Ad Hoc Networks 98 (2020) 102055

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Few shot learning for multi-class classification based on nested ensemble DSVM

Wei Wang*, Li Zhang, Mengjun Zhang, Zhixiong Wang

Tianjin Key Laboratory of Wireless Mobile Communications and Power Transmission, Tianjin Normal University, Tianjin 300387, China

ARTICLE INFO

Article history:
Received 3 November 2019
Revised 26 November 2019
Accepted 1 December 2019
Available online 3 December 2019

Keywords:
Few shot learning
Multi-class classification
Deep support vector machine
AdaBoost
Nested ensemble learning

ABSTRACT

Few shot learning (FSL) is a challenge task because a large amount of labeled samples are difficult to acquire due to privacy, ethic issues, safety concerns. Nevertheless, present approaches for multi-class classification usually require massive labeled samples. To bridge this gap, we propose a novel approach named the Nested Ensemble Deep Support Vector Machine (NE-DSVM). In this method, we firstly assign three different single-kernel functions to the deep support vector (DSVM) and ensemble a base classifier in the inner layer. Then, in the outer layer, we use the "one-to-rest" (OvR) strategy to convert the multi-class classification into a binary-class classification. In the last, we combine the AdaBoost framework to complete the classification task. To evaluate the effectiveness of the proposed method, we build a through wall human being detection system and conduct five-fold cross validation experiments on three datasets. We further compare it with SVM, DSVM, AdaBoost-SVM algorithm on the same training and testing data. The experimental results show that the proposed NE-DSVM algorithm lead to encouraging results for few shot learning.

1. Introduction

Deep learning has achieved remarkable achievements in many fields such as computer vision, natural language processing, voice recognition, etc. However, on the success of deep learning requires a large amount of labeled data, which is difficult to obtain in real-word scenarios. Therefore, how to learn effectively under the small-sample condition has become an important research direction, which is referred as few shot learning [1–4].

Generally, FSL refers to the procedure of learning a model with a very small amount of training data. Especially, one-shot learning and zero-learning are the cases of learning with one labeled example and with no labeled example respectively [5–11]. Due to the insufficient samples, FSL is more efficient and moreover can be applied to multiple model classifications [12,13].

Researchers propose three main schemes to solve FSL problem. The first one is based on data-level, where more data is used to avoid the "under-fitting" or "over-fitting" phenomenon. A common technique used is to tap into an extensive collection of external data sources. For example, Qi et al. [14] proposed a novel technique to synthesize samples with multiple tags for the multi-label few-shot classification task. Zhang et al. [15] presented a hallucinating additional data model from the limited exiting samples for employing a saliency network. They also used a saliency network to obtain available image samples and got data points directly in the feature space. Chu et al. [16] searched a deep learning framework based on maximum entropy reinforcement learning and soft Q-Learning to complete positive and negative sampling strategies for FSL. Chen et al. [17] combined a meta-learner with an image deformation network that synthesizes deformed images and achieves state-of-the-art performance on multiple one-shot learning benchmarks. Alfassy et al. [18] tried to combine pairs of exist examples in feature space, which aimed to the synthesized feature vectors would correspond the label sets.

However, based on the data level, the method of using external data tends to deviate sample estimation distribution, which further affects the authenticity of the classification performance. Hence, the metric learning scheme gradually becomes a hot topic. It attempts to learn the feature representation with better generalization ability, so that it can still be applied well in new tasks. In [19], an efficient framework named the Deep Nearest Neighbor Neural Network (DN4) for few-shot learning was explored. It utilized a k-nearest neighbor search over the deep local descriptors of convolutional feature maps to conduct online measure. In [20], three parameter-free improvements were introduced, including optimizing the training process by adapting cross-validation to meta-learning, setting an architecture to localize objects, and expanding the free parameters. In [21], two simple and effective solutions

* Corresponding author.
E-mail address: weiwang@tjnu.edu.cn (W. Wang).

https://doi.org/10.1016/j.adhoc.2019.102055
1570-8705/

were proposed. One is the dense classification over feature maps, and the other is to implant neurons. In [22], an approach was taken to learn a generalizable embedding space for image translation tasks, which was verified on two traffic sign datasets and three logo datasets.

Different from the above-mentioned two schemes, the meta-learning scheme is concentrated on the model parameters and it can handle with new objects under small samples. For example, Lee et al. [23] researched a method named MetaOptNet used these predictors as base classifiers to learn feature embedding for novel categories. Sun et al. [24] utilized deep neural networks to train multiple tasks and learning the functions weights for recognition objects.

The above studies have promoted the development of FSL. Nevertheless, these generating models initialized from prior knowledge training require a large amount of data and high computational costs. There are also certain difficulties in optimizing parameters.

In this paper, we proposed a novel method called nested ensemble Deep Support Vector Machine (NE-DSVM). Different from the traditional ensemble learning approach, the proposed NE-DSVM adopts a two-layer ensemble technology and a nested structure. Each layer performs different tasks based on the advantages of the selection strategy. Specifically, we first assign three different single-kernel functions to DSVMs with regularization capabilities to build the base classifier in the inner layer [25]. Secondly, in the outer layer, we use the "one-to-rest" (OvR) to convert the multi-class classification task into a binary class classification task [26]. Finally, we combine the AdaBoost framework to complete the multi-class classification [27].

To evaluate the performance of the proposed NE-DSVM, we establish the through wall human being detection system under three different statuses. We conduct five-fold cross validation experiments, and use the average accuracy as the metric. We compare the proposed NE-DSVM with the SVM [28,29] the DSVM-RBF and the AdaBoost-SVM algorithm [30,31] on the human target detection dataset, and the experimental results show that our method performs the state-of-the-art method.

The remain of this article is organized as follows. We briefly review the DSVM algorithm in Section 2 and detail the NE-DSVM algorithm in Section 3. In Section 4, we describe the experimental system and analyze the experimental results. The corresponding research summary will be presented in Section 5.

2. Introduction of DSVM

The deep support vector machine (DSVM) algorithm, proposed by Wiering et al. for classification, has been successfully applied to many engineer applications [32]. It is similar to the construction technique of multi-layer perception, which replaces the neurons of the perception with SVM. A two-layer DSVM algorithm module is shown in Fig.1.

In this module, we first input a tagged training dataset $D = \{(x_1, y_i)\}_{i=1}^{N}$, ,

$y_i \in \gamma = \{+1, -1\}$, $i = 1, 2..., N$. Here, x_i is the feature vector and y_i is the class label. These training data are extracted features given to $f_a(a = 1, 2...d)$ by taking $S_a(a = 1, 2...d)$ from the d DSVM kernel layer. Finally, the above features are processed and predicted by a main SVM named M. For the calculation of the feature layer, we use the formula (1):

$$f(x)_a = \sum_{i=1}^{l} (\alpha_i^*(a) - \alpha_i(a))(K(x_i, x)) + b_a \tag{1}$$

where $\alpha_i^*(a)$ and $\alpha_i(a)$ are the coefficients of the SVM$_s$ $S_a(a = 1, 2...d)$. b_a is its corresponding bias. $K(x_i,x)$ represents radial basis function (RBF). The formula (2) involved is:

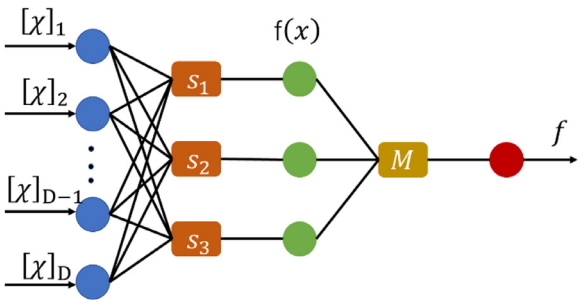


Fig. 1. Module of a two-layer DSVM.

$$K(x_i, x) = \exp\left(-\sum_{a} (x_i - x)^2 / \sigma_m\right) \tag{2}$$

Here σ_m determines the width of the M and the hidden layer RBF kernel function.

Feature layer S_a uses a similar back-propagation technique to obtain the next new input sample and uses formula (3) to reduce errors:

$$\begin{cases} \delta W / \delta [f(x_i)]_a = -(\alpha_i^* - \alpha_i) \sum_{j=1}^{l} (\alpha_j^* - \alpha_j) \delta K(f(x_i), f(x)) / \delta f(x_i)_a \\ \delta K(f(x_i), f(x)) / \delta [f(x_i)]_a = -2\left[(f(x_i)_a, f(x)) / \sigma_m\right] K(f(x_i), f(x_j)) \\ K(f(x_i), f(x)) = \exp\left(-\sum_a (f(x_i)_a - f(x)_a)^2 / \sigma_m\right) \end{cases} \tag{3}$$

The output calculation formula (4) for M is:

$$g(f(x)) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)(K(f(x_i), f(x)) + b \tag{4}$$

After the above calculation, a new min-max optimization problem is obtained, and its function (5) is:

$$\min_{f(x)} \max_{\alpha, \alpha^*} W(f(x), \alpha^{(*)}) = -\varepsilon \sum_{i=1}^{l} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) - 1/2 \sum_{i,j=1} (\alpha_i^* + \alpha_i)(\alpha_j^* - \alpha_j) K(f(x_i), f(x_j)) \tag{5}$$

where $W(f(x), \alpha^{(*)})$ represents the min-max function of the dual target of M. The function $f(x)$ represents the feature-layer, which is obtained by the gradient descent algorithm. $\alpha^{(*)}$, α_i^* and α_i are the parameters of SVMs. Here, the parameter $\alpha_i^{(*)}$ update rule is:

$$\alpha_i^{(*)} \leftarrow \alpha_i^{(*)} + \lambda \partial W / \partial \alpha_i^{(*)} \tag{6}$$

λ indicates the learning rate.

The gradient ascent algorithm is used to calculate the parameter $\alpha^{(*)}$. As described below:

$$\alpha_i = \alpha_i + \lambda\left(-\varepsilon - y_i + \sum_j (\alpha_j^* - \alpha_j) K(f(x_i), f(x_j))\right) \tag{7}$$

Here ε is the fault tolerance value of the loss function.

In summary, the DSVM algorithm is very flexible in adjusting the kernel function. It is easily implemented by the gradient ascent algorithm, the gradient descent algorithm, and the back propagation-like technique. Its output layer SVM has a strong regularization ability, effectively reducing the risk of over-fitting. However, to some extent, the method selects kernel functions to be relatively single and there are still some shortcomings in challenging research tasks such as dealing with few shot learning for multi-class classification.

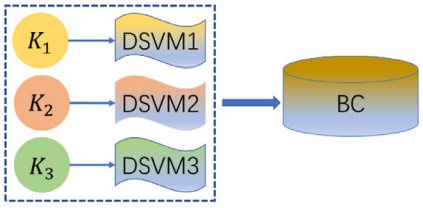


Fig. 2. Construction and Ensemble of the base classifier.

3. Nested ensemble algorithm for multi-class classification

In this section, we first introduce the base classifier construction, and then present the nested ensemble classification algorithm in detail. Finally, we use the "one-versus-rest" (OvR) strategy to solve the multi-classification problem of small samples.

3.1. Base classifier construction

Considering the characteristics of ensemble learning, we choose different kernel functions to obtain sub-classers. For the selection of kernel functions, it is judged by two points: one is to study whether the problem is linear, and the other is that the weak classifier performance is not optimal. In this article, we have selected three different kernel functions: (1) Radial basis function,K_1; (2) Sigmod kernel function,K_2; (3) Polynomial Kernel function, K_3. The corresponding description and expression, as follows:

(1) Radial basis function, also known as Gaussian radial basis function:

$$K_1{=}Kernel\left(X, X'\right) = \exp\left(-c\left\|X - X'\right\|^2\right) \tag{8}$$

Where, the parameter c is the range of the kernel function.

(2) Sigmod function:

$$K_2{=}Kernel\left(X, X'\right) = \tanh\left(\eta\left\langle X, X'\right\rangle + \theta\right) \tag{9}$$

Typically, the parameter$\eta$is required greater than 0, and the parameter θ which meets the condition less than 0.

(3) Polynomial Kernel function:

$$K_3 = Kernel\left(X, X'\right) = \left[\left(X \cdot X'\right) + 1\right]^d, d \in Z^+ \tag{10}$$

The parameter$d$represents the dimension of the kernel function. The parameter$d$gets larger, representing the high-dimension, which makes classification easier.

According to the mentioned kernel functions, we get the DSVM1, DSVM2, DSVM3 models. Then, the ensemble algorithm is used to construct the base classifier (BC), as shown in Fig.2.

Considering that the distribution difference of the dataset and the selection of different kernel functions will affect the experimental results reality. If we set the same weight coefficient for each sub-classifier and utilize the linear weighted method to reflect the results, it can't guarantee higher accuracy. Therefore, we take the following steps and show the pseudo-code named Algorithm 1:

Step1: Choose a kernel function for the SVM model;
Step2: Using the same training and testing data to train and test each SVM model, obtain the correlation coefficient and hyperplane equation expression;
Step3: According to the parameters obtained in *Step2*, the geometric distance $d_i$of all testing samples to the hyperplane can be gotten, as shown in Fig.3. Where, $d_i = {}^{\xi_i}/_{\|w\|}$, ξ_i is a relaxation variable and it should meet$\xi_i \geq 0$, $w$is the normal vector of the hyperplane;

Algorithm 1 Base classifier construction.

Input: Training and Testing data; the kernel functions asK_1,K_2,K_3
Output: the weight of the kernel function$\beta_i(i = 1, 2, 3)$and the final base classifier function$h(x)$
1, Initialize: The sum of the distance from the testing data to the hyperplane:D
The distance from the correct testing data to the hyperplane:d_m($m$is the number of correct testing data)
2, (1) Select the kernel function asK_1
(2) For i = 1to$N$ // N is the number of testing data.
$D = D + d_i$; // $d_i = {}^{\xi_i}/_{\|w\|}(i = 1, 2, \ldots, N)$represents the distance that each //testing data to the hyperplane.
End
(3) Calculate the average distance from all testing data to hyperplanes $\overline{d}$: $\overline{d} = {}^{D}/_{N}$
(4) Calculate the mean square error (MSE) for all testing data to the hyperplanes, MSE:
$MSE = {}^{1}/_{m}\sum_{t=1}^{m}(d_m - \overline{d})^2$ // $d_m$represents the distance from the correct testing data to
// the hyperplane. $m$is the number of correct testing data.
(5) Obtain the weight of the kernel function,β_i : $\beta_i = {}^{1}/_{MSE}(i = 1, 2, 3)$
3, Replace the kernel functionK_2,K_3separately
4, Repeat *Step2*
5, Obtain the final base classifier function:$h(x) = \sum_{i=1}^{I}\beta_i f_i(x), (I = 1, 2, 3)$
// the output function of the sub-classifier$DSVMi(i = 1, 2, 3)$as
//$f_i(x)(i = 1, 2, 3)$

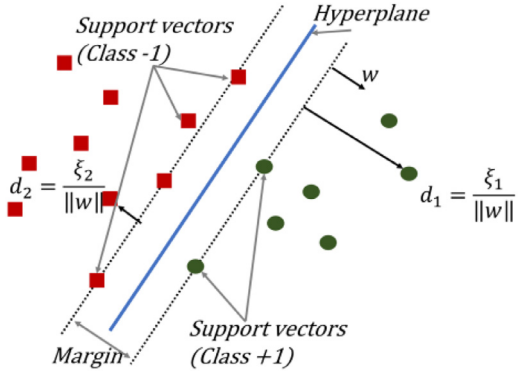


Fig. 3. Example$x_i$to interval boundary distance$d_i$after nonlinear transformation.

Step4: Using the reciprocal of the mean square error (MSE) of all correct testing data geometric distance obtained in *Step3* as the kernel function weight, $\beta_i(i = 1, 2, 3)$. The corresponding expressions is $\beta_i = {}^{1}/_{MSE}(i = 1, 2, 3)$. Here, $MSE = {}^{1}/_{m}\sum_{t=1}^{m}(d_m - \overline{d})^2$. The parameter$m$is the number of correct testing data. The parameter $d_m$represents the distance from the correct testing data to the hyperplane. The parameter $\overline{d}$ is obtained by the equation $\overline{d} = {}^{D}/_{N}$.The parameters $D$and N represent the sum of the distance from the testing data to the hyperplane and the number of testing data, respectively
Step5: Replace the kernel functionK_2,K_3separately.
Step6: Repeat the above-mentioned steps, obtain the sub-classifier$DSVMi(i = 1, 2, 3)$weight $\beta_i(i = 1, 2, 3)$
Step7: We define the output function of the sub-classifier$DSVMi(i = 1, 2, 3)$as$f_i(x)(i = 1, 2, 3)$.
Step8: Finally, construct a linear combination of sub-classifiers to get the final base classifier function:$h(x) = \sum_{i=1}^{I}\beta_i f_i(x), (I = 1, 2, 3)$.

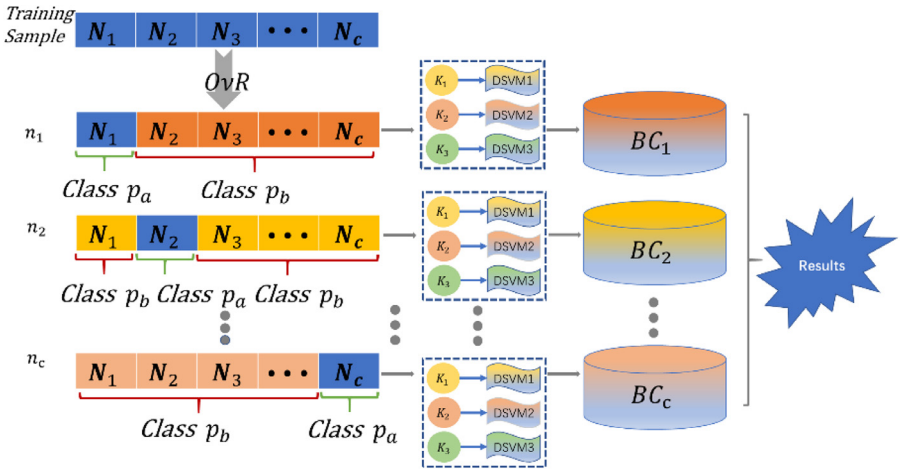


Fig. 4. Multi-classification structure diagram of the Nested Ensemble DSVM algorithm.

Algorithm 2 Nested ensemble classifier with DSVM.

Input: a set of tagged training samples: D ={(x_1,y_1),(x_2,y_2),..., (x_N,y_N)}, where $x_i \in \chi \subseteq R^n$,$y_i \in \gamma = \{+1, -1\}$, $i = 1, 2, ..., N$
Output: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$, $(T = 1, 2, 3)$; T is the number of base classifiers
1, Initialize: the weights of training samples: $w_i = 1/N (i = 1, 2, ..., N)$
2, while
(1) Calculate the base classifier error E_t on the training sample set $D$with the weight distribution:
$E_t = \sum_{i=1}^{I} \beta_i \varepsilon_i (I = 1, 2, 3)$ // ε_i is acquired by the $f_i(x)$ error.
(2) Set the weight of the base classifier$h_t(x)$,$(t = 1, 2, 3)$:
$\alpha_t = \frac{1}{2} \ln(1 - E_t / E_t)$, $(t = 1, 2, 3)$
(3) Update the weights of training samples:
$w_{t+1,i} = w_{t,i} \exp\{-\alpha_t y_i h_t(x_i)\} / Z_t$, $Z_t = \sum_{i=1}^{N} w_{t+1,i} \exp(-\alpha_t y_i h_t(x_i))$ Where, Z_t is a normalization constant and $\sum_{i=1}^{N} w_{t+1,i} = 1$
(4) If E_t>0.5, then break
End while

3.2. Nested ensemble classification algorithm

In this section, we attempt to focus on the nested ensemble algorithm, which is built with the base classifier model in Section 3.1 and the AdaBoost framework [27]. The algorithm description in detail and the pseudo-code named Algorithm 2 are as follows:

First, initialize the weight $w_i(i = 1, 2..., N)$ for the training dataset. Second, the training samples with weight distribution are substituted to the base classifier function $h_t(x)(t = 1, 2, 3)$to get the error rate E_t. Next, the base classifier weight α_i is obtained according to the error rate E_t on training sample using the AdaBoost framework. Then, update the weight $w_{t+1,i}$of the training sample. Change the training sample type and repeat the above steps to get the next base classifier weight. Finally, substitute the testing samples into the base classifier and conduct the linear combination to get the final results.

3.3. Few shot learning for multi-class classification

To solve the few shot learning on multi-class classification problem in practical applications, we propose a corresponding method based on the basic classifier and the AdaBoost framework established in Sections 3.1 and 3.2. We consider multi-classification problems as a set of multiple binary classification problems easy to solve. The three common “decomposition-combination” strategies are: one-versus-one (OvO), one-versus-rest (OvR), one-versus-one-versus-rest (OvOvR) [33–36].

First, we use the "one-versus rest" strategy to split the$c$-classclassification task into $c(c-1)/2$ binary sub-classification tasks. Then, the original data set D ={(x_1,y_1),(x_2,y_2),..., (x_N,y_N)} is subjected to $c$sampling to get c kinds of training data sets suitable for different classifiers, and we divide them into two categories: class p_a and class p_b. Among them, the class p_a is recorded as a positive class. The class p_b is the remaining class p_a, as a negative class. In this way, a binary sub-classifier is obtained. Next, create a base classifier for each training data set, as described in Section 3.1. Then, the final classification results are obtained by weighed average through the nested layers of Section 3.2. Fig.4. shows a multi-classification structure diagram of the nested ensemble algorithm.

4. Experiments and results

In this section, we verify the effectiveness of the Nested Ensemble DSVM (NE-DSVM) algorithm for multi-class classification based on the through wall human being detection experimental system. These experiments are performed on a server with the inter (R) Xeon (R) Silver 4110 CPU 2.10 GHz. Python 2.7, GCC-5.3.0, and MATLAB R2016a are used as programming tools.

4.1. Through wall human being detection experimental system

The experimental equipment used in this paper is a small size, low power operation named a P410MRM single base station radar module. It’s equipped with an antenna port for the dual antenna operation. In the experiment, the system uses the module to detect human being targets behind the wall and acquire the reflected human being status signal, as shown in Fig.5.

We have designed three indoor experimental scenarios, including the unmanned status behind the wall, one-person rapid breathing status behind the wall, and two-person walking slowly status at a speed of 0.1 m/s behind the wall. As shown in Fig.6. The wall involved in the experiment is a brick wall. The thickness of the wall is 25 cm. The distance of the human target from the wall is 100 cm. The distance between the radar equipment and the wall is 60 cm, and it is opposite to the brick wall of the human target.

4.2. Experimental data description

Based on the above experimental system, we have collected three raw datasets according to the three experimental scenarios. They are the unmanned status dataset, a one-person rapid breathing status dataset, and a two-person slowly walking status dataset.

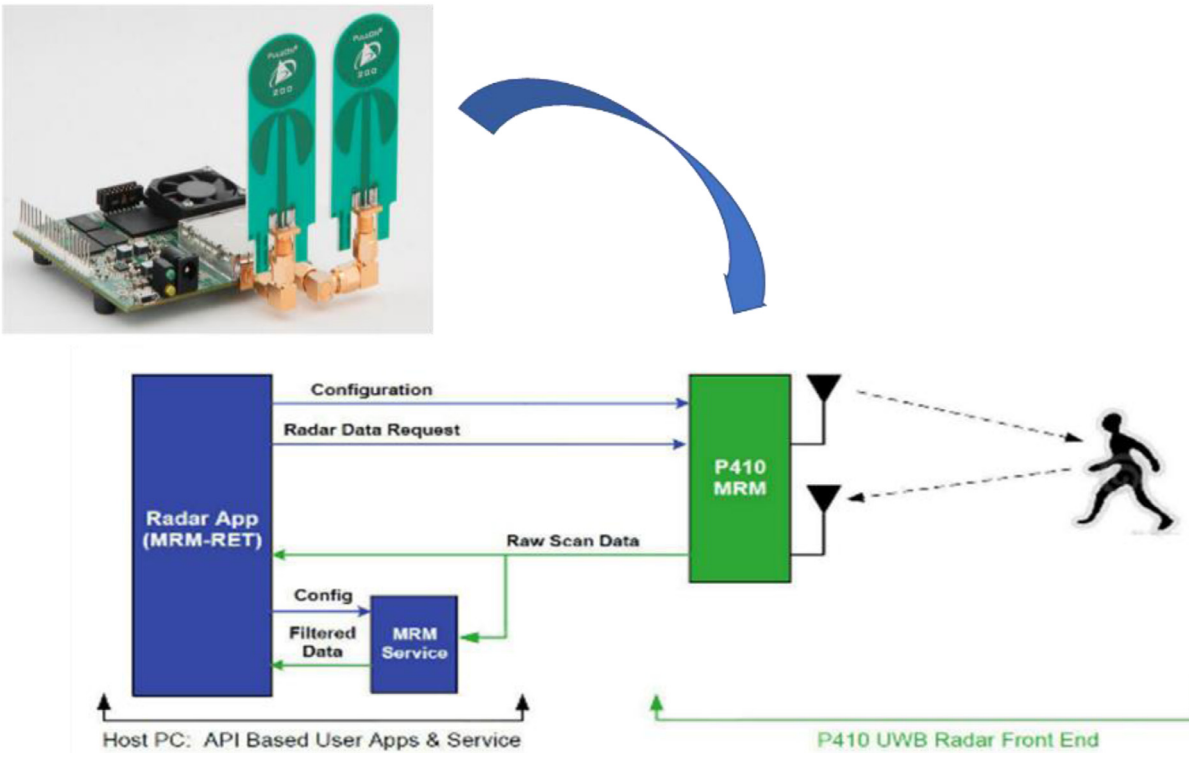


Fig. 5. Through wall human target detection system.

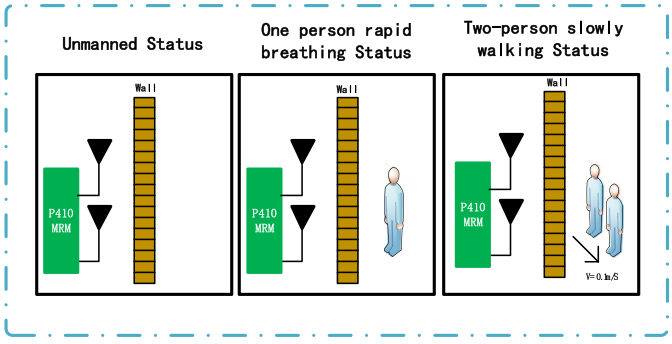


Fig. 6. Indoor experiment scenarios.

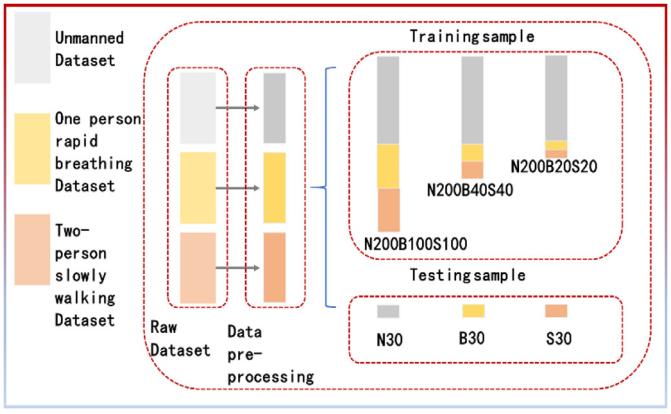


Fig. 7. Through wall human being detection experimental dataset.

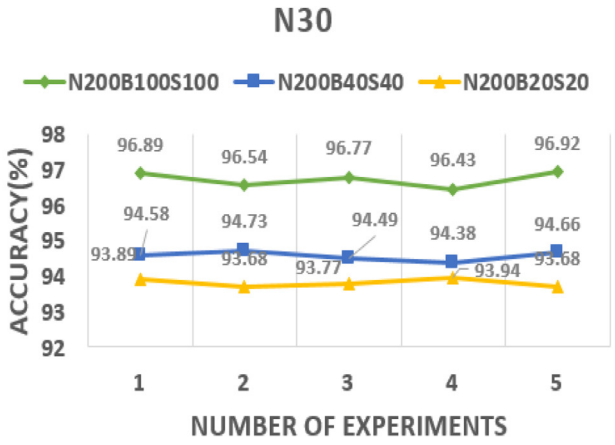
To reduce the computational complexity, we use the dimensionality pre-processing function pcaForSVM in the LibSVM toolbox developed by Prof. Lin of Taiwan University to process the raw dataset. Finally, we got the number of the feature attribute to 37.

According to the results of the discussion on the small sample issue between Jain and Raudys, and the correlation between the number of training samples and the classifier proposed by Fukunaga, it is concluded that the number of training samples n should satisfy the formula $n = \alpha \cdot k$. Where k is the dimension of the feature vector [37–39]. The typical values of a parameter α are 2, 5, 10, etc. In this paper, we have chosen the case where the parameter is 5.

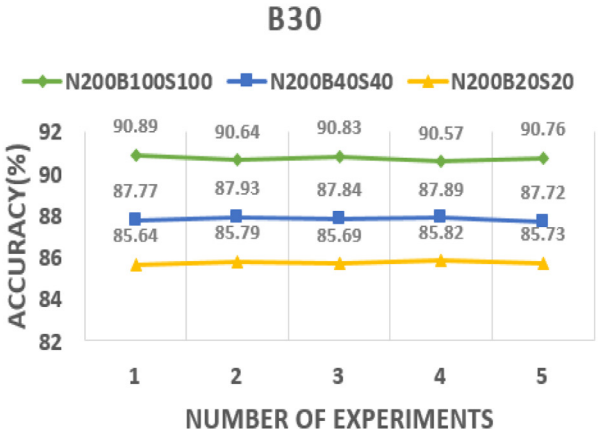
Therefore, the number of training data we selected are: (1) the number of unmanned status behind the wall is 200, the number of one-person rapid breathing status is 100, and the number of two-person walking slowly status is 100, named N200B100S100. (2) The number of unmanned status behind the wall is 200, the number of one-person rapid breathing status is 40, and the number of two-person walking slowly status is 40, named N200B40S40. (3) The number of unmanned status behind the wall is 200, the number of one-person rapid breathing status is 20, and the number of two-person walking slowly status is 20, named N200B20S20. For the testing samples, the number of unmanned status behind the wall is 30, named N30. The number of one-person rapid breathing status is 30, named B30. The number of two-person walking slowly status is 30, named S30. Fig.7 showed the combination of experimental data sets.

4.3. Experimental results and analysis

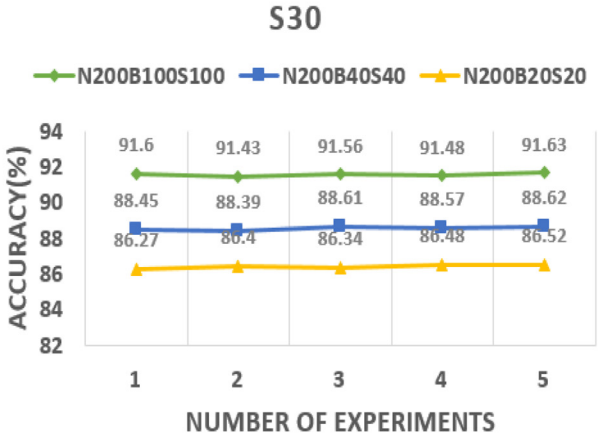
In this study, we firstly repeat each classification scenario five times with different randomly selected training sets, and we report the mean accuracy as the metric for the assessment of classification performance, as shown in Fig.8. It can be seen that the proposed NE-DSVM not only reduced the over-fitting phenomenon, but also achieved the better results.



(a) Five experimental results on the testing sample N30



(b) Five experimental results on the testing sample B30

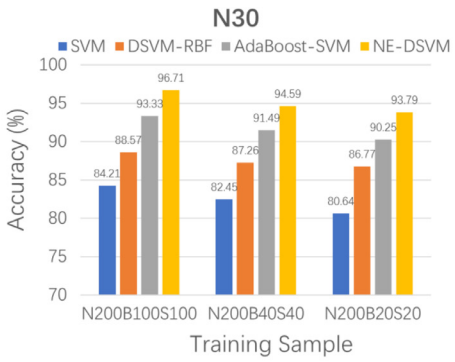


(c) Five experimental results on the testing sample S30

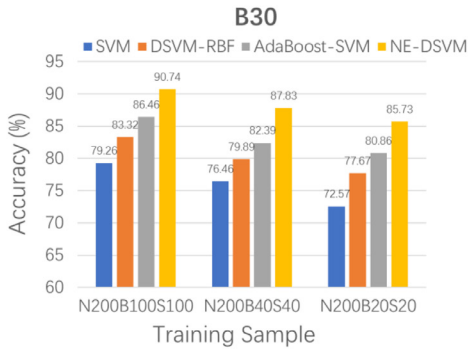
Fig. 8. Five experimental results on the NE-DSVM algorithm.

Secondly, based on the above experiments and results, we further compare it with other algorithms on the same training and testing data. The selected algorithms include SVM, DSVM with RBF kernel function and AdaBoost-SVM. Finally, the experimental results of the four algorithms are shown in Fig.9.

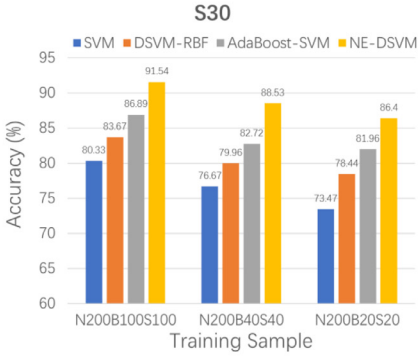
It can be seen that the NE-DSVM algorithm can achieve better accuracy in small samples. Specifically, the higher accuracies of the method on testing sample N30, B30, S30 are 96.71%, 90.74% and 91.54%, respectively. Further, the experimental results based on the approach are increased the range of 11.21% to 13.16%, compared with SVM. It yields improvements from 7.02% to 8.57%, compared with AdaBoost-SVM. Compared with AdaBoost-SVM, the classification accuracy increased ranges from 3.1% to 5.81%. Thus, in the experiment, the NE-DSVM algorithm is verified to be suitable for multi-class imbalance in small sample cases.



(a) Classification accuracy of four algorithms on the testing sample N30



(b) Classification accuracy of four algorithms on the testing sample B30



(c) Classification accuracy of four algorithms on the testing sample S30

Fig. 9. Classification accuracy values of four algorithms on the human target dataset.

5. Conclusion

In this paper, the nested ensemble DSVM (NE-DSVM) algorithm is proposed for imbalanced multi-classification issues in small sample cases. The algorithm adopts twice ensemble technology. First of all, we assign three different single-kernel functions to DSVMs with regularization ability which can build the base classifier in the inner layer. Then, the “one-to-rest” (OvR) strat-

egy and the AdaBoost framework are used in the outer layer to solve the multi-class classification task. We tested the algorithm on three human being detection datasets and compared it with prior methods including SVM, DSVM with the kernel function RBF and AdaBoost-SVM algorithm. The experimental results show that the proposed algorithm avoids the over-fitting phenomenon in small samples and improves classification accuracy effectively. However, the algorithm we proposed still has some limitations. For example, the high-order tensor data can't be used as the input sample, which should only be vector. Hence, we plan to expand this study by adopting tensor theory in the future [40,41]. That is, the model will be converted from the DSVM algorithm to the deep support tensor machine (DSTM) algorithm, which is suitable for many fields. And we also plan to explore the ensemble model to solve the multi-source heterogeneous data problem.

Authors' contributions

Wei Wang and Li Zhang are in charge of the major theoretical analysis, algorithm design, experimental simulation, and writing-original draft. Wang also holds the post of funding acquisition and project administration. Mengjun Zhang is responsible for data collation. Zhixiong Wang is responsible for writing - review & editing. At the same time, Mengjun Zhang and Zhixiong Wang assisted the experimental simulation. All authors read and approved the final manuscript.

Data for reference

Not available online. Please contact corresponding author for data requests.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work was partially supported by the Natural Science Foundation of China (61501326, 61971310).

References

[1] Qi G.J. and Luo. J., 2019. Small data challenges in big data era: a survey of recent progress on unsupervised and semi-supervised methods. arXiv:1903.11260.
[2] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087.
[3] M. Fink, Object classification from a single example utilizing class relevance metrics, in: Advances in Neural Information Processing Systems, 2005, pp. 449–456.
[4] Garcia V. and Bruna J., 2017. Few-shot learning with graph neural networks. arXiv:1711.04043.
[5] F.F. Li, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Trans. Pattern Anal. Mach. Intell. 28 (4) (2006) pp.594–pp.611.
[6] Y. Ji, Y Yang, X Xu, H.T. Shen, One-shot learning based pattern transition map for action early recognition, Signal Process. 143 (2018) pp.364–pp.370.
[7] Shaban A., Bansal S., Liu Z., Essa I. and Boots B., 2017. One-shot learning for semantic segmentation. arXiv:1709.03410.
[8] C.H. Lampert, H. Nickisch, S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 951–958.
[9] S. Changpinyo, W.L. Chao, F. Sha, Predicting visual exemplars of unseen classes for zero-shot learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3476–3485.
[10] B. Romera-Paredes, P. Torr, An embarrassingly simple approach to zero-shot learning, in: International Conference on Machine Learning, 2015, pp. 2152–2161.
[11] Y. Xian, C.H. Lampert, B. Schiele, Z. Akata, Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly, IEEE Trans. Pattern Anal. Mach. Intell. 41 (9) (2018) 2251–2265.
[12] X. Liu, M. Jia, Z. Na, W. Lu, F. Li, Multi-modal cooperative spectrum sensing based on dempster-shafer fusion in 5G-based cognitive radio, IEEE Access 6 (2017) 199–208.
[13] X. Liu, M. Jia, X. Zhang, W. Lu, A novel multi-channel internet of things based on dynamic spectrum sharing in 5G communication, IEEE IoT J. 6 (4) (2018) 5962–5970.
[14] H. Qi, M. Brown, D.G. Lowe, Low-shot learning with imprinted weights, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5822–5830.
[15] H. Zhang, J. Zhang, P. Koniusz, Few-shot learning via saliency-guided hallucination of samples, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2770–2779.
[16] W.H. Chu, Y.J. Li, J.C. Chang, Y.C.F. Wang, Spot and learn: a maximum-entropy patch sampler for few-shot image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 6251–6260.
[17] Z. Chen, Y. Fu, Y.X. Wang, L. Ma, W. Liu, M. Hebert, Image deformation meta-networks for one-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 8680–8689.
[18] A. Alfassy, L. Karlinsky, A. Aides, J. Shtok, S. Harary, R. Feris, R. Giryes, A.M. Bronstein, LaSO: label-Set operations networks for multi-label few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 6548–6557.
[19] W. Li, L. Wang, J Xu, J Huo, Y. Gao, J. Luo, Revisiting local descriptor based image-to-class measure for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. pp.7260–pp.7268.
[20] D. Wertheimer, B. Hariharan, Few-Shot learning with localization in realistic settings, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. pp.6558–pp.6567.
[21] Y. Lifchitz, Y. Avrithis, S. Picard, A. Bursuc, Dense classification and implanting for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9258–9267.
[22] J. Kim, T.H. Oh, S. Lee, F. Pan, I.S. Kweon, Variational prototyping-encoder: one-Shot learning with prototypical images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9462–9470.
[23] K. Lee, S. Maji, A. Ravichandran, S. Soatto, Meta-learning with differentiable convex optimization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10657–10665.
[24] Q. Sun, Y. Liu, T.S. Chua, B. Schiele, Meta-transfer learning for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 403–412.
[25] M.A. Wiering, M. Schutten, A. Millea, A. Meijster, L.R.B. Schomaker, Deep support vector machines for regression problems, in: International Workshop on Advances in Regularization, Optimization, Kernel Methods, and Support Vector Machines, 2013, pp. 53–54.
[26] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit. 44 (8) (2011) pp.1761–pp.1776.
[27] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) pp.119–pp.139.
[28] J.Z. Liang, SVM multi-classifier and web document classification, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), 3, 2004, pp. 1347–1351.
[29] L. Gonzalez-Abril, C. Angulo, F. Velasco, J.A. Ortega, A note on the bias in SVMs for multiclassification, IEEE Trans. Neural Netw. 19 (4) (2008) pp.723–pp.725.
[30] X. Li, L. Wang, E. Sung, AdaBoost with SVM-based component classifiers, Eng. Appl. Artif. Intell. 21 (5) (2008) pp.785–pp.795.
[31] Y.Q. Zhang, M. Ni, C.W. Zhang, S. Liang, S. Fang, R.J. Li, Z.Y. Tan, Research and application of AdaBoost algorithms based on SVM, IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2019.
[32] Wiering M.A. and Schomaker L.R., 2014. Multi-layer support vector machines. Regularization, optimization, kernels, and support vector machines, pp.457–475.
[33] Bennett K. and Mangasarian O.L., 1999. Combining support vector and mathematical programming methods for induction. Advances in Kernel Methods-SV Learning, pp.307–326.
[34] Krebel U.G., 1999. Pairwise classification and support vector machines. Advances in kernel methods: support vector learning, pp.255–268.
[35] C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. 13 (2) (2002) pp.415–pp.425.
[36] C. Angulo, X. Parra, A. Català, K-SVCR. a support vector machine for multi-class classification, Neurocomputing 1 (55) (2003) pp.57–pp.77.
[37] A.K. Jain, B. Chandrasekaran, Dimensionality and sample size considerations in pattern recognition practice, Handbook of statistics 2 (1982) 835–855.
[38] S.J. Raudys, A.K. Jain, Small sample size effects in statistical pattern recognition: recommendations for practitioners, IEEE Trans. Pattern Anal. Mach. Intell. 13 (1991) 252–264.
[39] K. Fukunaga, Introduction to Statistical Pattern Recognition, Elsevier, 2013.
[40] W. Wang, M. Zhang, L. Zhang, Classification of data stream in sensor network with small samples, IEEE IoT J. 6 (4) (2018) 6018–6025.
[41] L. Zhang, W. Wang, Y. Jiang, D. Wang, M. Zhang, Through wall human detection based on support tensor machines, in: International Conference in Communications, Signal Processing, and Systems, 2018, pp. 746–751.

Wei Wang is a Professor of Electronic and Communication Engineering in Tianjin Normal University. He obtained his BEng in Measuring and Controlling Technology and Instrument from University of Science and Technology Beijing in 2003, and he obtained his MS and PhD in Measuring Technology and Instrument from Tianjin University in 2006 and 2010 respectively. His research interests are data stream mining technology in sensor network, including feature extraction and selection, small samples classification, multi-class identification from sensor network data stream.

Li Zhang is currently working toward the Graduate degree in College of Electronic and Communication Engineering, Tianjin, Normal University, Tianjin, China. Her research interests include high-dimensional small sample, deep learning, tensor space and ensemble learning.

Mengjun Zhang is currently working toward the Graduate degree in College of Electronic and Communication Engineering, Tianjin, Normal University, Tianjin, China. Her research interests are multi-source heterogeneous dynamic data stream partitioning and sliding window adaptive adjustment.

Zhixiong Wang is currently working toward the Graduate degree in College of Electronic and Communication Engineering, Tianjin, Normal University, Tianjin, China. His main research direction is concept drift detection and classification algorithm.