\frac{1}{2}} (w_0 + Fu) \right\|_2, \tag{31}$$

In order to solve the above optimization problem (31), Lanckriet et al. [9] introduced two extra variables β and η and considered the following optimization problem:

$$\min_{u,\beta,\eta} \quad \beta + \frac{1}{\beta} \left\| \hat{\Sigma}_{z_+}^{\frac{1}{2}} (w_0 + Fu) \right\|_2^2 + \eta + \frac{1}{\eta} \left\| \hat{\Sigma}_{z_-}^{\frac{1}{2}} (w_0 + Fu) \right\|_2^2. \tag{32}$$

This optimization problem (32) is solved by an alternative iteration. The variables are divided into two sets: one is β and η, and the other is u. At the t-th iteration, first by fixing β and η to take the derivative of the optimization problem (32) with respect to u, we have the following updated iteration formula of u_t:

$$(\frac{1}{\beta_t} P + \frac{1}{\eta_t} Q) u_t = -(\frac{1}{\beta_t} p + \frac{1}{\eta_t} q), \tag{33}$$

where $P = F^{\mathrm{T}} \hat{\Sigma}_{z_+} F \in \mathbb{R}^{\frac{d^2+3d-2}{2} \times \frac{d^2+3d-2}{2}}$, $Q = F^{\mathrm{T}} \hat{\Sigma}_{z_-} F \in \mathbb{R}^{\frac{d^2+3d-2}{2} \times \frac{d^2+3d-2}{2}}$, $p = F^{\mathrm{T}} \hat{\Sigma}_{z_+} w_0 \in \mathbb{R}^{\frac{d^2+3d-2}{2}}$, $q = F^{\mathrm{T}} \hat{\Sigma}_{z_-} w_0 \in \mathbb{R}^{\frac{d^2+3d-2}{2}}$. To ensure the stability, the regularization term $\delta I_{\frac{d^2+3d-2}{2}} (\delta > 0)$ is added. Therefore, the Equation (33) is replaced by:

$$(\frac{1}{\beta_t} P + \frac{1}{\eta_t} Q + \delta I) u_t = -(\frac{1}{\beta_t} p + \frac{1}{\eta_t} q). \tag{34}$$

Next, by fixing $\boldsymbol{u}$ to take the derivative of the optimization problem (32) with respect to β and η, respectively, we have the following updated iteration formula of β_t and η_t:

$$\beta_t = \left\| \hat{\boldsymbol{\Sigma}}_{\boldsymbol{g}_+}^{\frac{1}{2}} (\boldsymbol{w}_0 + \boldsymbol{F}\boldsymbol{u}_t) \right\|_2, \eta_t = \left\| \hat{\boldsymbol{\Sigma}}_{\boldsymbol{g}_-}^{\frac{1}{2}} (\boldsymbol{w}_0 + \boldsymbol{F}\boldsymbol{u}_t) \right\|_2. \tag{35}$$

When the optimal solution $\boldsymbol{u}_*$ is obtained by the above two updated iteration Formulas (34) and (35), the optimal solution $\boldsymbol{w}_*$ of the optimization problem (28) is $\boldsymbol{w}_* = \boldsymbol{w}_0 + \boldsymbol{F}\boldsymbol{u}_*$. Then, we summarize the process of finding the optimal solution $\boldsymbol{A}_*$, $\boldsymbol{b}_*$, c_* of the optimization problem (18) in Algorithm 1.

Algorithm 1 Kernel-free quadratic surface minimax probability machine (QSMPM).

Input: Training set (1), $\delta = 1 \times 10^{-6}$, number of maximum iterations $\tau = 100$.

1: Initialize $\beta_1 = 1$, $\eta_1 = 1$, $t = 1$.
2: Obtain $\boldsymbol{z}_i$ by (19),$i = 1, 2, \ldots, m_+ + m_-$;
3: Calculate $\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_\pm}$ and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_\pm}$ by (25), and calculate $\boldsymbol{w}_0 = \frac{(\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_+} - \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_-})}{\left\| \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_+} - \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_-} \right\|_2^2}$;
4: Calculate $\boldsymbol{P} = \boldsymbol{F}^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_+} \boldsymbol{F}$, $\boldsymbol{Q} = \boldsymbol{F}^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_-} \boldsymbol{F}$, $\boldsymbol{p} = \boldsymbol{F}^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_+} \boldsymbol{w}_0$, $\boldsymbol{q} = \boldsymbol{F}^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{z}_-} \boldsymbol{w}_0$, where $\boldsymbol{F}$ is an orthogonal matrix whose columns span the subspace of vectors orthogonal $\hat{\boldsymbol{\mu}}_{\boldsymbol{z}_+} - \hat{\boldsymbol{\mu}}_{\boldsymbol{z}_-}$;
5: **while** $t < \tau$ **do**
6: Given β_t and η_t, update $\boldsymbol{u}_t$ by (34);
7: Given $\boldsymbol{u}_t$, update β_t and η_t by (35);
8: $t \leftarrow t + 1$.
9: **end**
10: Assign $\boldsymbol{w}_* = \boldsymbol{w}_0 + \boldsymbol{F}\boldsymbol{u}_t$, then obtain $\boldsymbol{A}_*$, $\boldsymbol{b}_*$ by (20), (21), and (22); further, obtain c_* by (29) or (30).

Output: $\boldsymbol{A}_*$, $\boldsymbol{b}_*$, c_*.

After obtaining the optimal solution $\boldsymbol{A}_*$, $\boldsymbol{b}_*$ and c_* to the optimization problem (18), for a given new sample $\boldsymbol{x} \in \mathbb{R}^d$, its label is assigned to either class +1 or class -1 by the decision function:

$$f(\boldsymbol{x}) = \operatorname{sgn}(\frac{1}{2}\boldsymbol{x}^{\mathrm{T}} \boldsymbol{A}_* \boldsymbol{x} + \boldsymbol{b}_*^{\mathrm{T}} \boldsymbol{x} - c_*). \tag{36}$$

It should be pointed out that our QSMPM is kernel-free, which avoids the time-consuming task of selecting the appropriate kernel function and its corresponding parameters. What is more, it does not require any choice of parameter, which makes its use simpler and convenient. Furthermore, from the geometric point of view, the quadratic hypersurface (17) determined by our method is allowed to be any general form of quadratic hypersurface, including hyperplanes, hyperparaboloids, hyperspheres, hyperellipsoids, hyperhyperboloids, and so on, which is shown clearly by five artificial examples in Section 5.

3.3. Computational Complexity

Here, we analyze the computational complexity of our QSMPM. Suppose that the number and the dimension of the samples are N and d, respectively. Before reformulating the QSMPM as an SOCP problem, all d-dimensional samples need to be projected into the $\frac{d^2+3d}{2}$-dimensional space. Therefore, the total computational complexity of the QSMPM is $O((\frac{d^2+3d}{2})^3 + N(\frac{d^2+3d}{2})^2 + Nd^2)$. In addition, we give the computational complexity of the MPM and the SVM. Their complexity is $O(d^3 + Nd^2)$ [9] and $O(N^3)$ [19], respectively. Then, by referencing the computational complexity of SVM, we obtain that the computational complexity of the QSSVM is $O(N^3 + Nd^2)$. According to the above analysis, assuming that N is much larger than d, we can see that the computational complexity of the QSMPM is higher than that of the MPM, but lower than that of the SVM and the QSSVM.

4. The Interpretability

In this section, we discuss the interpretability of our method QSMPM. Suppose we have obtained the optimal solution $\boldsymbol{A}_*$, $\boldsymbol{b}_*$, c_* to the optimization problem (18), then the quadratic hypersurface (17) has the following component form:

$$g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_*\boldsymbol{x} + \boldsymbol{b}_*^{\mathrm{T}}\boldsymbol{x} - c_* = \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij}^*[\boldsymbol{x}]_i[\boldsymbol{x}]_j + \sum_{i=1}^{d} b_i^*[\boldsymbol{x}]_i - c_* = 0, \tag{37}$$

where $[\boldsymbol{x}]_i$ is the i-th component of the vector $\boldsymbol{x} \in \mathbb{R}^d$, a_{ij}^* is the i-th row and the j-th column component of the matrix $\boldsymbol{A}_* \in \mathbb{S}^d$, and b_i^* is the i-th component of the vector $\boldsymbol{b}_* \in \mathbb{R}^d$. Each component of $\boldsymbol{x}$ produces the contribution of a quadratic polynomial function. Specifically, b_i^* is the linear effect coefficient of the i-th component, a_{ii}^* $(i = j)$ is the quadratic effect coefficient of the i-th component, and a_{ij}^* $(i \neq j)$ is the interaction coefficient between the i-th component and the j-th component. Therefore, for the i-th component of $\boldsymbol{x}$, consider that the larger $|a_{ii}^*| + |a_{ij}^*| + |b_i^*|$ $(j = 1, 2, \ldots, d, j \neq i)$, the greater the contribution of the i-th component is. In particular, when $|a_{ii}^*| + |a_{ij}^*| + |b_i^*| = 0$ $(j = 1, 2, \ldots, d, j \neq i)$, the i-th component of $\boldsymbol{x}$ would not work. Therefore, compared with the methods with the kernel function, the QSMPM has better interpretability.

5. Numerical Experiments

In this section, we provide some numerical experiments to verify the performance of our QSMPM. We compared it with the hard-margin support vector machine (H−SVM), the soft-margin support vector machine (S−SVM), and the MPM with the linear kernel, the quadratic polynomial kernel, and the RBF kernel (H−SVM−L, H−SVM−P, H−SVM−R, S−SVM−L, S−SVM−P, S−SVM−R, MPM−L, MPM−P, and MPM−R, respectively). In addition, we also compared it with the QSSVM and the SQSSVM. In all numerical experiments, the penalty parameter C in the S-SVM and the kernel parameter σ of the RBF kernel were selected from $\{2^{-7}, 2^{-6}, \cdots, 2^7\}$ by the 10-fold cross-validation method. All numerical experiments were conducted using MATLAB R2016 (b) on a computer equipped with a 2.50 GHz (I5-4210U) CPU, and 4G available memory.

5.1. Artificial Datasets

To show the geometric interpretation of the proposed method QSMPM and to compare it with the original methods, the MPM−L, the MPM−P, and the MPM−R, we performed the following numerical experiments on 4 artificial examples. These 5 artificial examples were all generated in the 2-dimensional space, and each generated 300 samples $\{\boldsymbol{x}_i = ([\boldsymbol{x}_i]_1, [\boldsymbol{x}_i]_2)^{\mathrm{T}}\}_{i=1}^{300}$, of which the first 150 were samples in class +1 and the last 150 samples in class −1. Here, we first illustrate the symbols in all figures. The red "+" represents the samples in class +1, and the blue "o" represents the samples in class −1. The results in the upper right express the accuracy of each method on the artificial example. The curve in bold black represents the hyperplane or quadratic hypersurface. Now, let us introduce the numerical experiments on each artificial example in turn.

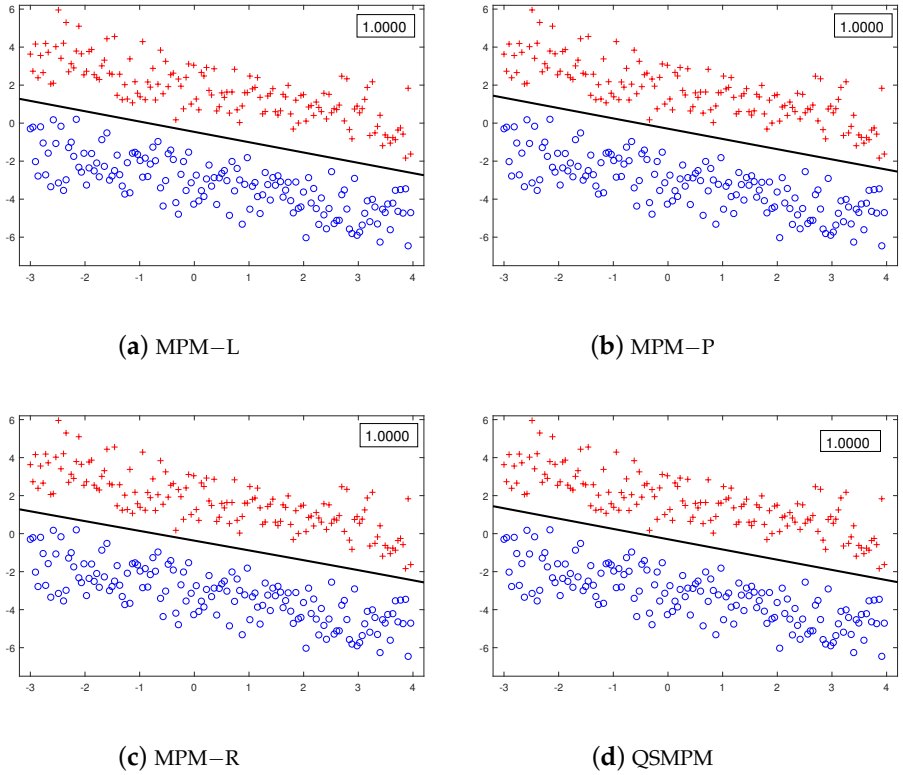
Example 1.

$$[\boldsymbol{x}_i]_2 = -\frac{1}{2}[\boldsymbol{x}_i]_1 + 2 + \xi_i, i = 1, \ldots, 150,$$
$$[\boldsymbol{x}_i]_2 = -\frac{1}{2}[\boldsymbol{x}_i]_1 - 3 + \xi_i, i = 151, \ldots, 300,$$

where $[\boldsymbol{x}_i]_1 \sim U[-3, 4]$, $\xi_i \sim N(0, 1)$.

Figure 1 illustrates the classification results of the MPM−L, the MPM−P, the MPM−R, and the QSMPM on Example 1, respectively. We can see from Figure 1 that our QSMPM can

obtain classification results as good as the other three methods. In addition, the quadratic hypersurface found by our QSMPM is a straight line, that is a linear separating hyperplane.



(**a**) MPM−L (**b**) MPM−P

(**c**) MPM−R (**d**) QSMPM

Figure 1. Classification results on Example 1.

Example 2.

$$[\boldsymbol{x}_i]_2 = \frac{1}{2}[\boldsymbol{x}_i]_1^2 + 2 + \xi_i, i = 1, \ldots, 150,$$
$$[\boldsymbol{x}_i]_2 = \frac{1}{2}[\boldsymbol{x}_i]_1^2 - 3 + \xi_i, i = 151, \ldots, 300,$$

where $[\boldsymbol{x}_i]_1 \sim U[-3, 4]$, $\xi_i \sim N(0, 1)$.

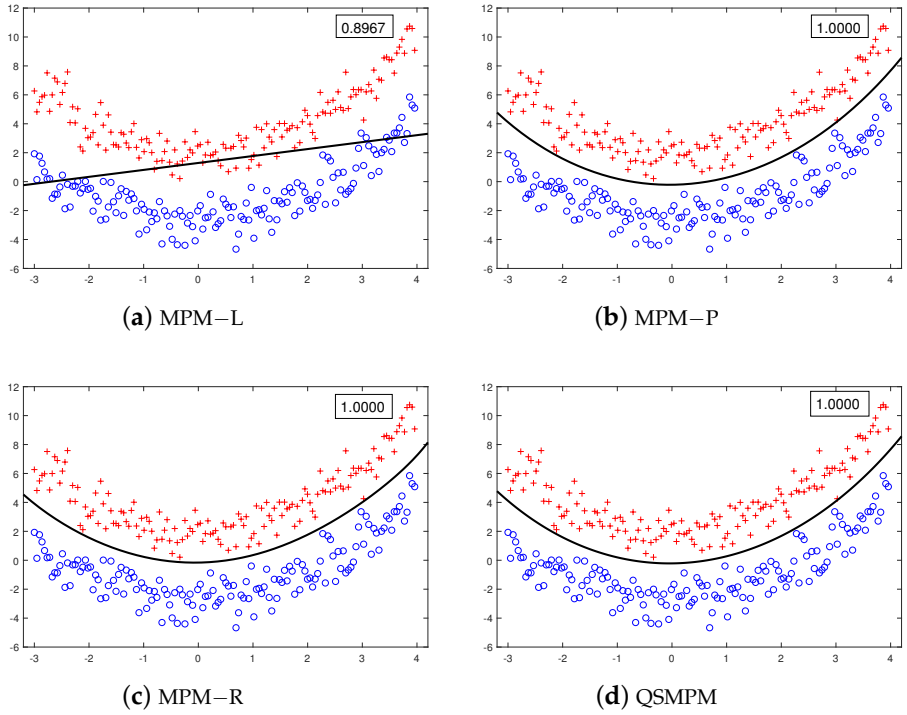
Figure 2 presents the classification results on Example 2. We can observe in Figure 2 that the classification result of our QSMPM is superior to the MPM−L and similar to the MPM−P and the MPM−R. Moreover, the quadratic hypersurface obtained by our QSMPM is a parabola.

Example 3.

$$[\boldsymbol{x}_i]_1 = r_i \cos(t_i), [\boldsymbol{x}_i]_2 = r_i \sin(t_i), i = 1, \ldots, 150,$$
$$[\boldsymbol{x}_i]_1 = 1.3 + r_i \cos(t_i), [\boldsymbol{x}_i]_2 = 1.3 + r_i \sin(t_i), i = 151, \ldots, 300,$$

where $r_i \sim U[0, 1]$, $t_i \sim U[0, 1]$.

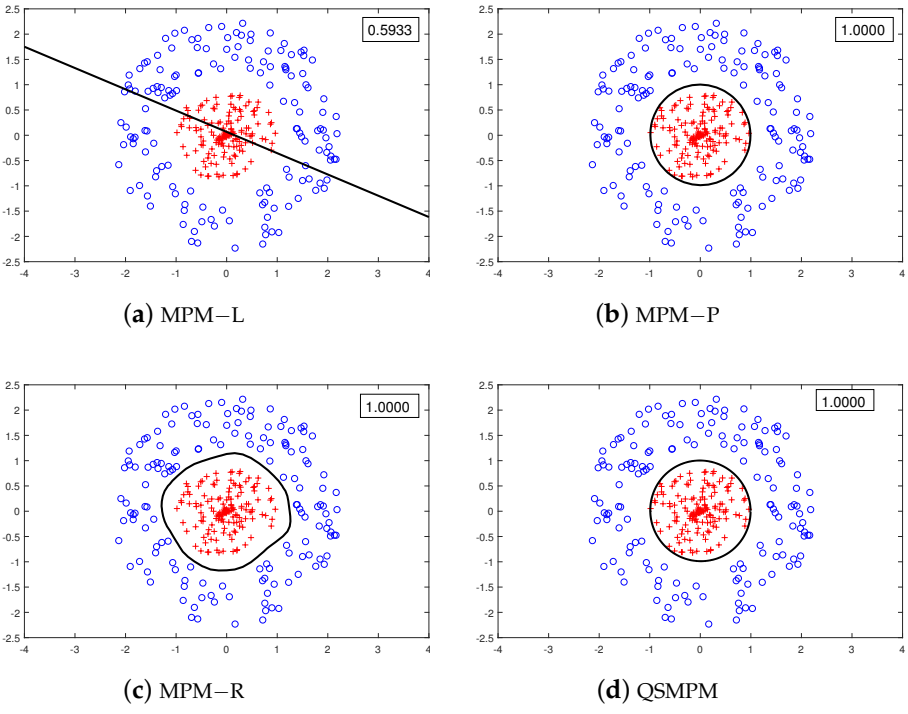
Figure 3 reports the classification results of the MPM−L, the MPM−P, the MPM−R, and the QSMPM on Example 3, respectively. Obviously, in Figure 3, we can see that the classification result of our QSMPM is superior to the MPM−L and is the same as the MPM−P and the MPM−R. Furthermore, the quadratic hypersurface of our QSMPM is a circle.



(**a**) MPM−L (**b**) MPM−P

(**c**) MPM−R (**d**) QSMPM

Figure 2. Classification results on Example 2.



(**a**) MPM−L (**b**) MPM−P

(**c**) MPM−R (**d**) QSMPM

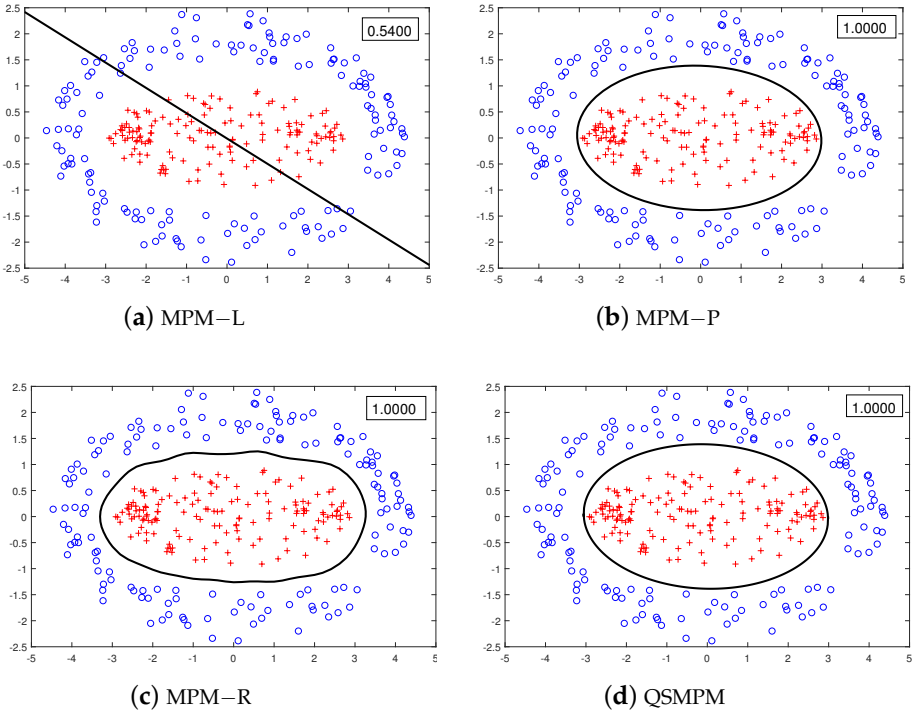
Figure 3. Classification results on Example 3.

Example 4.

$$[x_i]_1 = 2 + a_i \cos(t_i), [x_i]_2 = b_i \sin(t_i), i = 1, \ldots, 150,$$
$$[x_i]_1 = 3.5 + a_i \cos(t_i), [x_i]_2 = 1.5 + b_i \cos(t_i), i = 151, \ldots, 300,$$

where $a_i \sim U[0,1]$, $b_i \sim U[0,1]$, $t_i \sim U[0,1]$.

Figure 4 shows the classification results on Example 4. We can observe in Figure 4 that the QSMPM can obtain the same classification performance as the MPM−P and the MPM−R and is better than the MPM−L. Our QSMPM can find an ellipse.



(**a**) MPM−L (**b**) MPM−P

(**c**) MPM−R (**d**) QSMPM

Figure 4. Classification results on Example 4.

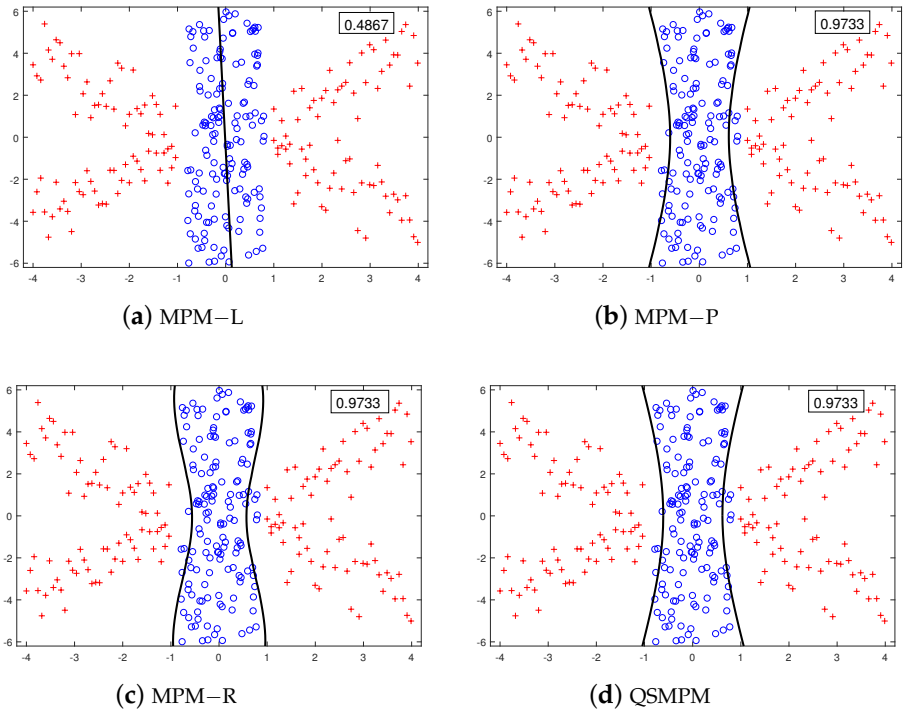
Example 5.

$$[\boldsymbol{x}_i]_2^2 = [\boldsymbol{x}_i]_1^2 - 1 + \xi_i, i = 1, \ldots, 150,$$
$$[\boldsymbol{x}_i]_1 \sim U[-0.6, 0.6], [\boldsymbol{x}_i]_2 \sim U[-6, 6], i = 151, \ldots, 300,$$

where $[\boldsymbol{x}_i]_1 \sim U[-4, -1]$ *and* $U[1, 4]$, $\xi_i \sim N(0, 1)$, $i = 1, \ldots, 150$.

Figure 5 reports the classification results of the MPM−L, the MPM−P, the MPM−R, and the QSMPM on Example 5, respectively. We can observe in Figure 5 that the classification performance of QSMPM is better than the MPM−L and is similar to the MPM−P and the MPM−R. In addition, our QSMPM finds a hyperbola.

In summary, from Figure 1 to Figure 5, we can see that our QSMPM can find any general form of the quadratic hypersurface, such as the line, parabola, circle, ellipse, and hyperbola found in sequence in the above numerical experiments. Moreover, our method can achieve as good classification performance as the MPM−P and the MPM−R. In addition, it can be seen from Figure 1d that our method can obtain the linear separating hyperplane.



(**a**) MPM−L (**b**) MPM−P

(**c**) MPM−R (**d**) QSMPM

Figure 5. Classification results on Example 5.

5.2. Benchmark Datasets

To verify the classification performance and computational efficiency of our QSMPM, we performed the following numerical experiments on 14 benchmark datasets. Table 1 summarizes the basic information of the 14 benchmark datasets in the UCI Machine Learning Repository.

Table 1. Basic information for the 14 benchmark datasets.

Datasets	Samples	Attributes	Class
Iirs	150	4	3
Hepatitis	155	19	2
Wine	178	13	3
Heart	270	13	2
Heart-c	303	14	2
Haberman	306	3	2
Bupa	345	6	2
Pima	768	8	2
QSAR	1055	41	2
Winequality-red	1599	11	6
Wireless	2000	7	4
Image	2310	19	7
Abalone	2649	8	2
Turkiye	5820	32	13

We divided the datasets in Table 1 into two groups for the numerical experiment. The first group was the first seven datasets, and the second group was the last seven datasets. All numerical experimental results were obtained through 10-times 10-fold cross-validation, as well as the numerical experimental results including the mean and standard deviation of accuracy and the CPU time of one experiment. The best results are highlighted in boldface. First of all, Table 2 shows the classification results on the first group.

Table 2. Classification results on the first group.

Methods	Iris	Hepatitis	Wine	Heart	Heart-c	Haberman	Bupa
H−SVM−L	0.6413 ± 0.0245 (4.1293)	0.5835 ± 0.0185 (2.1965)	0.9573 ± 0.0054 (1.8439)	0.6174 ± 0.0287 (7.70490)	**1.0000** ± 0.0000 (3.9203)	0.5189 ± 0.0024 (15.1649)	0.5573 ± 0.0021 (11.2976)
H−SVM−P	0.9447 ± 0.0055 (1.4087)	0.5150 ± 0.0399 (1.5304)	0.7571 ± 0.0365 (2.6005)	0.6985 ± 0.0126 (3.8797)	0.8678 ± 0.0386 (4.5568)	0.7351 ± 0.0005 (10.0121)	0.5798 ± 0.0003 (13.3459)
H−SVM−R	0.9467 ± 0.0000 (0.8250)	0.5858 ± 0.0280 (0.8734)	0.9341 ± 0.0116 (1.0813)	0.7144 ± 0.0183 (2.8344)	**1.0000** ± 0.0000 (2.4438)	0.7362 ± 0.0022 (4.0609)	0.5930 ± 0.0192 (3.7438)
S−SVM−L	0.8693 ± 0.0155 (1.4594)	0.6037 ± 0.0206 (1.3416)	0.9582 ± 0.0147 (2.6656)	0.8259 ± 0.0129 (9.2391)	0.9888 ± 0.0156 (8.2672)	0.7241 ± 0.0073 (7.8422)	0.6812 ± 0.0096 (1 0.7031)
S−SVM−P	0.9653 ± 0.0053 (1.4149)	0.6037 ± 0.0234 (1.4703)	0.7274 ± 0.0235 (1.9407)	**0.8296** ± 0.0097 (9.0609)	0.8605 ± 0.0087 (11.3328)	0.7228 ± 0.0103 (11.1953)	0.6788 ± 0.0095 (12.3328)
S−SVM−R	0.9547 ± 0.0061 (0.9406)	0.5588 ± 0.0183 (0.8656)	0.8921 ± 0.0131 (1.1641)	0.7989 ± 0.0058 (2.1344)	0.7642 ± 0.0101 (2.6000)	0.7261 ± 0.0080 (2.7719)	0.6826 ± 0.0088 (3.2813)
MPM−L	0.8280 ± 0.0082 (0.3073)	0.6010 ± 0.0114 (0.0244)	0.9731 ± 0.0052 (1.7831)	0.8133 ± 0.0080 (3.1887)	**1.0000** ± 0.0000 (1.7379)	0.7177 ± 0.0093 (3.4944)	0.6220 ± 0.0079 (0.1201)
MPM−P	0.9747 ± 0.0028 (1.9079)	0.5954 ± 0.0302 (1.7051)	**0.9759** ± 0.0046 (2.9640)	0.8026 ± 0.0133 (5.6504)	0.9681 ± 0.0066 (6.6425)	0.7159 ± 0.0075 (5.9514)	0.6891 ± 0.0114 (5.5131)
MPM−R	0.9620 ± 0.0063 (4.5256)	0.5382 ± 0.0412 (9.0683)	0.7860 ± 0.0121 (4.3758)	0.6578 ± 0.0105 (10.5238)	0.6981 ± 0.0101 (5.6452)	0.6879 ± 0.0114 (11.9107)	0.7212 ± 0.0088 (8.5723)
QSSVM	0.9533 ± 0.0070 (1.2371)	0.5626 ± 0.0342 (3.7612)	0.9608 ± 0.0052 (2.1109)	0.6989 ± 0.0182 (3.5241)	0.9980 ± 0.0023 (4.2947)	**0.7416** ± 0.0073 (5.6426)	0.7203 ± 0.0047 (8.0918)
SQSSVM	0.9527 ± 0.0049 (0.9266)	0.5650 ± 0.0117 (4.6098)	0.9622 ± 0.0071 (2.6832)	0.7970 ± 0.0098 (5.2822)	0.9954 ± 0.0024 (7.2993)	0.7296 ± 0.0050 (5.0482)	**0.7220** ± 0.0074 (7.0653)
QSMPM	**0.9767** ± 0.0035 (0.3089)	**0.6069** ± 0.0313 (2.0608)	**0.9759** ± 0.0053 (1.3884)	0.8293 ± 0.0114 (0.8580)	**1.0000** ± 0.0000 (2.5179)	0.7205 ± 0.0069 (0.2902)	0.7164 ± 0.0093 (0.2870)

It can be seen from Table 2 that compared with the other methods, our QSMPM obtained better accuracy on the first group of benchmark datasets, among which the accuracy was the best on four benchmark datasets. More specifically, except for Haberman and Bupa, the accuracy of our method was the best compared to the QSSVM and the SQSSVM. The accuracy of our QSMPM was the best compared to the three original kernel versions of the MPM except for Bupa. Furthermore, the accuracy of our method was the best compared to the H−SVM and the S−SVM with three kernel function except for Heart and Haberman. In addition, we can observe that QSMPM had a short CPU time.

Then, the classification results on the second group are reported in Table 3. The symbol "−" indicates that the corresponding method cannot obtain the classification results, because it cannot choose the optimal parameter in a limited amount of time or because the dimension and the number of the dataset are relatively large, resulting in insufficient memory.

Table 3. Classification results on the second group.

Methods	Pima	QSAR	Winequality-Red	Wireless	Image	Abalone	Turkiye
H−SVM−L	0.6799 ± 0.0035 (128.9255)	0.3675 ± 0.0014 (223.9172)	0.6156 ± 0.0014 (813.7250)	0.7280 ± 0.0094 (3956.2000)	0.6470 ± 0.0011 (947.0836)	0.7357 ± 0.0040 (2825.5000)	0.5051 ± 0.0005 (3013.1000)
H−SVM−P	0.5710 ± 0.0155 (75.1160)	0.8133 ± 0.0081 (451.2469)	0.4653 ± 0.0000 (475.2321)	0.8463 ± 0.0234 (4449.2000)	0.6979 ± 0.0020 (996.2086)	0.4934 ± 0.0000 (1429.9000)	0.5033 ± 0.0038 (23897.0000)
H−SVM−R	0.6919 ± 0.0117 (14.9234)	0.8171 ± 0.0069 (110.5313)	**0.7628** ± 0.0058 (3707.5000)	**0.9877** ± 0.0010 (404.7438)	0.9698 ± 0.0019 (567.4688)	0.8067 ± 0.0062 (693.4125)	– –
S−SVM−L	**0.7669** ± 0.0076 (82.0609)	0.8340 ± 0.0152 (211.0188)	0.7291 ± 0.0023 (640.9000)	0.9139 ± 0.0044) (1793.9000)	0.7531 ± 0.0304 (2595.7000)	0.8108 ± 0.0009 (1189.7000)	– –
S−SVM−P	0.7585 ± 0.0066 (27.8571)	**0.8677** ± 0.0058 (259.1922)	0.7492 ± 0.0021 (1123.8000)	0.9799 ± 0.0017 (3120.7000)	0.9602 ± 0.0015 (4820.8000)	0.8298 ± 0.0018 (3120.7000)	– –
S−SVM−R	0.6941 ± 0.0078 (16.5313)	0.8503 ± 0.0031 (103.9438)	0.7352 ± 0.0024 (294.0688)	0.9853 ± 0.0054 (717.2984)	**0.9715** ± 0.0020 (506.2984)	0.8242 ± 0.0008 (984.2250)	– –
MPM−L	0.7405 ± 0.0048 (0.1014)	0.8296 ± 0.0039 (0.8377)	0.7409 ± 0.0025 (0.2683)	0.9108 ± 0.0008 (0.1560)	0.8555 ± 0.0012 (0.2969)	0.8144 ± 0.0009 (0.1266)	0.5779 ± 0.0026 (0.3167)
MPM−P	0.7442 ± 0.0035 (32.9178)	0.8225 ± 0.0061 (68.1350)	0.7326 ± 0.0020 (122.7891)	0.9361 ± 0.0013 (187.2266)	0.8499 ± 0.0011 (257.7266)	0.8109 ± 0.0009 (413.9861)	0.5780 ± 0.0014 (2669.2000)
MPM−R	0.7356 ± 0.0054 (37.2874)	0.8373 ± 0.0037 (83.8022)	0.7234 ± 0.0020 (139.5734)	0.9850 ± 0.0006 (248.8594)	0.9413 ± 0.0018 (402.6281)	0.8264 ± 0.0014 (423.6125)	0.5689 ± 0.0016 (2386.7000)
QSSVM	0.7663 ± 0.0049 (70.7730)	– –	0.4722 ± 0.0032 (27.7094)	0.6315 ± 0.0078 (14.7250)	0.5714 ± 0.0000 (59.1266)	0.5125 ± 0.0012 (29.5344)	– –
SQSSVM	0.7589 ± 0.0036 (43.0610)	– –	0.7452 ± 0.0036 (36.8234)	0.9782 ± 0.0012 (45.3036)	0.5714 ± 0.0000 (63.7109)	0.8280 ± 0.0015 (50.1641)	– –
QSMPM	0.7530 ± 0.0049 (0.5585)	0.8482 ± 0.0047 (16.6328)	0.7470 ± 0.0027 (1.3525)	0.9427 ± 0.0012 (1.0608)	0.8731 ± 0.0013 (3.6953)	**0.8299** ± 0.0011 (1.1984)	**0.5852** ± 0.0018 (14.6360)

From Table 3, we can see that our QSMPM had good classification results on the second group of benchmark datasets. Especially on QSAR and Turkiye, the H−SVM−R, the three kernel versions of the S−SVM, the QSSVM, and the SQSSVM could not obtain the corresponding classification results, but our QSMPM could obtain good classification performance. Here, we mention the reason for this situation. According to the computational complexity of each method, we know that when the sample dimension and the number of samples are relatively large, the SVM and the QSSVM need a larger memory space. In addition, our QSMPM had the fastest running time except the MPM−L, and it ran quite fast when the number of samples or the dimension was large.

5.3. Statistical Analysis

To further compare the performance of the above 12 methods, the Friedman test and the post-hoc test were performed. The ranks of the 12 methods on all benchmark datasets is shown in Table 4.

First, the Friedman test was used to compare the average ranks of different methods. The null hypothesis states that all methods have the same performance, that is their average ranks are the same. Based on the average ranks displayed in Table 4, we can calculate the Friedman statistic τ_F by the following formula:

$$\begin{aligned} \tau_{\chi^2} &= \frac{12N}{k(k+1)}\left(\sum_{i=1}^{k} r_i - \frac{k(k+1)^2}{4}\right), \\ \tau_F &= \frac{(N-1)\tau_{\chi^2}}{N(k-1)-\tau_{\chi^2}}, \end{aligned} \tag{38}$$

where N and k are the number of datasets and methods, respectively. r_i is the average rank of the i-th method. According to the formula (38), $\tau_F = 4.1825$. For $\alpha = 0.05$, we can obtain $F_\alpha = 1.8526$. Since $\tau_F > F_\alpha$, we rejected the null hypothesis. Then, we proceeded with a post-hoc test (the Nemenyi test) to find out which methods significantly differed. To be more specific, the performance of two methods was considered to be significantly different if the difference of their average ranks was larger than the critical difference (CD). The CD can be calculated by:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}. \tag{39}$$

For α=0.05, we know $q_\alpha = 3.2680$. Thus, we obtained CD = 4.4535 by the formula (39).

Figure 6 visually displays the results of the Friedman test and Nemenyi post-hoc test, where the average ranks of each method are marked along an axis. The axis is turned so that the lowest (best) ranks are to the right. Groups of methods that are not significantly different are linked by a red line. In Figure 6, we can see that our QSMPM was the best one statistically among the compared methods. Furthermore, there was no significant difference in performance between the QSMPM and the MPM−R.

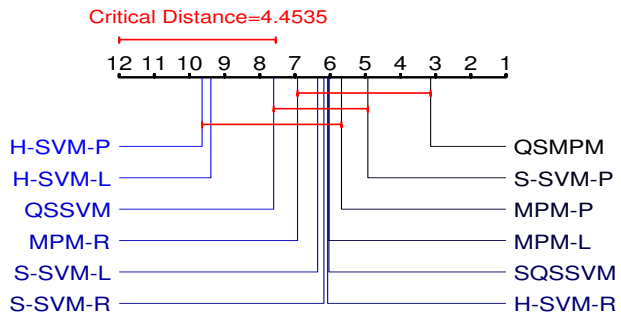


Figure 6. Results of the Friedman test and the Nemenyi post-hoc test.

Table 4. Ranks of accuracy.

Datasets	**H−SVM−L**	**H−SVM−P**	**H−SVM−R**	**S−SVM−L**	**S−SVM−P**	**S−SVM−R**	**MPM−L**	**MPM−P**	**MPM−R**	**QSSVM**	**SQSSVM**	**QSMPM**
Iirs	12	9	8	10	3	5	11	2	4	6	7	1
Hepatitis	7	12	6	2.5	2.5	10	4	5	11	9	8	1
Wine	7	11	8	6	12	9	3	1.5	10	5	4	1.5
Heart	12	10	8	3	1	6	4	5	11	9	7	2
Heart-c	2.5	9	2.5	7	10	11	2.5	8	12	5	6	2.5
Haberman	12	3	2	6	7	5	9	10	11	1	4	8
Bupa Liver	12	11	10	7	8	6	9	5	2	3	1	4
Pima	11	12	10	1	4	9	7	6	8	2	3	5
QSAR	10	9	8	5	1	2	6	7	4	11.5	11.5	3
Winequality-red	10	12	1	8	2	6	5	7	9	11	4	3
Wireless	11	10	1	8	4	2	9	7	3	12	5	6
Image Segmentation	10	9	2	8	3	1	6	7	4	11.5	11.5	5
Abalone	10	12	9	8	2	5	6	7	4	11	3	1
Turkiye	5	6	9.5	9.5	9.5	9.5	3	2	4	9.5	9.5	1
Average ranks	9.3929	9.6429	6.0714	6.3571	4.9286	6.1786	6.0357	5.6786	6.9286	7.6071	6.0357	3.1429

6. Conclusions

For the binary classification problem, a new classifier, called the kernel-free quadratic surface minimax probability machine (QSMPM), was proposed by using the kernel-free techniques of the QSSVM and the classification idea of the MPM. Specifically, our goal was to find a quadratic hypersurface that separates two classes of samples with maximum probability. However, the optimization problem derived directly was too difficult to solve. Therefore, a nonlinear transformation was introduced to change the quadratic function involved into a linear function. Through such processing, our optimization problem finally became a second-order cone programming problem, which was solved efficiently by an alternate iteration method. Here, we clarify the main contributions of this paper. Unlike the methods realizing nonlinear separation, our method was kernel-free and had better interpretability. Then, our method was easy to use because it did not have any parameters. Furthermore, numerical experiments on five artificial datasets showed that the quadratic hypersurfaces found by our method were rather general, including that it could obtain the linear separating hyperplane. In addition, numerical experiments on benchmark datasets confirmed that the proposed method was superior to some relevant methods in both accuracy and computational time. Especially when the number of samples or dimension was relatively large, our method could also quickly obtain good classification performance. Finally, the results of the statistical analysis showed that our QSMPM was statistically the best one compared with the corresponding methods. Our QSMPM focuses on the standard binary classification problem, which we will extend to the multiclassification problem.

In our future work, there will be some issues to be address to extend our QSMPM. For example, we need to investigate further how to add appropriate regularization terms to our method. Meanwhile, we need to consider that the worst-case accuracies for two classes are not the same, and that will be interesting. Furthermore, we will pay attention to how the QSMPM achieves the dual purpose of feature selection and classification simultaneously. In addition, we can apply our method to practical problems in many fields in the future, especially image recognition in the medical field.

Author Contributions: Conceptualization, X.Y.; Data curation, Y.W.; Formal analysis, X.Y.; Methodology, Y.W. and Z.Y.; Software, Y.W.; Supervision, Z.Y.; Writing—original draft, Y.W.; Writing—review and editing, Z.Y. and X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Xinjiang Provincial Natural Science Foundation of China (No. 2020D01C028) and the National Natural Science Foundation of China (No. 12061071).

Data Availability Statement: All of the benchmark datasets used in our numerical experiments are from the UCI Machine Learning Repository, which are available at http://archive.ics.uci.edu/ml/ (accessed on 10 June 2021).

Conflicts of Interest: The authors declare that they have no conflict of interest regarding this work.

References

1. Langley, P.; Simon, H.A. Applications of machine learning and rule induction. *Commun. ACM* **1995**, *38*, 54–64. [CrossRef]
2. Vapnik, V.; Sterin, A. On structural risk minimization or overall risk in a problem of pattern recognition. *Autom. Remote Control* **1977**, *10*, 1495–1503.
3. Forman, G. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* **2003**, *3*, 1289–1305.
4. Ying, S.H.; Qiao, H. Lie group method: A new approach to image matching with arbitrary orientations. *Int. J. Imaging Syst. Technol.* **2010**, *20*, 245–252. [CrossRef]
5. Cao, L.J.; Tay, F. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Trans. Neural. Netw.* **2003**, *14*, 1506–1518. [CrossRef] [PubMed]
6. Srinivasu, P.N.; Sivasai, J.G.; Ijaz, M.F.; Bhoi, A.K.; Kim, W.; Kang, J.J. Classification of skin disease using deep learning neural networks with MobileNet V2 and LSTM. *Sensors* **2021**, *21*, 2852. [CrossRef] [PubMed]
7. Panigrahi, R.; Borah, S.; Bhoi, A.K.; Ijaz, M.F.; Pramanik, M.; Jhaveri, R.H.; Chowdhary, C.L. Performance assessment of supervised classifiers for designing intrusion detection systems: A comprehensive review and recommendations for future research. *Mathematics* **2021**, *9*, 690. [CrossRef]
8. Lanckriet, G.R.G.; Ghaoui, L.E.; Bhattacharyya, C.; Jordan, M.I. Minimax probability machine. *Adv. Neural Inf. Process. Syst.* **2001**, *37*, 192–200.

9. Lanckriet, G.R.G.; Ghaoui, L.E.; Bhattacharyya, C.; Jordan, M.I. A robust minimax approach to classification. *J. Mach. Learn. Res.* **2003**, *3*, 555–582.
10. Johnny, K.C.; Zhong, Y.; Yang, S. A comparative study of minimax probability machine-based approaches for face recognition. *Pattern Recognit. Lett.* **2007**, *28*, 1995–2002.
11. Deng, Z.; Wang, S.; Chung, F.L. A minimax probabilistic approach to feature transformation for multi-class data. *Appl. Soft Comput.* **2013**, *13*, 116–127. [CrossRef]
12. Jiang, B.; Guo, Z.; Zhu, Q.; Huang, G. Dynamic minimax probability machine based approach for fault diagnosis using pairwise discriminate analysis. *IEEE Trans. Control Syst. Technol.* **2017**, *27*, 806–813. [CrossRef]
13. Yang, L.; Gao, Y.; Sun, Q. A new minimax probabilistic approach and its application in recognition the purity of hybrid seeds. *Comput. Model. Eng. Sci.* **2015**, *104*, 493–506.
14. Kwok, J.T.; Tsang, W.H.; Zurada, J.M. A Class of Single-Class Minimax Probability Machines for Novelty Detection. *IEEE. Trans. Neural Netw.* **2007**, *18*, 778–785. [CrossRef]
15. Strohmann, T.; Grudic, G.Z. A formulation for minimax probability machine regression. *Adv. Neural Inf. Process. Syst.* **2003**, *76*, 9–776.
16. Huang, k.; Yang, H.; King, I.; Lyu, M.R.; Chan, L. The minimum error minimax probability machine. *J. Mach. Learn. Res.* **2004**, *5*, 1253–1286.
17. Gu, B.; Sun, X.; Sheng, V.S. Structural minimax probability machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *1*, 1646–1656. [CrossRef] [PubMed]
18. Maldonado, S.; Carrasco, M.; Lopez, J. Regularized minimax probability machine. *Knowl. Based Syst.* **2019**, *177*, 127–135. [CrossRef]
19. Yang, L.M.; Wen, Y.K.; Zhag, M.; Wang, X. Twin minimax probability machine for pattern classification. *Neural Netw.* **2020**, *131*, 201–214. [CrossRef]
20. Cousins, S.; Shawe-Taylor, J. High-probability minimax probability machines. *Mach. Learn.* **2017**, *106*, 863–886. [CrossRef]
21. Yoshiyama, K.; Sakurai, A. Laplacian minimax probability machine. *Pattern Recognit. Lett.* **2014**, *37*, 192–200. [CrossRef]
22. He, K.X.; Zhong, M.Y.; Du, W.L. Weighted incremental minimax probability machine-based method for quality prediction in gasoline blending process. *Chemometr. Intell. Lab. Syst.* **2019**, *196*, 103909. [CrossRef]
23. Ma, J.; Yang, L.M.; Wen, Y.K.; Sun, Q. Twin minimax probability extreme learning machine for pattern recognition. *Knowl. Based Syst.* **2020**, *187*, 104806. [CrossRef]
24. Ma, J.; Shen J.M. A novel twin minimax probability machine for classification and regression. *Knowl. Based Syst.* **2020**, *196*, 105703. [CrossRef]
25. Deng, Z.H.; Chen, J.Y.; Zhang, T.; Cao, L.B.; Wang, S.T. Generalized hidden-mapping minimax probability machine for the training and reliability learning of several classical intelligent models. *Inf. Sci.* **2018**, *436–437*, 302–319. [CrossRef]
26. Dagher, I. Quadratic kernel-free non-linear support vector machine. *J. Glob. Optim.* **2008**, *41*, 15–30. [CrossRef]
27. Luo, J.; Fang, S.C.; Deng, Z.B.; Guo, X.L. Soft quadratic surface support vector machine for binary classification. *Asia Pac. J. Oper. Res.* **2016**, *33*, 1650046. [CrossRef]
28. Bai, Y.Q.; Han, X.; Chen, T.; Yu, H. Quadratic kernel-free least squares support vector machine for target diseases classification. *J. Comb. Optim.* **2015**, *30*, 850–870. [CrossRef]
29. Gao, Q.Q.; Bai, Y.R.; Zhan, Y.R. Quadratic kernel-free least square twin support vector machine for binary classification problems. *J. Oper. Res. Soc. China.* **2019**, *7*, 539–559. [CrossRef]
30. Mousavi, A.; Gao, Z.M.; Han, L.S.; Lim, A. Quadratic surface support vector machine with L1 norm regularization. *J. Ind. Manag. Optim.* **2019**. [CrossRef]
31. Gao, Z.M.; Fang, S.C.; Luo, J.; Medhin, N. A kernel-free double well potential support vector machine with applications. *Eur. J. Oper. Re.* **2020**, *290*, 248–262. [CrossRef]
32. Luo, J.; Fang, S.C.; Bai, Y.Q.; Deng, Z.B. Fuzzy quadratic surface support vector machine based on fisher discriminant analysis. *J. Ind. Mangn. Optim.* **2015**, *12*, 357–373. [CrossRef]
33. Yan, X.; Bai, Y.Q.; Fang, S.C.; Luo, J. A kernel-free quadratic surface support vector machine for semi-supervised learning. *J. Oper. Res. Soc.* **2016**, *67*, 1001–1011. [CrossRef]
34. Tian, Y.; Yong, Z.Y.; Luo, J. A new approach for reject inference in credit scoring using kernel-free fuzzy quadratic surface support vector machines. *Appl. Soft Comput.* **2018**, *73*, 96–105. [CrossRef]
35. Zhai, Q.R.; Tian, Y.; Zhou, J.Y. Linear twin quadratic surface support vector regression. *Math. Probl. Eng.* **2020**, *2020*, 1–18. [CrossRef]
36. Luo, J.; Tian, Y.; Yan, X. Clustering via fuzzy one-class quadratic surface support vector machine. *Soft Comput.* **2017**, *21*, 5859–5865. [CrossRef]