Available online at www.ijpe-online.com

**International Journal of Performability Engineering**

vol. 14, no. 5, May 2018, pp. 985-994
DOI: 10.23940/ijpe.18.05.p17.985994

# Classification Decision based on a Hybrid Method of Weighted *k*NN and Hyper-Sphere SVM

Peng Chen[a,b], Guoyou Shi[a], Shuang Liu[c,*], Yuanqiang Zhang[a], and Denis Špelič[d]

[a]*Navigation College, Dalian Maritime University, Dalian, 116026, China*
[b]*Department of Software Engineering, Dalian Neusoft University of Information, Dalian, 116030, China*
[c]*School of Computer Science & Engineering, Dalian Minzu University, Dalian, 116605, China*
[d]*Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, SI-2000, Slovenia*

**Abstract**

Hyper-sphere Support Vector Machine (SVM) is very effective for solving multi-class classification problems. Considering data distribution is very important for convergence of solving support vectors, a weight factor is imported into the original hyper-sphere SVM. After computing data for each training class, this weight factor is decided by its center-distance ratio. In the training process, data with bigger weight is put into the data processing thread first and is then followed by smaller ones. To save computation cost, a parallel genetic algorithm based SMO multi-threading is adopted. For a test sample, its class decision is based on its position with each classification of hyper-sphere. If all class-specific hyper-spheres are independent of each other, a new test sample can be classified correctly. But, if some hyper-spheres have common spaces, that is, one hyper-sphere intersects with one or more hyper-spheres, it is hard to decide the class of the test sample. Based on detailed analysis of three decision rules for the intersection data classification, one decision rule that combines the *k*NN method is put forward in this paper. For other simple inclusion cases, the simple decision rule is defined. Through two real experimental results of navigation tracking and ship meeting situations classification, our new proposed algorithm has a higher classification accuracy and boasts a lower computation cost than other algorithms.

*Keywords*: hyper-sphere SVM; weight factor; *k*NN; decision rule; multi-threading

(Submitted on February 8, 2018; Revised on March 12, 2018; Accepted on April 23, 2018)



## 1. Introduction

In the field of machine learning, support vector machines have been widely used in many pattern recognition applications, such as image classification [2], face recognition [4], target tracking [10] and so on. The purpose of the original support vector machine is to solve binary classification problems. But, there are many multi-class classification problems in real applications. Many researchers have studied how to extend binary SVM to multi-class classification problems, such as one-versus-one, one-versus-rest, directed acyclic graph (DAG), tree SVM and etc. Although these extended SVMs are successful in some cases, computation cost is still very high, especially for large amounts of data needed to be classified. The basic reason is its computation cost of solving quadratic programming. To solve the problem of high computation complexity, a hyper-sphere support vector machine is put forward in [11]. Based on one-class support vector machine, the basic idea of the hyper-sphere SVM is to compute the minimum bounding sphere for data that belongs to each class. The classification sphere is restricted with a minimum radius and contains as much data as possible of this class. Solving traditional SVM is to compute the optimal hyper-plane for each of the two classes of training samples. For hyper-sphere SVM, the optimal hyper-plane changes into hyper-spheres for multi-classes. Without computing all hyper-planes for each of the two classes, hyper-sphere SVM computes all restricting hyper-spheres directly. So, its computation complexity is less than that that of traditional combination binary SVM methods.

* Corresponding author.
*E-mail address*: liushuang@dlnu.edu.cn

Different training samples have different contributions to the resulting hyper-sphere [6]. We focus on distinguishing important samples from non-significant samples. To solve quadratic programming faster, samples with more contributions for the bounding sphere are chosen first. The contribution is based on a weight factor. This weight factor is decided by its center-distance ratio for each training class. In the training process, data with bigger weight is put into the data processing thread first, which is then followed by smaller ones. To save computation cost, a parallel genetic algorithm based SMO multi-threading is adopted in our algorithm [7,8].

If all class-specific hyper-spheres are independent of each other, a new test sample can be classified correctly. But, if one hyper-sphere intersects with one or more hyper-spheres, the decision for the new test sample is hard to say. There are three highly effective methods to solve this problem. The first one is the same or different error hyper-spheres rule, which is based on the same error data hyper-sphere and different error data hyper-sphere [9]. The second one is the multiple sub-hyper-spheres, which partitions the intersection into two or more sub-hyper-spheres with more restrictions [6]. The third one is the linear & nonlinear decision rule, which separates data by the separation hyper-plane or a new binary SVM hyper-plane [1]. Based on detailed analysis of these three decision rules for the intersection data classification, one improved decision rule that combines the *k*NN method is put forward in this paper.

The rest of this paper is organized as follows. An overview of basic hyper-sphere SVM and our weighted hyper-sphere SVM are given in Section 2. Based on detailed analysis of three decision rules for hyper-sphere intersections, a new decision rule combining the *k*NN method is put forward in Section 3. Two real applications of the proposed algorithm are discussed in Section 4. The conclusions are given in Section 5.

## 2. Introduction of Weighted Hyper-sphere SVM

### *2.1. Principles of Basic Hyper-sphere SVM*

Suppose there are *k*-class training samples in $n$ dimensional spaces and each class has $l_k$ samples, which can be denoted as $X_k = \{x_i \in R^n, i = 1,\mathrm{L}\ l_k\}(k = 1,\mathrm{L}\ ,m)$. Class identifier for each class is added in the following format as Equation (1).

$$X = \{(x_i, y_i), x_i \in R^n, y_i \in \{1,\mathrm{L}\ ,m\}\} \tag{1}$$

Finding the minimum bounding hyper-sphere of each class which that encloses all the training examples of that class can be computed by solving the following constrained quadratic optimization problem in Equation (2).

$$\begin{aligned} &\min_{c_k, R_k} \ R_k^2 + C\sum_{i=1}^{l_k} \xi_i \\ &s.t. \\ &\left\|\phi(\boldsymbol{x}_i) - c_k\right\|^2 \le R_k^2 + \xi_i \\ &\xi_i \ge 0, \ \ i = 1,\mathrm{L}\ ,l_k \end{aligned} \tag{2}$$

Here, a nonlinear mapping $\phi$ is introduced to transform the training samples into a high-dimensional feature space for solving nonlinear separation problems, and then the hyper-sphere in the feature space is computed. The minimum bounding hyper-sphere $S_k$ for class *k* is characterized by its center $c_k$ and radius $R_k$. $C$ is the penalty factor and $\xi_i \ge 0$ are slack variables. By introducing Lagrange multipliers $\alpha_i, \beta_i$, Lagrange polynomial can be written as Equation (3).

$$L(R_k, c_k, \xi_i, \alpha_i, \beta_i) = \ R_k^2 + C\sum_{i=1}^{l_k} \xi_i - \sum_{i=1}^{l_k} \alpha_i (R_k^2 + \xi_i - \left\|\phi(x_i) - c_k\right\|^2 - \sum_{i=1}^{l_k} \beta_i \xi_i \tag{3}$$

Taking the partial directives of *L* with respect to $R_k, c_k, \xi_i$ and substituting them back into Equation (3), the original optimization problem becomes a dual optimization problem in the following format as Equation (4).

$$\begin{aligned} &\min_{\alpha_i} \sum_{i,j=1}^{l_k} \alpha_i \alpha_j \mathrm{K}(x_i, x_j) - \sum_{i=1}^{l_k} \alpha_i \mathrm{K}(x_i, x_i) \\ &s.t. \quad \sum_{i=1}^{l_k} \alpha_i = 1 \\ &\quad 0 \le \alpha_i \le C, i = 1, \mathrm{L}\ , l_k \end{aligned} \tag{4}$$

The kernel trick is adopted to avoid computing the nonlinear map function. Inner products are computed in the feature space, that is, $\mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$. Lagrange multipliers can be solved by the dual quadratic programming problem. Support vectors are the vectors $\boldsymbol{x}_i$ with $\alpha_i > 0$. The center $c_k$ can be computed by the above deduction process, that is $c_k^2 = \sum_{i=1,j}^{l_k} \alpha_i \alpha_j \mathrm{K}(x_i, x_j)$ and the radius $R_k$ can be computed by equating $f_k(\boldsymbol{x})$ to zero for any support vector. So, the resulting decision function for a new sample $\boldsymbol{x}$ can be computed as Equation (5).

$$f_k(\boldsymbol{x}) = \mathrm{sgn}(R_k^{\ 2} - \sum_{i,j=1}^{l_k} \alpha_i \alpha_j \mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) + 2\sum_{i=1}^{l_k} \alpha_i \mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}) - \mathrm{K}(\boldsymbol{x}, \boldsymbol{x})) \tag{5}$$

Based on Equation (5), the new point $\boldsymbol{x}$ falls inside of the hyper-sphere if $f_k(\boldsymbol{x}) > 0$. $\boldsymbol{x}$ falls outside of the hyper-sphere if $f_k(\boldsymbol{x}) < 0$ and $\boldsymbol{x}$ lies on the hyper-sphere if $f_k(\boldsymbol{x}) = 0$. In 2D cases, resulting hyper-spheres are circles. In 3D cases, resulting hyper-spheres are like bubbles, which are shown in Figure 1. This example has five independent spheres. Each sphere can accurately classify all the data that belongs to its class.

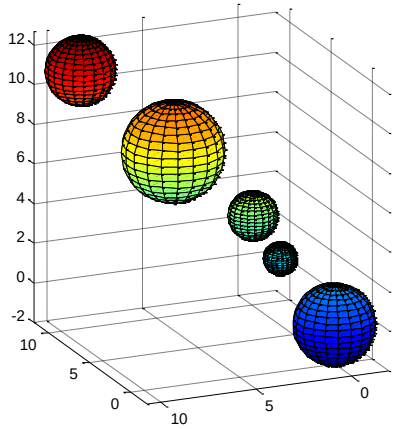


Figure 1. Five independent spheres in 3D dimension

*2.2. Weighted Hyper-sphere SVM*

All training samples are treated equally in basic hyper-sphere SVM. But we know different training samples give different contributions for the resulting hyper-sphere. For example, samples with closer distances to the hyper-sphere center have no contributions for the bounding sphere computation. Only those samples around the radius range are important for the bounding hyper-sphere computation. So, samples with more contributions for the bounding sphere are chosen first in our algorithm. The contribution is based on a weight factor.

In order to distinguish important samples from non-significant samples in the training process, a weight factor called center-distance ratio $s_{ki}$ is defined here. Data center $\bar{\boldsymbol{x}}_k$ for class *k* is computed first and the weight factor for sample $\boldsymbol{x}_i$ is defined in Equation (6).

$$s_{ki} = \frac{\left\| \boldsymbol{x}_i - \bar{\boldsymbol{x}}_k \right\|^2}{\max_j \left\| \boldsymbol{x}_j - \bar{\boldsymbol{x}}_k \right\|^2} \quad i, j = 1, \mathrm{L}\ , l_k \tag{6}$$

Here, $l_k$ is the total number of training samples in class *k*. All training samples are sorted by its weight factor to make sure convergence of the training algorithm is quick. By similar deduction of basic hyper-sphere SVM, the original constrained quadratic optimization problem of our weighted hyper-sphere SVM can be represented as Equation (7).

$$\begin{aligned} &\min_{c_k, R_k} \quad R_k^2 + C\sum_{i=1}^{m} s_{ki}\xi_i \\ &s.t. \\ &\left\|\phi(\boldsymbol{x}_i) - c_k\right\|^2 \le R_k^2 + \xi_i \\ &\xi_i \ge 0, \quad i = 1,\mathrm{L}\ ,l_k \end{aligned} \tag{7}$$

After introducing Lagrange multipliers and deduction, the final dual optimization problem is as Equation (8).

$$\begin{aligned} &\min_{\alpha_i} \quad \sum_{i,j=1}^{l_k} \alpha_i\alpha_j \mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{l_k} \alpha_i \mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}_i) \\ &s.t. \quad \sum_{i=1}^{l_k} \alpha_i = 1 \\ &\qquad 0 \le \alpha_i \le Cs_{ki}, i = 1,\mathrm{L}\ ,l_k \end{aligned} \tag{8}$$

The difference between Equation (4) and Equation (8) is the upper bound of Lagrange multipliers. As the penalty factor, a larger value means a higher penalty to errors, and a smaller value may cause an increase in the number of misclassification data points. With constant values, we can hardly influence the training process. Now, the upper bound of $\alpha_i$ becomes dynamical boundaries, which are functions of membership values. It is very simple to adjust the weighting for important training samples. Like basic hyper-sphere SVM, the resulting decision function for a new sample $\boldsymbol{x}$ can be computed by Equation (5) with restrictions of the hyper-sphere center and radius.

In the training process, data with bigger weight is put into the data processing thread first and then smaller ones to make sure the convergence of the solving quadratic program. To save computation cost, parallel genetic algorithm based SMO multi-threading is adopted for our algorithm, which is illustrated in Figure 2. Genetic algorithm and SMO algorithm are not the focus of this paper. Here, we only list them as one step of our solving process. As mentioned above, training samples of each class are sorted based on their weight factors before calling data processing multithreading. In each data processing thread, samples are added in turn by weight factors to get optimal solutions. When parameters for each hyper-sphere are obtained, all parameters are saved in a file for future test samples.

In the testing process, we will discuss how to classify the test sample falling in the intersection or the inclusion space in Section 3. Here, an inclusion case can be considered as one special intersection case, where one hyper-sphere falls in only another hyper-sphere. For other simple independent cases or tangent cases, the simple decision rule is defined as in [6].

## 3. Decision Rules for Hyper-spheres Intersection

Many researchers have discussed how to improve the generation of hyper-sphere SVMs while decreasing the computation cost at the same time. Considering four positions relationships between resulting classification hyper-spheres and test samples, the intersection is the most complex problem. Figure 3 shows one example of two intersected spheres and two tangent spheres.

If one new sample falls into the intersection, then the classification decision is hard to tell. Classifying samples belonging to the intersection of two or more hyper-spheres is very important for the performance of the hyper-sphere SVMs. There are three effective methods to solve this problem.

The first one is the same or different error hyper-spheres rule. The decision process is based on same error data hyper-sphere and different error data hyper-sphere. Data points that belong to the same class of the mother hyper-sphere are restricted into the same error data hyper-sphere. Data points that belong to the different class of the mother hyper-sphere are restricted into the different error data hyper-sphere. The decision result for new test sample is based on two kinds of error spheres. The second one is multiple sub-hyper-spheres, which partitions the intersection into two or more sub-hyper-spheres to get a more accurate classification space with smaller restricting bounding ranges. The third one is the linear & nonlinear decision rule, which is relatively simple. If training samples of two classes in the intersection are linearly separable by the separation hyper-plane, then this hyper-plane is the decision plane for new test samples. If training samples of two classes in the intersection are nonlinearly separable, then a new SVM resulting hyper-plane is computed for new test samples. Now, we will discuss these three decision rules in detail.

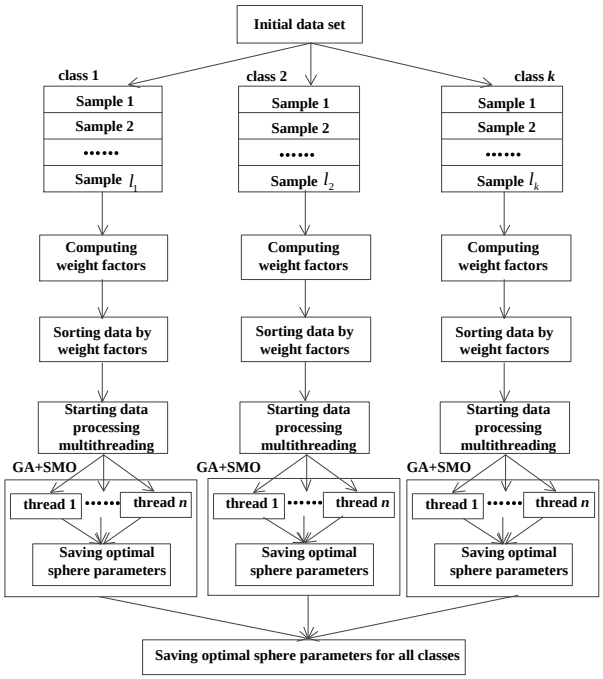


Figure 2. GA+SMO based multithreading training process

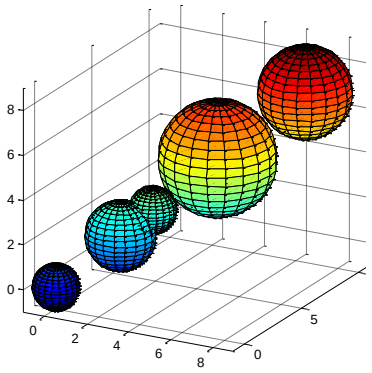

Figure 3. Examples of two intersectant spheres and two tangent spheres in 3D dimension

### *3.1. Same or Different Error Hyper-spheres Rule*

In Figure 4, suppose the minimum bounding sphere for class 1 and class 2 are $S_1$ and $S_2$, the $S_1'$ of class 1 belongs to sphere $S_2$, and the $S_2'$ of class 1 belongs to sphere $S_1$. Obviously, samples such as $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ that belong to $S_1'$ and $S_2'$ are support vectors. For example, $\boldsymbol{x}_2$ is a support vector of $S_2'$, and so $\boldsymbol{x}_2$ belongs to class 2. But, based on the decision function in (5), $f_1(\boldsymbol{x}_2)=0$ and $f_2(\boldsymbol{x}_2)=0$, so $\boldsymbol{x}_2$ belongs to class 1, which is obviously the wrong decision. So, using Equation (5) as the classification rule when two or more hyper-spheres intersect will influence the resulting classification accuracy.

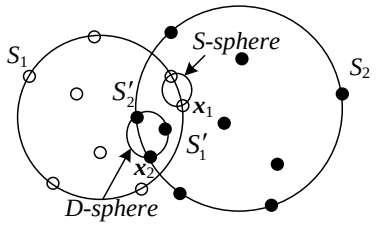


Figure 4. Illustration of same error hyper-sphere (denoted as S-sphere) & different error hyper-sphere (denoted as D-sphere)

To get accurate classification results, Wu proposed a so-called sub-hyper-sphere support vector machine to classify samples in the intersections in [9]. Its decision process is based on the same error data hyper-sphere (data points that belong to the same class of the mother hyper-sphere) and different error data hyper-sphere (data points that belong to different class of the mother hyper-sphere). In Figure 4, *S*-sphere and *D*-sphere are same error data sub-hyper-sphere and different error data hyper-sphere for class 1. If a new test sample lies in the intersection, *S*-sphere and *D*-sphere are used as the classification rule to get the right class.

### *3.2. Multiple Sub-hyper-spheres Classification Rule*

In [6], one multiple sub-hyper-sphere SVM is put forward. This algorithm computed hyper-spheres by using the sequential minimal optimization (SMO) algorithm for all classes first, and then computed position relationships between hyper-spheres based on the distance between sphere centers. For different position relationships, they adopted different decision rules. If two hyper-spheres intersect, they partitioned mother hyper-spheres into sub-hyper-spheres based on the overlap coefficient, which is computed based on the map of key value index. One partition sample is shown in Figure 5.

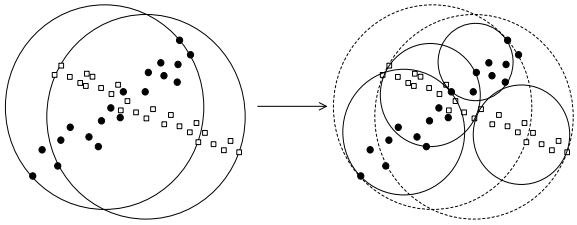

Figure 5. Sample of partition two mother hyper-spheres into multiple sub-hyper-spheres

For the new intersections, the exclusion method was used as a decision strategy if training samples in the intersection belonged to only one class. In the case of linear or nonlinear training samples in the intersections, one similarity function or same error sub-hyper-sphere or different error sub-hyper-sphere was used as decision rule. According to the experiments, this decision rule is effective for most cases. The only disadvantage for this new sub-hyper-sphere SVM is its computation cost. If the number of training samples is very large or the number of training samples that belong to the intersection of two or more hyper-spheres is very large, the proposed algorithm needs more training time and may be invalid in some special cases.

### *3.3. Linear & Nonlinear Classification Rule*

If training samples of two classes in the intersection are linearly separable by the separation hyper-plane, then this hyper-plane is the decision plane for new test samples, which is shown in Figure 6.

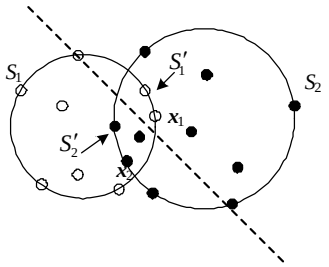


Figure 6. Separation hyper-plane for linear separable samples

If training samples of two classes in the intersection are nonlinearly separable, then a new SVM resulting hyper-plane is computed for new test samples, which is shown in Figure 7.

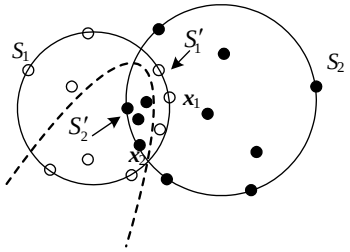


Figure 7. Optimal hyper-plane for non-linear separable samples

### *3.4. New Decision Rule based on kNN*

*k* Nearest Neighbor is a lazy learning algorithm, which defers the decision to generalize beyond the training examples until a new query is encountered. Whenever we have a new point to classify, we find its *k* nearest neighbors from the training data. The distance is measured by Euclidean Distance or Minkowski Distance. Given a query instance $\boldsymbol{x}_q$ to be classified and if $\boldsymbol{x}_1, \boldsymbol{x}_2, \text{L L } , \boldsymbol{x}_k$ denote the *k* instances from training examples that are nearest to $\boldsymbol{x}_q$, then the *k*NN algorithm returns the class that represents the maximum of the *k* instances. But, the similarity metrics in simple *k*NN do not consider the relation of attributes, which results in inaccurate distance and impacts classification precision. That is, the distance usually relates to all the attributes and assumes all of them have the same effects on distance. One better choice is weighted *k*NN, which assigns weights according to the relevance of attributes. It first assigns random weights and then calculates the classification error. According to the error, it adjusts the weights and repeats until an acceptable level of accuracy is reached. Details of weighted *k*NN are not discussed here but can be referenced from [3].

For the classification of test samples falling in the intersections or in the inclusion cases, a new decision rule based on *k*NN for weighted hyper-sphere SVM is shown in Figure 8. If $\boldsymbol{x}_2$ is the point needed to be classified, it belongs to class 2 whether *k* is 3 or 5. This example is relatively simple, but the classification result is satisfactory and can be extended to higher dimensions.

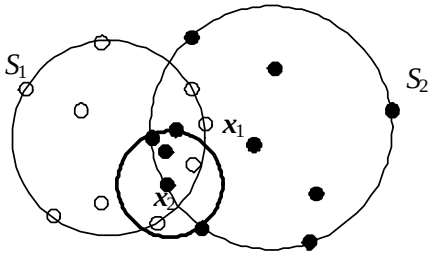


Figure 8. Example of weighted *k*NN for the intersection classification in 2D dimension

As we know, time complexity for *k*NN is O(n). Clearly, the efficiency of the *k*NN decision process is better than the other decision rules.

## 4. Experiments

To verify the validation and effectiveness of our weighted hyper-sphere SVM based on *k*NN decision rule, three real applications are chosen. The first and second application are navigation tracking data classification and ship meeting situations classification. The third one is image classification. Benchmark data sets are not used here because we want to apply our algorithm in real applications for guidance of practical applications, not in theoretical studies.

### *4.1. Navigation Tracking Data Classification based on AIS*

Vessel track plays a very important role in maritime traffic navigation, route planning and design. Wrong decisions for vessels may lead to ship collision accidents, which will lead to a huge loss of human casualties and environmental disasters. Avoiding collisions is an important part of safe and efficient navigation. Analyzing the tracks of different vessels in real-time efficiency will help reduce maritime accidents and its respective damages. Based on vessel tracks with AIS data, different traffic modes can be found and some abnormal cases can be found too. Compared with different traffic tracks shown in electronic chart displays, traffic problems are easily identified in visual formats.

There are three situations involving risk of collision, including Overtaking, Head-on and Crossing cases. From visual displays, the crew can take the right operations to avoid accidents, and the marine management department can find maritime sailing patterns in designated sea areas and can manage various problems. Figure 9 shows three classical classification results based on our new classification algorithm.

There are two vessels in each sub-image, which is denoted with red triangles and blue triangles. Triangle direction is the vessel course direction. Numbers beside each triangle are the time points. For example, time 1 for the red vessel and time 1 for the blue vessel are the same, and so on. The left sub-image is an overtaking case. That is, the red vessel overtakes the blue vessel. The middle sub-image is a head-on case. The red vessel changes its course to avoid the blue vessel. The right sub-

image is a crossing case. The red vessel increases its speed to avoid collision with the blue vessel. At time 4, the red vessel passed the crossing point safely. So, when the blue vessel arrives at the crossing point, there is no collision risk.

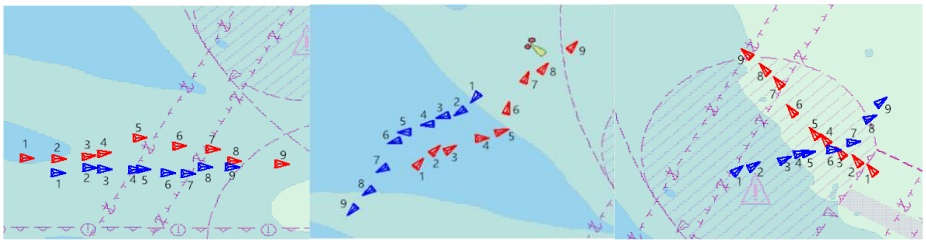

Figure 9. Classification for three collision avoidance situations

These AIS tracks classification results cannot only be used for finding different navigation cases, but also be used to find some abnormal vessels to decide whether there are marine accidents. It can even be used for future route design. Figure 10 shows an example of an abnormal track that turns in opposite directions compared to other normal tracks. So, this vessel must be abnormal and may be dangerous for the maritime department. With automatic processing steps, management of maritime traffic will become more and more intelligent compared to the traditional manual analyzing methods.

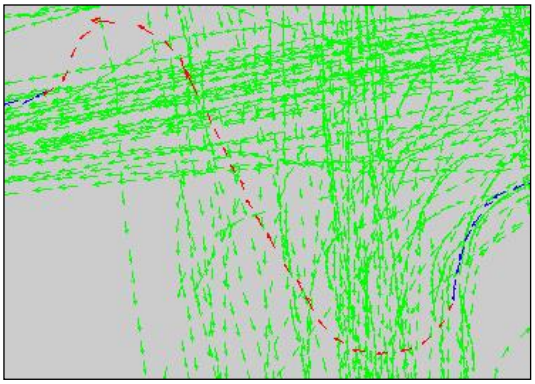

Figure 10. Abnormal track finding in navigation trajectories

Detailed classification is also easy to identify using our algorithm. Typical ship meeting situations such as head on situations and crossing situations are shown in Figure 11 on a large scale. A crossing situation for three vessels is shown in Figure 12.

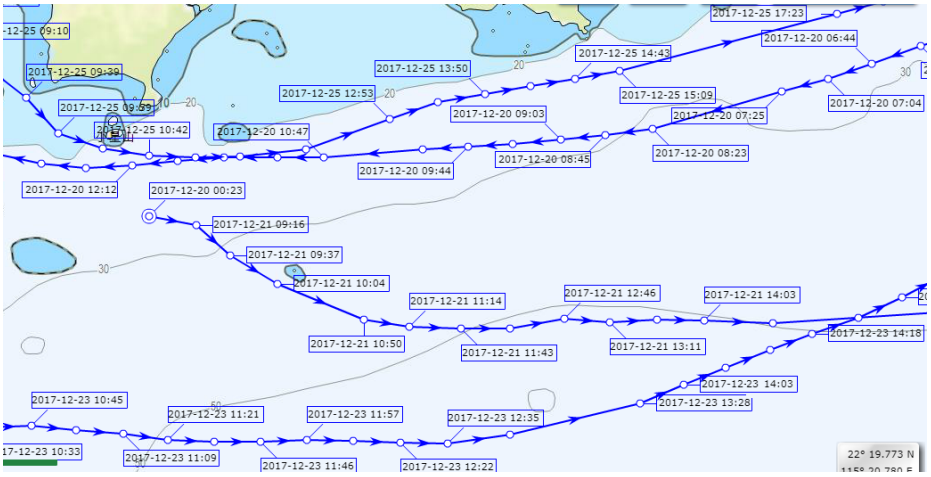


Figure 11. Head on situation & Crossing situation

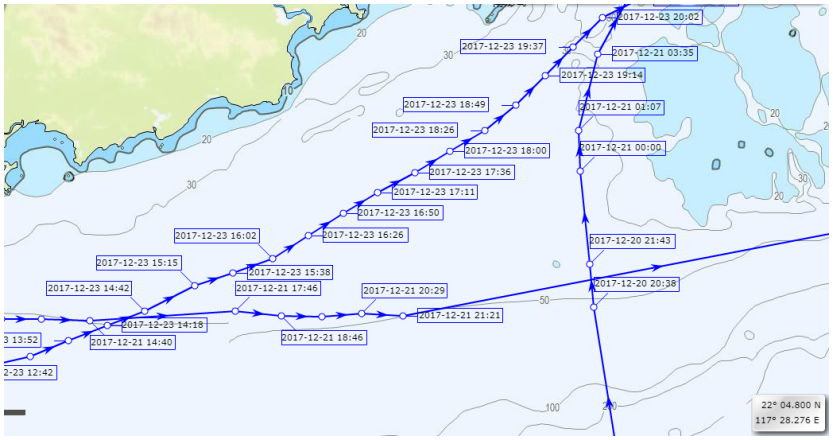


Figure 12. Crossing situation for three vessels

If such cases are identified, some measures should be taken to avoid collisions. According to the provisions of the International Regulations for Preventing Collisions at Sea (COLREGS), ships should adopt some actions to avoid collision. According to Rule 14, two power-driven vessels in a head-on situation when in sight of one another have the same responsibility to avoid collision, so each should alter her course. According to the Rule 15, the vessel that has the other on her own starboard side should keep out of the way if two power-driven vessels are in a crossing situation when in sight of another.

### *4.2. Image Classification*

Original images are collected by mobile phones, iPads or other intelligent terminals. There are 12 pre-defined classes, such as natural scenes, traffic, food, people, sports, arts, animals and so on. After denoising processing, all images are normalized into 800*600 pixels. HOG features 2*2 small blocks in 9 directions are computed for all images. One example of computing HOG features for natural scenes is shown in Figure 13.

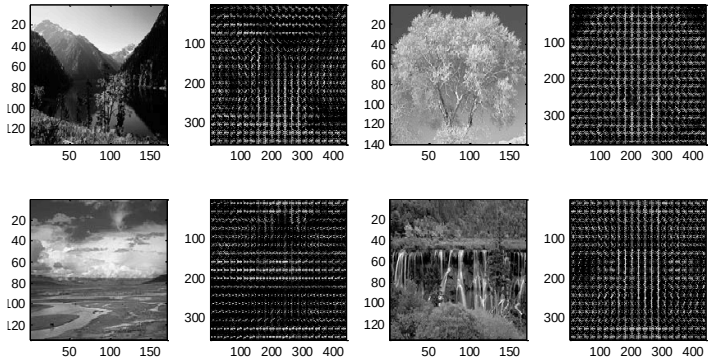

Figure 13. HOG features extraction for classification of natural scenes

800 images are randomly selected for each class and 200 images are selected for testing. RBF kernel function is adopted with $\delta = 1.56, C = 100$ for binary SVM. Comparisons of 1-versus-1 binary SVM, the traditional hyper-sphere SVM, and our new weighted hyper-sphere SVM, are shown in Table 1. Experiments for three classification SVMs are done 10 times and the average numbers are given. From this table, we know that our new algorithm runs faster than other SVM algorithms for the same data set. At the same time, a genetic algorithm is adopted to get optimal sphere parameters. So, accuracy of our algorithm is also higher than other SVMs. Weight factors are also useful for a small number of support vectors, which guarantees a relatively low computation cost for testing.

Table 1. Comparison of Different SVMs

| SVM type | 1-v-1 binary SVM | Traditional hyper-sphere SVM | Our weighted hyper-sphere SVM |
|---|---|---|---|
| Training time | 1026s | 723s | 602s |
| No. of support vectors | 403 | 301 | 276 |
| accuracy | 78.23% | 82.16% | 89.87% |

## 5. Conclusions

Because different training samples have different contributions for the resulting hyper-sphere, a weight factor called center-distance ratio is defined in the paper to distinguish important samples from non-significant samples in the training process. The computation is based on the data center for each training class, and this weight factor is decided by its center-distance ratio. In the training process, data with larger weight is put into the data processing thread first, which is then followed by smaller ones. To save computation cost, a parallel genetic algorithm based SMO multi-threading is adopted. For a test sample, its class decision is based on its position with each classification hyper-sphere. When some hyper-spheres have common spaces, that is, one hyper-sphere intersects with one or more hyper-spheres, a decision rule combining the *k*NN method is put forward in this paper. For other simple inclusion cases, independent cases or tangent cases, a simple decision rule is defined. Through two real experimental results of navigation tracking and ship meeting situations classification, our new proposed algorithm has been shown to be more effective for classification accuracy and boasts a lower computation cost than other algorithms.

## Acknowledgments

This work is partially supported by the National Nature Science Foundation of Liaoning Province with grant no. 2015020099 and the National Natural Science Foundation with grant no.71303031.

## References

1. P. Chen, S. Liu, "Research on Multiple Sub-hyper-sphere Support Vector Machine," *Microelectronics Computer*, vol.31, no.12, pp.28-33, 2014
2. S. F. Ding, F. L. Wu, Z. Z. Shi, "Wavelet Twin Support Vector Machine," *Neural Computing and Applications*, vol.25, no.6, pp. 1241-1247, 2014
3. R. Kohavi, P. Langley, Y. Yun, "The Utility of Feature Weighting in Nearest-Neighbor Algorithms," *Proceedings of the Ninth European Conference on Machine Learning*, Springer-Verlag, pp. 85-92, 1997
4. A. Chittora, O. Mishra, "Face Recognition Using RBF Kernel Based Support Vector Machine," *International Journal of Future Computer and Communication*, vol.1, no.3, pp.280-283, 2012
5. S. Liu, P. Chen, K. Q. Li, "Multiple Sub-hyper-spheres Support Vector Machine for Multi-class Classification," *International Journal of Wavelets Multiresolution and Information Processing*, vol.12, no.3, 1450035, 2014
6. S. Liu, P. Chen and J. Yun, "Fuzzy Hyper-sphere Support Vector Machine for Pattern Recognition," *ICIC Express Letters*, vol.9, no.1, pp. 87-92, 2015
7. John C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," *Technical Report MSR-TR-98-14*, 1998.
8. G. Y. Shi, S. Liu, "Model Selection of C-Support Vector Machines Based on Multi-threading Genetic Algorithm," *International Journal of Wavelets, Multiresolution and Information Processing*, vol.11, no.5, 1350041, 2013
9. Q. Wu, C. Y. Jia and A. F. Zhang, "An Improved Algorithm based on Sphere Structure SVMs and Simulation," *Journal of System Simulation*, 2008, vol.20, no.2, pp. 345–348.
10. Y. K. Xu, L. Qin, G. R. Li, etc. "Online Discriminative Structured Output SVM Learning for Multi-target Tracking," *IEEE Signal Processing Letters*, vol.21, no.2, pp. 190-194, 2014
11. M. L. Zhu, S. F. Chen and X. D. Liu, "Sphere-structured Support Vector Machines for Multi-class Pattern Recognition," *Lecture Notes in Computing Science*, vol.2369, pp.589-593, 2003

**Peng Chen** received his M.S. degree in Computer Science from Liaoning Shihua University, China, in 2003. Currently, he is a Professor with Dalian Neusoft University of Information, China.

**Guoyou Shi** is a Professor in Dalian Maritime University, China.

**Shuang Liu** received her Ph.D. in Traffic Information Engineering and Control from Dalian Maritime University, China. Now, she is an Associate Professor in Dalian Minzu University, China.

**Yuanqiang Zhang** is a Ph.D. candidate at Dalian Maritime University.

**Denis Špelič** obtained his Ph.D. in Computer Science in 2011 from Maribor University. Now, he is a Researcher at the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia.